DATA WAREHOUSE

A data warehouse is the main repository of the organization's historical data, its corporate memory. For example, an organization would use the information that's stored in its data warehouse to find out what day of the week they sold the most widgets in May 1992, or how employee sick leave the week before the winter break differed between California and New York from 2001-2005. In other words, the data warehouse contains the raw material for management's decision support system. The critical factor leading to the use of a data warehouse is that a data analyst can perform complex queries and analysis on the information without slowing down the operational systems.

While operational systems are optimized for simplicity and speed of modification (online transaction processing, or OLTP) through heavy use of database normalization and an entity-relationship model, the data warehouse is optimized for reporting and analysis (on line analytical processing, or OLAP). Frequently data in data warehouses is heavily denormalised, summarised and/or stored in a dimension-based model but this is not always required to achieve acceptable query response times.
More formally, Bill Inmon (one of the earliest and most influential practitioners) defined a data warehouse as follows:

Subject-oriented, meaning that the data in the database is organized so that all the data elements relating to the same real-world event or object are linked together;

Time-variant, meaning that the changes to the data in the database are tracked and recorded so that reports can be produced showing changes over time;

Non-volatile, meaning that data in the database is never over-written or deleted, once committed, the data is static, read-only, but retained for future reporting;

Integrated, meaning that the database contains data from most or all of an organization's operational applications, and that this data is made consistent History of data warehousing
Data Warehouses became a distinct type of computer database during the late 1980s and early 1990s. They were developed to meet a growing demand for management information and analysis that could not be met by operational systems. Operational systems were unable to meet this need for a range of reasons:

- The processing load of reporting reduced the response time of the operational systems,
- The database designs of operational systems were not optimized for information analysis and reporting,
- Most organizations had more than one operational system, so company-wide reporting could not be supported from a single system, and
- Development of reports in operational systems often required writing specific computer programs which was slow and expensive.

As a result, separate computer databases began to be built that were specifically designed to support management information and analysis purposes. These data warehouses were able to bring in data from a range of different data

sources, such as mainframe computers, minicomputers, as well as personal computers and office automation software such as spreadsheet, and integrate this information in a single place. This capability, coupled with user-friendly reporting tools and freedom from operational impacts, has led to a growth of this type of computer system.
As technology improved (lower cost for more performance) and user requirements increased (faster data load cycle times and more features), data warehouses have evolved through several fundamental stages:

Offline Operational Databases - Data warehouses in this initial stage are developed by simply copying the database of an operational system to an off-line server where the processing load of reporting does not impact on the operational system's performance.

Offline Data Warehouse - Data warehouses in this stage of evolution are updated on a regular time cycle (usually daily, weekly or monthly) from the operational systems and the data is stored in an integrated reporting-oriented data structure

Real Time Data Warehouse - Data warehouses at this stage are updated on a transaction or event basis, every time an operational system performs a transaction (e.g. an order or a delivery or a booking etc.)

Integrated Data Warehouse - Data warehouses at this stage are used to generate activity or transactions that are passed back into the operational systems for use in the daily activity of the organization.

DATA WAREHOUSE ARCHITECTURE

The term data warehouse architecture is primarily used today to describe the overall structure of a Business Intelligence system. Other historical terms include decision support systems (DSS), management information systems (MIS), and others.

The data warehouse architecture describes the overall system from various perspectives such as data, process, and infrastructure needed to communicate the structure, function and interrelationships of each component. The infrastructure or technology perspective details the various hardware and software products used to implement the distinct components of the overall system. The data perspectives typically diagrams the source and target data structures and aid the user in understanding what data assets are available and how they are related. The process perspective is primarily concerned with communicating the process and flow of data from the originating source system through the process of loading the data warehouse, and often the process that client products use to access and extract data from the warehouse.

DATA STORAGE METHOTS

In **OLTP** - online transaction processing systems relational database design use the discipline of data modeling and generally follow the Codd rules of data normalization in order to ensure absolute data integrity. Less

complex information is broken down into its most simple structures (a table) where all of the individual atomic level elements relate to each other and satisfy the normalization rules. Codd defines 5 increasing stringent rules of normalization and typically OLTP systems achieve a 3rd level normalization. Fully normalized OLTP database designs often result in having information from a business transaction stored in dozens to hundreds of tables. Relational database managers are efficient at managing the relationships between tables and result in very fast insert/update performance because only a little bit of data is affected in each relational transaction.

OLTP databases are efficient because they are typically only dealing with the information around a single transaction. In reporting and analysis, thousands to billions of transactions may need to be reassembled imposing a huge workload on the relational database. Given enough time the software can usually return the requested results, but because of the negative performance impact on the machine and all of its hosted applications, data warehousing professionals recommend that reporting databases be physically separated from the OLTP database.

In addition, data warehousing suggests that data be restructured and reformatted to facilitate query and analysis by novice users. OLTP databases are designed to provide good performance by rigidly defined applications built by programmers fluent in the constraints and conventions of the technology. Add in frequent enhancements, and to many a database is just a collection of cryptic names, seemingly unrelated and obscure structures that store data using incomprehensible coding schemes. All factors that while improving performance, complicate use by untrained people. Lastly, the data warehouse needs to support high volumes of data gathered over extended periods of time and are subject to complex queries and need to accommodate formats and definitions of inherited from independently designed package and legacy systems.

Designing the data warehouse data Architecture synergy is the realm of Data Warehouse Architects. The goal of a data warehouse is to bring data together from a variety of existing databases to support management and reporting needs. The generally accepted principle is that data should be stored at its most elemental level because this provides for the most useful and flexible basis for use in reporting and information analysis. However, because of different focus on specific requirements, there can be alternative methods for design and implementing data warehouses. There are two leading approaches to organizing the data in a data warehouse. The dimensional approach advocated by Ralph Kimball and the normalized approach advocated by Bill Inmon. Whilst the dimension approach is very useful in data mart design, it can result in a rats nest of long term data integration and abstraction complications when used in a data warehouse.

In the "dimensional" approach, transaction data is partitioned into either a measured "facts" which are generally numeric data that captures specific values or "dimensions" which contain the reference information that gives each transaction its context. As an example, a sales transaction would be broken up into facts such as the number of products ordered, and the price paid, and dimensions such as date, customer, product, geographical location and salesperson. The main advantages of a dimensional approach is that the data warehouse is easy for business staff with limited information technology experience to understand and use. Also, because the data is pre-joined into the dimensional form, the data warehouse tends to operate very quickly. The main disadvantage of the dimensional approach is that it is quite difficult to add or change later if the company changes the way in which it does business.
The "normalized" approach uses database normalization. In this method, the data in the data warehouse is stored in

third normal form. Tables are then grouped together by subject areas that reflect the general definition of the data (customer, product, finance, etc.). The main advantage of this approach is that it is quite straightforward to add new information into the database -- the primary disadvantage of this approach is that because of the number of tables involved, it can be rather slow to produce information and reports. Furthermore, since the segregation of facts and dimensions is not explicit in this type of data model, it is difficult for users to join the required data elements into meaningful information without a precise understanding of the data structure.

Subject areas are just a method of organizing information and can be defined along any lines. The traditional approach has subjects defined as the subjects or nouns within a problem space. For example, in a financial services business, you might have customers, products and contracts. An alternative approach is to organize around the business transactions, such as customer enrollment, sales and trades.

Advantages of using data warehouse

There are many advantages to using a data warehouse, some of them are:

- Enhances end-user access to a wide variety of data.
- Business decision makers can obtain various kinds of trend reports e.g. the item with the most sales in a particular area / country for the last two years.

A data warehouse can be a significant enabler of commercial business applications, most notably

Customer relationship management (CRM).

Concerns in using data warehouses

- Extracting, cleaning and loading data is time consuming.
- Data warehousing project scope must be actively managed to deliver a release of defined content and value.
- Compatibility problems with systems already in place.
- Security could develop into a serious issue, especially if the data warehouse is web accessible.
- Data Storage design controversy warrants careful consideration and perhaps prototyping of the data warehouse solution for each project's environments.

HISTORY OF DATA WAREHOUSING

Data warehousing emerged for many different reasons as a result of advances in the field of information systems.

A vital discovery that propelled the development of data warehousing was the fundamental differences between operational (transaction processing) systems and informational (decision support) systems. Operational systems are run in real time where in contrast informational systems support decisions on a historical point-in-time. Below is a comparison of the two.

Characteristic	Operational Systems (OLTP)	Informational Systems (OLAP)
Primary Purpose	Run the business on a current basis	Support managerial decision making
Type of Data	Real time based on current data	Snapshots and predictions

Primary Users	Clerks, salespersons, administrators	Managers, analysts, customers
Scope	Narrow, planned, and simple updates and queries	Broad, complex queries and analysis
Design Goal	Performance throughput, availability	Ease of flexible access and use
Database concept	Complex	simple
Normalization	High	Low
Time-focus	Point in time	Period of time
Volume	Many - constant updates and queries on one or a few table rows	Periodic batch updates and queries requiring many or all rows

Other aspects that also contributed for the need of data warehousing are:
• Improvements in database technology
o The beginning of relational data models and relational database management systems (RDBMS)
• Advances in computer hardware
o The abundant use of affordable storage and other architectures
• The importance of end-users in information systems
o The development of interfaces allowing easier use of systems for end users
• Advances in middleware products
o Enabled enterprise database connectivity across heterogeneous platforms

Data warehousing has evolved rapidly since its inception. Here is the story timeline of data warehousing:
1970's – Operational systems (such as data processing) were not able to handle large and frequent requests for data analyses. Data stored was in mainframe files and static databases. A request was processed from recorded tapes for specific queries and data gathering. This proved to be time consuming and an inconvenience.

1980's – Real time computer applications became decentralized. Relational models and database management systems started emerging and becoming the wave. Retrieving data from operational databases still a problem because of "islands of data."

1990's – Data warehousing emerged as a feasible solution to optimize and manipulate data both internally and externally to allow business' to make accurate decisions.
What is data warehousing?

After information technology took the world by storm, there were many revolutionary concepts that were created to make it more effective and helpful. During the nineties as new technology was being born and was becoming obsolete in no time, there was a need for a concrete fool proof idea that can help database administration more secure and reliable. The concept of data warehousing was thus, invented to help the business decision making

process. The working of data warehousing and its applications has been a boon to information technology professionals all over the world. It is very important for all these managers to understand the architecture of how it works and how can it be used as a tool to improve performance. The concept has revolutionized the business planning techniques.

Concept

Information processing and managing a database are the two important components for any business to have a smooth operation. Data warehousing is a concept where the information systems are computerized. Since there would be a lot of applications that run simultaneously, there is a possibility that each individual processes create an exclusive "secondary data" which originates from the source. The data warehouses are useful in tracking all the information down and are useful in analyzing this information and improve performance. They offer a wide variety of options and are highly compatible to virtually all working environments. They help the managers of companies to gauge the progress that is made by the company over a period of time and also explore new ways to improve the growth of the company. There are many "it's" in business and these data warehouses are read only integrated databases that help to answer these questions. They are useful to form a structure of operations and analyze the subject matter on a given time period.

The structure

As is the case with all computer applications there are various steps that are involved in planning a data warehouse. The need is analyzed and most of the time the end user is taken into consideration and their input forms an invaluable asset in building a customized database. The business requirements are analyzed and the "need" is discovered. That would then become the focus area. If a company wants to analyze all its records and use the research in improving performance.

A data warehouse allows the manager to focus on this area. After the need is zeroed in on then a conceptual data model is designed. This model is then used a basic structure that companies follow to build a physical database design. A number of iterations, technical decisions and prototypes are formulated. Then the systems development life cycle of design, development, implementation and support begins.

Collection of data

The project team analyzes various kinds of data that need to go into the database and also where they can find all this information that they can use to build the database. There are two different kinds of data. One which can be found internally in the company and the other is the data that comes from another source. There would be another team of professionals who would work on the creation, extraction programs that are used to collect all the information that is needed from a number of databases, Files or legacy systems. They identify these sources and ten copy them onto a staging area outside the database. They clean all the data which is described as cleansing and make sure that it does not contain any errors. They copy all the data into his data warehouse. This concept of data extraction from the source and the selection, transformation processes have been unique benchmarks of this concept. This is very important for the project to become successful. A lot of meticulous planning is involved in arriving at a step by step configuration of all the data from the source to the data warehouse.

Use of metadata

The whole process of extracting data and collecting it to make it effective component in the operation requires "metadata". The transformation of an analytical system from an operational system is achieved only with maps of Meta data. The transformational data includes the change in names, data changes and the physical characteristics that exist. It also includes the description of the data, its brigand updates. Algorithms are used in summarizing the data.Meta data provides graphical user interface that helps the non-technical end users. This offers richness in navigation and accessing the database. There is other form of Meta data called the operational Meta data. This forms the fundamental structure of accessing the procedures and monitoring the growth of data warehouse in relation with the available storage space. It also recognizes who would be responsible to access the data in the warehouse and in operational systems.

Data marts-specific data

In every data base systems, there is a need for updation. Some of them do it by the day and some by the minute. However if a specific department needs to monitor its own data in sync with the overall business process. They store it as data marts. These are not as big as data arehouse and are useful for storing the data and the information of a specific business module. The latest trend in data warehousing is to develop smaller data marts and then manage each of them individually and later integrate them into the overall business structure.

Security and reliability Similar to information system, trustworthiness of data is determined by the trustworthiness of the hardware, software, and the procedures that created them. The reliability and authenticity of the data and information extracted from the warehouse will be a function of the reliability and authenticity of the warehouse and the various source systems that it encompasses.
In data warehouse environments specifically, there needs to be a means to ensure the integrity of data first by having procedures to control the movement of data to the warehouse from operational systems and second by having controls to protect warehouse data from unauthorized changes. Data warehouse trustworthiness and security are contingent upon acquisition, transformation and access metadata and systems documentation

The basic need for every data base is that it needs to be secure and trustworthy. This is determined by the hardware components of the system the reliability and authenticity of the data and information extracted from the warehouse will be a function of the reliability and authenticity of the warehouse and the various source systems that it encompasses. In data warehouse environments specifically, there needs to be a means to ensure the integrity of data first by having procedures to control the movement of data to the warehouse from operational systems and second by having controls to protect warehouse data from unauthorized changes. Data warehouse trustworthiness and security are contingent upon acquisition, transformation and access metadata and systems documentation.

Han and Kamber (2001) define a data warehouse as "A repository of information collected from multiple sources, stored under a unified scheme, and which usually resides at a single site."
In educational terms, all past information available in electronic format about a school or district such as budget, payroll, student achievement and demographics is stored in one location where it can be accessed using a single set of inquiry tools.

These are some of the drivers that have been created to initiate data warehousing.

• **CRM:** Customer relationship management .there is a threat of losing customers due to poor quality and sometimes those unknown reasons that nobody ever explored. As a result of direct competition, this concept of customer relationship management has been on the forefront to provide the solutions. Data warehousing techniques have helped this cause enormously. Diminishing profit margins: Global competition has forced many companies that enjoyed generous profit margins on their products to reduce their prices to remain competitive. Since cost of goods sold remains constant, companies need to manage their operations better to improve their operating margins

• Data warehouses enable management decision support for managing business operations. Retaining the existing customers has been the most important feature of present day business. To facilitate good customer relationship management companies are investing a lot of money to find out the exact needs of the consumer. As a result of this direct competition the concept of customer relationship management came into existence. Data warehousing techniques have helped this cause enormously. Diminishing profit margins: Global competition has forced many companies that enjoyed generous profit margins on their products to reduce their prices to remain competitive. Since cost of goods sold remains constant, companies need to manage their operations better to improve their operating margins. Data warehouses enable management decision support for managing business operations.

• **Deregulation**: the ever growing competition and the diminishing profit margins have made companies to explore various new possibilities to play the game better. A company develops in one direction and establishes a particular core competency in the market. After they have their own speciality, they look for new avenues to go into a new market with a completely new set of possibilities. For a company to venture into developing a new core competency, the concept of deregulation is very important. . Data warehouses are used to provide this information. Data warehousing is useful in generating a cross reference data base that would help companies to get into cross selling. this is the single most effective way that this can hap

• **The complete life cycle**. The industry is very volatile where we come across a wide range of new products every day and then becoming obsolete in no time. The waiting time for the complete lifecycle often results in a heavy loss of resources of the company. There was a need to build a concept which would help in tracking all the volatile changes and update them by the minute. This allowed companies to be extra safe In regard to all their products. The system is useful in tracking all the changes and helps the business decision process to a great deal. These are also described as business intelligence systems in that aspect.

Merging of businesses: As described above, as a direct result of growing competition, companies join forces to carve a niche in a particular market. This would help the companies to work towards a common goal with twice the number of resources. In case of such an event, there is a huge amount of data that has to be integrated. This data might be on different platforms and different operating systems. To have a centralized authority over the data, it is important that a business tool has to be generated which not only is effective but also reliable. Data warehousing fits the need Relevance of Data Warehousing for organizations Enterprises today, both nationally and globally, are in perpetual search of competitive advantage. An incontrovertible axiom of business management is that information is the key to gaining this advantage. Within this explosion of data are the clues management needs to define its market strategy. Data Warehousing Technology is a means of discovering and unearthing these clues, enabling organizations to competitively position themselves within market sectors. It is an increasingly popular and powerful

concept of applying information technology to solving business problems. Companies use data warehouses to store information for marketing, sales and manufacturing to help managers get a feel for the data and run the business more effectively. Managers use sales data to improve forecasting and planning for brands, product lines and business areas. Retail purchasing managers use warehouses to track fast-moving lines and ensure an adequate supply of high-demand products. Financial analysts use warehouses to manage currency and exchange exposures, oversee cash flow and monitor capital expenditures.

Data warehousing has become very popular among organizations seeking competitive advantage by getting strategic information fast and easy (Adhikari, 1996). The reasons for organizations for having a data warehouse can be grouped into four sections:

• **Warehousing data outside the operational systems**:
The primary concept of data warehousing is that the data stored for business analysis can most effectively be accessed by separating it from the data in the operational systems. Many of the reasons for this separation has evolved over the years. A few years before legacy systems archived data onto tapes as it became inactive and many analysis reports ran from these tapes or data sources to minimize the performance on the operational systems.

• Integrating data from more than one operational system:
Data warehousing are more successful when data can be combined from more than one operational system. When data needs to be brought together from more than one application, it is natural that this integration be done at a place independent of the source application. Before the evolution of structured data warehouses, analysts in many instances would combine data extracted from more than one operational system into a single spreadsheet or a database. The data warehouse may very effectively combine data from multiple source applications such as sales, marketing, finance, and production.

• Data is mostly volatile:
Another key attribute of the data in a data warehouse system is that the data is brought to the warehouse after it has become mostly non-volatile. This means that after the data is in the data warehouse, there are no modifications to be made to this information.

• Data saved for longer periods than in transaction systems:
Data from most operational systems is archived after the data becomes inactive. For example, an order may become inactive after a set period from the fulfillment of the order; or a bank account may become inactive after it has been closed for a period of time. The primary reason for archiving the inactive data has been the performance of the operational system. Large amounts of inactive data mixed with operational live data can significantly degrade the performance of a transaction that is only processing the active data. Since the data warehouses are designed to be the archives for the operational data, the data here is saved for a very long period.

Advantages of data warehouse:

There are several advantages of data warehousing. When companies have a problem that requires necessary changes

in their transaction, they need the information and the transaction processing to make a decision.

• **Time reduction**

"The warehouse has enabled employee to shift their time from collecting information to analyzing it and that helps the company make better business decisions" A data warehouse turns raw information into a useful analytical tool for business decision-making. Most companies want to get the information or transaction processing quickly in order to make a decision-making. If companies are still using traditional online transaction processing systems, it will take longer time to get the information that needed. As a result, the decision-making will be made longer, and the companies will lose time and money. Data warehouse also makes the transaction processing easier.

• **Efficiency**

In order to minimize inconsistent reports and provide the capability for data sharing, the companies should provide a database technology that is required to write and maintain queries and reports. A data warehouse provides, in one central repository, all the metrics necessary to support decision-making throughout the queries and reports. Queries and reports make the management processing be efficient.

• **Complete Documentation**

A typical data warehouse objective is to store all the information including history. This objective comes with its own challenges. Historical data is seldom kept on the operational systems; and, even if it is kept, rarely is found in three or five years of history in one file. There are some reasons why companies need data warehouse to store historical data.

• **Data Integration**

Another primary goal for all data warehouses is to integrate data, because it is a primary deficiency in current decision support. Another reason to integrate data is that the data content in one file is at a different level of granularity than that in another file or that the same data in one file is updated at a different time period than that in another file.

Limitations:

Although data warehouse brings a lot of advantages to corporate, there are some disadvantages that apply to data warehouse.

• **High Cost**

Data warehouse system is too expensive. According to Phil Blackwood, “with the average cost of data warehouse systems valued at$1.8 million”. This limits small companies to buy data warehouse system. As a result, only big companies can afford to buy it. It means that not all companies have proper system to store data and transaction system databases.

Furthermore, because small companies do not have data warehouse, then it causes difficulty for small companies to store data and information in the system that may causes small companies to organize the data as one of the requirement for the company will grow.

• **Complexity**

Moreover, data warehouse is very complex system. The primary function of data warehouse is to integrate all the data and the transaction system database. Because integrate the system is complicated, data warehousing can

complicate business process significantly. For example, small change in the transaction processing system may have major impacts on all transaction processing system. Sometimes, adding, deleting, or changing the data and transaction can causes time consuming. The administrator need to control and check the correctness of changing transaction in order to impact on other transaction. Therefore, complexity of data warehouse prevents the companies from changing the data or transaction that are necessary to make.

Opportunities and Challenges for Data Warehousing

Data warehousing is facing tremendous opportunities and challenges which to a greater part decide the most immediate developments and future trends. Behind all these probable happenings is the impact that the Internet has upon ways of doing business and, consequently, upon data warehousing—a more and more important tool for today's and future's organizations and enterprises. The opportunities and challenges for data warehousing are mainly reflected in four aspects.

• **Data Quality**

Data warehousing has unearthed many previously hidden data-quality problems. Most companies have attempted data warehousing and discovered problems as they integrate information from different business units. Data that was apparently adequate for operational systems has often proved to be inadequate for data warehouses (Faden, 2000). On the other hand, the emergence of E-commerce has also opened up an entirely new source of data-quality problems. As we all know, data, now, may be entered at a Web site directly by a customer, a business partner, or, in some cases, by anyone who visits the site. They are more likely to make mistakes, but, in most cases, less likely to care if they do. All these are "elevating data cleansing from an obscure, specialized technology to a core requirement for data warehousing, cusomer-relationship management, and Web-based commerce "

• **Business Intelligence**

The second challenge comes from the necessity of integrating data warehousing with business intelligence to maximize profits and competency. We have been witnessing an ever-increasing demand to deploy data warehousing structures and business intelligence. The primary purpose of the data warehouse is experiencing a shift from a focus on data transformation into information to—most recently—transformation into intelligence.

All the way down this new development, people will expect more and more analytical function of the data warehouse. The customer profile will be extended with psycho-graphic, behavioral and competitive ownership information as companies attempt to go beyond understanding a customer's preference. In the end, data warehouses will be used to automate actions based on business intelligence. One example is to determine with which supplier the order should be placed in order to achieve delivery as promised to the customer.

• **E-business and the Internet**

Besides the data quality problem we mentioned above, a more profound impact of this new trend on data warehousing is in the nature of data warehousing itself.

On the surface, the rapidly expanding e-business has posed a threat to data warehouse practitioners. They may be concerned that the Internet has surpassed data warehousing in terms of strategic importance to their company, or that Internet development skills are more highly valued than those for data warehousing. They may feel that the Internet and e-business have captured the hearts and minds of business executives, relegating data warehousing to 'second class citizen' status. However, the opposite is true.

• Other trends

While data warehousing is facing so many challenges and opportunities, it also brings opportunities for other fields. Some trends that have just started are as follows:

- More and more small-tier and middle-tier corporations are looking to build their own decision support systems.
- The reengineering of decision support systems more often than not end up with the architecture that would help fuel the growth of their decision support systems.
- Advanced decision support architectures proliferate in response to companies' increasing demands to integrate their customer relationship management and e-business initiatives with their decision support systems.
- More organizations are starting to use data warehousing meta data standards, which allow the various decision support tools to share their data with one another.

Architectural Overview

In concept the architecture required is relatively simple as can be seen from the diagram below:

Source System(s)
Data
Mart
ETL
Transaction Repository
ETL
ETL
ETL
Reporting Tools
Data
Mart
Data
Mart

Figure 1 - Simple Architecture

However this is a very simple design concept and does not reflect what it takes to implement a data warehousing solution. In the next section we look not only at these core components but the additional elements required to make it all work.

White Paper - Overview Architecture for Enterprise Data Warehouses

Components of the Enterprise Data Warehouse

The simple architecture diagram shown at the start of the document shows four core components of an enterprise data warehouse. Real implementations however often have many more depending on the circumstances. In this section we look first at the core components and then look at what other additional components might be needed.
The core components
The core components are those shown on the diagram in Figure 1 – Simple Architecture. They are the ones that are most easily identified and described.

Source Systems
The first component of a data warehouse is the source systems, without which there would be no data. These provide the input into the solution and will require detailed analysis early in any project. Important considerations in looking at these systems include:

- Is this the master of the data you are looking for?
- Who owns/manages/maintains this system?
- Where is the source system in its lifecycle?
- What is the quality of the data in the system?
- What are the batch/backup/upgrade cycles on the system?
- Can we get access to it?

Source systems can broadly be categorised in five types:

On-line Transaction Processing (OLTP) Systems These are the main operational systems of the business and will normally include financial systems, manufacturing systems, and customer relationship management (CRM) systems. These systems will provide the core of any data warehouse but, whilst a large part of the effort will be expended on loading these systems it is the integration of the other sources that provides the value. Legacy Systems Organisations will often have systems that are at the end of their life, or archives of de-commissioned systems. One of the business case justifications for building a data warehouse may have been to remove these systems after the critical data has been moved into the data warehouse. This sort of data often adds to the historical richness of a solution.

Missing or Source-less Data

During the analysis it is often the case that data is identified as required but for which no viable source exists, e.g. exchange rates used on a given date or corporate calendar events, a source that is unusable for loading such as a document, or just that the answer is
in someone.s head. There is also data required for the basic operation such as descriptions of codes.

This is therefore an important category, which is frequently forgotten during the initial design stages, and then requires a last minute fix into the system, often achieved by direct manual changes to the data warehouse. The down side of this approach is that it loses the tracking, control and auditability of the information added to the warehouse. Our advice is therefore to create a system or systems that we call the Warehouse Support Application (WSA). This is normally a number of simple data entry type forms that can capture the data required. This is then treated as another OLTP source and managed in the same way. Organisations are often concerned about how much of this they will have to build. In reality it is a reflection of the level of good data capture during the existing business process and current systems. If
these are good then there will be little or no WSA components to build but if they are poor then significant development will be required and this should also raise a red flag about the readiness of the organisation to

undertake this type of build.

Transactional Repository (TR)

The Transactional Repository is the store of the lowest level of data and thus defines the scope and size of the database. The scope is defined by what tables are available in the data model and the size is defined by the amount of data put into the model. Data that is loaded here will be clean, consistent, and time variant. The design of the data model in this area is critical to the long term success of the data warehouse as it determines the scope and the cost of changes, makes mistakes expensive and inevitably causes delays.

As can be seen from the architecture diagram the transaction repository sits at the heart of the system; it is the point where all data is integrated and the point where history is held. If the model, once in production, is missing key business information and can not easily be xtended when the requirements or the sources change then this will mean significant rework. Avoiding this cost is a factor in the choice of design for this data Model.

In order to design the Transaction Repository there are three data modelling approaches that can be identified. Each lends itself to different organisation types and each has its own advantages and disadvantages, although a detailed discussion of these is outside the scope of this document.

The three approaches are:

Enterprise Data Modelling (Bill Inmon)

This is a data model that starts by using conventional relational modelling techniques and often will describe the business in a conventional normalised database. There may then be a series of de-normalisations for performance and to assist extraction into the
data marts.

This approach is typically used by organisations that have a corporate-wide data model and strong central control by a group such as a strategy team. These organisations will tend also to have more internally developed systems rather than third party products.

Data Bus (Ralph Kimball)

The data model for this type of solution is normally made up of a series of star schemas that have evolved over time, with dimensions becoming .conformed. as they are re-used. The transaction repository is made up of these base star schemas and their associated dimensions. The data marts in the architecture will often just be views either directly onto these schemas or onto aggregates of these star schemas. This approach is particularly suitable for companies which have evolved from a number of independent data marts and growing and evolving into a more mature data warehouse environment. Process Neutral Model

A Process Neutral Data Model is a data model in which all embedded business rules have been removed. If this is done correctly then as business processes change there should be little or no change required to the data model. Business Intelligence solutions designed around such a model should therefore not be subject to limitations as the business changes.

This is achieved both by making many relationships optional and have multiple cardinality, and by carefully making sure the model is generic rather then reflecting only the views and needs of one or more specific business areas. Although this sounds simple (and it is once you get used to it) in reality it takes a little while to fully understand and

to be able to achieve. This type of data model has been used by a number of very large organisations where it combines some of the best features of both the data bus approach and enterprise data modelling. As with enterprise data modelling it sets out to describe the entire business
but rather than normalise data it uses an approach that embeds the metadata (or data about data) in the data model and often contains natural star schemas. This approach is generally used by large corporations that have one or more of the following attributes: many legacy systems, a number of systems as a result of business acquisitions, no central data model, or
a rapidly changing corporate environment.

Data Marts

The data marts are areas of a database where the data is organised for user queries, reporting and analysis. Just as with the design of the Transaction Repository there are a number of design types for data mart. The choice depends on factors such as the design of transaction repository and which tools are to be used to query the data marts.
The most commonly used models are star schemas and snowflake schemas where direct database access is made, whilst data cubes are favoured by some tool vendors. It is also possible to have single table solution sets if this meets the business requirement. There is no need for all data marts to have the same design type, as they are user facing it is important that they are fit for purpose for the user and not what suits a purist architecture.

Extract . Transform- Load (ETL) Tools
ETL tools are the backbone of the data warehouse, moving data from source to transaction repository

and on to data marts. They must deal with issues of performance of load for large volumes and with complex transformation of data, in a repeatable, scheduled environment. These tools build the interfaces between components in the architecture and will also often work with data cleansing elements to ensure that the most accurate data is available. The need for a standard approach to ETL design within a project is paramount. Developers will often create an intricate and complicated solution for which there is a simple solution, often requiring little compromise. Any compromise in the deliverable is usually accepted by the
business once they understand these simple approaches will save them a great deal of cash in terms of time taken to design, develop, test and ultimately support.

Analysis and Reporting Tools
Collecting all of the data into a single place and making it available is useless without the ability for users to access the information. This is done with a set of analysis and reporting tools. Any given data warehouse is likely to have more than one tool. The types of tool can be qualified in broadly four categories:

- Simple reporting tools that either produce fixed or simple parameterised reports.
- Complex ad hoc query tools that allow users to build and specify their own queries.
- Statistical and data mining packages that allow users to delve into the information contained within the data.
- .What-if. tools that allow users to extract data and then modify it to role play or simulate scenarios.

Additional Components

In addition to the core components a real data warehouse may require any or all of these components to deliver the solution. The requirement to use a component should be considered by each programme on its own merits.

Literal Staging Area (LSA)

Occasionally, the implementation of the data warehouse encounters environmental problems, particularly with legacy systems (e.g. a mainframe system, which is not easily accessible by applications and tools). In this case it might be necessary to implement a Literal Staging

Area, which creates a literal copy of the source system.s content but in a more convenient environment (e.g. moving mainframe data into an ODBC accessible relational database). This literal staging area then acts as a surrogate for the source system for use by the downstream ETL interfaces.

There are some important benefits associated with implementing an LSA:

- It will make the system more accessible to downstream ETL products.
- It creates a quick win for projects that have been trying to get data off, for example a Mainframe, in a more laborious fashion.
- It is a good place to perform data quality profiling.
- It can be used as a point close to the source to perform data quality cleaning.

Transaction Repository Staging Area (TRS)

ETL loading will often need an area to put intermediate data sets, or working tables, Somewhere which for clarity and ease of management should not be in the same area as the main model. This area is used when bringing data from a source system or its surrogate into the transaction repository.

Data Mart Staging Area (DMS)

As with the transaction repository staging area there is a need for space between the transaction repository and data marts for intermediate data sets. This area provides that space.

Operational Data Store (ODS)

An operational data store is an area that is used to get data from a source and, if required, lightly aggregate it to make it quickly available. This is required for certain types of reporting which need to be available in .realtime . (updated within 15 minutes) or .near-time. (for example 15 to 60 minutes old). The ODS will not normally clean, integrate, or fully aggregate

data (as the data warehouse does) but it will provide rapid answers, and the data will then become available via the data warehouse once the cleaning, integration and aggregation has taken place in the next batch cycle.

Tools & Technology

The component diagrams above show all the areas and the elements needed. This translates into a significant list of tools and technology that are required to build and operationally run a data warehouse solution. These include:

- Operating system
- Database
- Backup and Recovery
- Extract, Transform, Load (ETL)
- Data Quality Profiling
- Data Quality Cleansing

- Scheduling
- Analysis & Reporting
- Data Modelling
- Metadata Repository
- Source Code Control
- Issue Tracking
- Web based solution integration

The tools selected should operate together to cover all of these areas. The technology choices will also be influenced by whether the organisation needs to operate a homogeneous
(all systems of the same type) or heterogeneous (systems may be of differing types)
environment, and also whether the solution is to be centralised or distributed.

Operating System

The server side operating system is usually an easy decision, normally following the recommendation in the organisation.s Information System strategy. The operating system choice for enterprise data warehouses tends to be a Unix/Linux variant, although some organisations do use Microsoft operating systems. It is not the purpose of this paper to make any recommendations for the above and the choice should be the result of the organisation.s normal procurement procedures.

Database

The database falls into a very similar category to the operating system in that for most organisations it is a given from a select few including Oracle, Sybase, IBM DB2 or Microsoft SQLServer.

Backup and Recovery

This may seem like an obvious requirement but is often overlooked or slipped in at the
end. From .Day 1. of development there will be a need to backup and recover the databases from time to time. The backup poses a number of issues:

- Ideally backups should be done whilst allowing the database to stay up.
- It is not uncommon for elements to be backed up during the day as this is the point of least load on the system and it is often read-only at that point.
- It must handle large volumes of data.
- It must cope with both databases and source data in flat files.

The recovery has to deal with the related consequence of the above:

- Recovery of large databases quickly to a point in time.

Extract - Transform - Load (ETL)

The purpose of the extract, transform and load (ETL) software, to create interfaces, has been described above and is at the core of the data warehouse. The market for such tools is constantly moving, with a trend for database vendors to include this sort of technology in their core product. Some of the considerations for selection of an ETL tool include:

- Ability to access source systems
- Ability to write to target systems
- Cost of development (it is noticeable that some of the easy to deploy and operate tools are not easy to develop with)
- Cost of deployment (it is also noticeable that some of the easiest tools to develop with are not easy to deploy or operate)
- Integration with scheduling tools Typically, only one ETL is needed, however it is common for specialist tools to be used from a source system to a literal staging area as a way of overcoming a limitation in the main ETL

Data Quality Profiling

Data profiling tools look at the data and identify issues with it. It does this by some of the following techniques:

- Looking at individual values in a column to check that they are valid; Validating data types within a column
- Looking for rules about uniqueness or frequencies of certain values
- Validating primary and foreign key constraints +++++
- Validating that data within a row is consistent
- Validating that data is consistent within a table
- Validating that data is consistent across tables; etc.

This is important for both the analysts when examining the system and developers when building the system. It also will identify data quality cleansing rules that can be applied to the data before loading. It is worth noting that good analysts will often do this without tools especially if good analysis templates are available.

Data Quality Cleansing

This tool updates data to improve the overall data quality, often based on the output of the data quality profiling tool. There are essentially two types of cleansing tools:

- **Rule-based cleansing;** this performs updates on the data based on rules (e.g. make everything uppercase; replace two spaces with a single space, etc.). These rules can be very simple or quite complex depending on the tool used and the business requirement.

- **Heuristic cleansing;** this performs cleansing by being given only an approximate method of solving the problem within the context of some goal, and then uses feedback from the effects of the solution to improve its own performance. This is commonly used for address matching type problems.

An important consideration when implementing a cleansing tool is that the process should be performed as closely as possible to the source system. If it is performed further downstream, data will be repeatedly presented for cleansing.

Scheduling

With backup, ETL and batch reporting runs the data warehouse environment has a large number of jobs to be scheduled (typically in the hundreds per day) with many Dependencies, for example:

.The backup can only start at the end of the business day and provided that the source system has generated a flat file, if the file does not exist then it must poll for thirty minutes to see if it arrives otherwise notify an operator. The data mart load can not start until the transaction repository load is complete but then can run six different data mart

loads in parallel.

This should be done via a scheduling tool that integrates into the environment.

Analysis & Reporting

The analysis and reporting tools are the user.s main interface into the system. As has already been discussed there are four main types

- Simple reporting tools
- Complex ad hoc query tools
- Statistical and data mining packages
- What-if tools

Whilst the market for such tools changes constantly the recognised source of information is The OLAP Report2.

Data Modelling

With all the data models that have been discussed it is obvious that a tool in which to build data models is required. This will allow designers to graphically manage data models and generate the code to create the database objects. The tool should be capable of both logical and physical data modelling. Metadata Repository Metadata is data about data. In the case of the data warehouse this will include information about the sources, targets, loading procedures, when those procedures were run, and information about what certain terms mean and how they relate to the data in the database. The metadata required is defined in a subsequent section on documentation however the information itself will need to be held somewhere. Most tools have some elements of a metadata repository but there is a need to identify what constitutes the entire repository by identifying which parts are held in which tools.

2 The OLAP Report by Nigel Pendse and Richard Creeth is an independent research resource for organizations buying and implementing OLAP applications.

Source Code Control

Up to this point you will have noticed that we have steadfastly remained vendor independent and we remain so here. However the issue of source control is one of the biggest impacts on a data warehouse. If the tools that you use do not have versioning control or if your tools do not integrate to allow version control across them and your organisation does not have a source code control tool then download and use CVS, it is free, multi-platform and we have found can be made to work with most of the tools in other categories. There are also Microsoft Windows clients for CVS and web based tools for CVS available.

Issue Tracking

In a similar vein to the issue of source code control most projects do not deal with issue tracking well. The worst nightmare being a spreadsheet that is mailed around once a week to get updates. We again recommend that if a suitable tool is not already available then you consider an open source tool called Bugzilla.

Web Based Solution Integration

Running a programme such as the one described will bring much information together. It is important to bring everything together in an accessible fashion. Fortunately web technologies provide an easy way to do this.

An ideal environment would allow communities to see some or all of the following via a secure web based interface:

- Static reports
- Parameterised reports
- Web based reporting tools
- Balanced Scorecards
- Analysis
- Documentation
- Requirements Library
- Business Terms Definitions
- Schedules
- Metadata Reports
- Data Quality profiles
- Data Quality rules
- Data Quality Reports
- Issue tracking
- Source code

There are two similar but different technologies that are available to do this depending on the corporate approach or philosophy:

- Portals: these provide personalised websites and make use of distributed applications to provide a collaborative workspace.
- Wiki3: which provide a website that allows users to easily add and edit contents and link to other web applications

Both can be very effective in developing common understanding of what the data warehouse does and how it operates which in turn leads to a more engaged user community and greater return on investment.

3 A wiki is a type of website that allows users to easily add and edit content and is especially suited for collaborative writing. In essence, wiki is a simplification of the process of creating HTML web pages combined with a system that records each individual change that occurs over time, so that at any time, a page can be reverted to any of its previous states. A wiki system may also provide various tools that allow the user community to easily monitor the constantly changing state of the wiki and discuss the issues that emerge in trying to achieve a general consensus about wiki content.

Documentation Requirements

Given the size and complexity of the Enterprise Data Warehouse, a core set of documentation is required, which is described in the following section. If a structured project approach is adopted, these documents would be produced as a natural byproduct however we would recommend the following set of documents as a minimum. To facilitate this, at Data Management & Warehousing, we have developed our own set of templates for this purpose.

Requirements Gathering

This is a document managed using a word-processor.

Timescales: At start of project 40 days effort plus on-going updates.

There are four sections to our requirement templates:

- **Facts:** these are the key figures that a business requires. Often these will be associated with Key Performance Indicators (KPIs) and the information required to calculate them i.e. the Metrics required for running the company. An example of a fact might be the number of products sold in a store.
- **Dimensions:** this is the information used to constrain or qualify the facts. An example of this might be the list of products or the date of a transaction or some attribute of the customer who purchased product.
- **Queries:** these are the typical questions that a user might want to ask for example .How many cans of soft drink were sold to male customers on the 2nd February?. This uses information from the requirements sections on available facts and dimensions. Non-functional: these are the requirements that do not directly relate to the data, such as when must the system be available to users, how often does it need to be refreshed, what quality metrics should be recorded about the data, who should be able to access it, etc.

Note that whilst an initial requirements document will come early in the project it will undergo a number of versions as the user community matures in both its use and understanding of the system and data available to it. Key Design Decisions This is a document managed using a word-processor.

Timescales: 0.5 days effort as and when required.

This is a simple one or two page template used to record the design decisions that are made during the project. It contains the issue, the proposed outcome, any counterarguments and why they were rejected and the impact on the various teams within the project. It is important because given the long term nature of such projects there is often a revisionist element that queries why such decisions were made and spends time revisiting them.

Data Model

This is held in the data modelling tool.s internal format.

Timescales: At start of project 20 days effort plus on-going updates. Both logical and physical data models will be required. The logical data model is an abstract representation of a set of data entities and their relationship, usually including their key attributes. The logical data model is intended to facilitate analysis of the function of the data design, and is not intended to be a full representation of the physical database. It is typically produced early in system design, and it is frequently a precursor to the physical data model that documents the actual implementation of the database.

In parallel with the gathering of requirements the data models for the transaction repository and the initial data marts will be developed. These will be constantly maintained throughout the life of the solution.

Analysis

These are documents managed using a word-processor. The analysis phase of the project is broken down into three main templates, each serving as a step in the progression of understanding required to build the system. During the system analysis part of the project, the following three areas must be covered and documented:

Source System Analysis (SSA)

Timescales: 2-3 days effort per source system.

This is a simple high-level overview of each source system to understand its value as a potential source of business information, and to clarify its ownership and longevity. This is normally done for all systems that are potential sources. As the name implies this looks at the .system. level and identifies .candidate. systems.

These documents are only updated at the start of each phase when candidate systems are being identified.

Source Entity Analysis (SEA)

Timescales: 7-10 days effort per system.

This is a detailed look at the .candidate. systems, examining the data, the data quality issues, frequency of update, access rights, etc. The output is a list of tables and fields that are required to populate the data warehouse. These documents are updated at the start of each phase when candidate systems are being examined and as part of the impact analysis of any upgrades to a system that has been used for a previous phase and is being upgraded.

Target Oriented Analysis (TOA)

Timescales: 15-20 days effort for the Transaction Repository, 3-5 days effort for each data mart. This is a document that describes the mappings and transformations that are required to populate a target object. It is important that this is target focused as a common failing is to look at the source and ask the question .Where do I put all these bits of information ?. rather than the correct question which is .I need to populate this object where do I get the information from ?.

Operations Guide

This is a document managed using a word-processor. Timescales: 20 days towards the end of the development phase. This document describes how to operate the system; it will include the schedule for running all the ETL jobs, including dependencies on other jobs and external factors such as the backups or a source system. It will also include instructions on how to

recover from failure and what the escalation procedures for technical problem resolution are. Other sections will include information on current sizing, predicted growth and key data inflection points (e.g. year end where there are a particularly large number of journal entries) It will also include the backup and recovery plan identifying what should be backed up and how to perform system recoveries from backup. Security Model This is a document managed using a word-processor.

Timescales: 10 days effort after the data model is complete, 5 days effort toward the development phase.

This document should identify who can access what data when and where. This can be a complex issue, but the above architecture can simplify this as most access control needs to be around the data marts and nearly everything else will only be visible to the ETL tools extracting and loading data into them.

Issue log

This is held in the issue logging system's internal format.

Timescales: Daily as required.

As has already been identified the project will require an issue log that tracks issues during the development and operation of the system.

Metadata

There are two key categories of metadata as discussed below:

Business Metadata

This is a document managed using a word-processor or a Portal or Wiki if available.

Business Definitions Catalogue4

Timescales: 20 days effort after the requirements are complete and ongoing maintenance.

This is a catalogue of business terms and their definitions. It is all about adding context to data and making meaning explicit and providing definitions to business terms, data elements, acronyms and abbreviations. It will often include information about who owns the definition and who maintains it and where appropriate what formula is required to calculate it. Other useful elements will include synonyms, related terms and preferred terms. Typical examples can include definitions of business terms such as .Net Sales Value. or .Average revenue per customer. as well as definitions of hierarchies and common terms such as customer.
Technical Metadata This is the information created by the system as it is running. It will either be held in server log files or databases.

Server & Database availability

This includes all information about which servers and databases were available when and serves two purposes, firstly monitoring and management of service level agreements (SLAs) and secondly performance optimisation to fit the ETL into the available batch window and to ensure that users have good reporting performance.

ETL Information

This is all the information generated by the ETL process and will include items such as:

- When was a mapping created or changed?
- When was it last run?
- How long did it run for?
- Did it succeed or fail?
- How many records inserted, updated, deleted>

This information is again used to monitor the effective running and operation of the system not only in failure but also by identifying trends such as mappings or transformations whose Performance characteristics are changing.

Query Information

This gathers information about which queries the users are making. The information will include:

- What are the queries that are being run?
- Which tables do they access?
- Which fields are being used?
- How long do queries take to execute?

This information is used to optimise the users experience but also to remove redundant information that is no longer being queried by users.

Some additional high-level guidelines

The following items are just some of the common issues that arise in delivering data warehouse solutions. Whilst not exhaustive they are some of the most important factors to
consider:
Programme or project?
For data warehouse solutions to be successful (and financially viable), it is important for organisations to view the development as a long term programme of work and examine how the work can be broken up into smaller component projects for delivery. This enables many smaller quick wins at different stages of the programme whilst retaining focus on the overall objective.
Examples of this approach may include the development of tactical independent data marts, a literal staging area to

facilitate reporting from a legacy system, or prioritization of the Development of particular reports which can significantly help a particular business function. Most successful data warehouse programmes will have an operational life in excess of ten years with peaks and troughs in development.

The technology trap

At the outset of any data warehouse project organisations frequently fall into the trap of wanting to design the largest, most complex and functionally all-inclusive solution. This will often tempt the technical teams to use the latest, greatest technology promised by a vendor.

However, building a data warehouse is not about creating the biggest database or using the cleverest technology, it is about putting lots of different, often well established, components together so that they can function successfully to meet the organisation.s data management requirements. It also requires sufficient design such that when the next enhancement or extension of the requirement comes along, there is a known and well understood business process and technology path to meet that requirement.

Vendor Selection

This document presents a vendor-neutral view. However, it is important (and perhaps obvious) to note that the products which an organisation chooses to buy will dramatically affect the design and development of the system. In particular most vendors are looking to spread their coverage in the market space. This means that two selected products may have overlapping functionality and therefore which product to use for a given piece of functionality must be identified. It is also important to differentiate between strategic and tactical tools

The other major consideration is that this technology market space changes rapidly. The process, whereby vendors constantly add features similar to those of another competing product, means that few vendors will have a significant long term advantage on features alone. Most features that you will require (rather than those that are sometimes desired) will become available during the lifetime of the programme in market leading products if they are not already there.

The rule of thumb is therefore when assessing products to follow the basic Gartner5 type magic quadrant of .ability to execute. and .completeness of vision. and combine that with your organisations view of the long term relationship it has with the vendor and the fact that a series of rolling upgrades to the technology will be required over the life of the programme.

Development partners

This is one of the thorniest issues for large organisations as they often have policies that outsource development work to third parties and do not want to create internal teams.

In practice the issue can be broken down with programme management and business requirements being sourced internally. Technical design authority is either an external domain expert who transitions to an internal person or an internal person if suitable skills exist.

It is then possible for individual development projects to be outsourced to development partners. In general the market place has more contractors with this type of experience than permanent staff with specialist domain/technology knowledge and so some contractor base either internally or at the development partner is almost inevitable. Ultimately it comes down to the individuals and how they come together as a team, regardless of the supplier and the best teams will be a blend of the best people.

The development and implementation sequence

Data Warehousing on this scale requires a top down approach to requirements and a bottom up approach to the build. In order to deliver a solution it is important to understand what is required of the reports, where that is sourced from in the transaction repository and how in turn the transaction repository is populated from the source system. Conversely the build must start at the bottom and build up through the transaction repository and on to the data marts.

Each build phase will look to either build up (i.e. add another level) or build out (i.e. add another source) This approach means that the project manager can firstly be assured that the final destination will meet the users requirement and that the build can be optimized by using different teams to build up in some areas whilst other teams are building out the underlying levels. Using this model it is also possible to change direction after each completed phase.

Homogeneous & Heterogeneous Environments

This architecture can be deployed using homogeneous or heterogeneous technologies. In a homogeneous environment all the operating systems, databases and other components are built using the same technology, whilst a heterogeneous solution would allow multiple technologies to be used, although it is usually advisable to limit this to one technology per component.

For example using Oracle on UNIX everywhere would be a homogeneous environment, whilst using Sybase for the transaction repository and all staging areas on a UNIX environment and Microsoft SQLServer on Microsoft Windows for the data marts would be an example of a heterogeneous environment.

The trade off between the two deployments is the cost of integration and managing additional skills with a heterogeneous environment compared with the suitability of a single product to fulfil all roles in a homogeneous environment. There is obviously a spectrum of solutions Between the two end points, such as the same operating system but different databases.

Centralised vs. Distributed solutions

This architecture also supports deployment in either a centralised or distributed mode. In a centralised solution all the systems are held at a central data centre, this has the advantage of easy management but may result in a performance impact where users that are remote from the central solution suffer problems over the network. Conversely a distributed solution provides local solutions, which may have a better performance profile for local users but might be more difficult to administer and will suffer from capacity issues when loading the data. Once again there is a spectrum of solutions and therefore there are degrees to which this can be applied. It is normal that centralised solutions are associated with homogeneous environments whilst distributed environments are usually heterogeneous, however this need not always be the case.

Converting Data from Application Centric to User Centric Systems such as ERP systems are effectively systems designed to pump data through a particular business process (application-centric). A data warehouse is designed to look across systems (user-centric) to allow the user to view the data they need to perform their job.

As an example: raising a purchase order in the ERP system is optimised to get the purchase order from being raised, through approval to being sent out. Whilst the data warehouse user may want to look at who is raising orders, the average value, who approves them and how long do they take to do the approval. Requirements should therefore reflect the view of the data warehouse user and not what a single application can provide.

Analysis and Reporting Tool Usage

When buying licences etc. for the analysis and reporting tools a common mistake is to require many thousands for a given reporting tool. Once delivered the number of users never rises to the original estimates. The diagram below illustrates why this occurs:

Flexibility in data access and complexity of tool
Size of user community
Data
Mining
Ad Hoc
Reporting Tools
Parameterised Reporting
Fixed Reporting
Web Based Desktop Tools
Senior Analysts
Business Analysts
Business Users
Customers and Suppliers
Researchers

Figure 5 - Analysis and Reporting Tool Usage

What the diagram shows is that there is a direct, inverse relationship between the degree of reporting flexibility required by a user and the number of users requiring this access.

There will be very few people, typically business analysts and planners at the top but these individuals will need to have tools that really allow them to manipulate and mine the data. At the next level down, there will be a somewhat larger group of users who require ad hoc reporting access, these people will normally be developing or improving reports that get presented to management. The remainder but largest community of the user base will only have a requirement to be presented with data in the form of pre-defined reports with varying degrees of inbuilt flexibility: for instance, managers, sales staff or even suppliers and customers coming into the solution over the internet. This broad community will also influence the choice of tool to reflect the skills of the users. Therefore no individual tool will be perfect and it is a case of fitting the users and a selection of tools together to give the best results.

Data Warehouse .A data warehouse is a subject-oriented, integrated, time-variant and non-volatile collection of data in support of management's decision making process. (Bill Inmon).

Design Pattern A design pattern provides a generic approach, rather than a specific solution for building a particular system or systems.

Dimension Table Dimension tables contain attributes that describe fact records in the fact table.

Distributed Solution A system architecture where the system components are distributed over a number of sites to provide local solutions. DMS Data Mart Staging, a component in the data warehouse architecture for staging data. ERP Enterprise Resource Planning, a business management system that integrates all facets of the business, including planning, manufacturing, sales, and marketing. ETL Extract, Transform and Load. Activities required to populate data warehouses and OLAP applications with clean, consistent, integrated and properly summarized data. Also a component in the data warehouse architecture. Fact Table In an organisation, the .facts. are the key figures that a business requires. Within that organisation.s data mart, the fact table is the foundation from which everything

else arises.

Term Description

Heterogeneous System An environment in which all or any of the operating systems, databases and other components are built using the different technology, and are the integrated by means of customized interfaces.

Heuristic Cleansing Cleansing by means of an approximate method for solving a Problem within the context of a goal. Heuristic cleansing then uses feedback from the effects of its solution to improve its own performance. Homogeneous System An environment in which the operating systems, databases and other components are built using the same technology. KDD Key Design Decision, a project template. KPI Key Performance Indicators. KPIs help an organization define and measure progress toward organizational goals. LSA Literal Staging Area. Data from a legacy system is taken and stored in a database in order to make this data more readily accessible to the downstream systems. A component in the data warehouse architecture. Middleware Software that connects or serves as the "glue" between two otherwise separate applications. Near-time Refers to data being updated by means of batch processing at intervals of in between 15 minutes and 1 hour (in contrast to .Real-time. data, which needs to be updated within 15 minute intervals). Normalisation Database normalization is a process of eliminating duplicated data in a relational database. The key idea is to store data in one location, and provide links to it wherever needed. ODS Operational Data Store, also a component in the data warehouse architecture that allows near-time reporting. OLAP On-Line Analytical Processing. A category of applications and technologies for collecting, managing, processing and presenting multidimensional data for analysis and management purposes.
OLTP OLTP (Online Transaction Processing) is a form of transaction processing conducted via a computer network. Portal A Web site or service that offers a broad array of resources and services, such as e-mail, forums, search engines. Process Neutral Model A Process Neutral Data Model is a data model in which all embedded business rules have been removed. If this is done correctly then as business processes change there should be little or no change required to the data model. Business
Intelligence solutions designed around such a model should therefore not be subject to limitations as the business changes.
Rule Based Cleansing A data cleansing method, which performs updates on the data
based on rules.
SEA Source Entity Analysis, an analysis template.
Snowflake Schema A variant of the star schema with normalized dimension tables.
SSA Source System Analysis, an analysis template.

Term Description

Star Schema A relational database schema for representing multidimensional data. The data is stored in a central fact table, with one or more tables holding information on each dimension. Dimensions have levels, and all levels are usually shown as colum ns in each dimension table.

TOA Target Oriented Analysis, an analysis template. TR Transactional Repository. The collated, clean repository for the lowest level of data held by the organisation and a component in the data warehouse architecture. TRS Transaction Repository Staging, a component in the data warehouse architecture used to stage data. Wiki A wiki is a type of website, or the software needed to operate this website, that allows users to easily add and edit content, and that is particularly suited to collaborative content creation. WSA Warehouse Support Application, a component in the data warehouse architecture that supports missing data. Designing the Star Schema Database

Creating a Star Schema Database is one of the most important, and sometimes the final, step in creating a data warehouse. Given how important this process is to our data warehouse, it is important to understand how me move from a standard, on-line transaction processing (OLTP) system to a final star schema (which here, we will call an OLAP system).

This paper attempts to address some of the issues that have no doubt kept you awake at night. As you stared at the ceiling, wondering how to build a data warehouse, questions began swirling in your mind:

- What is a Data Warehouse? What is a Data Mart?
- What is a Star Schema Database?
- Why do I want/need a Star Schema Database?
- The Star Schema looks very denormalized. Won't I get in trouble for that?
- What do all these terms mean?
- Should I repaint the ceiling?

These are certainly burning questions. This paper will attempt to answer these questions, and show you how to build a star schema database to support decision support within your organization.

Usually, you are bored with terminology at the end of a chapter, or buried in an appendix at the back of the book. Here, however, I have the thrill of presenting some terms up front. The intent is not to bore you earlier than usual, but to present a baseline off of which we can operate. The problem in data warehousing is that the terms are often used loosely by different parties. The Data Warehousing Institute (http://www.dw-institute.com) has attempted to standardize some terms and concepts. I will present my best understanding of the terms I will use throughout this lecture. Please note, however, that I do not speak for the Data Warehousing Institute.

OLTP

OLTP stand for Online Transaction Processing. This is a standard, normalized database structure. OLTP is designed for transactions, which means that inserts, updates, and deletes must be fast. Imagine a call center that takes orders. Call takers are continually taking calls and entering orders that may contain numerous items. Each order and each item must be inserted into a database. Since the performance of the database is critical, we want to maximize the speed of inserts (and updates and deletes). To maximize performance, we typically try to hold as few records in the database as possible.

OLAP and Star Schema

OLAP stands for Online Analytical Processing. OLAP is a term that means many things to many people. Here, we will use the term OLAP and Star Schema pretty much interchangeably. We will assume that a star schema database is an OLAP system. This is not the same thing that Microsoft calls OLAP; they extend OLAP to mean the cube structures built using their product, OLAP Services. Here, we will assume that any system of read-only, historical, aggregated data is an OLAP system.

In addition, we will assume an OLAP/Star Schema can be the same thing as a data warehouse. It can be, although often data warehouses have cube structures built on top of them to speed queries.

Data Warehouse and Data Mart

Before you begin grumbling that I have taken two very different things and lumped them together, let me explain

that Data Warehouses and Data Marts are conceptually different – in scope. However, they are built using the exact same methods and procedures, so I will define them together here, and then discuss the differences.

A data warehouse (or mart) is way of storing data for later retrieval. This retrieval is almost always used to support decision-making in the organization. That is why many data warehouses are considered to be DSS (Decision-Support Systems). You will hear some people argue that not all data warehouses are DSS, and that's fine. Some data warehouses are merely archive copies of data. Still, the full benefit of taking the time to create a star schema, and then possibly cube structures, is to speed the retrieval of data. In other words, it supports queries. These queries are often across time. And why would anyone look at data across time? Perhaps they are looking for trends. And if they are looking for trends, you can bet they are making decisions, such as how much raw material to order. Guess what: that's decision support!

Enough of the soap box. Both a data warehouse and a data mart are storage mechanisms for read-only, historical, aggregated data. By read-only, we mean that the person looking at the data won't be changing it. If a user wants to look at the sales yesterday for a certain product, they should not have the ability to change that number. Of course, if we know that number is wrong, we need to correct it, but more on that later.

The "historical" part may just be a few minutes old, but usually it is at least a day old. A data warehouse usually holds data that goes back a certain period in time, such as five years. In contrast, standard OLTP systems usually only hold data as long as it is "current" or active. An order table, for example, may move orders to an archive table once they have been completed, shipped, and received by the customer.

When we say that data warehouses and data marts hold aggregated data, we need to stress that there are many levels of aggregation in a typical data warehouse. In this section, on the star schema, we will just assume the "base" level of aggregation: all the data in our data warehouse is aggregated to a certain point in time.

Let's look at an example: we sell 2 products, dog food and cat food. Each day, we record sales of each product. At the end of a couple of days, we might have data that looks like this:

		Quantity Sold	
Date	Order Number	Dog Food	Cat Food
4/24/99	1	5	2
	2	3	0
	3	2	6
	4	2	2
	5	3	3
4/25/99	1	3	7
	2	2	1
	3	4	0

Table 1

Now, as you can see, there are several transactions. This is the data we would find in a standard OLTP system. However, our data warehouse would usually not record this level of detail. Instead, we summarize, or aggregate, the data to daily totals. Our records in the data warehouse might look something like this:

	Quantity Sold	
Date	Dog Food	Cat Food
4/24/99	15	13
4/25/99	9	8

Table 2

You can see that we have reduced the number of records by aggregating the individual transaction records into daily records that show the number of each product purchased each day.

We can certainly get from the OLTP system to what we see in the OLAP system just by running a query. However, there are many reasons not to do this, as we will see later.

Aggregations

There is no magic to the term "aggregations." It simply means a summarized, additive value. The level of aggregation in our star schema is open for debate. We will talk about this later. Just realize that almost every star schema is aggregated to some base level, called the grain.

OLTP Systems

OLTP, or Online Transaction Processing, systems are standard, normalized databases. OLTP systems are optimized for inserts, updates, and deletes; in other words, transactions. Transactions in this context can be thought of as the entry, update, or deletion of a record or set of records.

OLTP systems achieve greater speed of transactions through a couple of means: they minimize repeated data, and they limit the number of indexes. First, let's examine the minimization of repeated data.

If we take the concept of an order, we usually think of an order header and then a series of detail records. The header contains information such as an order number, a bill-to address, a ship-to address, a PO number, and other fields. An order detail record is usually a product number, a product description, the quantity ordered, the unit price, the total price, and other fields. Here is what an order might look like:

Order **Order Number 12345**

Ship-To	Bill-To
ACME Products	Jane Doe
123 Main Street	ACME Products
Louisville, KY 40202	123 Main Street
	Louisville, KY 40202

Order Date: 4/24/99 Terms: 2/10 net 30

Product Number	Product Description	Quantity	Price Each	Total
A13J2	1/4" Brass Widget	200	$1.00	$ 200.00
J44RR	1" Copper Sprocket	300	$2.00	$ 600.00
GR1AE	Plastic Gizmo	1000	$0.20	$ 200.00
Total				$1,000.00

Figure 1

Now, the data behind this looks very different. If we had a flat structure, we would see the detail records looking like this:

Order Number	Order Date	Customer ID	Customer Name	Customer Address	Customer City
12345	4/24/99	451	ACME Products	123 Main Street	Louisville
Customer State	Customer	Contact	Contact	Product ID	Product Name

	Zip	Name	Number		
KY	40202	Jane Doe	502-555-1212	A13J2	Widget
Product Description	Category	SubCategory	Product Price	Quantity Ordered	Etc…
¼" Brass Widget	Brass Goods	Widgets	$1.00	200	Etc…

Table 3

Notice, however, that for each detail, we are repeating a lot of information: the entire customer address, the contact information, the product information, etc. We need all of this information for each detail record, but we don't want to have to enter the customer and product information for each record. Therefore, we use relational technology to tie each detail to the header record, without having to repeat the header information in each detail record. The new detail records might look like this:

Order Number	Product Number	Quantity Ordered
12473	A4R12J	200

Table 4

A simplified logical view of the tables might look something like this:

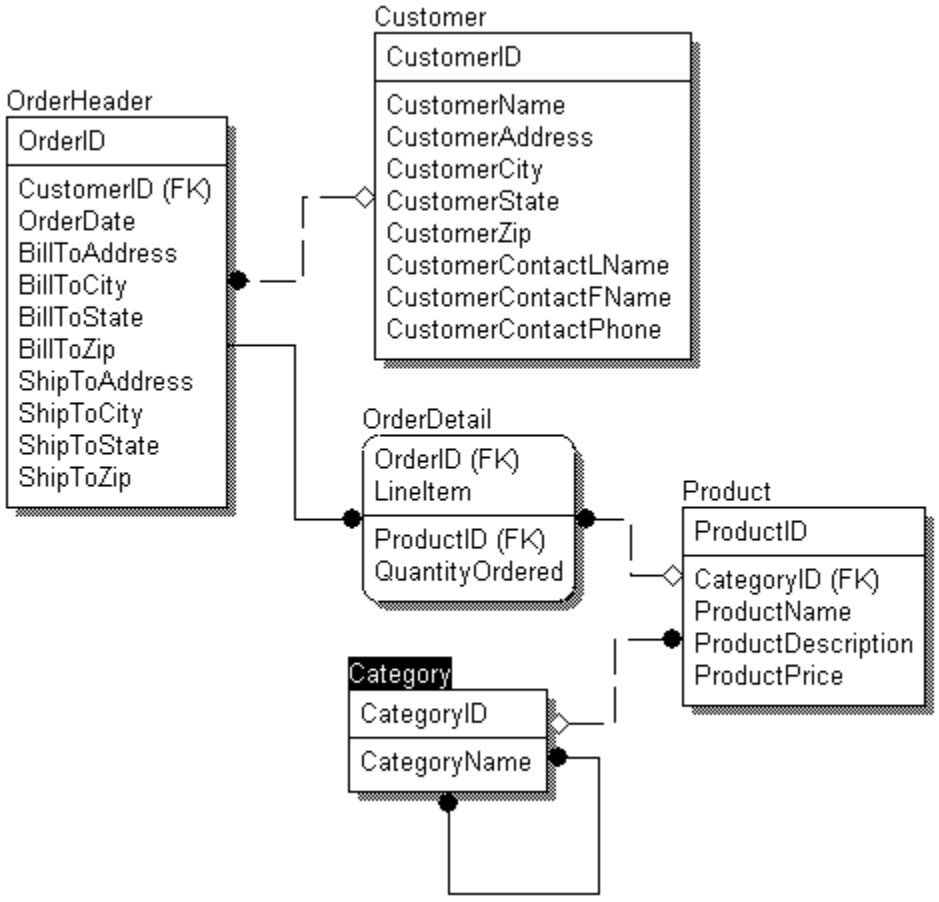


Figure 2

Notice that we do not have the extended cost for each record in the OrderDetail table. This is because we store as

little data as possible to speed inserts, updates, and deletes. Therefore, any number that can be calculated is calculated and not stored.

We also minimize the number of indexes in an OLTP system. Indexes are important, of course, but they slow down inserts, updates, and deletes. Therefore, we use just enough indexes to get by. Over-indexing can significantly decrease performance.

Normalization

Database normalization is basically the process of removing repeated information. As we saw above, we do not want to repeat the order header information in each order detail record. There are a number of rules in database normalization, but we will not go through the entire process.

First and foremost, we want to remove repeated records in a table. For example, we don't want an order table that looks like this:

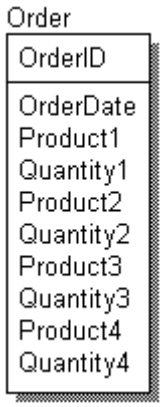


Figure 3

In this example, we will have to have some limit of order detail records in the Order table. If we add 20 repeated sets of fields for detail records, we won't be able to handle that order for 21 products. In addition, if an order just has one product ordered, we still have all those fields wasting space.

So, the first thing we want to do is break those repeated fields into a separate table, and end up with this:

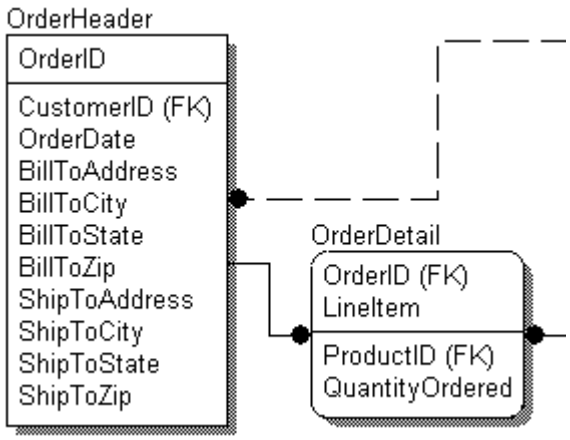


Figure 4

Now, our order can have any number of detail records.

OLTP Advantages

As stated before, OLTP allows us to minimize data entry. For each detail record, we only have to enter the primary key value from the OrderHeader table, and the primary key of the Product table, and then add the order quantity. This greatly reduces the amount of data entry we have to perform to add a product to an order.

Not only does this approach reduce the data entry required, it greatly reduces the size of an OrderDetail record. Compare the size of the records in Table 3 as to that in Table 4. You can see that the OrderDetail records take up much less space when we have a normalized table structure. This means that the table is smaller, which helps speed inserts, updates, and deletes.

In addition to keeping the table smaller, most of the fields that link to other tables are numeric. Queries generally perform much better against numeric fields than they do against text fields. Therefore, replacing a series of text fields with a numeric field can help speed queries. Numeric fields also index faster and more efficiently.

With normalization, we may also have fewer indexes per table. This means that inserts, updates, and deletes run faster, because each insert, update, and delete may affect one or more indexes. Therefore, with each transaction, these indexes must be updated along with the table. This overhead can significantly decrease our performance.

OLTP Disadvantages

There are some disadvantages to an OLTP structure, especially when we go to retrieve the data for analysis. For one, we now must utilize joins and query multiple tables to get all the data we want. Joins tend to be slower than reading from a single table, so we want to minimize the number of tables in any single query. With a normalized structure, we have no choice but to query from multiple tables to get the detail we want on the report.

One of the advantages of OLTP is also a disadvantage: fewer indexes per table. Fewer indexes per table are great for speeding up inserts, updates, and deletes. In general terms, the fewer indexes we have, the faster inserts, updates, and deletes will be. However, again in general terms, the fewer indexes we have, the slower select queries will run. For the purposes of data retrieval, we want a number of indexes available to help speed that retrieval. Since one of our design goals to speed transactions is to minimize the number of indexes, we are limiting ourselves when it comes to doing data retrieval. That is why we look at creating two separate database structures: an OLTP system for transactions, and an OLAP system for data retrieval.

Last but not least, the data in an OLTP system is not user friendly. Most IT professionals would rather not have to create custom reports all day long. Instead, we like to give our customers some query tools and have them create reports without involving us. Most customers, however, don't know how to make sense of the relational nature of the database. Joins are something mysterious, and complex table structures (such as associative tables on a bill-of-material system) are hard for the average customer to use. The structures seem obvious to us, and we sometimes wonder why our customers can't get the hang of it. Remember, however, that our customers know how to do a FIFO-to-LIFO revaluation and other such tasks that we don't want to deal with; therefore, understanding relational concepts just isn't something our customers should have to worry about.

If our customers want to spend the majority of their time performing analysis by looking at the data, we need to support their desire for fast, easy queries. On the other hand, we need to meet the speed requirements of our transaction-processing activities. If these two requirements seem to be in conflict, they are, at least partially. Many companies have solved this by having a second copy of the data in a structure reserved for analysis. This copy is more heavily indexed, and it allows customers to perform large queries against the data without impacting the inserts, updates, and deletes on the main data. This copy of the data is often not just more heavily indexed, but also

denormalized to make it easier for customers to understand.

Reasons to Denormalize

Whenever I ask someone why you would ever want to denormalize, the first (and often only) answer is: speed. We've already discussed some disadvantages to the OLTP structure; it is built for data inserts, updates, and deletes, but not data retrieval. Therefore, we can often squeeze some speed out of it by denormalizing some of the tables and having queries go against fewer tables. These queries are faster because they perform fewer joins to retrieve the same recordset.

Joins are slow, as we have already mentioned. Joins are also confusing to many end users. By denormalizing, we can present the user with a view of the data that is far easier for them to understand. Which view of the data is easier for a typical end-user to understand:

ProductID	ProductName	SupplierID	CategoryID
1	Chai	1	1
2	Chang	1	1
3	Aniseed Syrup	1	2
4	Chef Anton's Cajun Seasoning	2	2
5	Chef Anton's Gumbo Mix	2	2
6	Grandma's Boysenberry Spread	3	2
7	Uncle Bob's Organic Dried Pears	3	7
8	Northwoods Cranberry Sauce	3	2
9	Mishi Kobe Niku	4	6
10	Ikura	4	8
11	Queso Cabrales	5	4

Figure 5

ProductName	SupplierName	CategoryName
Chai	Exotic Liquids	Beverages
Chang	Exotic Liquids	Beverages
Aniseed Syrup	Exotic Liquids	Condiments
Chef Anton's Cajun Seasoning	New Orleans Cajun Delights	Condiments
Chef Anton's Gumbo Mix	New Orleans Cajun Delights	Condiments
Grandma's Boysenberry Spread	Grandma Kelly's Homestead	Condiments
Uncle Bob's Organic Dried Pears	Grandma Kelly's Homestead	Produce
Northwoods Cranberry Sauce	Grandma Kelly's Homestead	Condiments
Mishi Kobe Niku	Tokyo Traders	Meat/Poultry
Ikura	Tokyo Traders	Seafood
Queso Cabrales	Cooperativa de Quesos 'Las ...	Dairy Products

Figure 6

The second view is much easier for the end user to understand. We had to use joins to create this view, but if we put all of this in one table, the user would be able to perform this query without using joins. We could create a view that looks like this, but we are still using joins in the background and therefore not achieving the best performance on the

query.

How We View Information

All of this leads us to the real question: how do we view the data we have stored in our database? This is not the question of how we view it with queries, but how do we logically view it? For example, are these intelligent questions to ask:

- How many bottles of Aniseed Syrup did we sell last week?
- Are overall sales of Condiments up or down this year compared to previous years?
- On a quarterly and then monthly basis, are Dairy Product sales cyclical?
- In what regions are sales down this year compared to the same period last year? What products in those regions account for the greatest percentage of the decrease?

All of these questions would be considered reasonable, perhaps even common. They all have a few things in common. First, there is a time element to each one. Second, they all are looking for aggregated data; they are asking for sums or counts, not individual transactions. Finally, they are looking at data in terms of "by" conditions.

When I talk about "by" conditions, I am referring to looking at data by certain conditions. For example, if we take the question "On a quarterly and then monthly basis, are Dairy Product sales cyclical" we can break this down into this: "We want to see total sales by category (just Dairy Products in this case), by quarter or by month."

Here we are looking at an aggregated value, the sum of sales, by specific criteria. We could add further "by" conditions by saying we wanted to see those sales by brand and then the individual products.

Figuring out the aggregated values we want to see, like the sum of sales dollars or the count of users buying a product, and then figuring out these "by" conditions is what drives the design of our star schema.

Making the Database Match our Expectations

If we want to view our data as aggregated numbers broken down along a series of "by" criteria, why don't we just store data in this format?

That's exactly what we do with the star schema. It is important to realize that OLTP is not meant to be the basis of a decision support system. The "T" in OLTP stands for transactions, and a transaction is all about taking orders and depleting inventory, and not about performing complex analysis to spot trends. Therefore, rather than tie up our OLTP system by performing huge, expensive queries, we build a database structure that maps to the way we see the world.

We see the world much like a cube. We won't talk about cube structures for data storage just yet. Instead, we will talk about building a database structure to support our queries, and we will speed it up further by creating cube structures later.

Facts and Dimensions

When we talk about the way we want to look at data, we usually want to see some sort of aggregated data. These data are called measures. These measures are numeric values that are measurable and additive. For example, our sales dollars are a perfect measure. Every order that comes in generates a certain sales volume measured in some currency. If we sell twenty products in one day, each for five dollars, we generate 100 dollars in total sales. Therefore, sales dollars is one measure we may want to track. We may also want to know how many customers we had that day. Did we have five customers buying an average of four products each, or did we have just one customer buying twenty products? Sales dollars and customer counts are two measures we will want to track.

Just tracking measures isn't enough, however. We need to look at our measures using those "by" conditions. These "by" conditions are called dimensions. When we say we want to know our sales dollars, we almost always mean by day, or by quarter, or by year. There is almost always a time dimension on anything we ask for. We may

also want to know sales by category or by product. These by conditions will map into dimensions: there is almost always a time dimension, and product and geographic dimensions are very common as well.
Therefore, in designing a star schema, our first order of business is usually to determine what we want to see (our measures) and how we want to see it (our dimensions).

Mapping Dimensions into Tables

Dimension tables answer the "why" portion of our question: how do we want to slice the data? For example, we almost always want to view data by time. We often don't care what the grand total for all data happens to be. If our data happen to start on June 14, 1989, do we really care how much our sales have been since that date, or do we really care how one year compares to other years? Comparing one year to a previous year is a form of trend analysis and one of the most common things we do with data in a star schema.
We may also have a location dimension. This allows us to compare the sales in one region to those in another. We may see that sales are weaker in one region than any other region. This may indicate the presence of a new competitor in that area, or a lack of advertising, or some other factor that bears investigation.
When we start building dimension tables, there are a few rules to keep in mind. First, all dimension tables should have a single-field primary key. This key is often just an identity column, consisting of an automatically incrementing number. The value of the primary key is meaningless; our information is stored in the other fields. These other fields contain the full descriptions of what we are after. For example, if we have a Product dimension (which is common) we have fields in it that contain the description, the category name, the sub-category name, etc. These fields do not contain codes that link us to other tables. Because the fields are the full descriptions, the dimension tables are often fat; they contain many large fields.
Dimension tables are often short, however. We may have many products, but even so, the dimension table cannot compare in size to a normal fact table. For example, even if we have 30,000 products in our product table, we may track sales for these products each day for several years. Assuming we actually only sell 3,000 products in any given day, if we track these sales each day for ten years, we end up with this equation: 3,000 products sold X 365 day/year * 10 years equals almost 11,000,000 records! Therefore, in relative terms, a dimension table with 30,000 records will be short compared to the fact table.
Given that a dimension table is fat, it may be tempting to denormalize the dimension table. Resist the urge to do so; we will see why in a little while when we talk about the snowflake schema.

Dimensional Hierarchies

We have been building hierarchical structures in OLTP systems for years. However, hierarchical structures in an OLAP system are different because the hierarchy for the dimension is actually all stored in the dimension table.
The product dimension, for example, contains individual products. Products are normally grouped into categories, and these categories may well contain sub-categories. For instance, a product with a product number of X12JC may actually be a refrigerator. Therefore, it falls into the category of major appliance, and the sub-category of refrigerator. We may have more levels of sub-categories, where we would further classify this product. The key here is that all of this information is stored in the dimension table.
Our dimension table might look something like this:

Figure 7

Notice that both Category and Subcategory are stored in the table and not linked in through joined tables that store the hierarchy information. This hierarchy allows us to perform "drill-down" functions on the data. We can perform a query that performs sums by category. We can then drill-down into that category by calculating sums for the subcategories for that category. We can the calculate the sums for the individual products in a particular subcategory.

The actual sums we are calculating are based on numbers stored in the fact table. We will examine the fact table in more detail later.

Consolidated Dimensional Hierarchies (Star Schemas)

The above example (Figure 7) shows a hierarchy in a dimension table. This is how the dimension tables are built in a star schema; the hierarchies are contained in the individual dimension tables. No additional tables are needed to hold hierarchical information.

Storing the hierarchy in a dimension table allows for the easiest browsing of our dimensional data. In the above example, we could easily choose a category and then list all of that category's subcategories. We would drill-down into the data by choosing an individual subcategory from within the same table. There is no need to join to an external table for any of the hierarchical informaion.

In this overly-simplified example, we have two dimension tables joined to the fact table. We will examine the fact table later. For now, we will assume the fact table has only one number: SalesDollars.

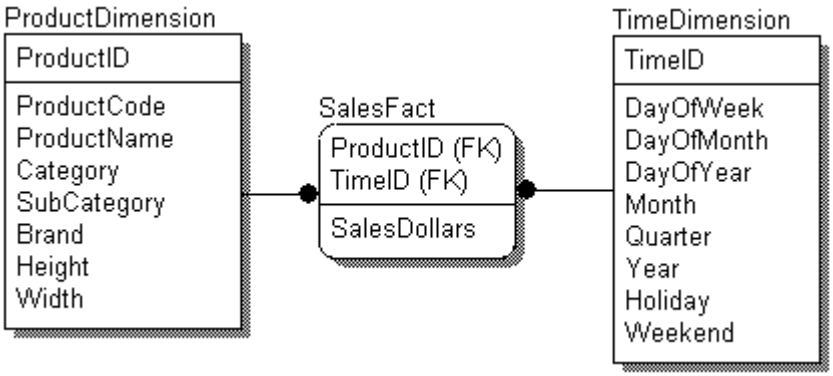


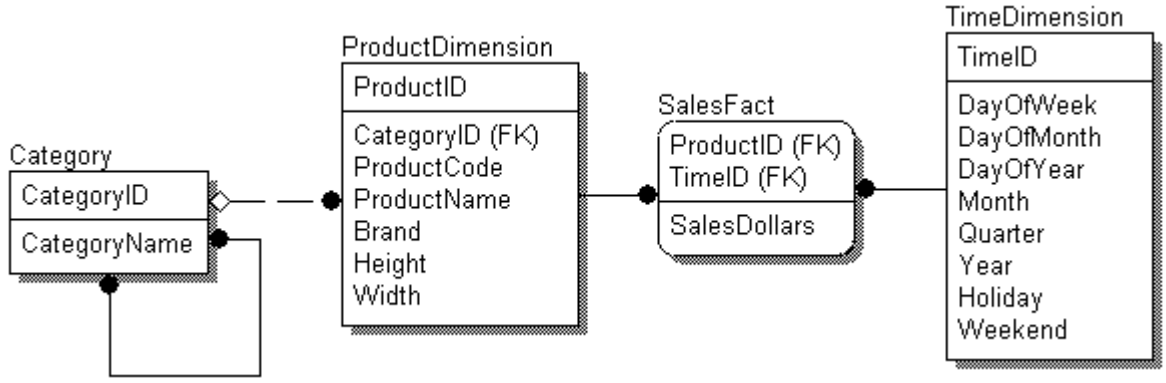
Figure 8

In order to see the total sales for a particular month for a particular category, our SQL query has to be defined.

Snowflake Schemas

Sometimes, the dimension tables have the hierarchies broken out into separate tables. This is a more normalized structure, but leads to more difficult queries and slower response times.

Figure 9 represents the beginning of the snowflake process. The category hierarchy is being broken out of the ProductDimension table. You can see that this structure increases the number of joins and can slow queries. Since the purpose of our OLAP system is to speed queries, snowflaking is usually not something we want to do. Some people try to normalize the dimension tables to save space. However, in the overall scheme of the data warehouse, the dimension tables usually only hold about 1% of the records. Therefore, any space savings from normalizing, or snowflaking, are negligible.



igure 9

Building the Fact Table

The Fact Table holds our measures, or facts. The measures are numeric and additive across some or all of the dimensions. For example, sales are numeric and we can look at total sales for a product, or category, and we can look at total sales by any time period. The sales figures are valid no matter how we slice the data.

While the dimension tables are short and fat, the fact tables are generally long and skinny. They are long because they can hold the number of records represented by the product of the counts in all the dimension tables.

For example, take the following simplified star schema:

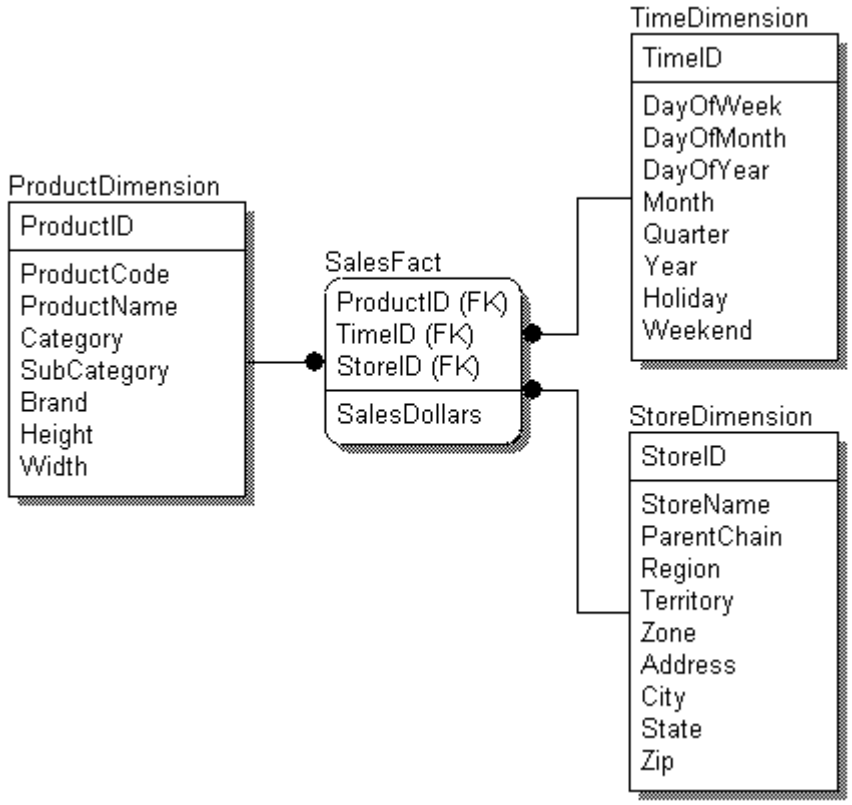


Figure 10

In this schema, we have product, time and store dimensions. If we assume we have ten years of daily data, 200 stores, and we sell 500 products, we have a potential of 365,000,000 records (3650 days * 200 stores * 500 products). As you can see, this makes the fact table long.

The fact table is skinny because of the fields it holds. The primary key is made up of foreign keys that have migrated from the dimension tables. These fields are just some sort of numeric value. In addition, our measures are also numeric. Therefore, the size of each record is generally much smaller than those in our dimension tables. However, we have many, many more records in our fact table.

Fact Granularity

One of the most important decisions in building a star schema is the granularity of the fact table. The granularity, or frequency, of the data is usually determined by the time dimension. For example, you may want to only store weekly or monthly totals. The lower the granularity, the more records you will have in the fact table. The granularity also determines how far you can drill down without returning to the base, transaction-level data.

Many OLAP systems have a daily grain to them. The lower the grain, the more records that we have in the fact table. However, we must also make sure that the grain is low enough to support our decision support needs.

One of the major benefits of the star schema is that the low-level transactions are summarized to the fact table grain. This greatly speeds the queries we perform as part of our decision support. This aggregation is the heart of our OLAP system.

Fact Table Size

We have already seen how 500 products sold in 200 stores and tracked for 10 years could produce 365,000,000 records in a fact table with a daily grain. This, however, is the maximum size for the table. Most of the time, we do not have this many records in the table. One of the things we do not want to do is store zero values. So, if a product

did not sell at a particular store for a particular day, we would not store a zero value. We only store the records that have a value. Therefore, our fact table is often sparsely populated.

Even though the fact table is sparsely populated, it still holds the vast majority of the records in our database and is responsible for almost all of our disk space used. The lower our granularity, the larger the fact table. You can see from the previous example that moving from a daily to weekly grain would reduce our potential number of records to only slightly more than 52,000,000 records.

The data types for the fields in the fact table do help keep it as small as possible. In most fact tables, all of the fields are numeric, which can require less storage space than the long descriptions we find in the dimension tables.

Finally, be aware that each added dimension can greatly increase the size of our fact table. If we added one dimension to the previous example that included 20 possible values, our potential number of records would reach 7.3 billion.

Changing Attributes

One of the greatest challenges in a star schema is the problem of changing attributes. As an example, we will use the simplified star schema in Figure 10. In the StoreDimension table, we have each store being in a particular region, territory, and zone. Some companies realign their sales regions, territories, and zones occasionally to reflect changing business conditions. However, if we simply go in and update the table, and then try to look at historical sales for a region, the numbers will not be accurate. By simply updating the region for a store, our total sales for that region will not be historically accurate.

In some cases, we do not care. In fact, we want to see what the sales would have been had this store been in that other region in prior years. More often, however, we do not want to change the historical data. In this case, we may need to create a new record for the store. This new record contains the new region, but leaves the old store record, and therefore the old regional sales data, intact. This approach, however, prevents us from comparing this stores current sales to its historical sales unless we keep track of it's previous StoreID. This can require an extra field called PreviousStoreID or something similar.

There are no right and wrong answers. Each case will require a different solution to handle changing attributes.

Aggregations

Finally, we need to discuss how to handle aggregations. The data in the fact table is already aggregated to the fact table's grain. However, we often want to aggregate to a higher level. For example, we may want to sum sales to a monthly or quarterly number. In addition, we may be looking for total just for a product or a category.

These numbers must be calculated on the fly using a standard SQL statement. This calculation takes time, and therefore some people will want to decrease the time required to retrieve higher-level aggregations.

Some people store higher-level aggregations in the database by pre-calculating them and storing them in the database. This requires that the lowest-level records have special values put in them. For example, a TimeDimension record that actually holds weekly totals might have a 9 in the DayOfWeek field to indicate that this particular record holds the total for the week.

This approach has been used in the past, but better alternatives exist. These alternatives usually consist of building a cube structure to hold pre-calculated values. We will examine Microsoft's OLAP Services, a tool designed to build cube structures to speed our access to warehouse data.

Slowly changing dimensions are used to describe the date effectivity of the data. It describe the dimensions whose attribute value vary over time.

This term is commonly used in the Data Warehousing world. However, the problem exists in the OLTP, relational data modeling as well.

Example:

The sales representative assigned to a customer may change over time. Linda was the salesrep for ABC, inc. before March last year. Kathy later becomes the representative for this account.
You may want to track the data "as is", "as was", or both. If you show the year total sales, you can either report as the sales are all generated by Kathy, or actually break the number down between Linda and Kathy.

Slowly changing dimensions

Slowly changing dimensions (1)

- The dimensional attribute record is overwritten with the new value
- No changes are needed elsewhere in the dimension record
- No keys are affected anywhere in the database
- Very easy to implement but the historical data is now inconsistent

Slowly changing dimensions (2)

- Introduce a new record for the same dimensional entity in order to reflect its changed state
- A new instance of the dimensional key is created which references the new record
- In order to is best dealt with by using version digits at the end of the key.
- All these keys need to be created, maintained and managed by someone and tracked in the metadata
- The database maintains its consistency and the versions can be said to partition history

Slowly changing dimensions (3)

- Use slightly different design of dimension table which has fields for:
  - original status of dimensional attribute
  - current status of dimensional attribute
  - an effective date of change field
- This allows the analyst to compare the as-is and as-was states against each other
- Only two states can be traced, the current and the original
- Some inconsistencies are created in the data as time is not properly partitioned

Introduction to the Series

Oracle9i provides a new set of ETL options that can be effectively integrated into the ETL architecture. In order to develop the correct approach to implementing new technology in the ETL architecture, it is important to understand the components, architectural options and best practices when designing and developing a data warehouse. With this background, each option will be explored and how it is best suited for the ETL architecture.
Through this series of articles, an overview of the ETL architecture will be discussed as well as a detailed look at each option. Each ETL option's syntax, behavior and performance (where appropriate) will be examined. Based on the results of examples, combined with a solid understanding of the ETL architecture, strategies and approaches to leverage the new options in the ETL architecture will be outlined. The final article in the series will provide a look at all of the ETL options working together, stemming from examples throughout the series.
Individual articles in the series include:

- Part 1 – Overview of the Extract, Transform and Load (ETL) Architecture
- Part 2 – External Tables
- Part 3 – Multiple Table Insert

- Part 4 – Upsert /MERGE INTO (Add and Update Combined Statement)
- Part 5 – Table Functions
- Part 6 – Bring it All Together: A Look at the Combined Use of the ETL Options.

The information in the series is targeted for data warehouse developers, data warehouse architects and information technology managers.

Overview of the Extract, Transform and Load Architecture (ETL)

The warehouse architect can assemble ETL architectures in many different forms using an endless variety of technologies. Due to this fact, the warehouse can take advantage of the software, skill sets, hardware and standards already in place within an organization. The potential weakness of the warehouse arises when a loosely managed project, which does not adhere to a standard approach, results in an increase in scope, budget and maintenance. This weakness may result in vulnerabilities to unforeseen data integrity limitations in the source systems as well. The key to eliminating this weakness is to develop a technical design that employs solid warehouse expertise and data warehouse best practices. Professional experience and the data warehouse fundamentals are key elements to eliminating failure on a warehouse project.

Potential problems are exposed in this article not to deliver fear or confirm the popular cliché that "warehouse projects fail." It is simply important to understand that new technologies, such as database options, are not a replacement for the principles of data warehousing and ETL processing. New technologies should, and many times will, advance or complement the warehouse. They should make its architecture more efficient, scalable and stable. That is where the new Oracle9i features play nicely. These features will be explored while looking at their appropriate uses in the ETL architecture. In order to determine where the new Oracle9i features may fit into the ETL architecture, it is important to look at ETL approaches and components.

Approaches to ETL Architecture

Within the ETL architecture two distinct, but not mutually exclusive, approaches are traditionally used in the ETL design. The custom approach is the oldest and was once the only approach for data warehousing. In effect, this approach takes the technologies and hardware that an organization has on hand and develops a data warehouse using those technologies. The second approach includes the use of packaged ETL software. This approach focuses on performing the majority of connectivity, extraction, transformation and data loading within the ETL tool itself. However, this software comes with an additional cost. The potential benefits of an ETL package include a reduction in development time as well as a reduction in maintenance overhead.

ETL Components

The ETL architecture is traditionally designed into two components:

- The source to stage component is intended to focus the efforts of reading the source data (sourcing) and replicating the data to the staging area. The staging area is typically comprised of several schemas that house individual source systems or sets of related source systems. Within each schema, all of the source system tables are usually "mirrored." The structure of the stage table is identical to that of the source table with the addition of data elements to support referential integrity and future ETL processing.
- The stage to warehouse component focuses the effort of standardizing and centralizing the data from the source systems into a single view of the organization's information. This centralized target can be a data warehouse, data mart, operational data store, customer list store, reporting database or any other reporting/data environment. (The examples in this article assume the final target is a data warehouse.) This portion of the architecture should not be concerned with translation, data formats, or data type conversion. It can now focus on the complex task of

cleansing, standardizing and transforming the source data according to the business rules.

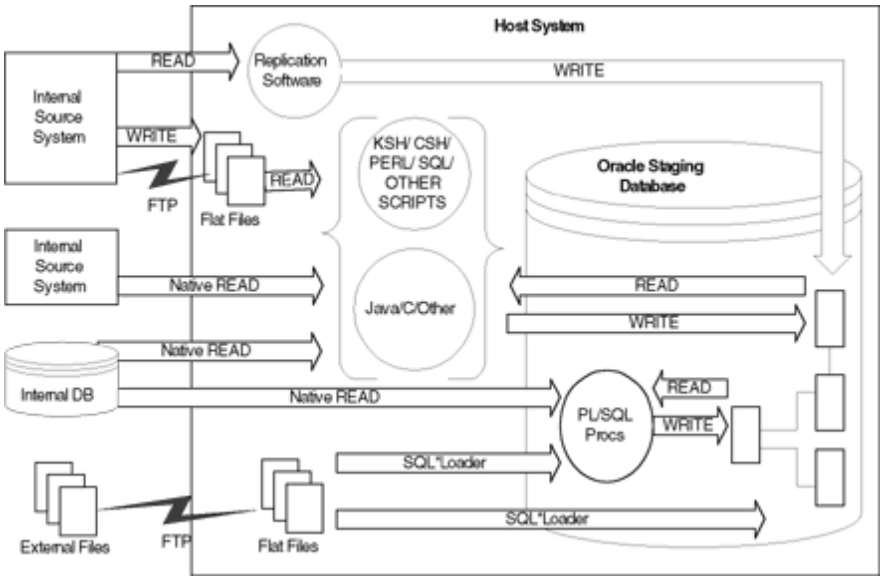
It is important to note that an ETL tool strictly "extracts, transforms and loads." Separate tools or external service organizations, which may require additional cost, accomplish the work of name and address cleansing and standardization. These data cleansing tools can work in conjunction with the ETL packaged software in a variety of ways. Many organizations exist that are able to perform the same work offsite under a contractual basis. The task of data cleansing can occur in the staging environment prior to or during stage to warehouse processing. In any case, it is a good practice to house a copy of the data cleansing output in the staging area for auditing purposes.

The following sections include diagrams and overviews of:

- Custom source to stage,
- Packaged ETL tool source to stage,
- Custom stage to warehouse, and
- Packaged ETL tool stage to warehouse architectures.

This article assumes that the staging and warehouse databases are Oracle9i instances hosted on separate systems.

Custom ETL – Source to Stage



Figur e 1: Source to Stage Portion of Custom ETL Architecture

Figure 1 outlines the source to stage portion of a custom ETL architecture and exposes several methods of data "connections." These methods include:

- Replicating data through the use of data replication software (mirroring software) that detects or "sniffs" changes from a database or file system logs.
- Generating flat files by pulling or pushing data from a client program connected to the source system.
- "FTPing" internal data from the source system in a native or altered format.
- Connecting natively to source system data and/or files (i.e., a DB2 connection to a AS/400 file systems).
- Reading data from a native database connection.
- Reading data over a database link from an Oracle instance to the target Oracle staging instance and FTPing data from an external site to the staging host system.

Other data connection options may include a tape delivered on site and copied, reading data from a queue (i.e.,

MQSeries), reading data from an enterprise application integration (EAI) message, reading data via a database bridge or other third-party broker for data access (i.e., DB2 Connect, DBAnywhere), etc …

After a connection is established to the source systems, many methods are used to read and load the data into the staging area as described in the diagram. These methods include the use of:

- Replication software (combines read and write replication into a single software package),
- A shell or other scripting tool such as KSH, CSH, PERL and SQL reading data from a flat file,
- A shell or other scripting tool reading data from a database connection (i.e., over PERL DBI),
- A packaged or custom executable such as C, C++, AWK, SED or Java reading data from a flat file,
- A packaged or custom executable reading data from a database connection and SQL*Loader reading from a flat file.

Packaged ETL Tool – Source to Stage

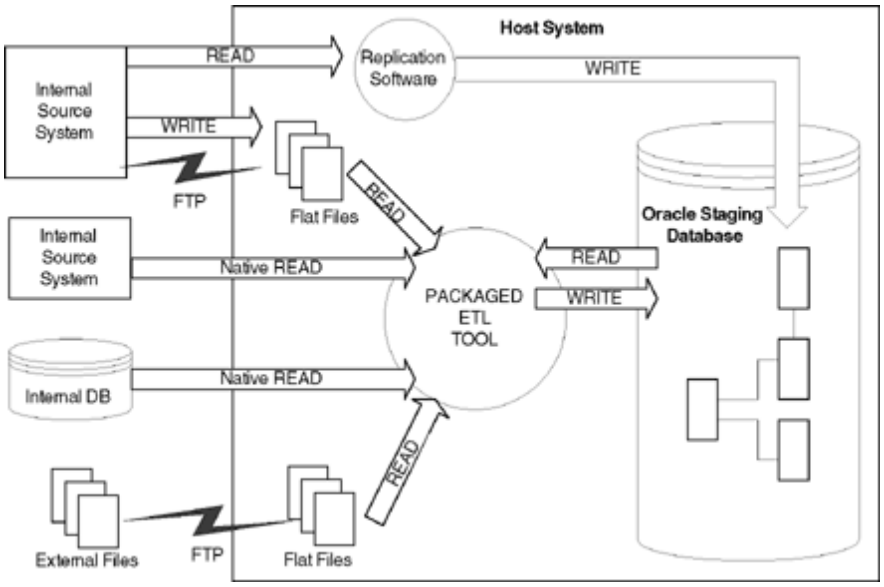


Figure 2: Source to Stage Portion Using a Packaged ETL Tool

Figure 2 outlines the source to stage portion of an ETL architecture using a packaged ETL tool and exposes several methods of data "connections" which are similar to those used in the custom source to stage processing model. The connection method with a packaged ETL tool typically allows for all of the connections one would expect from a custom development effort. In most cases, each type of source connection requires a license. For example if a connection is required to a Sybase, DB2 and Oracle database, three separate licenses are needed. If licensing is an issue, the ETL architecture typically embraces a hybrid solution using other custom methods to replicate source data in addition to the packaged ETL tool.

Connection methods include:

- Replicating data using data replication software (mirroring software) that detects or sniffs changes. from the database or file system logs.
- FTPing internal data from the source system in native or altered format.
- Connecting natively to the system data and/or files (i.e., a DB2 connection to a AS/400 file systems).
- Reading data from a native database connection.
- Reading data over a database link by an Oracle staging database from the target Oracle instance.
- FTPing data from an external site to the staging host system.

Other options may include a tape delivered on site and copied, reading data from a queue (i.e., MQSeries), reading

data from an enterprise application integration (EAI) message/queue, reading data via a database bridge or other third-party broker for data access (i.e., DB2 Connect, DBAnywhere), etc…

After a connection is established to the source systems, the ETL tool is used to read, perform simple transformations such as rudimentary cleansing (i.e., trimming spaces), perform data type conversion, convert data formats and load the data into the staging area. Advanced transformations are recommended to take place in the stage to warehouse component and not in the source to stage processing (explained in the next section). Because the package ETL tool is designed to handle all of the transformations and conversions, all the work is done within the ETL server itself. Within the ETL tool's server repository, separate mappings exist to perform the individual ETL tasks.

Custom ETL – Stage to Warehouse

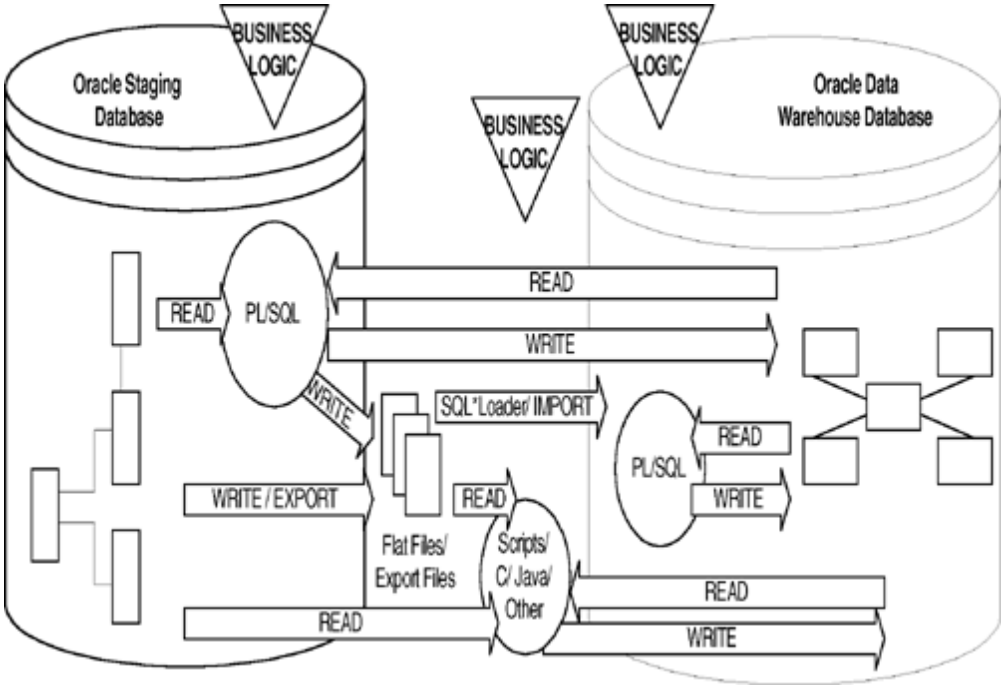


Figure 3: Stage to Warehouse Portion of Custom ETL Architecture

Figure 3 outlines the stage to warehouse portion of a custom ETL architecture. The ETL stage to warehouse component is where the data standardization and centralization occur. The work of gathering, formatting and converting data types is has been completed by the source to stage component. Now the ETL work can focus on the task of creating a single view of the organization's data in the warehouse.

This diagram exposes several typical methods of standardizing and/or centralizing data to the data warehouse. These methods include the use of a:

- PL/SQL procedure reading and writing directly to the data warehouse from the staging database (this could be done just as easily if the procedure was located in the warehouse database).
- PL/SQL procedure reading from the staging database and writing to flat files (i.e., via a SQL script).
- SQL*Plus client writing data to a flat file from stage, SQL*Loader importing files into the warehouse for loading or additional processing by a PL/SQL procedure.
- An Oracle table export-import process from staging to the warehouse for loading or additional processing by a PL/SQL procedure.
- Shell or other scripting tool such as KSH, CSH, PERL or SQL reading data natively or from a flat file and writing data into the warehouse.

- Packaged or custom executable such as C, C++, AWK, SED or Java reading data natively or from a flat file and writing data into the warehouse.

Packaged ETL Tool – Stage to Warehouse

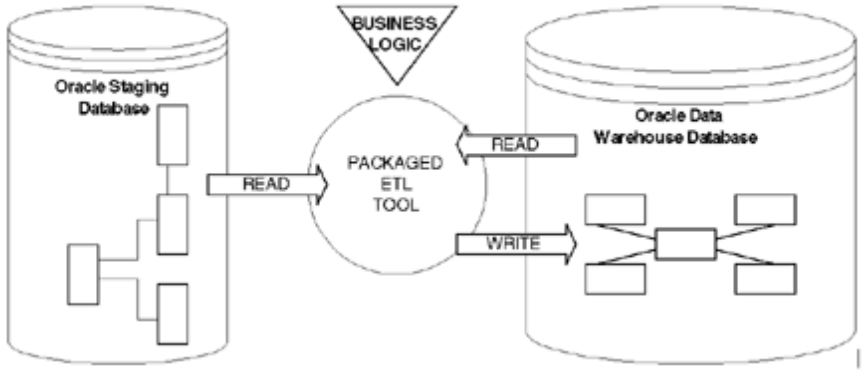


Figure 4: Stage to Warehouse Portion of a Packaged ETL Tool Architecture

Figure 4 outlines the stage to warehouse portion of a packaged ETL tool architecture. Figure 4 diagrams the packaged ETL application performing the standardization and centralization of data to the warehouse all within one application. This is the strength of a packaged ETL tool. In addition, this is the component of ETL architecture where the ETL tool is best suited to apply the organization's business rules. The packaged ETL tool will source the data through a native connection to the staging database. It will perform transformations on each record after pulling the data from the stage database through a pipe. From there it will load each record into the warehouse through a native connection to the database. Again, not all packaged ETL architectures look like this due to many factors. Typically a deviation in the architecture is due to requirements that the ETL software cannot, or is not licensed, to fulfill. In these instances one of the custom stage to warehouse methods is most commonly used.

Business Logic and the ETL Architecture

In any warehouse development effort, the business logic is the core of the warehouse. The business logic is applied to the proprietary data from the organization's internal and external data sources. The application process combines the heterogeneous data into a single view of the organization's information. The logic to create a central view of the information is often a complex task. In order to properly manage this task, it is important to consolidate the business rules into the stage to warehouse ETL component, regardless of the ETL architecture. If this best practice is ignored, much of the business logic may be spread throughout the source to stage and stage to warehouse components. This will ultimately hamper the organization's ability to maintain the warehouse solution long term and may lead to an error prone system.

Within the packaged ETL tool architecture, the centralization of the business logic becomes a less complex task. Due to the fact that the mapping and transformation logic is managed by the ETL software package, the centralization of rules is offered as a feature of the software. However, using packaged ETL tools does not guarantee a proper ETL implementation. Good warehouse development practices are still necessary when developing any type of ETL architecture.

In the custom ETL architecture, it becomes critical to place the application of business logic in the stage to warehouse component due to the large number of individual modules. The custom solution will typically store business logic in a custom repository or in the code of the ETL transformations. This is the greatest disadvantage to the custom warehouse. Developing a custom repository requires additional development effort, solid warehousing design experience and strict attention to detail. Due to this difficulty, the choice may be made to develop the rules

into the ETL transformation code to speed the time of delivery. Whether or not the decision is made to store the rules in a custom repository or not, it is important to have a well-thought-out design. The business rules are the heart of the warehouse. Any problems with the rules will create errors in the system.
It is important to understand some of the best practices and risks when developing ETL architectures to better appreciate how the new technology will fit into the architecture. With this background it is apparent that new technology or database options will not be silver bullet for ETL processing. New technology will not increase a solution's effectiveness nor replace the need for management of the business rules. However, the new Oracle 9i ETL options provide a great complement to custom and packaged ETL tool architectures.

Extract, transform, and load (ETL) is a process in data warehousing that involves extracting data from outside sources,
transforming it to fit business needs, and ultimately loading it into the data warehouse.
ETL is important, as it is the way data actually gets loaded into the warehouse. This article assumes that data is always loaded into a data warehouse, whereas the term ETL can in fact refer to a process that loads any database.
Contents
[hide]
1 Extract
2 Transform
3 Load
4 Challenges
5 Tools

Extract
The first part of an ETL process is to extract the data from the source systems. Most data warehousing projects consolidate data from different source systems. Each separate system may also use a different data organization / format. Common data source formats are relational databases and flat files, but may include non-relational database structures such as IMS or other data structures such as VSAM or ISAM. Extraction converts the data into a format for transformation processing.

Transform
The transform stage applies a series of rules or functions to the extracted data to derive the data to be loaded. Some data sources will require very little manipulation of data. In other cases, one or more of the following transformations types may be required:
Selecting only certain columns to load (or selecting null columns not to load)
Translating coded values (e.g., if the source system stores M for male and F for female, but the warehouse stores 1 for male and 2 for female)
Encoding free-form values (e.g., mapping "Male" and "M" and "Mr" onto 1)
Deriving a new calculated value (e.g., sale_amount = qty * unit_price)
Joining together data from multiple sources (e.g., lookup, merge, etc.)
Summarizing multiple rows of data (e.g., total sales for each region)
Generating surrogate key values
Transposing or pivoting (turning multiple columns into multiple rows or vice versa)
Splitting a column into multiple columns (e.g., putting a comma-separated list specified as a string in one column as

individual values in different columns)

Load

The load phase loads the data into the data warehouse. Depending on the requirements of the organization, this process ranges widely. Some data warehouses merely overwrite old information with new data. More complex systems can maintain a history and audit trail of all changes to the data.

Challenges

ETL processes can be quite complex, and significant operational problems can occur with improperly designed ETL systems.

The range of data values or data quality in an operational system may be outside the expectations of designers at the time validation and transformation rules are specified. Data profiling of a source during data analysis is recommended to identify the data conditions that will need to be managed by transform rules specifications.

The scalability of an ETL system across the lifetime of its usage needs to be established during analysis. This includes understanding the volumes of data that will have to be processed within Service Level Agreements, (SLAs). The time available to extract from source systems may change, which may mean the same amount of data may have to be processed in less time. Some ETL systems have to scale to process terabytes of data to update data warehouses with tens of terabytes of data. Increasing volumes of data may require designs that can scale from daily batch to intra-day micro-batch to integration with message queues for continuous transformation and update.

A recent development in ETL software is the implementation of parallel processing. This has enabled a number of methods to improve overall performance of ETL processes when dealing with large volumes of data.

There are 3 main types of parallelisms as implemented in ETL applications:

Data: By splitting a single sequential file into smaller data files to provide parallel access.

Pipeline: Allowing the simultaneous running of several components on the same data stream. An example would be looking up a value on record 1 at the same time as adding together two fields on record 2.

Component: The simultaneous running of multiple processes on different data streams in the same job. Sorting one input file while performing a deduplication on another file would be an example of component parallelism.

All three types of parallelism are usually combined in a single job.An additional difficulty is making sure the data being uploaded is relatively consistent. Since multiple source databases all have different update cycles (some may be updated every few minutes, while others may take days or weeks), an ETL system may be required to hold back certain data until all sources are synchronized. Likewise, where a warehouse may have to be reconciled to the contents in a source system or with the general ledger, establishing synchronization and reconciliation points is necessary.

Tools

While an ETL process can be created using almost any programming language, creating them from scratch is quite complex. Increasingly, companies are buying ETL tools to help in the creation of ETL processes.

A good ETL tool must be able to communicate with the many different relational databases and read the various file formats used throughout an organization. ETL tools have started to migrate into Enterprise Application Integration, or even Enterprise Service Bus, systems that now cover much more than just the extraction, transformation and loading of data. Many ETL vendors now have data profiling, data quality and metadata capabilities

WHAT IS AN ETL PROCESS?

WHAT IS AN ETL PROCESS?

ETL process - acronymic for extraction, transformation and loading operations are a fundamental phenomenon in a data warehouse. Whenever DML (data manipulation language) operations such as INSERT, UPDATE OR DELETE are issued on the source database, data extraction occurs.

After data extraction and transformation have taken place, data are loaded into the data warehouse. Incremental loading is beneficial in the sense that only that have changed after the last data extraction and transformation are loaded.

ORACLE CHANGE DATA CAPTURE FRAMEWORK

The change data framework is designed for capturing only insert, delete and update operations on the oracle database, that is to say they are 'DML sensitive'. Below is architecture of change data capture framework. Below is architecture illustrating the flow of information in an oracle data capture framework.

Figure 1.Change data capture framework architecture.

Implementing oracle change data capture is very simple. Following the following steps, guides you through the whole implementation process.

Source table identification: Firstly, the source tables must be identified.

Choose a publisher: The publisher is responsible for creating and managing the change tables. Note that the publisher must be granted SELECT_CATALOG_ROLE, which enables the publisher to select data from any SYS-owned dictionary tables or views and EXECUTE_CATALOG_ROLE, which enables the publisher to receive execute privileges on any SYS-owned packages. He also needs select privilege on the source tables

Change tables creation: When data extraction occurs, change data are stored in the change tables. Also stored in the change tables are system metadata, imperative for the smooth functioning of the change tables. In order to create the change tables, the procedure DBMS_LOGMNR_CDC_PUBLISH.CREATE_CHANGE_TABLE is executed. It is important to note that each source table must have its own change table.

Choose the subscriber: The publisher must grant select privilege on the change tables and source tables to the subscriber. You might have more than one subscriber as the case may be.

Subscription handle creation: Creating the subscription handle is very pertinent because it is used to specifically identify a particular subscription. Irrespective of the number of tables subscribed to, one and only one subscription handle must be created. To create a subscription handle, first define a variable, and then execute the DBMS_LOGMNR_CDC_SUBSCRIBE.GET_SUBSCRIPTION HANDLE procedure.

Subscribe to the change tables: The data in the change tables are usually enormous, thus only data of interest should be subscribed to. To subscribe, the DBMS_LOGMNR_CDC_SUBSCRIBE.SUBSCRIBE procedure is executed.

Subscription activation: Subscription is activated only once and after activation, subscription cannot be modified. Activate your subscription using the DBMS_LOGMNR_CDC_SUBSCRIBE.ACTIVATE_SUBSCRIPTION procedure.

Subscription window creation: Since subscription to the change tables does not stop data extraction from the

source table, a window is set up using the DBMS_LOGMNR_CDC_SUBSCRIBE.EXTEND_WINDOW procedure. However, it is to be noted that changes effected on the source system after this procedure is executed will not be available until the window is flushed and re-extended.

Subscription views creation: In order to view and query the change data, a subscriber view is prepared for individual source tables that the subscriber subscribes to using DBMS_LOGMNR_CDC_SUBSCRIBE.PREPARE_SUBSCRIBER_VIEW procedure. However, you need to define the variable in which the subscriber view name would be returned. Also, you would be prompted for the subscription handle, source schema name and source table name.

Query the change tables: Resident in the subscriber view are not only the change data needed but also metadata, fundamental to the efficient use of the change data such as OPERATION$, CSCN$, USERNAME$ etc. Since you already know the view name, you can describe the view and then query it using the conventional select statement.

Drop the subscriber view: The dropping of the subscriber view is carried out only when you are sure you are done with the data in the view and they are no longer needed (i.e. they've been viewed and extracted). It is imperative to note that each subscriber view must be dropped individually using the DBMS_LOGMNR_CDC_SUBSCRIBE.DROP_SUBSCRIBE_VIEW procedure. Purge the subscription view: To facilitate the extraction of change data again, the subscription window must be purged using the DBMS_LOGMNR_CDC_SUBSCRIBE.PURGE_WINDOW procedure.

ETL Process

Here is the typical ETL Process:

Specify metadata for sources, such as tables in an operational system

Specify metadata for targets—the tables and other data stores in a data warehouse

Specify how data is extracted, transformed, and loaded from sources to targets

Schedule and execute the processes

Monitor the execution

A ETL tool thus involves the following components:

A design tool for building the mapping and the process flows

A monitor tool for executing and monitoring the process

The process flows are sequences of steps for the extraction, transformation, and loading of data. The data is extracted from sources (inputs to an operation) and loaded into a set of targets (outputs of an operation) that make up a data warehouse or a data mart.

A good ETL design tool should provide the change management features that satisfies the following criteria:

A metadata respository that stores the metdata about sources, targets, and the transformations that connect them.

Enforce metadata source control for team-based development : Multiple designers should be able to work with the same metadata repository at the same time without overwriting each other’s changes. Each developer should be able to check out metadata from the respository into their project or workspace, modify them, and check the changes back into the respository.

After a metadata object has been checked out by one person, it is locked so that it cannot be updated by another person until the object has been checked back in.

Overview of ETL in Data Warehouses

You need to load your data warehouse regularly so that it can serve its purpose of facilitating business analysis. To do this, data from one or more operational systems needs to be extracted and copied into the data warehouse. The process of extracting data from source systems and bringing it into the data warehouse is commonly called ETL, which stands for extraction, transformation, and loading. The acronym ETL is perhaps too simplistic, because it omits the transportation phase and implies that each of the other phases of the process is distinct. We refer to the entire process, including data loading, as ETL. You should understand that ETL refers to a broad process, and not three well-defined steps.

The methodology and tasks of ETL have been well known for many years, and are not necessarily unique to data warehouse environments: a wide variety of proprietary applications and database systems are the IT backbone of any enterprise. Data has to be shared between applications or systems, trying to integrate them, giving at least two applications the same picture of the world. This data sharing was mostly addressed by mechanisms similar to what we now call ETL.

Data warehouse environments face the same challenge with the additional burden that they not only have to exchange but to integrate, rearrange and consolidate data over many systems, thereby providing a new unified information base for business intelligence. Additionally, the data volume in data warehouse environments tends to be very large.

What happens during the ETL process? During extraction, the desired data is identified and extracted from many different sources, including database systems and applications. Very often, it is not possible to identify the specific subset of interest, therefore more data than necessary has to be extracted, so the identification of the relevant data will be done at a later point in time. Depending on the source system's capabilities (for example, operating system resources), some transformations may take place during this extraction process. The size of the extracted data varies from hundreds of kilobytes up to gigabytes, depending on the source system and the business situation. The same is true for the time delta between two (logically) identical extractions: the time span may vary between days/hours and minutes to near real-time. Web server log files for example can easily become hundreds of megabytes in a very short period of time.

After extracting data, it has to be physically transported to the target system or an intermediate system for further processing. Depending on the chosen way of transportation, some transformations can be done during this process, too. For example, a SQL statement which directly accesses a remote target through a gateway can concatenate two columns as part of the SELECT statement.

The emphasis in many of the examples in this section is scalability. Many long-time users of Oracle Database are experts in programming complex data transformation logic using PL/SQL. These chapters suggest alternatives for many such data manipulation operations, with a particular emphasis on implementations that take advantage of Oracle's new SQL functionality, especially for ETL and the parallel query infrastructure.

ETL Tools for Data Warehouses

Designing and maintaining the ETL process is often considered one of the most difficult and resource-intensive portions of a data warehouse project. Many data warehousing projects use ETL tools to manage this process. Oracle Warehouse Builder (OWB), for example, provides ETL capabilities and takes advantage of inherent database abilities. Other data warehouse builders create their own ETL tools and processes, either inside or outside the database.

Besides the support of extraction, transformation, and loading, there are some other tasks that are important for a

successful ETL implementation as part of the daily operations of the data warehouse and its support for further enhancements. Besides the support for designing a data warehouse and the data flow, these tasks are typically addressed by ETL tools such as OWB.

Oracle is not an ETL tool and does not provide a complete solution for ETL. However, Oracle does provide a rich set of capabilities that can be used by both ETL tools and customized ETL solutions. Oracle offers techniques for transporting data between Oracle databases, for transforming large volumes of data, and for quickly loading new data into a data warehouse.

Daily Operations in Data Warehouses

The successive loads and transformations must be scheduled and processed in a specific order. Depending on the success or failure of the operation or parts of it, the result must be tracked and subsequent, alternative processes might be started. The control of the progress as well as the definition of a business workflow of the operations are typically addressed by ETL tools such as Oracle Warehouse Builder.

Evolution of the Data Warehouse

As the data warehouse is a living IT system, sources and targets might change. Those changes must be maintained and tracked through the lifespan of the system without overwriting or deleting the old ETL process flow information. To build and keep a level of trust about the information in the warehouse, the process flow of each individual record in the warehouse can be reconstructed at any point in time in the future in an ideal case.

Overview of Extraction in Data Warehouses

Extraction is the operation of extracting data from a source system for further use in a data warehouse environment. This is the first step of the ETL process. After the extraction, this data can be transformed and loaded into the data warehouse.

The source systems for a data warehouse are typically transaction processing applications. For example, one of the source systems for a sales analysis data warehouse might be an order entry system that records all of the current order activities.

Designing and creating the extraction process is often one of the most time-consuming tasks in the ETL process and, indeed, in the entire data warehousing process. The source systems might be very complex and poorly documented, and thus determining which data needs to be extracted can be difficult. The data has to be extracted normally not only once, but several times in a periodic manner to supply all changed data to the data warehouse and keep it up-to-date. Moreover, the source system typically cannot be modified, nor can its performance or availability be adjusted, to accommodate the needs of the data warehouse extraction process.

These are important considerations for extraction and ETL in general. This chapter, however, focuses on the technical considerations of having different kinds of sources and extraction methods. It assumes that the data warehouse team has already identified the data that will be extracted, and discusses common techniques used for extracting data from source databases.

Designing this process means making decisions about the following two main aspects:

Which extraction method do I choose?

This influences the source system, the transportation process, and the time needed for refreshing the warehouse.

How do I provide the extracted data for further processing?

This influences the transportation method, and the need for cleaning and transforming the data.

Introduction to Extraction Methods in Data Warehouses

The extraction method you should choose is highly dependent on the source system and also from the business needs in the target data warehouse environment. Very often, there is no possibility to add additional logic to the source systems to enhance an incremental extraction of data due to the performance or the increased workload of these systems. Sometimes even the customer is not allowed to add anything to an out-of-the-box application system.
The estimated amount of the data to be extracted and the stage in the ETL process (initial load or maintenance of data) may also impact the decision of how to extract, from a logical and a physical perspective. Basically, you have to decide how to extract data logically and physically.

Logical Extraction Methods

There are two types of logical extraction:
Full Extraction
Incremental Extraction

Full Extraction

The data is extracted completely from the source system. Because this extraction reflects all the data currently available on the source system, there's no need to keep track of changes to the data source since the last successful extraction. The source data will be provided as-is and no additional logical information (for example, timestamps) is necessary on the source site. An example for a full extraction may be an export file of a distinct table or a remote SQL statement scanning the complete source table.

Incremental Extraction

At a specific point in time, only the data that has changed since a well-defined event back in history will be extracted. This event may be the last time of extraction or a more complex business event like the last booking day of a fiscal period. To identify this delta change there must be a possibility to identify all the changed information since this specific time event. This information can be either provided by the source data itself such as an application column, reflecting the last-changed timestamp or a change table where an appropriate additional mechanism keeps track of the changes besides the originating transactions. In most cases, using the latter method means adding extraction logic to the source system.
Many data warehouses do not use any change-capture techniques as part of the extraction process. Instead, entire tables from the source systems are extracted to the data warehouse or staging area, and these tables are compared with a previous extract from the source system to identify the changed data. This approach may not have significant impact on the source systems, but it clearly can place a considerable burden on the data warehouse processes, particularly if the data volumes are large.
Oracle's Change Data Capture mechanism can extract and maintain such delta information.

Physical Extraction Methods

Depending on the chosen logical extraction method and the capabilities and restrictions on the source side, the extracted data can be physically extracted by two mechanisms. The data can either be extracted online from the source system or from an offline structure. Such an offline structure might already exist or it might be generated by an extraction routine.

There are the following methods of physical extraction:
Online Extraction
Offline Extraction

Online Extraction

The data is extracted directly from the source system itself. The extraction process can connect directly to the source system to access the source tables themselves or to an intermediate system that stores the data in a preconfigured manner (for example, snapshot logs or change tables). Note that the intermediate system is not necessarily physically different from the source system.
With online extractions, you need to consider whether the distributed transactions are using original source objects or prepared source objects.

Offline Extraction

The data is not extracted directly from the source system but is staged explicitly outside the original source system. The data already has an existing structure (for example, redo logs, archive logs or transportable tablespaces) or was created by an extraction routine.
You should consider the following structures:

Flat files

Data in a defined, generic format. Additional information about the source object is necessary for further processing.

Dump files

Oracle-specific format. Information about the containing objects may or may not be included, depending on the chosen utility. Redo and archive logs Information is in a special, additional dump file.
Transportable tablespaces
A powerful way to extract and move large volumes of data between Oracle databases. Oracle Corporation recommends that you use transportable tablespaces whenever possible, because they can provide considerable advantages in performance and manageability over other extraction techniques.

Change Data Capture

An important consideration for extraction is incremental extraction, also called Change Data Capture. If a data warehouse extracts data from an operational system on a nightly basis, then the data warehouse requires only the data that has changed since the last extraction (that is, the data that has been modified in the past 24 hours). Change Data Capture is also the key-enabling technology for providing near real-time, or on-time, data warehousing.
When it is possible to efficiently identify and extract only the most recently changed data, the extraction process (as well as all downstream operations in the ETL process) can be much more efficient, because it must extract a much smaller volume of data. Unfortunately, for many source systems, identifying the recently modified data may be difficult or intrusive to the operation of the system. Change Data Capture is typically the most challenging technical issue in data extraction.
Because change data capture is often desirable as part of the extraction process and it might not be possible to use the Change Data Capture mechanism, this section describes several techniques for implementing a self-developed change capture on Oracle Database source systems:

Timestamps
Partitioning
Triggers

These techniques are based upon the characteristics of the source systems, or may require modifications to the source systems. Thus, each of these techniques must be carefully evaluated by the owners of the source system prior to implementation.
Each of these techniques can work in conjunction with the data extraction technique discussed previously. For example, timestamps can be used whether the data is being unloaded to a file or accessed through a distributed query.

Timestamps
The tables in some operational systems have timestamp columns. The timestamp specifies the time and date that a given row was last modified. If the tables in an operational system have columns containing timestamps, then the latest data can easily be identified using the timestamp columns. For example, the following query might be useful for extracting today's data from an orders table:

```
SELECT * FROM orders
WHERE TRUNC(CAST(order_date AS date),'dd') =
  TO_DATE(SYSDATE,'dd-mon-yyyy');
```

If the timestamp information is not available in an operational source system, you will not always be able to modify the system to include timestamps. Such modification would require, first, modifying the operational system's tables to include a new timestamp column and then creating a trigger to update the timestamp column following every operation that modifies a given row.

Partitioning
Some source systems might use range partitioning, such that the source tables are partitioned along a date key, which allows for easy identification of new data. For example, if you are extracting from an orders table, and the orders table is partitioned by week, then it is easy to identify the current week's data.

Data Warehousing Extraction ways

You can extract data in two ways:
Extraction Using Data Files
Extraction Through Distributed Operations

Extraction Using Data Files
Most database systems provide mechanisms for exporting or unloading data from the internal database format into flat files. Extracts from mainframe systems often use COBOL programs, but many databases, as well as third-party software vendors, provide export or unload utilities.
Data extraction does not necessarily mean that entire database structures are unloaded in flat files. In many cases, it may be appropriate to unload entire database tables or objects. In other cases, it may be more appropriate to unload only a subset of a given table such as the changes on the source system since the last extraction or the results of

joining multiple tables together. Different extraction techniques vary in their capabilities to support these two scenarios.
When the source system is an Oracle database, several alternatives are available for extracting data into files:
Extracting into Flat Files Using SQL*Plus
Extracting into Flat Files Using OCI or Pro*C Programs
Exporting into Export Files Using the Export Utility
Extracting into Export Files Using External Tables

Extracting into Flat Files Using SQL*Plus
The most basic technique for extracting data is to execute a SQL query in SQL*Plus and direct the output of the query to a file. For example, to extract a flat file, country_city.log, with the pipe sign as delimiter between column values, containing a list of the cities in the US in the tables countries and customers, the following SQL script could be run:

```
SET echo off SET pagesize 0 SPOOL country_city.log
SELECT distinct t1.country_name ||'|'|| t2.cust_city
FROM countries t1, customers t2 WHERE t1.country_id = t2.country_id
AND t1.country_name= 'United States of America';
SPOOL off
```

The exact format of the output file can be specified using SQL*Plus system variables.
This extraction technique offers the advantage of storing the result in a customized format. Note that using the external table data pump unload facility, you can also extract the result of an arbitrary SQL operation. The example previously extracts the results of a join.
This extraction technique can be parallelized by initiating multiple, concurrent SQL*Plus sessions, each session running a separate query representing a different portion of the data to be extracted. For example, suppose that you wish to extract data from an orders table, and that the orders table has been range partitioned by month, with partitions orders_jan1998, orders_feb1998, and so on. To extract a single year of data from the orders table, you could initiate 12 concurrent SQL*Plus sessions, each extracting a single partition. The SQL script for one such session could be:

```
SPOOL order_jan.dat
SELECT * FROM orders PARTITION (orders_jan1998);
SPOOL OFF
```

These 12 SQL*Plus processes would concurrently spool data to 12 separate files. You can then concatenate them if necessary (using operating system utilities) following the extraction. If you are planning to use SQL*Loader for loading into the target, these 12 files can be used as is for a parallel load with 12 SQL*Loader sessions.
Even if the orders table is not partitioned, it is still possible to parallelize the extraction either based on logical or physical criteria. The logical method is based on logical ranges of column values, for example:

```
SELECT ... WHERE order_date
BETWEEN TO_DATE('01-JAN-99') AND TO_DATE('31-JAN-99');
```

The physical method is based on a range of values. By viewing the data dictionary, it is possible to identify the

Oracle Database data blocks that make up the orders table. Using this information, you could then derive a set of rowid-range queries for extracting data from the orders table:
SELECT * FROM orders WHERE rowid BETWEEN value1 and value2;

Parallelizing the extraction of complex SQL queries is sometimes possible, although the process of breaking a single complex query into multiple components can be challenging. In particular, the coordination of independent processes to guarantee a globally consistent view can be difficult. Unlike the SQL*Plus approach, using the new external table data pump unload functionality provides transparent parallel capabilities.
Note that all parallel techniques can use considerably more CPU and I/O resources on the source system, and the impact on the source system should be evaluated before parallelizing any extraction technique.

Extracting into Flat Files Using OCI or Pro*C Programs
OCI programs (or other programs using Oracle call interfaces, such as Pro*C programs), can also be used to extract data. These techniques typically provide improved performance over the SQL*Plus approach, although they also require additional programming. Like the SQL*Plus approach, an OCI program can extract the results of any SQL query. Furthermore, the parallelization techniques described for the SQL*Plus approach can be readily applied to OCI programs as well.
When using OCI or SQL*Plus for extraction, you need additional information besides the data itself. At minimum, you need information about the extracted columns. It is also helpful to know the extraction format, which might be the separator between distinct columns.

Exporting into Export Files Using the Export Utility
The Export utility allows tables (including data) to be exported into Oracle Database export files. Unlike the SQL*Plus and OCI approaches, which describe the extraction of the results of a SQL statement, Export provides a mechanism for extracting database objects. Thus, Export differs from the previous approaches in several important ways:
The export files contain metadata as well as data. An export file contains not only the raw data of a table, but also information on how to re-create the table, potentially including any indexes, constraints, grants, and other attributes associated with that table.
A single export file may contain a subset of a single object, many database objects, or even an entire schema.
Export cannot be directly used to export the results of a complex SQL query. Export can be used only to extract subsets of distinct database objects.
The output of the Export utility must be processed using the Import utility.
Oracle provides the original Export and Import utilities for backward compatibility and the data pump export/import infrastructure for high-performant, scalable and parallel extraction. See Oracle Database Utilities for further details.

Extracting into Export Files Using External Tables
In addition to the Export Utility, you can use external tables to extract the results from any SELECT operation. The data is stored in the platform independent, Oracle-internal data pump format and can be processed as regular external table on the target system. The following example extracts the result of a join operation in parallel into the four specified files. The only allowed external table type for extracting data is the Oracle-internal format ORACLE_DATAPUMP.
CREATE DIRECTORY def_dir AS '/net/dlsun48/private/hbaer/WORK/FEATURES/et';

```
DROP TABLE extract_cust;
CREATE TABLE extract_cust
ORGANIZATION EXTERNAL
(TYPE ORACLE_DATAPUMP DEFAULT DIRECTORY def_dir ACCESS PARAMETERS
(NOBADFILE NOLOGFILE)
LOCATION ('extract_cust1.exp', 'extract_cust2.exp', 'extract_cust3.exp',
      'extract_cust4.exp'))
PARALLEL 4 REJECT LIMIT UNLIMITED AS
SELECT  c.*, co.country_name, co.country_subregion, co.country_region
FROM customers c, countries co where co.country_id=c.country_id;
```

The total number of extraction files specified limits the maximum degree of parallelism for the write operation. Note that the parallelizing of the extraction does not automatically parallelize the SELECT portion of the statement.
Unlike using any kind of export/import, the metadata for the external table is not part of the created files when using the external table data pump unload. To extract the appropriate metadata for the external table, use the DBMS_METADATA package, as illustrated in the following statement:

```
SET LONG 2000
SELECT DBMS_METADATA.GET_DDL('TABLE','EXTRACT_CUST') FROM DUAL;
```

Extraction Through Distributed Operations

Using distributed-query technology, one Oracle database can directly query tables located in various different source systems, such as another Oracle database or a legacy system connected with the Oracle gateway technology. Specifically, a data warehouse or staging database can directly access tables and data located in a connected source system. Gateways are another form of distributed-query technology. Gateways allow an Oracle database (such as a data warehouse) to access database tables stored in remote, non-Oracle databases. This is the simplest method for moving data between two Oracle databases because it combines the extraction and transformation into a single step, and requires minimal programming. However, this is not always feasible.
Suppose that you wanted to extract a list of employee names with department names from a source database and store this data into the data warehouse. Using an Oracle Net connection and distributed-query technology, this can be achieved using a single SQL statement:

```
CREATE TABLE country_city AS SELECT distinct t1.country_name, t2.cust_city
FROM countries@source_db t1, customers@source_db t2
WHERE t1.country_id = t2.country_id
AND t1.country_name='United States of America';
```

This statement creates a local table in a data mart, country_city, and populates it with data from the countries and customers tables on the source system.
This technique is ideal for moving small volumes of data. However, the data is transported from the source system to the data warehouse through a single Oracle Net connection. Thus, the scalability of this technique is limited. For larger data volumes, file-based data extraction and transportation techniques are often more scalable and thus more appropriate.
13 Transportation in Data Warehouses
The following topics provide information about transporting data into a data warehouse:

Overview of Transportation in Data Warehouses
Introduction to Transportation Mechanisms in Data Warehouses

Transportation in Data Warehouses

Transportation is the operation of moving data from one system to another system. In a data warehouse environment, the most common requirements for transportation are in moving data from:
A source system to a staging database or a data warehouse database
A staging database to a data warehouse
A data warehouse to a data mart
Transportation is often one of the simpler portions of the ETL process, and can be integrated with other portions of the process.

Introduction to Transportation Mechanisms in Data Warehouses
You have three basic choices for transporting data in warehouses:
Transportation Using Flat Files
Transportation Through Distributed Operations
Transportation Using Transportable Tablespaces

Transportation Using Flat Files

The most common method for transporting data is by the transfer of flat files, using mechanisms such as FTP or other remote file system access protocols. Data is unloaded or exported from the source system into flat files using techniques, and is then transported to the target platform using FTP or similar mechanisms.
Because source systems and data warehouses often use different operating systems and database systems, using flat files is often the simplest way to exchange data between heterogeneous systems with minimal transformations. However, even when transporting data between homogeneous systems, flat files are often the most efficient and most easy-to-manage mechanism for data transfer.

Transportation Through Distributed Operations

Distributed queries, either with or without gateways, can be an effective mechanism for extracting data. These mechanisms also transport the data directly to the target systems, thus providing both extraction and transformation in a single step. Depending on the tolerable impact on time and system resources, these mechanisms can be well suited for both extraction and transformation.
As opposed to flat file transportation, the success or failure of the transportation is recognized immediately with the result of the distributed query or transaction.

Transportation Using Transportable Tablespaces

Oracle transportable tablespaces are the fastest way for moving large volumes of data between two Oracle databases. Previous to the introduction of transportable tablespaces, the most scalable data transportation mechanisms relied on moving flat files containing raw data. These mechanisms required that data be unloaded or exported into files from the source database, Then, after transportation, these files were loaded or imported into the target database. Transportable tablespaces entirely bypass the unload and reload steps.
Using transportable tablespaces, Oracle data files (containing table data, indexes, and almost every other Oracle

database object) can be directly transported from one database to another. Furthermore, like import and export, transportable tablespaces provide a mechanism for transporting metadata in addition to transporting data.
Transportable tablespaces have some limitations: source and target systems must be running Oracle8i (or higher), must use the same character set, and, prior to Oracle Database 10g, must run on the same operating system. For details on how to transport tablespace between operating systems.
The most common applications of transportable tablespaces in data warehouses are in moving data from a staging database to a data warehouse, or in moving data from a data warehouse to a data mart.

Transportable Tablespaces Example
Suppose that you have a data warehouse containing sales data, and several data marts that are refreshed monthly. Also suppose that you are going to move one month of sales data from the data warehouse to the data mart.

Step 1 Place the Data to be Transported into its own Tablespace
The current month's data must be placed into a separate tablespace in order to be transported. In this example, you have a tablespace ts_temp_sales, which will hold a copy of the current month's data. Using the CREATE TABLE ... AS SELECT statement, the current month's data can be efficiently copied to this tablespace:

```
CREATE TABLE temp_jan_sales NOLOGGING TABLESPACE ts_temp_sales
AS SELECT * FROM sales
WHERE time_id BETWEEN '31-DEC-1999' AND '01-FEB-2000';
```

Following this operation, the tablespace ts_temp_sales is set to read-only:

```
ALTER TABLESPACE ts_temp_sales READ ONLY;
```

A tablespace cannot be transported unless there are no active transactions modifying the tablespace. Setting the tablespace to read-only enforces this.
The tablespace ts_temp_sales may be a tablespace that has been especially created to temporarily store data for use by the transportable tablespace features. this tablespace can be set to read/write, and, if desired, the table temp_jan_sales can be dropped, or the tablespace can be re-used for other transportations or for other purposes.
In a given transportable tablespace operation, all of the objects in a given tablespace are transported. Although only one table is being transported in this example, the tablespace ts_temp_sales could contain multiple tables. For example, perhaps the data mart is refreshed not only with the new month's worth of sales transactions, but also with a new copy of the customer table. Both of these tables could be transported in the same tablespace. Moreover, this tablespace could also contain other database objects such as indexes, which would also be transported.
Additionally, in a given transportable-tablespace operation, multiple tablespaces can be transported at the same time. This makes it easier to move very large volumes of data between databases. Note, however, that the transportable tablespace feature can only transport a set of tablespaces which contain a complete set of database objects without dependencies on other tablespaces. For example, an index cannot be transported without its table, nor can a partition be transported without the rest of the table. You can use the DBMS_TTS package to check that a tablespace is transportable.

In this step, we have copied the January sales data into a separate tablespace; however, in some cases, it may be possible to leverage the transportable tablespace feature without even moving data to a separate tablespace. If the sales table has been partitioned by month in the data warehouse and if each partition is in its own tablespace, then it

may be possible to directly transport the tablespace containing the January data. Suppose the January partition, sales_jan2000, is located in the tablespace ts_sales_jan2000. Then the tablespace ts_sales_jan2000 could potentially be transported, rather than creating a temporary copy of the January sales data in the ts_temp_sales.
However, the same conditions must be satisfied in order to transport the tablespace ts_sales_jan2000 as are required for the specially created tablespace. First, this tablespace must be set to READ ONLY. Second, because a single partition of a partitioned table cannot be transported without the remainder of the partitioned table also being transported, it is necessary to exchange the January partition into a separate table (using the ALTER TABLE statement) to transport the January data. The EXCHANGE operation is very quick, but the January data will no longer be a part of the underlying sales table, and thus may be unavailable to users until this data is exchanged back into the sales table after the export of the metadata. The January data can be exchanged back into the sales table after you complete step 3.

Step 2 Export the Metadata
The Export utility is used to export the metadata describing the objects contained in the transported tablespace. For our example scenario, the Export command could be:
EXP TRANSPORT_TABLESPACE=y TABLESPACES=ts_temp_sales FILE=jan_sales.dmp
This operation will generate an export file, jan_sales.dmp. The export file will be small, because it contains only metadata. In this case, the export file will contain information describing the table temp_jan_sales, such as the column names, column datatype, and all other information that the target Oracle database will need in order to access the objects in ts_temp_sales.

Step 3 Copy the Datafiles and Export File to the Target System
Copy the data files that make up ts_temp_sales, as well as the export file jan_sales.dmp to the data mart platform, using any transportation mechanism for flat files. Once the datafiles have been copied, the tablespace ts_temp_sales can be set to READ WRITE mode if desired.
Step 4 Import the Metadata
Once the files have been copied to the data mart, the metadata should be imported into the data mart:
IMP TRANSPORT_TABLESPACE=y DATAFILES='/db/tempjan.f'
TABLESPACES=ts_temp_sales FILE=jan_sales.dmp

At this point, the tablespace ts_temp_sales and the table temp_sales_jan are accessible in the data mart. You can incorporate this new data into the data mart's tables.
You can insert the data from the temp_sales_jan table into the data mart's sales table in one of two ways:
INSERT /*+ APPEND */ INTO sales SELECT * FROM temp_sales_jan;

Following this operation, you can delete the temp_sales_jan table (and even the entire ts_temp_sales tablespace).
Alternatively, if the data mart's sales table is partitioned by month, then the new transported tablespace and the temp_sales_jan table can become a permanent part of the data mart. The temp_sales_jan table can become a partition of the data mart's sales table:
ALTER TABLE sales ADD PARTITION sales_00jan VALUES
LESS THAN (TO_DATE('01-feb-2000','dd-mon-yyyy'));
ALTER TABLE sales EXCHANGE PARTITION sales_00jan
WITH TABLE temp_sales_jan INCLUDING INDEXES WITH VALIDATION;

Other Uses of Transportable Tablespaces

The previous example illustrates a typical scenario for transporting data in a data warehouse. However, transportable tablespaces can be used for many other purposes. In a data warehousing environment, transportable tablespaces should be viewed as a utility (much like Import/Export or SQL*Loader), whose purpose is to move large volumes of data between Oracle databases. When used in conjunction with parallel data movement operations such as the CREATE TABLE ... AS SELECT and INSERT ... AS SELECT statements, transportable tablespaces provide an important mechanism for quickly transporting data for many purposes.

Overview of Loading and Transformation in Data Warehouses

Data transformations are often the most complex and, in terms of processing time, the most costly part of the extraction, transformation, and loading (ETL) process. They can range from simple data conversions to extremely complex data scrubbing techniques. Many, if not all, data transformations can occur within an Oracle database, although transformations are often implemented outside of the database (for example, on flat files) as well.

This chapter introduces techniques for implementing scalable and efficient data transformations within the Oracle Database. The examples in this chapter are relatively simple. Real-world data transformations are often considerably more complex. However, the transformation techniques introduced in this chapter meet the majority of real-world data transformation requirements, often with more scalability and less programming than alternative approaches.

This chapter does not seek to illustrate all of the typical transformations that would be encountered in a data warehouse, but to demonstrate the types of fundamental technology that can be applied to implement these transformations and to provide guidance in how to choose the best techniques.

Transformation Flow

From an architectural perspective, you can transform your data in two ways:

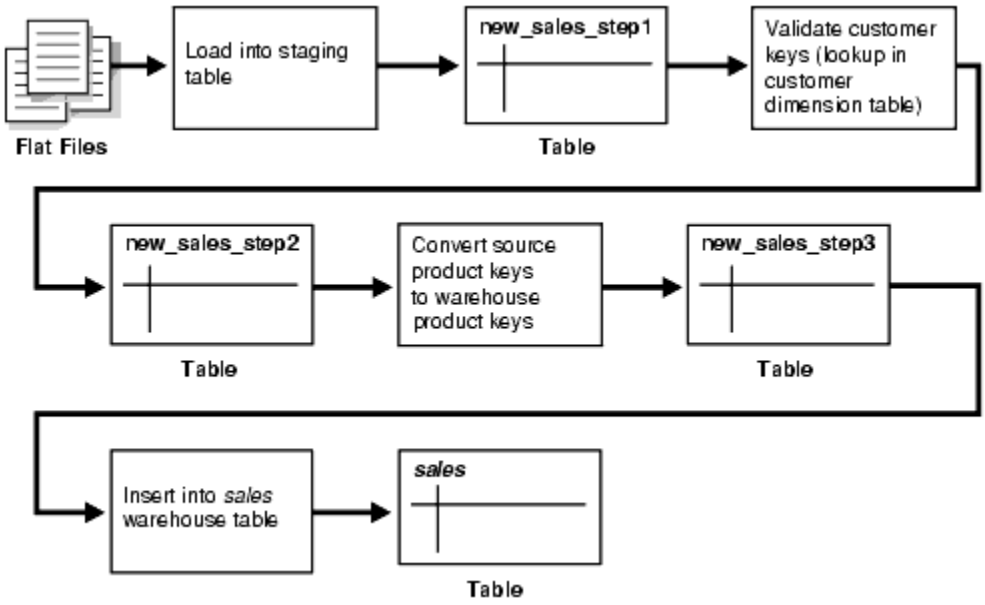
1) Multistage Data Transformation
2) Pipelined Data Transformation

Multistage Data Transformation

The data transformation logic for most data warehouses consists of multiple steps. For example, in transforming new records to be inserted into a sales table, there may be separate logical transformation steps to validate each dimension key.

Figure 14-1 offers a graphical way of looking at the transformation logic.

Figure 14-1 Multistage Data Transformation



When using Oracle Database as a transformation engine, a common strategy is to implement each transformation as a separate SQL operation and to create a separate, temporary staging table (such as the tables new_sales_step1 and new_sales_step2 in Figure 14-1) to store the incremental results for each step. This load-then-transform strategy also provides a natural checkpointing scheme to the entire transformation process, which enables to the process to be more easily monitored and restarted. However, a disadvantage to multistaging is that the space and time requirements increase.

It may also be possible to combine many simple logical transformations into a single SQL statement or single PL/SQL procedure. Doing so may provide better performance than performing each step independently, but it may also introduce difficulties in modifying, adding, or dropping individual transformations, as well as recovering from failed transformations.

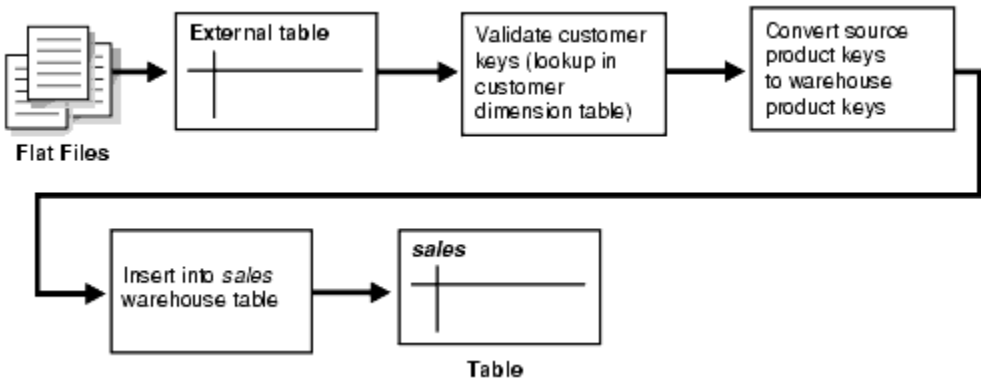
Pipelined Data Transformation

The ETL process flow can be changed dramatically and the database becomes an integral part of the ETL solution.

The new functionality renders some of the former necessary process steps obsolete while some others can be remodeled to enhance the data flow and the data transformation to become more scalable and non-interruptive. The task shifts from serial transform-then-load process (with most of the tasks done outside the database) or load-then-transform process, to an enhanced transform-while-loading.

Oracle offers a wide variety of new capabilities to address all the issues and tasks relevant in an ETL scenario. It is important to understand that the database offers toolkit functionality rather than trying to address a one-size-fits-all solution. The underlying database has to enable the most appropriate ETL process flow for a specific customer need, and not dictate or constrain it from a technical perspective. Figure 14-2 illustrates the new functionality, which is discussed throughout later sections.

Figure 14-2 ***Pipelined Data Transformation***



Description of the illustration dwhsg107.gif

Loading Mechanisms

You can use the following mechanisms for loading a data warehouse:

- Loading a Data Warehouse with SQL*Loader
- Loading a Data Warehouse with External Tables
- Loading a Data Warehouse with OCI and Direct-Path APIs
- Loading a Data Warehouse with Export/Import

Schemas in Data Warehouses

A **schema** is a collection of database objects, including tables, views, indexes, and synonyms.

There is a variety of ways of arranging schema objects in the schema models designed for data warehousing. One data warehouse schema model is a star schema.

Star Schemas

The **star schema** is perhaps the simplest data warehouse schema. It is called a star schema because the entity-relationship diagram of this schema resembles a star, with points radiating from a central table. The center of the star consists of a large fact table and the points of the star are the dimension tables.

A **star query** is a join between a fact table and a number of dimension tables. Each dimension table is joined to the fact table using a primary key to foreign key join, but the dimension tables are not joined to each other. The optimizer recognizes star queries and generates efficient execution plans for them.

A typical fact table contains keys and measures. For example, in the sh sample schema, the fact table, sales, contain the measures quantity_sold, amount, and cost, and the keys cust_id, time_id, prod_id, channel_id, and promo_id. The dimension tables are customers, times, products, channels, and promotions. The products dimension table, for example, contains information about each product number that appears in the fact table.

A star join is a primary key to foreign key join of the dimension tables to a fact table.

The main advantages of star schemas are that they:

Provide a direct and intuitive mapping between the business entities being analyzed by end users and the schema design.

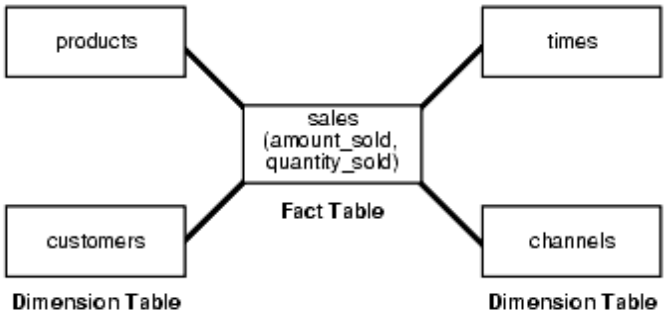
Provide highly optimized performance for typical star queries.

Are widely supported by a large number of business intelligence tools, which may anticipate or even require that the data warehouse schema contain dimension tables.

Star schemas are used for both simple data marts and very large data warehouses.

Figure 19-2 presents a graphical representation of a star schema.

Figure 19-2 Star Schema



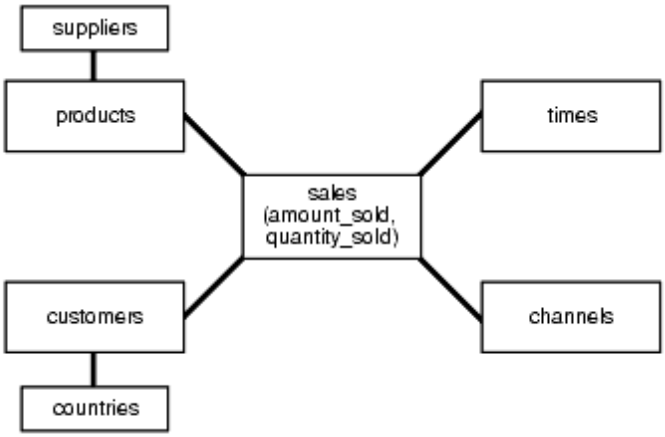
Snowflake Schemas

The snowflake schema is a more complex data warehouse model than a star schema, and is a type of star schema. It is called a snowflake schema because the diagram of the schema resembles a snowflake.

Snowflake schemas normalize dimensions to eliminate redundancy. That is, the dimension data has been grouped into multiple tables instead of one large table. For example, a product dimension table in a star schema might be normalized into a products table, a product_category table, and a product_manufacturer table in a snowflake

schema. While this saves space, it increases the number of dimension tables and requires more foreign key joins. The result is more complex queries and reduced query performance. Figure 19-3 presents a graphical representation of a snowflake schema.

Figure 19-3 Snowflake Schema



Description of the illustration dwhsg008.gif

Optimizing Star Queries

You should consider the following when using star queries:

- Tuning Star Queries
- Using Star Transformation

Tuning Star Queries

To get the best possible performance for star queries, it is important to follow some basic guidelines:

- A bitmap index should be built on each of the foreign key columns of the fact table or tables.
- The initialization parameter STAR_TRANSFORMATION_ENABLED should be set to TRUE. This enables an important optimizer feature for star-queries. It is set to FALSE by default for backward-compatibility.

When a data warehouse satisfies these conditions, the majority of the star queries running in the data warehouse will use a query execution strategy known as the star transformation. The star transformation provides very efficient query performance for star queries.

Using Star Transformation

The star transformation is a powerful optimization technique that relies upon implicitly rewriting (or transforming) the SQL of the original star query. The end user never needs to know any of the details about the star transformation. Oracle's query optimizer automatically chooses the star transformation where appropriate.

The star transformation is a query transformation aimed at executing star queries efficiently. Oracle processes a star query using two basic phases. The first phase retrieves exactly the necessary rows from the fact table (the result set). Because this retrieval utilizes bitmap indexes, it is very efficient. The second phase joins this result set to the dimension tables. An example of an end user query is: "What were the sales and profits for the grocery department of stores in the west and southwest sales districts over the last three quarters?" This is a simple star query.

How Oracle Chooses to Use Star Transformation

The optimizer generates and saves the best plan it can produce without the transformation. If the transformation is enabled, the optimizer then tries to apply it to the query and, if applicable, generates the best plan using the transformed query. Based on a comparison of the cost estimates between the best plans for the two versions of the query, the optimizer will then decide whether to use the best plan for the transformed or untransformed version.

If the query requires accessing a large percentage of the rows in the fact table, it might be better to use a full table scan and not use the transformations. However, if the constraining predicates on the dimension tables are sufficiently selective that only a small portion of the fact table needs to be retrieved, the plan based on the transformation will probably be superior.

Note that the optimizer generates a subquery for a dimension table only if it decides that it is reasonable to do so based on a number of criteria. There is no guarantee that subqueries will be generated for all dimension tables. The optimizer may also decide, based on the properties of the tables and the query, that the transformation does not merit being applied to a particular query. In this case the best regular plan will be used.

Star Transformation Restrictions

Star transformation is not supported for tables with any of the following characteristics:

Queries with a table hint that is incompatible with a bitmap access path

Queries that contain bind variables

Tables with too few bitmap indexes. There must be a bitmap index on a fact table column for the optimizer to generate a subquery for it.

Remote fact tables. However, remote dimension tables are allowed in the subqueries that are generated.

Anti-joined tables

Tables that are already used as a dimension table in a subquery

Tables that are really unmerged views, which are not view partitions

The star transformation may not be chosen by the optimizer for the following cases:

Tables that have a good single-table access path

Tables that are too small for the transformation to be worthwhile

In addition, temporary tables will not be used by star transformation under the following conditions:

The database is in read-only mode

The star query is part of a transaction that is in serializable mode

Informatica

Informatica is a tool, supporting all the steps of Extraction, Transformation and Load process. Now a days Informatica is also being used as an Integration tool.

Informatica is an easy to use tool. It has got a simple visual interface like forms in visual basic. You just need to drag and drop different objects (known as transformations) and design process flow for Data extraction transformation and load. These process flow diagrams are known as **mappings**. Once a mapping is made, it can be scheduled to run as and when required. In the background Informatica server takes care of fetching data from source, transforming it, & loading it to the target systems/databases.

Informatica can communicate with all major data sources (mainframe/RDBMS/Flat Files/XML/VSM/SAP etc), can move/transform data between them. It can move huge volumes of data in a very effective way, many a times better than even bespoke programs written for specific data movement only. It can throttle the transactions (do big updates in small chunks to avoid long locking and filling the transactional log). It can effectively join data from two distinct data sources (even a xml file can be joined with a relational table). In all, Informatica has got the ability to effectively integrate heterogeneous data sources & converting raw data into useful information.

Before we start actually working in Informatica, let's have an idea about the company owning this wonderful product.

Some facts and figures about Informatica Corporation:

- Founded in 1993, based in Redwood City, California
- 1400+ Employees; 3450 + Customers; 79 of the Fortune 100 Companies
- NASDAQ Stock Symbol: INFA; Stock Price: $18.74 (09/04/2009)
- Revenues in fiscal year 2008: $455.7M

Informatica Developer Networks: 20000 Members

Informatica Software Architecture illustrated

Informatica ETL product, known as Informatica Power Center consists of 3 main components.

1. Informatica PowerCenter Client Tools:

These are the development tools installed at developer end. These tools enable a developer to

- Define transformation process, known as mapping. **(Designer)**
- Define run-time properties for a mapping, known as sessions **(Workflow Manager)**
- Monitor execution of sessions **(Workflow Monitor)**
- Manage repository, useful for administrators **(Repository Manager)**
- Report Metadata **(Metadata Reporter)**

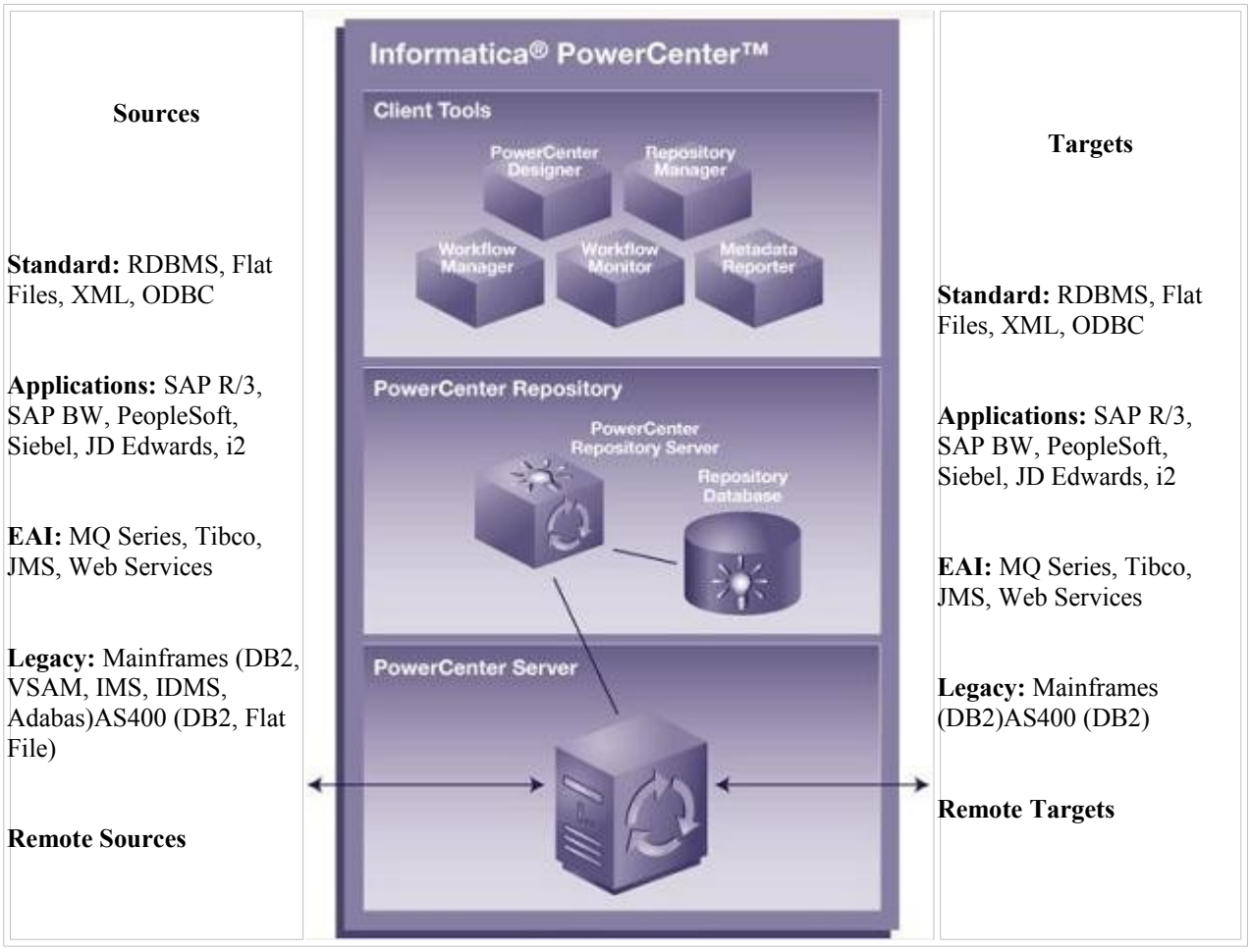
2. Informatica PowerCenter Repository:

Repository is the heart of Informatica tools. Repository is a kind of data inventory where all the data related to mappings, sources, targets etc is kept. This is the place where all the metadata for your application is stored. All the client tools and Informatica Server fetch data from Repository. Informatica client and server without repository is same as a PC without memory/harddisk, which has got the ability to process data but has no data to process. This can be treated as backend of Informatica.

3. Informatica PowerCenter Server:

Server is the place, where all the executions take place. Server makes physical connections to sources/targets, fetches data, applies the transformations mentioned in the mapping and loads the data in the target system.

This architecture is visually explained in diagram below:



Informatica Product Line

Informatica is a powerful ETL tool from Informatica Corporation, a leading provider of enterprise data integration software and ETL softwares.

The important products provided by Informatica Corporation is provided below:

- Power Center
- Power Mart
- Power Exchange
- Power Center Connect
- Power Channel
- Metadata Exchange
- Power Analyzer
- Super Glue

Power Center & Power Mart: Power Mart is a departmental version of Informatica for building, deploying, and managing data warehouses and data marts. Power center is used for corporate enterprise data warehouse and power mart is used for departmental data warehouses like data marts. Power Center supports global repositories and networked repositories and it can be connected to several sources. Power Mart supports single repository and it can be connected to fewer sources when compared to Power Center. Power Mart can extensibily grow to an enterprise implementation and it is easy for developer productivity through a codeless environment.

Power Exchange: Informatica Power Exchange as a stand alone service or along with Power Center, helps organizations leverage data by avoiding manual coding of data extraction programs. Power Exchange supports batch, real time and changed data capture options in main frame(DB2, VSAM, IMS etc.,), mid range (AS400 DB2 etc.,), and for relational databases (oracle, sql server, db2 etc) and flat files in unix, linux and windows systems.

Power Center Connect: This is add on to Informatica Power Center. It helps to extract data and metadata from ERP systems like IBM's MQSeries, Peoplesoft, SAP, Siebel etc. and other third party applications.

Power Channel: This helps to transfer large amount of encrypted and compressed data over LAN, WAN, through Firewalls, tranfer files over FTP, etc.

Meta Data Exchange: Metadata Exchange enables organizations to take advantage of the time and effort already invested in defining data structures within their IT environment when used with Power Center. For example, an organization may be using data modeling tools, such as Erwin, Embarcadero, Oracle designer, Sybase Power Designer etc for developing data models. Functional and technical team should have spent much time and effort in creating the data model's data structures(tables, columns, data types, procedures, functions, triggers etc). By using meta deta exchange, these data structures can be imported into power center to identifiy source and target mappings which leverages time and effort. There is no need for informatica developer to create these data structures once again.

Power Analyzer: Power Analyzer provides organizations with reporting facilities. PowerAnalyzer makes accessing, analyzing, and sharing enterprise data simple and easily available to decision makers. PowerAnalyzer enables to gain insight into business processes and develop business intelligence.

With PowerAnalyzer, an organization can extract, filter, format, and analyze corporate information from data stored in a data warehouse, data mart, operational data store, or otherdata storage models. PowerAnalyzer is best with a dimensional data warehouse in a relational database. It can also run reports on data in any table in a relational database that do not conform to the dimensional model.

Super Glue: Superglue is used for loading metadata in a centralized place from several sources. Reports can be run against this superglue to analyze meta data.

TRANSFORMATIONS

Informatica Transformations

A transformation is a repository object that generates, modifies, or passes data. The Designer provides a set of transformations that perform specific functions. For example, an Aggregator transformation performs calculations on groups of data.

Transformations can be of two types:

Active Transformation

An active transformation can change the number of rows that pass through the transformation, change the transaction boundary, can change the row type. For example, Filter, Transaction Control and Update Strategy are active transformations.

Note: The key point is to note that Designer does not allow you to connect multiple active transformations or an active and a passive transformation to the same downstream transformation or transformation input group because the Integration Service may not be able to concatenate the rows passed by active transformations However, Sequence Generator transformation(SGT) is an exception to this rule. A SGT does not receive data. It generates unique numeric values. As a result, the Integration Service does not encounter problems concatenating rows passed by a SGT and an active transformation.

Passive Transformation.

A passive transformation does not change the number of rows that pass through it, maintains the transaction boundary, and maintains the row type.

The key point is to note that Designer allows you to connect multiple transformations to the same downstream transformation or transformation input group only if all transformations in the upstream branches are passive. The transformation that originates the branch can be active or passive.

Transformations can be Connected or UnConnected to the data flow.

Connected Transformation

Connected transformation is connected to other transformations or directly to target table in the mapping.

UnConnected Transformation

An unconnected transformation is not connected to other transformations in the mapping. It is called within another transformation, and returns a value to that transformation

1.Expression Transformation Connected/passive

You can use the Expression transformation to calculate values in a single row before you write to the target. For example, you might need to adjust employee salaries, concatenate first and last names, or convert strings to numbers. You can use the Expression transformation to perform any non-aggregate calculations. You can also use the Expression transformation to test conditional statements before you output the results to target tables or other transformations.

Calculating Values

To use the Expression transformation to calculate values for a single row, you must include the following ports:

- **Input or input/output ports for each value used in the calculation.** For example, when calculating the total price for an order, determined by multiplying the unit price by the quantity ordered, the input or input/output ports. One port provides the unit price and the other provides the quantity ordered.
- **Output port for the expression.** You enter the expression as a configuration option for the output port. The return value for the output port needs to match the return value of the expression. For information on entering expressions, see "Transformations" in the *Designer Guide*. Expressions use the transformation language, which includes SQL-like functions, to perform calculations

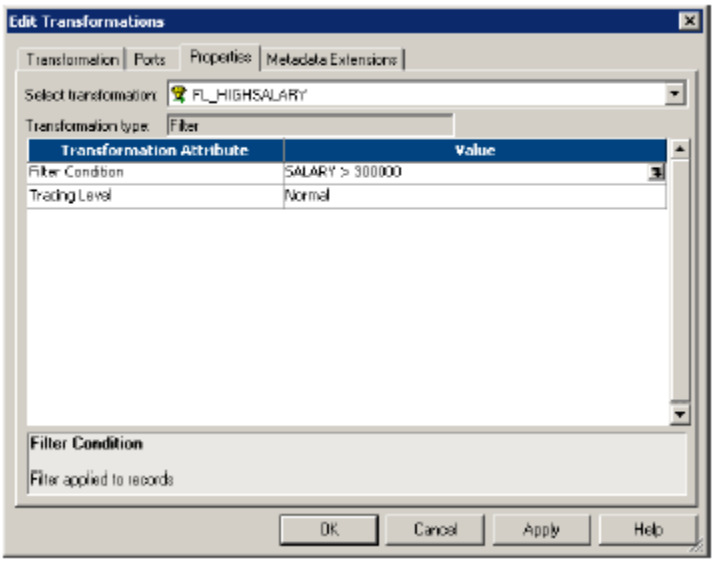
You can enter multiple expressions in a single Expression transformation. As long as you enter only one expression for each output port, you can create any number of output ports in the transformation. In this way, you can use one Expression transformation rather than creating separate transformations for each calculation that requires the same set of data.

2.Filter transformation Connecte / Active

The Filter transformation allows you to filter rows in a mapping. You pass all the rows from a source transformation through the Filter transformation, and then enter a filter condition for the transformation. All ports in a Filter transformation are input/output, and only rows that meet the condition pass through the Filter transformation.

In some cases, you need to filter data based on one or more conditions before writing it to targets. For example, if you have a human resources target containing information about current employees, you might want to filter out employees who are part-time and hourly

Figure 6-2. Specifying a Filter Condition in a Filter Transformation



With the filter of SALARY > 30000, only rows of data where employees that make salaries greater than $30,000 pass through to the target.

As an active transformation, the Filter transformation may change the number of rows passed through it. A filter condition returns TRUE or FALSE for each row that passes through the transformation, depending on whether a row meets the specified condition. Only rows that return TRUE pass through this transformation. Discarded rows do not appear in the session log or reject files.You use the transformation language to enter the filter condition. The condition is an expression that returns TRUE or FALSE.

To maximize session performance, include the Filter transformation as close to the sources in the mapping as possible. Rather than passing rows you plan to discard through the mapping, you then filter out unwanted data early in the flow of data from sources to targets

Use the Filter transformation early in the mapping.To maximize session performance, keep the Filter transformation as close as possible to the sources in the mapping. Rather than passing rows that you plan to discard through the mapping, you can filter out unwanted data early in the flow of data from sources to targets.

To filter out rows containing null values or spaces, use the ISNULL and IS_SPACES functions to test the value of the port. For example, if you want to filter out rows that contain NULLs in the FIRST_NAME port, use the following condition: IIF(ISNULL(FIRST_NAME),FALSE,TRUE)

3.Joiner Transformation Connected/Active

You can use the Joiner transformation to join source data from two related heterogeneous sources residing in different locations or file systems. Or, you can join data from the same source.

The Joiner transformation joins two sources with at least one matching port. The Joiner transformation uses a condition that matches one or more pairs of ports between the two sources. If you need to join more than two sources, you can add more Joiner transformations to the mapping.The Joiner transformation requires input from two separate pipelines or two branches from one pipeline.

The Joiner transformation accepts input from most transformations. However, there are some limitations on the pipelines you connect to the Joiner transformation. You cannot use a Joiner transformation in the following situations:

- Either input pipeline contains an Update Strategy transformation.
- You connect a Sequence Generator transformation directly before the Joiner transformation

The join condition contains ports from both input sources that must match for the PowerCenter Server to join two rows. Depending on the type of join selected, the Joiner transformation either adds the row to the result set or discards the row. The Joiner produces result sets based on the join type, condition, and input data sources.

Before you define a join condition, verify that the master and detail sources are set for optimal performance. During a session, the PowerCenter Server compares each row of the master source against the detail source. The fewer unique rows in the master, the fewer iterations of the join comparison occur, which speeds the join process. To improve performance, designate the source with the smallest count of distinct values as the master.

You define the join type on the Properties tab in the transformation. The Joiner transformation supports the following types of joins:

- Normal
- Master Outer
- Detail Outer
- Full Outer

You can improve session performance by configuring the Joiner transformation to use sorted input. When you configure the Joiner transformation to use sorted data, the PowerCenter Server improves performance by minimizing disk input and output. You see the greatest performance improvement when you work with large data sets.

When you use a Joiner transformation in a mapping, you must configure the mapping according to the number of pipelines and sources you intend to use. You can configure a mapping to join the following types of data:

- **Data from multiple sources**. When you want to join more than two pipelines, you must configure the mapping using multiple Joiner transformations.
- **Data from the same source**. When you want to join data from the same source, you must configure the mapping to use the same source

Unsorted Joiner Transformation

When the PowerCenter Server processes an unsorted Joiner transformation, it reads all master rows before it reads the detail rows. To ensure it reads all master rows before the detail rows, the PowerCenter Server blocks the detail source while it caches rows from the master source. Once the PowerCenter Server reads and caches all master rows, it unblocks the detail source and reads the detail rows

Sorted Joiner Transformation

When the PowerCenter Server processes a sorted Joiner transformation, it blocks data based on the mapping configuration.

When the PowerCenter Server can block and unblock the source pipelines connected to the Joiner transformation without blocking all sources in the target load order group simultaneously, it uses blocking logic to process the Joiner transformation. Otherwise, it does not use blocking logic and instead it stores more rows in the cache.

Perform joins in a database when possible.

Performing a join in a database is faster than performing a join in the session. In some cases, this is not possible, such as joining tables from two different databases or flat file systems. If you want to perform a join in a database, you can use the following options:

- Create a pre-session stored procedure to join the tables in a database.
- Use the Source Qualifier transformation to perform the join.

Join sorted data when possible.

You can improve session performance by configuring the Joiner transformation to use sorted input. When you configure the Joiner transformation to use sorted data, the PowerCenter Server improves performance by minimizing disk input and output. You see the greatest performance improvement when you work with large data sets.

For an unsorted Joiner transformation, designate as the master source the source with fewer rows.

For optimal performance and disk storage, designate the master source as the source with the fewer rows. During a session, the Joiner transformation compares each row of the master source against the detail source. The fewer unique rows in the master, the fewer iterations of the join comparison occur, which speeds the join process.

For a sorted Joiner transformation, designate as the master source the source with fewer duplicate key values.

For optimal performance and disk storage, designate the master source as the source with fewer duplicate key values. When the PowerCenter Server processes a sorted Joiner transformation, it caches rows for one hundred keys at a time. If the master source contains many rows with the same key value, the PowerCenter Server must cache more rows, and performance can be slowed.

4.Rank Transformation active/connected

The Rank transformation allows you to select only the top or bottom rank of data. You can use a Rank transformation to return the largest or smallest numeric value in a port or group. You can also use a Rank transformation to return the strings at the top or the bottom of a session sort order. During the session, the PowerCenter Server caches input data until it can perform the rank calculations.

You connect all ports representing the same row set to the transformation. Only the rows that fall within that rank, based on some measure you set when you configure the transformation, pass through the Rank transformation. You can also write expressions to transform data or perform calculations

As an active transformation, the Rank transformation might change the number of rows passed through it. You might pass 100 rows to the Rank transformation, but select to rank only the top 10 rows, which pass from the Rank transformation to another transformation.

You can connect ports from only one transformation to the Rank transformation. The Rank transformation allows you to create local variables and write non-aggregate expressions.

Rank Caches

During a session, the PowerCenter Server compares an input row with rows in the data cache. If the input row out-ranks a cached row, the PowerCenter Server replaces the cached row with the input row. If you configure the Rank transformation to rank across multiple groups, the PowerCenter Server ranks incrementally for each group it finds.

The PowerCenter Server stores group information in an index cache and row data in a data cache. If you create multiple partitions in a pipeline, the PowerCenter Server creates separate caches for each partition

Rank Transformation Properties

When you create a Rank transformation, you can configure the following properties:

- Enter a cache directory.
- Select the top or bottom rank.
- Select the input/output port that contains values used to determine the rank. You can select only one port to define a rank.
- Select the number of rows falling within a rank.
- Define groups for ranks, such as the 10 least expensive products for each manufacturer.

the Rank transformation changes the number of rows in two different ways. By filtering all but the rows falling within a top or bottom rank, you reduce the number of rows that pass through the transformation. By defining groups, you create one set of ranked rows for each group.

5.Router Transformation connected/active

A Router transformation is similar to a Filter transformation because both transformations allow you to use a condition to test data. A Filter transformation tests data for one condition and drops the rows of data that do not meet the condition. However, a Router transformation tests data for one or more conditions and gives you the option to route rows of data that do not meet any of the conditions to a default output group.

If you need to test the same input data based on multiple conditions, use a Router transformation in a mapping instead of creating multiple Filter transformations to perform the same task. The Router transformation is more efficient. For example, to test data based on three conditions, you only need one Router transformation instead of three filter transformations to perform this task. Likewise, when you use a Router transformation in a mapping, the PowerCenter Server processes the incoming data only once. When you use multiple Filter transformations in a mapping, the PowerCenter Server processes the incoming data for each transformation.

Using Group Filter Conditions

You can test data based on one or more group filter conditions. You create group filter conditions on the Groups tab using the Expression Editor. You can enter any expression that returns a single value. You can also specify a constant for the condition. A group filter condition returns TRUE or FALSE for each row that passes through the transformation, depending on whether a row satisfies the specified condition. Zero (0) is the equivalent of FALSE, and any non-zero value is the equivalent of TRUE. The PowerCenter Server passes the rows of data that evaluate to TRUE to each transformation or target that is associated with each user-defined group

A Router transformation has input ports and output ports. Input ports are in the input group, and output ports are in the output groups. You can create input ports by copying them from another transformation or by manually creating them on the Ports tab.

6.Sequence Generator Transformation passive/connected

The Sequence Generator transformation generates numeric values. You can use the Sequence Generator to create unique primary key values, replace missing primary keys, or cycle through a sequential range of numbers.

The Sequence Generator transformation is a connected transformation. It contains two output ports that you can connect to one or more transformations. The PowerCenter Server generates a block of sequence numbers each time a block of rows enters a connected transformation. If you connect CURRVAL, the PowerCenter Server processes one row in each block. When NEXTVAL is connected to the input port of another transformation, the PowerCenter Server generates a sequence of numbers. When CURRVAL is connected to the input port of another transformation, the PowerCenter Server generates the NEXTVAL value plus the Increment By value.

When creating primary or foreign keys, only use the Cycle option to prevent the PowerCenter Server from creating duplicate primary keys. You might do this by selecting the Truncate Target Table option in the session properties (if appropriate) or by creating composite keys.

To create a composite key, you can configure the PowerCenter Server to cycle through a smaller set of values. For example, if you have three stores generating order numbers, you might have a Sequence Generator cycling through values from 1 to 3, incrementing by 1. When you pass the following set of foreign keys, the generated values then create unique composite keys:

The Sequence Generator transformation provides two output ports: NEXTVAL and CURRVAL. You cannot edit or delete these ports. Likewise, you cannot add ports to the transformation.

NEXTVAL

Connect NEXTVAL to multiple transformations to generate unique values for each row in each transformation. Use the NEXTVAL port to generate sequence numbers by connecting it to a transformation or target. You connect the NEXTVAL port to a downstream transformation to generate the sequence based on the Current Value and Increment By properties.

For example, you might connect NEXTVAL to two target tables in a mapping to generate unique primary key values. The PowerCenter Server creates a column of unique primary key values for each target table. The column of unique primary key values is sent to one target table as a block of sequence numbers. The second targets receives a block of sequence numbers from the Sequence Generator transformation only after the first target table receives the block of sequence numbers.

If you want the *same* values to go to more than one target that receives data from a single transformation, you can connect a Sequence Generator transformation to that preceding transformation. The PowerCenter Server Sequence Generator transformation processes the values into a block of sequence numbers. This allows the PowerCenter Server to pass unique values to the transformation, and then route rows from the transformation to targets.

Start Value and Cycle

You can use Cycle to generate a repeating sequence, such as numbers 1 through 12 to correspond to the months in a year.

To cycle the PowerCenter Server through a sequence:

1. Enter the lowest value in the sequence that you want the PowerCenter Server to use for the Start Value.

2. Then enter the highest value to be used for End Value.

3. Select Cycle.

As it cycles, the PowerCenter Server reaches the configured end value for the sequence, it wraps around and starts the cycle again, beginning with the configured Start Value.

Number of Cached Values

Number of Cached Values determines the number of values the PowerCenter Server caches at one time. When Number of Cached Values is greater than zero, the PowerCenter Server caches the configured number of values and updates the current value each time it caches values.

When multiple sessions use the same reusable Sequence Generator transformation at the same time, there might be multiple instances of the Sequence Generator transformation. To avoid generating the same values for each session, reserve a range of sequence values for each session by configuring Number of Cached Values.

Reset If you select Reset for a non-reusable Sequence Generator transformation, the PowerCenter Server generates values based on the original current value each time it starts the session. Otherwise, the PowerCenter Server updates the current value to reflect the last-generated value plus one, and then uses the updated value the next time it uses the Sequence Generator transformation

7.Sorter Transformation connected/active

The Sorter transformation allows you to sort data. You can sort data in ascending or descending order according to a specified sort key. You can also configure the Sorter transformation for case-sensitive sorting, and specify whether the output rows should be distinct. The Sorter transformation is an active transformation. It must be connected to the data flow.

You can sort data from relational or flat file sources. You can also use the Sorter transformation to sort data passing through an Aggregator transformation configured to use sorted input.

When you create a Sorter transformation in a mapping, you specify one or more ports as a sort key and configure each sort key port to sort in ascending or descending order. You also configure sort criteria the PowerCenter Server applies to all sort key ports and the system resources it allocates to perform the sort operation.

The Sorter transformation contains only input/output ports. All data passing through the Sorter transformation is sorted according to a sort key. The sort key is one or more ports that you want to use as the sort criteria.

Sorter Cache Size

The PowerCenter Server uses the Sorter Cache Size property to determine the maximum amount of memory it can allocate to perform the sort operation. The PowerCenter Server passes all incoming data into the Sorter transformation before it performs the sort operation.

You can specify any amount between 1 MB and 4 GB for the Sorter cache size. If the total configured session cache size is 2 GB (2,147,483,648 bytes) or greater, you must run the session on a 64-bit PowerCenter Server.

Distinct Output Rows

You can configure the Sorter transformation to treat output rows as distinct. If you configure the Sorter transformation for distinct output rows, the Mapping Designer configures all ports as part of the sort key. When the PowerCenter Server runs the session, it discards duplicate rows compared during the sort operation.

Transformation Scope

The transformation scope specifies how the PowerCenter Server applies the transformation logic to incoming data:

- **Transaction.** Applies the transformation logic to all rows in a transaction. Choose Transaction when a row of data depends on all rows in the same transaction, but does not depend on rows in other transactions.
- **All Input.** Applies the transformation logic on all incoming data. When you choose All Input, the PowerCenter drops incoming transaction boundaries. Choose All Input when a row of data depends on all rows in the source.

8.Transaction Control Transformation Active/connected

PowerCenter allows you to control commit and rollback transactions based on a set of rows that pass through a Transaction Control transformation. A transaction is the set of rows bound by commit or rollback rows. You can define a transaction based on a varying number of input rows. You might want to define transactions based on a group of rows ordered on a common key, such as employee ID or order entry date.

In PowerCenter, you define transaction control at two levels:

- **Within a mapping.** Within a mapping, you use the Transaction Control transformation to define a transaction. You define transactions using an expression in a Transaction Control transformation. Based on the return value of the expression, you can choose to commit, roll back, or continue without any transaction changes.
- **Within a session.** When you configure a session, you configure it for user-defined commit. You can choose to commit or roll back a transaction if the PowerCenter Server fails to transform or write any row to the target.

The expression contains values that represent actions the PowerCenter Server performs based on the return value of the condition. The PowerCenter Server evaluates the condition on a row-by-row basis. The return value determines whether the PowerCenter Server commits, rolls back, or makes no transaction changes to the row. When the PowerCenter Server issues a commit or rollback based on the return value of the expression, it begins a new transaction. Use the following built-in variables in the Expression Editor when you create a transaction control expression:

- **TC_CONTINUE_TRANSACTION.** The PowerCenter Server does not perform any transaction change for this row. This is the default value of the expression.
- **TC_COMMIT_BEFORE.** The PowerCenter Server commits the transaction, begins a new transaction, and writes the current row to the target. The current row is in the new transaction.

- **TC_COMMIT_AFTER.** The PowerCenter Server writes the current row to the target, commits the transaction, and begins a new transaction. The current row is in the committed transaction.
- **TC_ROLLBACK_BEFORE.** The PowerCenter Server rolls back the current transaction, begins a new transaction, and writes the current row to the target. The current row is in the new transaction.
- **TC_ROLLBACK_AFTER.** The PowerCenter Server writes the current row to the target, rolls back the transaction, and begins a new transaction. The current row is in the rolled back transaction.

If the transaction control expression evaluates to a value other than commit, rollback, or continue, the PowerCenter Server fails the session.

Mapping Guidelines and Validation

Consider the following rules and guidelines when you create a mapping with a Transaction Control transformation:

- If the mapping includes an XML target, and you choose to append or create a new document on commit, the input groups must receive data from the same transaction control point.
- Transaction Control transformations connected to any target other than relational, XML, or dynamic IBM MQSeries targets are ineffective for those targets.
- You must connect each target instance to a Transaction Control transformation.
- You can connect multiple targets to a single Transaction Control transformation.
- You can connect only one effective Transaction Control transformation to a target.
- You cannot place a Transaction Control transformation in a pipeline branch that starts with a Sequence Generator transformation.
- If you use a dynamic Lookup transformation and a Transaction Control transformation in the same mapping, a rolled-back transaction might result in unsynchronized target data.
- A Transaction Control transformation may be effective for one target and ineffective for another target. If each target is connected to an effective Transaction Control transformation, the mapping is validEither all targets or none of the targets in the mapping should be connected to an effective Transaction Control transformation.

9.Aggregator Transformation connected/active

The Aggregator transformation allows you to perform aggregate calculations, such as averages and sums. The Aggregator transformation is unlike the Expression transformation, in that you can use the Aggregator transformation to perform calculations on groups. The Expression transformation permits you to perform calculations on a row-by-row basis only.

Components of the Aggregator Transformation

The Aggregator is an active transformation, changing the number of rows in the pipeline. The Aggregator transformation has the following components and options:

- **Aggregate expression.** Entered in an output port. Can include non-aggregate expressions and conditional clauses.
- **Group by port.** Indicates how to create groups. The port can be any input, input/output, output, or variable port. When grouping data, the Aggregator transformation outputs the last row of each group unless otherwise specified.
- **Sorted input.** Use to improve session performance. To use sorted input, you must pass data to the Aggregator transformation sorted by group by port, in ascending or descending order.
- **Aggregate cache.** The PowerCenter Server stores data in the aggregate cache until it completes aggregate calculations. It stores group values in an index cache and row data in the data cache.

The Designer allows aggregate expressions only in the Aggregator transformation. An aggregate expression can include conditional clauses and non-aggregate functions. It can also include one aggregate function nested within another aggregate function, such as:

MAX(COUNT(ITEM))

The result of an aggregate expression varies depending on the group by ports used in the transformation. For example, when the PowerCenter Server calculates the following aggregate expression with no group by ports defined, it finds the total quantity of items sold:

SUM(QUANTITY)

Aggregate Functions

You can use the following aggregate functions within an Aggregator transformation. You can nest one aggregate function within another aggregate function.

The transformation language includes the following aggregate functions:

- AVG
- COUNT
- FIRST
- LAST
- MAX
- MEDIAN
- MIN
- PERCENTILE
- STDDEV
- SUM
- VARIANCE

Non-Aggregate Functions

You can also use non-aggregate functions in the aggregate expression.

The following expression returns the highest number of items sold for each item (grouped by item). If no items were sold, the expression returns 0.

IIF(MAX(QUANTITY) > 0, MAX(QUANTITY), 0))

Null Values in Aggregate Functions

When you configure the PowerCenter Server, you can choose how you want the PowerCenter Server to handle null values in aggregate functions. You can choose to treat null values in aggregate functions as NULL or zero. By default, the PowerCenter Server treats null values as NULL in aggregate functions.

Group By Ports

The Aggregator transformation allows you to define groups for aggregations, rather than performing the aggregation across all input data. For example, rather than finding the total company sales, you can find the total sales grouped by region.

To define a group for the aggregate expression, select the appropriate input, input/output, output, and variable ports in the Aggregator transformation. You can select multiple group by ports, creating a new group for each unique combination of groups. The PowerCenter Server then performs the defined aggregation for each group

Using Sorted Input

You can improve Aggregator transformation performance by using the sorted input option. When you use sorted input, the PowerCenter Server assumes all data is sorted by group. As the PowerCenter Server reads rows for a group, it performs aggregate calculations. When necessary, it stores group information in memory. To use the Sorted Input option, you must pass sorted data to the Aggregator transformation. You can gain performance with sorted ports when you configure the session with multiple partitions.

When you do not use sorted input, the PowerCenter Server performs aggregate calculations as it reads. However, since data is not sorted, the PowerCenter Server stores data for each group until it reads the entire source to ensure all aggregate calculations are accurate.

Aggregator Transformation Tips

You can use the following guidelines to optimize the performance of an Aggregator transformation.

Use sorted input to decrease the use of aggregate caches.

Sorted input reduces the amount of data cached during the session and improves session performance. Use this option with the Sorter transformation to pass sorted data to the Aggregator transformation.

Limit connected input/output or output ports.

Limit the number of connected input/output or output ports to reduce the amount of data the Aggregator transformation stores in the data cache.

Filter before aggregating.

If you use a Filter transformation in the mapping, place the transformation before the Aggregator transformation to reduce unnecessary aggregation.

10.Update Strategy Transformation active/connected

When you design your data warehouse, you need to decide what type of information to store in targets. As part of your target table design, you need to determine whether to maintain all the historic data or just the most recent changes.

For example, you might have a target table, T_CUSTOMERS, that contains customer data. When a customer address changes, you may want to save the original address in the table instead of updating that portion of the customer row. In this case, you would create a new row containing the updated address, and preserve the original row with the old customer address. This illustrates how you might store historical information in a target table. However, if you want the T_CUSTOMERS table to be a snapshot of current customer data, you would update the existing customer row and lose the original address.

The model you choose determines how you handle changes to existing rows. In PowerCenter, you set your update strategy at two different levels:

- **Within a session.** When you configure a session, you can instruct the PowerCenter Server to either treat all rows in the same way (for example, treat all rows as inserts), or use instructions coded into the session mapping to flag rows for different database operations.
- **Within a mapping.** Within a mapping, you use the Update Strategy transformation to flag rows for insert, delete, update, or reject

Setting the Update Strategy

Use the following steps to define an update strategy:

1. To control how rows are flagged for insert, update, delete, or reject within a mapping, add an Update Strategy transformation to the mapping. Update Strategy transformations are essential if you want to flag rows destined for the same target for different database operations, or if you want to reject rows.

2. Define how to flag rows when you configure a session. You can flag all rows for insert, delete, or update, or you can select the data driven option, where the PowerCenter Server follows instructions coded into Update Strategy transformations within the session mapping.

3. Define insert, update, and delete options for each target when you configure a session. On a target-by-target basis, you can allow or disallow inserts and deletes, and you can choose three different ways to handle updates

- For the greatest degree of control over your update strategy, you add Update Strategy transformations to a mapping. The most important feature of this transformation is its update strategy expression, used to flag individual rows for insert, delete, update, or reject.

Operation	Constant	Numeric Value
Insert	DD_INSERT	0
Update	DD_UPDATE	1
Delete	DD_DELETE	2
Reject	DD_REJECT	3

-

Forwarding Rejected Rows

You can configure the Update Strategy transformation to either pass rejected rows to the next transformation or drop them. By default, the PowerCenter Server forwards rejected rows to the next transformation. The PowerCenter Server flags the rows for reject and writes them to the session reject file. If you do not select Forward Rejected Rows, the PowerCenter Server drops rejected rows and writes them to the session log file

Update Strategy Expressions

Frequently, the update strategy expression uses the IIF or DECODE function from the transformation language to test each row to see if it meets a particular condition. If it does, you can then assign each row a numeric code to flag it for a particular database operation. For example, the following IIF statement flags a row for reject if the entry date is after the apply date. Otherwise, it flags the row for update:

IIF((ENTRY_DATE > APPLY_DATE), DD_REJECT, DD_UPDATE)

Setting	Use To
Insert	Populate the target tables for the first time, or maintain a historical data warehouse. In the latter case, you must set this strategy for the entire data warehouse, not just a select group of target tables.
Delete	Clear target tables.
Update	Update target tables. You might choose this setting whether your data warehouse contains historical data or a snapshot. Later, when you configure how to update individual target tables, you can determine whether to insert updated rows as new rows or use the updated information to modify existing rows in the target.
Data Driven	Exert finer control over how you flag rows for insert, delete, update, or reject. Choose this setting if rows destined for the same table need to be flagged on occasion for one operation (for example, update), or for a different operation (for example, reject). In addition, this setting provides the only way you can flag rows for reject.

Specifying Operations for Individual Target Tables

Once you determine how to treat all rows in the session, you also need to set update strategy options for individual targets. Define the update strategy options in the Transformations view on Mapping tab of the session properties.

You can set the following update strategy options:

- **Insert.** Select this option to insert a row into a target table.
- **Delete.** Select this option to delete a row from a table.
- **Update.** You have the following options in this situation:
    - **Update as update.** Update each row flagged for update if it exists in the target table.
    - **Update as insert.** Inset each row flagged for update.
    - **Update else Insert.** Update the row if it exists. Otherwise, insert it.
- **Truncate table.** Select this option to truncate the target table before loading data.

Update Strategy Checklist

Choosing an update strategy requires setting the right options within a session and possibly adding Update Strategy transformations to a mapping. This section summarizes what you need to implement different versions of an update strategy.

Only perform inserts into a target table.

When you configure the session, select Insert for the Treat Source Rows As session property. Also, make sure that you select the Insert option for all target instances in the session.

Delete all rows in a target table.

When you configure the session, select Delete for the Treat Source Rows As session property. Also, make sure that you select the Delete option for all target instances in the session.

Only perform updates on the contents of a target table.

When you configure the session, select Update for the Treat Source Rows As session property. When you configure the update options for each target table instance, make sure you select the Update option for each target instance.

Perform different database operations with different rows destined for the same target table.

Add an Update Strategy transformation to the mapping. When you write the transformation update strategy expression, use either the DECODE or IIF function to flag rows for different operations (insert, delete, update, or reject). When you configure a session that uses this mapping, select Data Driven for the Treat Source Rows As session property. Make sure that you select the Insert, Delete, or one of the Update options for each target table instance.

Reject data.

Add an Update Strategy transformation to the mapping. When you write the transformation update strategy expression, use DECODE or IIF to specify the criteria for rejecting the row. When you configure a session that uses this mapping, select Data Driven for the Treat Source Rows As session property.

11.Lookup Transformation Passive connected/unconnected

Use a Lookup transformation in a mapping to look up data in a flat file or a relational table, view, or synonym. You can import a lookup definition from any flat file or relational database to which both the PowerCenter Client and Server can connect. You can use multiple Lookup transformations in a mapping.

The PowerCenter Server queries the lookup source based on the lookup ports in the transformation. It compares Lookup transformation port values to lookup source column values based on the lookup condition. Pass the result of the lookup to other transformations and a target.

You can use the Lookup transformation to perform many tasks, including:

- **Get a related value.** For example, your source includes employee ID, but you want to include the employee name in your target table to make your summary data easier to read.
- **Perform a calculation.** Many normalized tables include values used in a calculation, such as gross sales per invoice or sales tax, but not the calculated value (such as net sales).
- **Update slowly changing dimension tables.** You can use a Lookup transformation to determine whether rows already exist in the target.

You can configure the Lookup transformation to perform the following types of lookups:

- **Connected or unconnected.** Connected and unconnected transformations receive input and send output in different ways.
- **Relational or flat file lookup.** When you create a Lookup transformation, you can choose to perform a lookup on a flat file or a relational table.

  When you create a Lookup transformation using a relational table as the lookup source, you can connect to the lookup source using ODBC and import the table definition as the structure for the Lookup transformation.

  When you create a Lookup transformation using a flat file as a lookup source, the Designer invokes the Flat File Wizard. For more information about using the Flat File Wizard, see "Working with Flat Files" in the *Designer Guide*.

- **Cached or uncached.** Sometimes you can improve session performance by caching the lookup table. If you cache the lookup, you can choose to use a dynamic or static cache. By default, the lookup cache remains static and does not change during the session. With a dynamic cache, the PowerCenter Server inserts or updates rows in the cache during the session. When you cache the target table as the lookup, you can look up values in the target and insert them if they do not exist, or update them if they do.

  Note: If you use a flat file lookup, you must use a static cache.

Connected and Unconnected Lookups

You can configure a connected Lookup transformation to receive input directly from the mapping pipeline, or you can configure an unconnected Lookup transformation to receive input from the result of an expression in another transformation.

. Differences Between Connected and Unconnected Lookups

Connected Lookup	Unconnected Lookup
Receives input values directly from the pipeline.	Receives input values from the result of a :LKP expression in another transformation.
You can use a dynamic or static cache.	You can use a static cache.
Cache includes all lookup columns used in the mapping (that is, lookup source columns included in the lookup condition and lookup source columns linked as output ports to other transformations).	Cache includes all lookup/output ports in the lookup condition and the lookup/return port.
Can return multiple columns from the same row or insert into the dynamic lookup cache.	Designate one return port (R). Returns one column from each row.

If there is no match for the lookup condition, the PowerCenter Server returns the default value for all output ports. If you configure dynamic caching, the PowerCenter Server inserts rows into the cache or leaves it unchanged.	If there is no match for the lookup condition, the PowerCenter Server returns NULL.
If there is a match for the lookup condition, the PowerCenter Server returns the result of the lookup condition for all lookup/output ports. If you configure dynamic caching, the PowerCenter Server either updates the row the in the cache or leaves the row unchanged.	If there is a match for the lookup condition, the PowerCenter Server returns the result of the lookup condition into the return port.
Pass multiple output values to another transformation. Link lookup/output ports to another transformation.	Pass one output value to another transformation. The lookup/output/return port passes the value to the transformation calling :LKP expression.
Supports user-defined default values.	Does not support user-defined default values.

Relational and Flat File Lookups

When you create a Lookup transformation, you can choose to use a relational table or a flat file for the lookup source.

Relational Lookups

When you create a Lookup transformation using a relational table as a lookup source, you can connect to the lookup source using ODBC and import the table definition as the structure for the Lookup transformation.

You can use the following options with relational lookups only:

- You can override the default SQL statement if you want to add a WHERE clause or query multiple tables.
- You can use a dynamic lookup cache with relational lookups.

Flat File Lookups

When you use a flat file for a lookup source, you can use any flat file definition in the repository, or you can import it. When you import a flat file lookup source, the Designer invokes the Flat File Wizard.

You can use the following options with flat file lookups only:

- You can use indirect files as lookup sources by specifying a file list as the lookup file name.
- You can use sorted input for the lookup.
- You can sort null data high or low. With relational lookups, this is based on the database support.
- You can use case-sensitive string comparison with flat file lookups. With relational lookups, the case-sensitive comparison is based on the database support.

Using Sorted Input

When you configure a flat file Lookup transformation for sorted input, the condition columns must be grouped. If the condition columns are not grouped, the PowerCenter Server cannot cache the lookup and fails the session. For best caching performance, sort the condition columns.

The Lookup transformation also enables an associated ports property that you configure when you use a dynamic cache.

. Session Properties for Flat File Lookups

Property	Description
Lookup Source File Directory	Enter the directory name. By default, the PowerCenter Server looks in the server variable directory, $PMLookupFileDir, for lookup files. You can enter the full path and file name. If you specify both the directory and file name in the Lookup Source Filename field, clear this field. The PowerCenter Server concatenates this field with the Lookup Source Filename field when it runs the session. You can also use the $InputFile*Name* session parameter to specify the file name.
Lookup Source Filename	The name of the lookup file. If you use an indirect file, specify the name of the indirect file you want the PowerCenter Server to read. You can also use the lookup file parameter, $LookupFile*Name*, to change the name of the lookup file a session uses. If you specify both the directory and file name in the Source File Directory field, clear this field. The PowerCenter Server concatenates this field with the Lookup Source File Directory field when it runs the session. For example, if you have “C:\lookup_data\” in the Lookup Source File Directory field, then enter “filename.txt” in the Lookup Source Filename field. When the PowerCenter Server begins the session, it looks for “C:\lookup_data\filename.txt”.
Lookup Source Filetype	Indicates whether the lookup source file contains the source data or a list of files with the same file properties. Choose Direct if the lookup source file contains the source data. Choose Indirect if the lookup source file contains a list of files. When you select Indirect, the PowerCenter Server creates one cache for all files. If you use sorted input with indirect files, verify that the range of data in the files do not overlap. If the range of data overlaps, the PowerCenter Server processes the lookup as if you did not configure for sorted input.

Configuring Relational Lookups in a Session

When you configure a session, you specify the connection for the lookup database in the Connection node on the Mapping tab (Transformation view). You have the following options to specify a connection:

- Choose any relational connection.

- Use the connection variable, $DBConnection.
- Specify a database connection for $Source or $Target information.

If you use $Source or $Target for the lookup connection, configure the $Source Connection Value and $Target Connection Value in the session properties. This ensures that the PowerCenter Server uses the correct database connection for the variable when it runs the session.

If you use $Source or $Target and you do not specify a Connection Value in the session properties, the PowerCenter Server determines the database connection to use when it runs the session. It uses a source or target database connection for the source or target in the pipeline that contains the Lookup transformation. If it cannot determine which database connection to use, it fails the session.

Lookup Condition

The PowerCenter Server uses the lookup condition to test incoming values. It is similar to the WHERE clause in an SQL query. When you configure a lookup condition for the transformation, you compare transformation input values with values in the lookup source or cache, represented by lookup ports. When you run a workflow, the PowerCenter Server queries the lookup source or cache for all incoming values based on the condition.

You must enter a lookup condition in all Lookup transformations. Some guidelines for the lookup condition apply for all Lookup transformations, and some guidelines vary depending on how you configure the transformation.

Use the following guidelines when you enter a condition for a Lookup transformation:

- The datatypes in a condition must match.
- Use one input port for each lookup port used in the condition. You can use the same input port in more than one condition in a transformation.
- When you enter multiple conditions, the PowerCenter Server evaluates each condition as an AND, not an OR. The PowerCenter Server returns only rows that match all the conditions you specify.
- The PowerCenter Server matches null values. For example, if an input lookup condition column is NULL, the PowerCenter Server evaluates the NULL equal to a NULL in the lookup.
- If you configure a flat file lookup for sorted input, the PowerCenter Server fails the session if the condition columns are not grouped. If the columns are grouped, but not sorted, the PowerCenter Server processes the lookup as if you did not configure sorted input. For more information about sorted input.

The lookup condition guidelines and the way the PowerCenter Server processes matches can vary, depending on whether you configure the transformation for a dynamic cache or an uncached or static cache. For more information about lookup caches.

Lookup Caches

You can configure a Lookup transformation to cache the lookup file or table. The PowerCenter Server builds a cache in memory when it processes the first row of data in a cached Lookup transformation. It allocates memory for the cache based on the amount you configure in the transformation or session properties. The PowerCenter Server stores condition values in the index cache and output values in the data cache. The PowerCenter Server queries the cache for each row that enters the transformation.

The PowerCenter Server also creates cache files by default in the $PMCacheDir. If the data does not fit in the memory cache, the PowerCenter Server stores the overflow values in the cache files. When the session completes, the PowerCenter Server releases cache memory and deletes the cache files unless you configure the Lookup transformation to use a persistent cache.

When configuring a lookup cache, you can specify any of the following options:

- Persistent cache
- Recache from lookup source
- Static cache
- Dynamic cache
- Shared cache

Note: You can use a dynamic cache for relational lookups only.

Configuring Unconnected Lookup Transformations

An unconnected Lookup transformation is separate from the pipeline in the mapping. You write an expression using the :LKP reference qualifier to call the lookup within another transformation. Some common uses for unconnected lookups include:

- Testing the results of a lookup in an expression
- Filtering rows based on the lookup results
- Marking rows for update based on the result of a lookup, such as updating slowly changing dimension tables
- Calling the same lookup multiple times in one mapping

Complete the following steps when you configure an unconnected Lookup transformation:

1. Add input ports.
2. Add the lookup condition.
3. Designate a return value.
4. Call the lookup from another transformation.

Call the Lookup Through an Expression

You supply input values for an unconnected Lookup transformation from a :LKP expression in another transformation. The arguments are local input ports that match the Lookup transformation input ports used in the lookup condition. Use the following syntax for a :LKP expression:

:LKP.lookup_transformation_name(*argument, argument, ...*)

To continue the example about the retail store, when you write the update strategy expression, the order of ports in the expression must match the order in the lookup condition. In this case, the ITEM_ID condition is the first lookup condition, and therefore, it is the first argument in the update strategy expression.

IIF(ISNULL(:LKP.lkpITEMS_DIM(ITEM_ID, PRICE)), DD_UPDATE, DD_REJECT)

Lookup Caches Overview

You can configure a Lookup transformation to cache the lookup table. The PowerCenter Server builds a cache in memory when it processes the first row of data in a cached Lookup transformation. It allocates memory for the cache based on the amount you configure in the transformation or session properties. The PowerCenter Server stores condition values in the index cache and output values in the data cache. The PowerCenter Server queries the cache for each row that enters the transformation.

The PowerCenter Server also creates cache files by default in the $PMCacheDir. If the data does not fit in the memory cache, the PowerCenter Server stores the overflow values in the cache files. When the session completes, the PowerCenter Server releases cache memory and deletes the cache files unless you configure the Lookup transformation to use a persistent cache.

If you use a flat file lookup, the PowerCenter Server always caches the lookup source. If you configure a flat file lookup for sorted input, the PowerCenter Server cannot cache the lookup if the condition columns are not grouped. If the columns are grouped, but not sorted, the PowerCenter Server processes the lookup as if you did not configure sorted input. For more information about sorted input.

When configuring a lookup cache, you can specify any of the following options:

- **Persistent cache.** You can save the lookup cache files and reuse them the next time the PowerCenter Server processes a Lookup transformation configured to use the cache. **Recache from source.** If the persistent cache is not synchronized with the lookup table, you can configure the Lookup transformation to rebuild the lookup cache. For more information.
- **Static cache.** You can configure a static, or read-only, cache for any lookup source. By default, the PowerCenter Server creates a static cache. It caches the lookup file or table and looks up values in the cache for each row that comes into the transformation. When the lookup condition is true, the PowerCenter Server returns a value from the lookup cache. The PowerCenter Server does not update the cache while it processes the Lookup transformation.

- **dynamic cache.** If you want to cache the target table and insert new rows or update existing rows in the cache and the target, you can create a Lookup transformation to use a dynamic cache. The PowerCenter Server dynamically inserts or updates data in the lookup cache and passes data to the target table. You cannot use a dynamic cache with a flat file lookup.
- **Shared cache.** You can share the lookup cache between multiple transformations. You can share an unnamed cache between transformations in the same mapping. You can share a named cache between transformations in the same or different mappings. For more information..

When you do not configure the Lookup transformation for caching, the PowerCenter Server queries the lookup table for each input row. The result of the Lookup query and processing is the same, whether or not you cache the lookup table. However, using a lookup cache can increase session performance. Optimize performance by caching the lookup table when the source table is large.

Cache Comparison

The differences between an uncached lookup, a static cache, and a dynamic cache:

Table 9-1. Lookup Caching Comparison

Uncached	Static Cache	Dynamic Cache
You cannot insert or update the cache.	You cannot insert or update the cache.	You can insert or update rows in the cache as you pass rows to the target.
You cannot use a flat file lookup.	You can use a relational or a flat file lookup.	You can use a relational lookup only.
When the condition is true, the PowerCenter Server returns a value from the lookup table or cache. When the condition is not true, the PowerCenter Server returns the default value for connected transformations and NULL for unconnected transformations. .	When the condition is true, the PowerCenter Server returns a value from the lookup table or cache. When the condition is not true, the PowerCenter Server returns the default value for connected transformations and NULL for unconnected transformations. .	When the condition is true, the PowerCenter Server either updates rows in the cache or leaves the cache unchanged, depending on the row type. This indicates that the row is in the cache and target table. You can pass updated rows to the target table. When the condition is not true, the PowerCenter Server either inserts rows into the cache or leaves the cache unchanged, depending on the row type. This indicates that the row is not in the cache or target table. You can pass inserted rows to the target table. .

Note: The PowerCenter Server uses the same transformation logic to process a Lookup transformation whether you configure it to use a static cache or no cache. However, when you configure the transformation to use no cache, the PowerCenter Server queries the lookup table instead of the lookup cache.

Working with an Uncached Lookup or Static Cache

By default, the PowerCenter Server creates a static lookup cache when you configure a Lookup transformation for caching. The PowerCenter Server builds the cache when it processes the first lookup request. It queries the cache based on the lookup condition for each row that passes into the transformation. The PowerCenter Server does not update the cache while it processes the transformation. The PowerCenter Server processes an uncached lookup the same way it processes a cached lookup except that it queries the lookup source instead of building and querying the cache.

Working with a Dynamic Lookup Cache

For relational lookups, you might want to configure the transformation to use a dynamic cache when the target table is also the lookup table. The PowerCenter Server builds the cache when it processes the first lookup request. It queries the cache based on the lookup condition for each row that passes into the transformation. When you use a dynamic cache, the PowerCenter Server updates the lookup cache as it passes rows to the target.

When the PowerCenter Server reads a row from the source, it updates the lookup cache by performing one of the following actions:

- **Inserts the row into the cache.** The row is not in the cache and you specified to insert rows into the cache. You can configure the transformation to insert rows into the cache based on input ports or generated sequence IDs. The PowerCenter Server flags the row as insert.
- **Updates the row in the cache.** The row exists in the cache and you specified to update rows in the cache. The PowerCenter Server flags the row as update. The PowerCenter Server updates the row in the cache based on the input ports.
- **Makes no change to the cache.** The row exists in the cache and you specified to insert new rows only. Or, the row is not in the cache and you specified to update existing rows only. Or, the row is in the cache, but based on the lookup condition, nothing changes. The PowerCenter Server flags the row as unchanged.

Lookup Cache Tips

Use the following tips when you configure the Lookup transformation to cache the lookup table:

Cache small lookup tables.

Improve session performance by caching small lookup tables. The result of the lookup query and processing is the same, whether or not you cache the lookup table.

Use a persistent lookup cache for static lookup tables.

If the lookup table does not change between sessions, configure the Lookup transformation to use a persistent lookup cache. The PowerCenter Server then saves and reuses cache files from session to session, eliminating the time required to read the lookup table.

12.Source Qualifier Transformation active/connected

When you add a relational or a flat file source definition to a mapping, you need to connect it to a Source Qualifier transformation. The Source Qualifier transformation represents the rows that the PowerCenter Server reads when it runs a session.

You can use the Source Qualifier transformation to perform the following tasks:

- **Join data originating from the same source database.** You can join two or more tables with primary key-foreign key relationships by linking the sources to one Source Qualifier transformation.
- **Filter rows when the PowerCenter Server reads source data.** If you include a filter condition, the PowerCenter Server adds a WHERE clause to the default query.
- **Specify an outer join rather than the default inner join.** If you include a user-defined join, the PowerCenter Server replaces the join information specified by the metadata in the SQL query.
- **Specify sorted ports.** If you specify a number for sorted ports, the PowerCenter Server adds an ORDER BY clause to the default SQL query.
- **Select only distinct values from the source.** If you choose Select Distinct, the PowerCenter Server adds a SELECT DISTINCT statement to the default SQL query.
- **Create a custom query to issue a special SELECT statement for the PowerCenter Server to read source data.** For example, you might use a custom query to perform aggregate calculations.

Parameters and Variables

You can use mapping parameters and variables in the SQL query, user-defined join, and source filter of a Source Qualifier transformation. You can also use the system variable $$$SessStartTime.

The PowerCenter Server first generates an SQL query and replaces each mapping parameter or variable with its start value. Then it runs the query on the source database.

When you use a string mapping parameter or variable in the Source Qualifier transformation, use a string identifier appropriate to the source system. Most databases use a single quotation mark as a string identifier. For example, to use the string parameter $$IPAddress in a source filter for a Microsoft SQL Server database table, enclose the parameter in single quotes as follows, '$$IPAddress'.

- For relational sources, the PowerCenter Server generates a query for each Source Qualifier transformation when it runs a session. The default query is a SELECT statement for each source column used in the mapping. In other words, the PowerCenter Server reads only the columns that are connected to another transformation

Overriding the Default Query

You can alter or override the default query in the Source Qualifier transformation by changing the default settings of the transformation properties. Do not change the list of selected ports or the order in which they appear in the query. This list must match the connected transformation output ports.

When you edit transformation properties, the Source Qualifier transformation includes these settings in the default query. However, if you enter an SQL query, the PowerCenter Server uses only the defined SQL statement. The SQL Query overrides the User-Defined Join, Source Filter, Number of Sorted Ports, and Select Distinct settings in the Source Qualifier transformation.

Note: When you override the default SQL query, you must enclose all database reserved words in quotes.

Joining Source Data

You can use one Source Qualifier transformation to join data from multiple relational tables. These tables must be accessible from the same instance or database server.

When a mapping uses related relational sources, you can join both sources in one Source Qualifier transformation. During the session, the source database performs the join before passing data to the PowerCenter Server. This can increase performance when source tables are indexed.

Tip: Use the Joiner transformation for heterogeneous sources and to join flat files.

Custom Joins

If you need to override the default join, you can enter contents of the WHERE clause that specifies the join in the custom query.

You might need to override the default join under the following circumstances:

- Columns do not have a primary key-foreign key relationship.
- The datatypes of columns used for the join do not match.
- You want to specify a different type of join, such as an outer join.

Heterogeneous Joins

To perform a heterogeneous join, use the Joiner transformation. Use the Joiner transformation when you need to join the following types of sources:

- Join data from different source databases
- Join data from different flat file systems
- Join relational sources and flat files

Adding an SQL Query

The Source Qualifier transformation provides the SQL Query option to override the default query. You can enter an SQL statement supported by your source database. Before entering the query, connect all the input and output ports you want to use in the mapping.

When you edit the SQL Query, you can generate and edit the default query. When the Designer generates the default query, it incorporates all other configured options, such as a filter or number of sorted ports. The resulting query overrides all other options you might subsequently configure in the transformation.

You can include mapping parameters and variables in the SQL Query. When including a string mapping parameter or variable, use a string identifier appropriate to the source system. For most databases, you should enclose the name of a string parameter or variable in single quotes.

You can use the Source Qualifier and the Application Source Qualifier transformations to perform an outer join of two sources in the same database. When the PowerCenter Server performs an outer join, it returns all rows from one source table and rows from the second source table that match the join condition.

Locations for Entering Outer Join Syntax

Transformation	Transformation Setting	Description
Source Qualifier transformation	User-Defined Join	Create a join override. During the session, the PowerCenter Server appends the join override to the WHERE clause of the default query.
	SQL Query	Enter join syntax immediately after the WHERE in the default query.
Application Source Qualifier transformation	Join Override	Create a join override. During the session, the PowerCenter Server appends the join override to the WHERE clause of the default query.
	Extract Override	Enter join syntax immediately after the WHERE in the default query.

You can combine left outer and right outer joins with normal joins in a single source qualifier. You can use multiple normal joins and multiple left outer joins.

You can enter a source filter to reduce the number of rows the PowerCenter Server queries. If you include the string 'WHERE' or large objects in the source filter, the PowerCenter Server fails the session

When you use sorted ports, the PowerCenter Server adds the ports to the ORDER BY clause in the default query. The PowerCenter Server adds the configured number of ports, starting at the top of the Source Qualifier transformation. You might use sorted ports to improve performance when you include any of the following transformations in a mapping:

If you want the PowerCenter Server to select unique values from a source, you can use the Select Distinct option. You might use this feature to extract unique customer IDs from a table listing total sales. Using Select Distinct filters out unnecessary data earlier in the data flow, which might improve performance.

By default, the Designer generates a SELECT statement. If you choose Select Distinct, the Source Qualifier transformation includes the setting in the default SQL query.

You can add pre- and post-session SQL commands on the Properties tab in the Source Qualifier transformation. You might want to use pre-session SQL to write a timestamp row to the source table when a session begins.

The PowerCenter Server runs pre-session SQL commands against the source database before it reads the source. It runs post-session SQL commands against the source database after it writes to the target.

13 Stored Procedure Transformation Passive connected/unconnected

A Stored Procedure transformation is an important tool for populating and maintaining databases. Database administrators create stored procedures to automate tasks that are too complicated for standard SQL statements.

A stored procedure is a precompiled collection of Transact-SQL, PL-SQL or other database procedural statements and optional flow control statements, similar to an executable script. Stored procedures are stored and run within the database. You can run a stored procedure with the EXECUTE SQL statement in a database client tool, just as you can run SQL statements. Unlike standard SQL, however, stored procedures allow user-defined variables, conditional statements, and other powerful programming features.

One of the most useful features of stored procedures is the ability to send data to the stored procedure, and receive data from the stored procedure.

Input/Output Parameters

For many stored procedures, you provide a value and receive a value in return. These values are known as input and output parameters. For example, a sales tax calculation stored procedure can take a single input parameter, such as the price of an item. After performing the calculation, the stored procedure returns two output parameters, the amount of tax, and the total cost of the item including the tax.

The Stored Procedure transformation sends and receives input and output parameters using ports, variables, or by entering a value in an expression

The Stored Procedure transformation captures return values in a similar manner as input/output parameters, depending on the method that the input/output parameters are captured. In some instances, only a parameter or a return value can be captured.

Connected and Unconnected

Stored procedures run in either connected or unconnected mode. The mode you use depends on what the stored procedure does and how you plan to use it in your session. You can configure connected and unconnected Stored Procedure transformations in a mapping.

- **Connected.** The flow of data through a mapping in connected mode also passes through the Stored Procedure transformation. All data entering the transformation through the input ports affects the stored procedure. You should use a connected Stored Procedure transformation when you need data from an input port sent as an input parameter to the stored procedure, or the results of a stored procedure sent as an output parameter to another transformation.
- **Unconnected.** The unconnected Stored Procedure transformation is not connected directly to the flow of the mapping. It either runs before or after the session, or is called by an expression in another transformation in the mapping.

Comparison of Connected and Unconnected Stored Procedure Transformations

If you want to	Use this mode
Run a stored procedure before or after your session.	Unconnected
Run a stored procedure once during your mapping, such as pre- or post-session.	Unconnected
Run a stored procedure every time a row passes through the Stored Procedure transformation.	Connected or Unconnected
Run a stored procedure based on data that passes through the mapping, such as when a specific port does not contain a null value.	Unconnected
Pass parameters to the stored procedure and receive a single output parameter.	Connected or Unconnected
Pass parameters to the stored procedure and receive multiple output parameters. Note: To get multiple output parameters from an unconnected Stored Procedure	Connected or Unconnected

transformation, you must create variables for each output parameter. For details..	
Run nested stored procedures.	Unconnected
Call multiple times within a mapping.	Unconnected

To use a Stored Procedure transformation:

1. Create the stored procedure in the database.

   Before using the Designer to create the transformation, you must create the stored procedure in the database. You should also test the stored procedure through the provided database client tools.

2. Import or create the Stored Procedure transformation.

   Use the Designer to import or create the Stored Procedure transformation, providing ports for any necessary input/output and return values.

3. Determine whether to use the transformation as connected or unconnected.

   You must determine how the stored procedure relates to the mapping before configuring the transformation.

4. If connected, map the appropriate input and output ports.

   You use connected Stored Procedure transformations just as you would most other transformations. Click and drag the appropriate input flow ports to the transformation, and create mappings from output ports to other transformations.

5. If unconnected, either configure the stored procedure to run pre- or post-session, or configure it to run from an expression in another transformation.

   Since stored procedures can run before or after the session, you may need to specify when the unconnected transformation should run. On the other hand, if the stored procedure is called from another transformation, you write the expression in another transformation that calls the stored procedure. The expression can contain variables, and may or may not include a return value.

6. Configure the session.

   The session properties in the Workflow Manager includes options for error handling when running stored procedures and several SQL override options.

Oracle

In Oracle, any stored procedure that returns a value is called a stored function. Rather than using the CREATE PROCEDURE statement to make a new stored procedure based on the example, you use the CREATE FUNCTION statement. In this sample, the variables are declared as IN and OUT, but Oracle also supports an INOUT parameter type, which allows you to pass in a parameter, modify it, and return the modified value:

Configuring an Unconnected Transformation

An unconnected Stored Procedure transformation is not directly connected to the flow of data through the mapping. Instead, the stored procedure runs either:

- **From an expression.** Called from an expression written in the Expression Editor within another transformation in the mapping.
- **Pre- or post-session.** Runs before or after a session.

When using an unconnected Stored Procedure transformation in an expression, you need a method of returning the value of output parameters to a port. Use one of the following methods to capture the output values:

- Assign the output value to a local variable.
- Assign the output value to the system variable PROC_RESULT.

By using PROC_RESULT, you assign the value of the return parameter directly to an output port, which can apply directly to a target. You can also combine the two options by assigning one output parameter as PROC_RESULT, and the other parameter as a variable.

Use PROC_RESULT only within an expression. If you do not use PROC_RESULT or a variable, the port containing the expression captures a NULL. You cannot use PROC_RESULT in a connected Lookup transformation or within the Call Text for a Stored Procedure transformation.

If you require nested stored procedures, where the output parameter of one stored procedure passes to another stored procedure, use PROC_RESULT to pass the value.

The PowerCenter Server calls the unconnected Stored Procedure transformation from the Expression transformation. Notice that the Stored Procedure transformation has two input ports and one output port. All three ports are string datatypes.

Unconnected Stored Procedure transformations can be called from an expression in another transformation. Use the following rules and guidelines when configuring the expression:

- A single output parameter is returned using the variable PROC_RESULT.
- When you use a stored procedure in an expression, use the :SP reference qualifier. To avoid typing errors, select the Stored Procedure node in the Expression Editor, and double-click the name of the stored procedure.

- However, the same instance of a Stored Procedure transformation cannot run in both connected and unconnected mode in a mapping. You must create different instances of the transformation.
- The input/output parameters in the expression must match the input/output ports in the Stored Procedure transformation. If the stored procedure has an input parameter, there must also be an input port in the Stored Procedure transformation.
- When you write an expression that includes a stored procedure, list the parameters in the same order that they appear in the stored procedure and the Stored Procedure transformation.
- The parameters in the expression must include all of the parameters in the Stored Procedure transformation. You cannot leave out an input parameter. If necessary, pass a dummy variable to the stored procedure.
- The arguments in the expression must be the same datatype and precision as those in the Stored Procedure transformation.
- Use PROC_RESULT to apply the output parameter of a stored procedure expression directly to a target. You cannot use a variable for the output parameter to pass the results directly to a target. Use a local variable to pass the results to an output port within the same transformation.
- Nested stored procedures allow passing the return value of one stored procedure as the input parameter of another stored procedure

14.Union Transformation connected/active

The Union transformation is a multiple input group transformation that you can use to merge data from multiple pipelines or pipeline branches into one pipeline branch. It merges data from multiple sources similar to the UNION ALL SQL statement to combine the results from two or more SQL statements. Similar to the UNION ALL statement, the Union transformation does not remove duplicate rows.

You can connect heterogeneous sources to a Union transformation. The Union transformation merges sources with matching ports and outputs the data from one output group with the same ports as the input groups

A Union transformation has multiple input groups and one output group. Create input groups on the Groups tab, and create ports on the Group Ports tab.

You can create one or more input groups on the Groups tab. The Designer creates one output group by default. You cannot edit or delete the output group.

Using a Union Transformation in Mappings

The Union transformation is a non-blocking multiple input group transformation. You can connect the input groups to different branches in a single pipeline or to different source pipelines.

When you add a Union transformation to a mapping, you must verify that you connect the same ports in all input groups. If you connect all ports in one input group, but do not connect a port in another input group, the PowerCenter Server passes NULLs to the unconnected port.

15.Custom Transformation (active/passive) /connected

Custom transformations operate in conjunction with procedures you create outside of the Designer interface to extend PowerCenter functionality. You can create a Custom transformation and bind it to a procedure that you develop using the functions You can use the Custom transformation to create transformation applications, such as sorting and aggregation, which require all input rows to be processed before outputting any output rows. To support this process, the input and output functions occur separately in Custom transformations compared to External Procedure transformations.

The PowerCenter Server passes the input data to the procedure using an input function. The output function is a separate function that you must enter in the procedure code to pass output data to the PowerCenter Server. In contrast, in the External Procedure transformation, an external procedure function does both input and output, and its parameters consist of all the ports of the transformation.

Code Page Compatibility

The Custom transformation procedure code page is the code page of the data the Custom transformation procedure processes. The following factors determine the Custom transformation procedure code page:

- PowerCenter Server data movement mode
- The INFA_CTChangeStringMode() function
- The INFA_CTSetDataCodePageID() function

The Custom transformation procedure code page must be two-way compatible with the PowerCenter Server code page. The PowerCenter Server passes data to the procedure in the Custom transformation procedure code page. Also, the data the procedure passes to the PowerCenter Server must be valid characters in the Custom transformation procedure code page.

A Custom transformation has both input and output groups. It also can have input ports, output ports, and input/output ports. You create and edit groups and ports on the Ports tab of the Custom transformation. You can also define the relationship between input and output ports on the Ports tab.

Creating Groups and Ports

You can create multiple input groups and multiple output groups in a Custom transformation. You must create at least one input group and one output group. To create an input group, click the Create Input Group icon. To create an output group, click the Create Output Group icon. When you create a group, the Designer adds it as the last group. When you create a passive Custom transformation, you can only create one input group and one output group.

To create a port, click the Add button. When you create a port, the Designer adds it below the currently selected row or group. Each port contains attributes defined on the Port Attribute Definitions tab. You can edit the attributes for each port. For more information about creating and editing user-defined port attributes

By default, an output port in a Custom transformation depends on all input ports. However, you can define the relationship between input and output ports in a Custom transformation. When you do this, you can view link paths in a mapping containing a Custom transformation and you can see which input ports an output port depends on. You can also view source column dependencies for target ports in a mapping containing a Custom transformation.

To define the relationship between ports in a Custom transformation, create a port dependency. A port dependency is the relationship between an output or input/output port and one or more input or input/output ports. When you create a port dependency, base it on the procedure logic in the code.

Custom Transformation Properties

Option	Description
Module Identifier	The module name. Enter only ASCII characters in this field. You cannot enter multibyte characters. This property is the base name of the DLL or the shared library that contains the procedure. The Designer uses this name to create the C file when you generate the external procedure code.
Function Identifier	The name of the procedure in the module. Enter only ASCII characters in this field. You cannot enter multibyte characters. The Designer uses this name to create the C file where you enter the procedure code.
Runtime Location	The location that contains the DLL or shared library. The default is $PMExtProcDir. Enter a path relative to the PowerCenter Server machine that runs the session using the Custom transformation. If you make this property blank, the PowerCenter Server uses the environment variable defined on the PowerCenter Server machine to locate the DLL or shared library. You must copy all DLLs or shared libraries to the runtime location or to the environment variable defined on the PowerCenter Server machine. The PowerCenter Server fails to load the procedure when it cannot locate the DLL, shared library, or a referenced file.
Tracing Level	Amount of detail displayed in the session log for this transformation. The default is Normal.
Is Partitionable	Specifies whether or not you can create multiple partitions in a pipeline that uses this transformation. This property is disabled by default.
Inputs Must Block	Specifies whether or not the procedure associated with the transformation must be able to block incoming data. This property is enabled by default. For more information about blocking data..
Is Active	Specifies whether this transformation is an active or passive transformation.

	You cannot change this property after you create the Custom transformation. If you need to change this property, create a new Custom transformation and select the correct property value.
Update Strategy Transformation	Specifies whether or not this transformation defines the update strategy for output rows. This property is disabled by default. You can enable this for active Custom transformations. For more information about this property..
Transformation Scope	Specifies how the PowerCenter Server applies the transformation logic to incoming data: • Row • Transaction • All Input When the transformation is passive, this property is always Row. When the transformation is active, this property is All Input by default. For more information about working with transaction control.
Generate Transaction	Specifies whether or not this transformation can generate transactions. When a Custom transformation generates transactions, it does so for all output groups. This property is disabled by default. You can only enable this for active Custom transformations. For more information about working with transaction control.
Output is Repeatable	Specifies whether the order of the output data is consistent between session runs. • Never. The order of the output data is inconsistent between session runs. This is the default for active transformations. • Based On Input Order. The output order is consistent between session runs when the input data order is consistent between session runs. This is the default for passive transformations. • Always. The order of the output data is consistent between session runs even if the order of the input data is inconsistent between session runs.

By default, the PowerCenter Server concurrently reads sources in a target load order group. However, you can write the external procedure code to block input data on some input groups. Blocking is the suspension of the data flow into an input group of a multiple input group transformation. For more information about blocking source data, To use a Custom transformation to block input data, you must write the procedure code to block and unblock data. You must also enable blocking on the Properties tab for the Custom transformation.

16.XML Source Qualifierransformation Active/connected

You can add an XML Source Qualifier transformation to a mapping by dragging an XML source definition to the Mapping Designer workspace or by manually creating one. When you add an XML source definition to a mapping, you need to connect it to an XML Source Qualifier transformation. The XML Source Qualifier transformation defines the data elements that the PowerCenter Server reads when it executes a session. It determines how the PowerCenter reads the source data.

An XML Source Qualifier transformation always has one input or output port for every column in the XML source. When you create an XML Source Qualifier transformation for a source definition, the Designer links each port in the XML source definition to a port in the XML Source Qualifier transformation. You cannot remove or edit any of the links. If you remove an XML source definition from a mapping, the Designer also removes the corresponding XML Source Qualifier transformation. You can link one XML source definition to one XML Source Qualifier transformation

17 Normalizer Transformation active/connected

Normalization is the process of organizing data. In database terms, this includes creating normalized tables and establishing relationships between those tables according to rules designed to both protect the data and make the database more flexible by eliminating redundancy and inconsistent dependencies.

The Normalizer transformation normalizes records from COBOL and relational sources, allowing you to organize the data according to your own needs. A Normalizer transformation can appear anywhere in a pipeline when you normalize a relational source. Use a Normalizer transformation instead of the Source Qualifier transformation when you normalize a COBOL source. When you drag a COBOL source into the Mapping Designer workspace, the Mapping Designer creates a Normalizer transformation with input and output ports for every column in the source.

You primarily use the Normalizer transformation with COBOL sources, which are often stored in a denormalized format. The OCCURS statement in a COBOL file nests multiple records of information in a single record. Using the Normalizer transformation, you break out repeated data within a record into separate records. For each new record it creates, the Normalizer transformation generates a unique identifier. You can use this key value to join the normalized records.

You can also use the Normalizer transformation with relational sources to create multiple rows from a single row of data.

Performance Tuning

The goal of performance tuning is to optimize session performance by eliminating performance bottlenecks. To tune the performance of a session, first you identify a performance bottleneck, eliminate it, and then identify the next performance bottleneck until you are satisfied with the session performance. You can use the test load option to run sessions when you tune session performance.

The most common performance bottleneck occurs when the PowerCenter Server writes to a target database. You can identify performance bottlenecks by the following methods:

- **Running test sessions.** You can configure a test session to read from a flat file source or to write to a flat file target to identify source and target bottlenecks.
- **Studying performance details.** You can create a set of information called performance details to identify session bottlenecks. Performance details provide information such as buffer input and output efficiency **Monitoring system performance.** You can use system monitoring tools to view percent CPU usage, I/O waits, and paging to identify system bottlenecks.

Once you determine the location of a performance bottleneck, you can eliminate the bottleneck by following these guidelines:

- **Eliminate source and target database bottlenecks.** Have the database administrator optimize database performance by optimizing the query, increasing the database network packet size, or configuring index and key constraints.
- **Eliminate mapping bottlenecks.** Fine tune the pipeline logic and transformation settings and options in mappings to eliminate mapping bottlenecks.
- **Eliminate session bottlenecks.** You can optimize the session strategy and use performance details to help tune session configuration.
- **Eliminate system bottlenecks.** Have the system administrator analyze information from system monitoring tools and improve CPU and network performance.

If you tune all the bottlenecks above, you can further optimize session performance by increasing the number of pipeline partitions in the session. Adding partitions can improve performance by utilizing more of the system hardware while processing the session.

Because determining the best way to improve performance can be complex, change only one variable at a time, and time the session both before and after the change. If session performance does not improve, you might want to return to your original configurations.

Identifying the Performance Bottleneck

The first step in performance tuning is to identify the performance bottleneck. Performance bottlenecks can occur in the source and target databases, the mapping, the session, and the system. Generally, you should look for performance bottlenecks in the following order:

1. Target
2. Source
3. Mapping
4. Session
5. System

You can identify performance bottlenecks by running test sessions, viewing performance details, and using system monitoring tools.

Identifying Target Bottlenecks

The most common performance bottleneck occurs when the PowerCenter Server writes to a target database. You can identify target bottlenecks by configuring the session to write to a flat file target. If the session performance increases significantly when you write to a flat file, you have a target bottleneck.

If your session already writes to a flat file target, you probably do not have a target bottleneck. You can optimize session performance by writing to a flat file target local to the PowerCenter Server.

Causes for a target bottleneck may include small check point intervals, small database network packet size, or problems during heavy loading operations. For details about eliminating a target bottleneck.

Identifying Source Bottlenecks

Performance bottlenecks can occur when the PowerCenter Server reads from a source database. If your session reads from a flat file source, you probably do not have a source bottleneck. You can improve session performance by setting the number of bytes the PowerCenter Server reads per line if you read from a flat file source.

If the session reads from relational source, you can use a filter transformation, a read test mapping, or a database query to identify source bottlenecks.

Using a Filter Transformation

You can use a filter transformation in the mapping to measure the time it takes to read source data.

Add a filter transformation in the mapping after each source qualifier. Set the filter condition to false so that no data is processed past the filter transformation. If the time it takes to run the new session remains about the same, then you have a source bottleneck.

Using a Read Test Session

You can create a read test mapping to identify source bottlenecks. A read test mapping isolates the read query by removing the transformation in the mapping. Use the following steps to create a read test mapping:

1. Make a copy of the original mapping.
2. In the copied mapping, keep only the sources, source qualifiers, and any custom joins or queries.
3. Remove all transformations.
4. Connect the source qualifiers to a file target.

Use the read test mapping in a test session. If the test session performance is similar to the original session, you have a source bottleneck.

Using a Database Query

You can identify source bottlenecks by executing the read query directly against the source database.

Copy the read query directly from the session log. Execute the query against the source database with a query tool such as isql. On Windows, you can load the result of the query in a file. On UNIX systems, you can load the result of the query in /dev/null.

Measure the query execution time and the time it takes for the query to return the first row. If there is a long delay between the two time measurements, you can use an optimizer hint to eliminate the source bottleneck.

Causes for a source bottleneck may include an inefficient query or small database network packet sizes. For details about eliminating source bottlenecks.

Identifying Mapping Bottlenecks

If you determine that you do not have a source or target bottleneck, you might have a mapping bottleneck. You can identify mapping bottlenecks by using a Filter transformation in the mapping.

If you determine that you do not have a source bottleneck, you can add a Filter transformation in the mapping before each target definition. Set the filter condition to false so that no data is loaded into the target tables. If the time it takes to run the new session is the same as the original session, you have a mapping bottleneck.

You can also identify mapping bottlenecks by using performance details. High errorrows and rowsinlookupcache counters indicate a mapping bottleneck.

High Rowsinlookupcache Counters

Multiple lookups can slow down the session. You might improve session performance by locating the largest lookup tables and tuning those lookup expressions.

High Errorrows Counters

Transformation errors impact session performance. If a session has large numbers in any of the *Transformation*_errorrows counters, you might improve performance by eliminating the errors..

Identifying a Session Bottleneck

If you do not have a source, target, or mapping bottleneck, you may have a session bottleneck. You can identify a session bottleneck by using the performance details. The PowerCenter Server creates performance details when you enable Collect Performance Data in the Performance settings on the Properties tab of the session properties.

Performance details display information about each Source Qualifier, target definition, and individual transformation. All transformations have some basic counters that indicate the number of input rows, output rows, and error rows.

Any value other than zero in the readfromdisk and writetodisk counters for Aggregator, Joiner, or Rank transformations indicate a session bottleneck.

Small cache size, low buffer memory, and small commit intervals can cause session bottlenecks.

Aggregator, Rank, and Joiner Readfromdisk and Writetodisk Counters

If a session contains Aggregator, Rank, or Joiner transformations, examine each *Transformation*_readfromdisk and *Transformation*_writetodisk counter.

If these counters display any number other than zero, you can improve session performance by increasing the index and data cache sizes. The PowerCenter Server uses the index cache to store group information and the data cache to store transformed data, which is typically larger. Therefore, although both the index cache and data cache sizes affect performance, you will most likely need to increase the data cache size more than the index cache size. If the session performs incremental aggregation, the PowerCenter Server reads historical aggregate data from the local disk during the session and writes to disk when saving historical data. As a result, the Aggregator_readtodisk and writetodisk counters display a number besides zero. However, since the PowerCenter Server writes the historical data to a file at the end of the session, you can still evaluate the counters during the session. If the counters show any number other than zero during the session run, you can increase performance by tuning the index and data cache sizes.

To view the session performance details while the session runs, right-click the session in the Workflow Monitor and choose Properties. Click the Properties tab in the details dialog box.

Source and Target BufferInput_efficiency and BufferOutput_efficiency Counters

If the BufferInput_efficiency and the BufferOutput_efficiency counters are low for all sources and targets, increasing the session DTM buffer size may improve performance. For information on when and how to tune this parameter.

Under certain circumstances, tuning the buffer block size may also improve session performance.

Identifying a System Bottleneck

After you tune the source, target, mapping, and session, you may consider tuning the system. You can identify system bottlenecks by using system tools to monitor CPU usage, memory usage, and paging.

The PowerCenter Server uses system resources to process transformation, session execution, and reading and writing data. The PowerCenter Server also uses system memory for other data such as aggregate, joiner, rank, and cached lookup tables. You can use system performance monitoring tools to monitor the amount of system resources the PowerCenter Server uses and identify system bottlenecks.

On Windows, you can use system tools in the Task Manager or Administrative Tools.

On UNIX systems you can use system tools such as vmstat and iostat to monitor system performance.

On Windows, you can view the Performance and Processes tab in the Task Manager (use Ctrl-Alt-Del and choose Task Manager). The Performance tab in the Task Manager provides a quick look at CPU usage and total memory used. You can view more detailed performance information by using the Performance Monitor on Windows (use Start-Programs-Administrative Tools and choose Performance Monitor).

Use the Windows Performance Monitor to create a chart that provides the following information:

- **Percent processor time**. If you have several CPUs, monitor each CPU for percent processor time. If the processors are utilized at more than 80%, you may consider adding more processors.
- **Pages/second**. If pages/second is greater than five, you may have excessive memory pressure (thrashing). You may consider adding more physical memory.
- **Physical disks percent time**. This is the percent time that the physical disk is busy performing read or write requests. You may consider adding another disk device or upgrading the disk device.
- **Physical disks queue length**. This is the number of users waiting for access to the same disk device. If physical disk queue length is greater than two, you may consider adding another disk device or upgrading the disk device.
- **Server total bytes per second**. This is the number of bytes the server has sent to and received from the network. You can use this information to improve network bandwidth.

Identifying System Bottlenecks on UNIX

You can use UNIX tools to monitor user background process, system swapping actions, CPU loading process, and I/O load operations. When you tune UNIX systems, tune the server for a major database system. Use the following UNIX tools to identify system bottlenecks on the UNIX system:

- **lsattr -E -I sys0**. Use this tool to view current system settings. This tool shows maxuproc, the maximum level of user background processes. You may consider reducing the amount of background process on your system.
- **iostat**. Use this tool to monitor loading operation for every disk attached to the database server. Iostat displays the percentage of time that the disk was physically active. High disk utilization suggests that you may need to add more disks.

If you use disk arrays, use utilities provided with the disk arrays instead of iostat.

- **vmstat** or **sar -w**. Use this tool to monitor disk swapping actions. Swapping should not occur during the session. If swapping does occur, you may consider increasing your physical memory or reduce the number of memory-intensive applications on the disk.
- **sar -u**. Use this tool to monitor CPU loading. This tool provides percent usage on user, system, idle time, and waiting time. If the percent time spent waiting on I/O (%wio) is high, you may consider using other under-utilized disks. For example, if your source data, target data, lookup, rank, and aggregate cache files are all on the same disk, consider putting them on different disks.

Optimizing the Target Database

If your session writes to a flat file target, you can optimize session performance by writing to a flat file target that is local to the PowerCenter Server. If your session writes to a relational target, consider performing the following tasks to increase performance:

- Drop indexes and key constraints.
- Increase checkpoint intervals.
- Use bulk loading.
- Use external loading.
- Increase database network packet size.
- Optimize Oracle target databases.

Dropping Indexes and Key Constraints

When you define key constraints or indexes in target tables, you slow the loading of data to those tables. To improve performance, drop indexes and key constraints before running your session. You can rebuild those indexes and key constraints after the session completes.

If you decide to drop and rebuild indexes and key constraints on a regular basis, you can create pre- and post-load stored procedures to perform these operations each time you run the session.

Note: To optimize performance, use constraint-based loading only if necessary.

Increasing Checkpoint Intervals

The PowerCenter Server performance slows each time it waits for the database to perform a checkpoint. To increase performance, consider increasing the database checkpoint interval. When you increase the database checkpoint

interval, you increase the likelihood that the database performs checkpoints as necessary, when the size of the database log file reaches its limit.

For details on specific database checkpoints, checkpoint intervals, and log files, consult your database documentation.

Bulk Loading

You can use bulk loading to improve the performance of a session that inserts a large amount of data to a DB2, Sybase, Oracle, or Microsoft SQL Server database. Configure bulk loading on the Mapping tab.

When bulk loading, the PowerCenter Server bypasses the database log, which speeds performance. Without writing to the database log, however, the target database cannot perform rollback. As a result, you may not be able to perform recovery. Therefore, you must weigh the importance of improved session performance against the ability to recover an incomplete session.

External Loading

You can use the External Loader session option to integrate external loading with a session.

If you have a DB2 EE or DB2 EEE target database, you can use the DB2 EE or DB2 EEE external loaders to bulk load target files. The DB2 EE external loader uses the PowerCenter Server db2load utility to load data. The DB2 EEE external loader uses the DB2 Autoloader utility.

If you have a Teradata target database, you can use the Teradata external loader utility to bulk load target files.

If your target database runs on Oracle, you can use the Oracle SQL*Loader utility to bulk load target files. When you load data to an Oracle database using a pipeline with multiple partitions, you can increase performance if you create the Oracle target table with the same number of partitions you use for the pipeline.

If your target database runs on Sybase IQ, you can use the Sybase IQ external loader utility to bulk load target files. If your Sybase IQ database is local to the PowerCenter Server on your UNIX system, you can increase performance by loading data to target tables directly from named pipes.

Increasing Database Network Packet Size

You can increase the network packet size in the Informatica Workflow Manager to reduce target bottleneck. For Sybase and Microsoft SQL Server, increase the network packet size to 8K - 16K. For Oracle, increase the network packet size in tnsnames.ora and listener.ora. If you increase the network packet size in the PowerCenter Server configuration, you also need to configure the database server network memory to accept larger packet sizes.

See your database documentation about optimizing database network packet size.

Optimizing Oracle Target Databases

If your target database is Oracle, you can optimize the target database by checking the storage clause, space allocation, and rollback segments.

When you write to an Oracle database, check the storage clause for database objects. Make sure that tables are using large initial and next values. The database should also store table and index data in separate tablespaces, preferably on different disks.

When you write to Oracle target databases, the database uses rollback segments during loads. Make sure that the database stores rollback segments in appropriate tablespaces, preferably on different disks. The rollback segments should also have appropriate storage clauses.

You can optimize the Oracle target database by tuning the Oracle redo log. The Oracle database uses the redo log to log loading operations. Make sure that redo log size and buffer size are optimal. You can view redo log properties in the init.ora file.

If your Oracle instance is local to the PowerCenter Server, you can optimize performance by using IPC protocol to connect to the Oracle database. You can set up Oracle database connection in listener.ora and tnsnames.ora.

Optimizing the Source Database

If your session reads from a flat file source, you can improve session performance by setting the number of bytes the PowerCenter Server reads per line. By default, the PowerCenter Server reads 1024 bytes per line. If each line in the source file is less than the default setting, you can decrease the Line Sequential Buffer Length setting in the session properties.

If your session reads from a relational source, review the following suggestions for improving performance:

- Optimize the query.
- Create tempdb as in-memory database.
- Use conditional filters.
- Increase database network packet size.
- Connect to Oracle databases using IPC protocol.

Optimizing the Query

If a session joins multiple source tables in one Source Qualifier, you might be able to improve performance by optimizing the query with optimizing hints. Also, single table select statements with an ORDER BY or GROUP BY clause may benefit from optimization such as adding indexes.

Usually, the database optimizer determines the most efficient way to process the source data. However, you might know properties about your source tables that the database optimizer does not. The database administrator can create optimizer hints to tell the database how to execute the query for a particular set of source tables.

The query the PowerCenter Server uses to read data appears in the session log. You can also find the query in the Source Qualifier transformation. Have your database administrator analyze the query, and then create optimizer hints and/or indexes for the source tables.

Use optimizing hints if there is a long delay between when the query begins executing and when PowerCenter receives the first row of data. Configure optimizer hints to begin returning rows as quickly as possible, rather than returning all rows at once. This allows the PowerCenter Server to process rows parallel with the query execution.

Queries that contain ORDER BY or GROUP BY clauses may benefit from creating an index on the ORDER BY or GROUP BY columns. Once you optimize the query, use the SQL override option to take full advantage of these modifications. For details on using SQL override, see "Source Qualifier Transformation" in the *Transformation Guide*.

You can also configure the source database to run parallel queries to improve performance. See your database documentation for configuring parallel query.

Using tempdb to Join Sybase and Microsoft SQL Server Tables

When joining large tables on a Sybase or Microsoft SQL Server database, you might improve performance by creating the tempdb as an in-memory database to allocate sufficient memory. Check your Sybase or Microsoft SQL Server manual for details.

Using Conditional Filters

A simple source filter on the source database can sometimes impact performance negatively because of lack of indexes. You can use the PowerCenter conditional filter in the Source Qualifier to improve performance.

Whether you should use the PowerCenter conditional filter to improve performance depends on your session. For example, if multiple sessions read from the same source simultaneously, the PowerCenter conditional filter may improve performance.

However, some sessions may perform faster if you filter the source data on the source database. You can test your session with both the database filter and the PowerCenter filter to determine which method improves performance.

Increasing Database Network Packet Sizes

You can improve the performance of a source database by increasing the network packet size, allowing larger packets of data to cross the network at one time. To do this you must complete the following tasks:

- Increase the database server network packet size.
- Change the packet size in the Workflow Manager database connection to reflect the database server packet size.

For Oracle, increase the packet size in listener.ora and tnsnames.ora. For other databases, check your database documentation for details on optimizing network packet size.

Connecting to Oracle Source Databases

If your Oracle instance is local to the PowerCenter Server, you can optimize performance by using IPC protocol to connect to the Oracle database. You can set up Oracle database connection in listener.ora and tnsnames.ora.

Optimizing the Mapping

Mapping-level optimization may take time to implement but can significantly boost session performance. Focus on mapping-level optimization only after optimizing on the target and source databases.

Generally, you reduce the number of transformations in the mapping and delete unnecessary links between transformations to optimize the mapping. You should configure the mapping with the least number of transformations and expressions to do the most amount of work possible. You should minimize the amount of data moved by deleting unnecessary links between transformations.

For transformations that use data cache (such as Aggregator, Joiner, Rank, and Lookup transformations), limit connected input/output or output ports. Limiting the number of connected input/output or output ports reduces the amount of data the transformations store in the data cache.

You can also perform the following tasks to optimize the mapping:

- Configure single-pass reading.
- Optimize datatype conversions.
- Eliminate transformation errors.
- Optimize transformations.
- Optimize expressions.

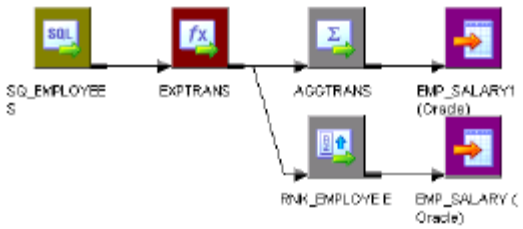
Configuring Single-Pass Reading

Single-pass reading allows you to populate multiple targets with one source qualifier. Consider using single-pass reading if you have several sessions that use the same sources. If you join the separate mappings and use only one source qualifier for each source, the PowerCenter Server then reads each source only once, then sends the data into separate data flows. A particular row can be used by all the data flows, by any combination, or by none, as the situation demands.

For example, you have the PURCHASING source table, and you use that source daily to perform an aggregation and a ranking. If you place the Aggregator and Rank transformations in separate mappings and sessions, you force the PowerCenter Server to read the same source table twice. However, if you join the two mappings, using one source qualifier, the PowerCenter Server reads PURCHASING only once, then sends the appropriate data to the two separate data flows.

When changing mappings to take advantage of single-pass reading, you can optimize this feature by factoring out any functions you do on both mappings. For example, if you need to subtract a percentage from the PRICE ports for both the Aggregator and Rank transformations, you can minimize work by subtracting the percentage *before* splitting the pipeline as shown in Figure 25-1:

Figure 25-1. Single-Pass Reading



Optimizing Datatype Conversions

Forcing the PowerCenter Server to make unnecessary datatype conversions slows performance. For example, if your mapping moves data from an Integer column to a Decimal column, then back to an Integer column, the unnecessary datatype conversion slows performance. Where possible, eliminate unnecessary datatype conversions from mappings.

Some datatype conversions can improve system performance. Use integer values in place of other datatypes when performing comparisons using Lookup and Filter transformations.

For example, many databases store U.S. zip code information as a Char or Varchar datatype. If you convert your zip code data to an Integer datatype, the lookup database stores the zip code 94303-1234 as 943031234. This helps increase the speed of the lookup comparisons based on zip code.

Eliminating Transformation Errors

In large numbers, transformation errors slow the performance of the PowerCenter Server. With each transformation error, the PowerCenter Server pauses to determine the cause of the error and to remove the row causing the error from the data flow. Then the PowerCenter Server typically writes the row into the session log file.

Transformation errors occur when the PowerCenter Server encounters conversion errors, conflicting mapping logic, and any condition set up as an error, such as null input. Check the session log to see where the transformation errors occur. If the errors center around particular transformations, evaluate those transformation constraints.

If you need to run a session that generates a large numbers of transformation errors, you might improve performance by setting a lower tracing level. However, this is not a recommended long-term response to transformation errors.

Optimizing Lookup Transformations

If a mapping contains a Lookup transformation, you can optimize the lookup. Some of the things you can do to increase performance include caching the lookup table, optimizing the lookup condition, or indexing the lookup table.

Caching Lookups

If a mapping contains Lookup transformations, you might want to enable lookup caching. In general, you want to cache lookup tables that need less than 300MB.

When you enable caching, the PowerCenter Server caches the lookup table and queries the lookup cache during the session. When this option is not enabled, the PowerCenter Server queries the lookup table on a row-by-row basis. You can increase performance using a shared or persistent cache:

- **Shared cache.** You can share the lookup cache between multiple transformations. You can share an unnamed cache between transformations in the same mapping. You can share a named cache between transformations in the same or different mappings.
- **Persistent cache.** If you want to save and reuse the cache files, you can configure the transformation to use a persistent cache. Use this feature when you know the lookup table does not change between session runs. Using a persistent cache can improve performance because the PowerCenter Server builds the memory cache from the cache files instead of from the database.

Reducing the Number of Cached Rows

Use the Lookup SQL Override option to add a WHERE clause to the default SQL statement. This allows you to reduce the number of rows included in the cache.

Optimizing the Lookup Condition

If you include more than one lookup condition, place the conditions with an equal sign first to optimize lookup performance.

Indexing the Lookup Table

The PowerCenter Server needs to query, sort, and compare values in the lookup condition columns. The index needs to include every column used in a lookup condition. You can improve performance for both cached and uncached lookups:

- **Cached lookups.** You can improve performance by indexing the columns in the lookup ORDER BY. The session log contains the ORDER BY statement.
- **Uncached lookups.** Because the PowerCenter Server issues a SELECT statement for each row passing into the Lookup transformation, you can improve performance by indexing the columns in the lookup condition.

Optimizing Multiple Lookups

If a mapping contains multiple lookups, even with caching enabled and enough heap memory, the lookups can slow performance. By locating the Lookup transformations that query the largest amounts of data, you can tune those lookups to improve overall performance.

To see which Lookup transformations process the most data, examine the Lookup_rowsinlookupcache counters for each Lookup transformation. The Lookup transformations that have a large number in this counter might benefit from tuning their lookup expressions. If those expressions can be optimized, session performance improves.

Optimizing Filter Transformations

If you filter rows from the mapping, you can improve efficiency by filtering early in the data flow. Instead of using a Filter transformation halfway through the mapping to remove a sizable amount of data, use a source qualifier filter to remove those same rows at the source.

If you cannot move the filter into the source qualifier, move the Filter transformation as close to the source qualifier as possible to remove unnecessary data early in the data flow.

In your filter condition, avoid using complex expressions. You can optimize Filter transformations by using simple integer or true/false expressions in the filter condition.

Use a Filter or Router transformation to drop rejected rows from an Update Strategy transformation if you do not need to keep rejected rows.

Optimizing Aggregator Transformations

Aggregator transformations often slow performance because they must group data before processing it. Aggregator transformations need additional memory to hold intermediate group results. You can optimize Aggregator transformations by performing the following tasks:

- Group by simple columns.
- Use sorted input.
- Use incremental aggregation.

Group By Simple Columns

You can optimize Aggregator transformations when you group by simple columns. When possible, use numbers instead of string and dates in the columns used for the GROUP BY. You should also avoid complex expressions in the Aggregator expressions.

Use Sorted Input

You can increase session performance by sorting data and using the Aggregator Sorted Input option.

The Sorted Input decreases the use of aggregate caches. When you use the Sorted Input option, the PowerCenter Server assumes all data is sorted by group. As the PowerCenter Server reads rows for a group, it performs aggregate calculations. When necessary, it stores group information in memory.

The Sorted Input option reduces the amount of data cached during the session and improves performance. Use this option with the Source Qualifier Number of Sorted Ports option to pass sorted data to the Aggregator transformation.

You can benefit from better performance when you use the Sorted Input option in sessions with multiple partitions.

Use Incremental Aggregation

If you can capture changes from the source that changes less than half the target, you can use Incremental Aggregation to optimize the performance of Aggregator transformations.

When using incremental aggregation, you apply captured changes in the source to aggregate calculations in a session. The PowerCenter Server updates your target incrementally, rather than processing the entire source and recalculate the same calculations every time you run the session.

Optimizing Joiner Transformations

Joiner transformations can slow performance because they need additional space at run time to hold intermediate results. You can view Joiner performance counter information to determine whether you need to optimize the Joiner transformations.

Joiner transformations need a data cache to hold the master table rows and an index cache to hold the join columns from the master table. You need to make sure that you have enough memory to hold the data and the index cache so the system does not page to disk. To minimize memory requirements, you can also use the smaller table as the master table or join on as few columns as possible.

The type of join you use can affect performance. Normal joins are faster than outer joins and result in fewer rows. When possible, use database joins for homogenous sources.

Optimizing Sequence Generator Transformations

You can optimize Sequence Generator transformations by creating a reusable Sequence Generator and use it in multiple mappings simultaneously. You can also optimize Sequence Generator transformations by configuring the Number of Cached Values property.

The Number of Cached Values property determines the number of values the PowerCenter Server caches at one time. Make sure that the Number of Cached Value is not too small. You may consider configuring the Number of Cached Values to a value greater than 1,000.

Optimizing Expressions

As a final step in tuning the mapping, you can focus on the expressions used in transformations. When examining expressions, focus on complex expressions for possible simplification. Remove expressions one-by-one to isolate the slow expressions.

Once you locate the slowest expressions, take a closer look at how you can optimize those expressions.

Factoring Out Common Logic

If the mapping performs the same task in several places, reduce the number of times the mapping performs the task by moving the task earlier in the mapping. For example, you have a mapping with five target tables. Each target requires a Social Security number lookup. Instead of performing the lookup five times, place the Lookup transformation in the mapping before the data flow splits. Then pass lookup results to all five targets.

Minimizing Aggregate Function Calls

When writing expressions, factor out as many aggregate function calls as possible. Each time you use an aggregate function call, the PowerCenter Server must search and group the data. For example, in the following expression, the PowerCenter Server reads COLUMN_A, finds the sum, then reads COLUMN_B, finds the sum, and finally finds the sum of the two sums:

SUM(COLUMN_A) + SUM(COLUMN_B)

If you factor out the aggregate function call, as below, the PowerCenter Server adds COLUMN_A to COLUMN_B, then finds the sum of both.

SUM(COLUMN_A + COLUMN_B)

Replacing Common Sub-Expressions with Local Variables

If you use the same sub-expression several times in one transformation, you can make that sub-expression a local variable. You can use a local variable only within the transformation, but by calculating the variable only once, you can speed performance. For details, see “Transformations” in the *Designer Guide*.

Choosing Numeric versus String Operations

The PowerCenter Server processes numeric operations faster than string operations. For example, if you look up large amounts of data on two columns, EMPLOYEE_NAME and EMPLOYEE_ID, configuring the lookup around EMPLOYEE_ID improves performance.

Optimizing Char-Char and Char-Varchar Comparisons

When the PowerCenter Server performs comparisons between CHAR and VARCHAR columns, it slows each time it finds trailing blank spaces in the row. You can use the Treat CHAR as CHAR On Read option in the PowerCenter Server setup so that the PowerCenter Server does not trim trailing spaces from the end of Char source fields.

Choosing DECODE versus LOOKUP

When you use a LOOKUP function, the PowerCenter Server must look up a table in a database. When you use a DECODE function, you incorporate the lookup values into the expression itself, so the PowerCenter Server does not have to look up a separate table. Therefore, when you want to look up a small set of unchanging values, using DECODE may improve performance.

Using Operators Instead of Functions

The PowerCenter Server reads expressions written with operators faster than expressions with functions. Where possible, use operators to write your expressions. For example, if you have an expression that involves nested CONCAT calls such as:

CONCAT(CONCAT(CUSTOMERS.FIRST_NAME, ' ') CUSTOMERS.LAST_NAME)

you can rewrite that expression with the || operator as follows:

CUSTOMERS.FIRST_NAME || ' ' || CUSTOMERS.LAST_NAME

Optimizing IIF Expressions

IIF expressions can return a value as well as an action, which allows for more compact expressions. For example, say you have a source with three Y/N flags: FLG_A, FLG_B, FLG_C, and you want to return values such that: If FLG_A = “Y”, then return = VAL_A. If FLG_A = “Y” AND FLG_B = “Y”, then return = VAL_A + VAL_B, and so on for all the permutations.

One way to write the expression is as follows:

```
IIF( FLG_A = 'Y' and FLG_B = 'Y' AND FLG_C = 'Y',
 VAL_A + VAL_B + VAL_C,
  IIF( FLG_A = 'Y' and FLG_B = 'Y' AND FLG_C = 'N',
```

```
  VAL_A + VAL_B ,
   IIF( FLG_A = 'Y' and FLG_B = 'N' AND FLG_C = 'Y',
    VAL_A + VAL_C,
    IIF( FLG_A = 'Y' and FLG_B = 'N' AND FLG_C = 'N',
     VAL_A ,
     IIF( FLG_A = 'N' and FLG_B = 'Y' AND FLG_C = 'Y',
      VAL_B + VAL_C,
      IIF( FLG_A = 'N' and FLG_B = 'Y' AND FLG_C = 'N',
       VAL_B ,
       IIF( FLG_A = 'N' and FLG_B = 'N' AND FLG_C = 'Y',
        VAL_C,
        IIF( FLG_A = 'N' and FLG_B = 'N' AND FLG_C = 'N',
         0.0,
))))))))
```

This first expression requires 8 IIFs, 16 ANDs, and at least 24 comparisons.

But if you take advantage of the IIF function's ability to return a value, you can rewrite that expression as:

IIF(FLG_A='Y', VAL_A, 0.0)+ IIF(FLG_B='Y', VAL_B, 0.0)+ IIF(FLG_C='Y', VAL_C, 0.0)

This results in three IIFs, two comparisons, two additions, and a faster session.

Evaluating Expressions If you are not sure which expressions slow performance, the following steps can help isolate the problem.

To evaluate expression performance:

1. Time the session with the original expressions.
2. Copy the mapping and replace half of the complex expressions with a constant.
3. Run and time the edited session.
4. Make another copy of the mapping and replace the other half of the complex expressions with a constant.
5. Run and time the edited session.

Optimizing the Session

Once you optimize your source database, target database, and mapping, you can focus on optimizing the session. You can perform the following tasks to improve overall performance:

- Increase the number of partitions.
- Reduce errors tracing.
- Remove staging areas.
- Tune session parameters.

Table 25-1 lists the settings and values you can use to improve session performance:

Table 25-1. Session Tuning Parameters

Setting	Default Value	Suggested Minimum Value	Suggested Maximum Value
DTM Buffer Size	12,000,000 bytes	6,000,000 bytes	128,000,000 bytes
Buffer block size	64,000 bytes	4,000 bytes	128,000 bytes
Index cache size	1,000,000 bytes	1,000,000 bytes	12,000,000 bytes
Data cache size	2,000,000 bytes	2,000,000 bytes	24,000,000 bytes
Commit interval	10,000 rows	N/A	N/A
High Precision	Disabled	N/A	N/A
Tracing Level	Normal	Terse	N/A

Pipeline Partitioning

If you purchased the partitioning option, you can increase the number of partitions in a pipeline to improve session performance. Increasing the number of partitions allows the PowerCenter Server to create multiple connections to sources and process partitions of source data concurrently.

When you create a session, the Workflow Manager validates each pipeline in the mapping for partitioning. You can specify multiple partitions in a pipeline if the PowerCenter Server can maintain data consistency when it processes the partitioned data.

Allocating Buffer Memory

When the PowerCenter Server initializes a session, it allocates blocks of memory to hold source and target data. The PowerCenter Server allocates at least two blocks for each source and target partition. Sessions that use a large number of sources and targets might require additional memory blocks. If the PowerCenter Server cannot allocate enough memory blocks to hold the data, it fails the session.

By default, a session has enough buffer blocks for 83 sources and targets. If you run a session that has more than 83 sources and targets, you can increase the number of available memory blocks by adjusting the following session parameters:

- **DTM Buffer Size.** Increase the DTM buffer size found in the Performance settings of the Properties tab. The default setting is 12,000,000 bytes.
- **Default Buffer Block Size.** Decrease the buffer block size found in the Advanced settings of the Config Object tab. The default setting is 64,000 bytes.

To configure these settings, first determine the number of memory blocks the PowerCenter Server requires to initialize the session. Then, based on default settings, you can calculate the buffer size and/or the buffer block size to create the required number of session blocks.

If you have XML sources or targets in your mapping, use the number of groups in the XML source or target in your calculation for the total number of sources and targets.

For example, you create a session that contains a single partition using a mapping that contains 50 sources and 50 targets.

1. You determine that the session requires 200 memory blocks:

   [(total number of sources + total number of targets)* 2] = (session buffer blocks)

   100 * 2 = 200

2. Next, based on default settings, you determine that you can change the DTM Buffer Size to 15,000,000, or you can change the Default Buffer Block Size to 54,000:

   (session Buffer Blocks) = (.9) * (DTM Buffer Size) / (Default Buffer Block Size) * (number of partitions)

   200 = .9 * *14222222* / 64000 * 1

   or

   200 = .9 * 12000000 / *54000* * 1

Increasing DTM Buffer Size

The DTM Buffer Size setting specifies the amount of memory the PowerCenter Server uses as DTM buffer memory. The PowerCenter Server uses DTM buffer memory to create the internal data structures and buffer blocks used to bring data into and out of the PowerCenter Server. When you increase the DTM buffer memory, the PowerCenter Server creates more buffer blocks, which improves performance during momentary slowdowns.

Increasing DTM buffer memory allocation generally causes performance to improve initially and then level off. When you increase the DTM buffer memory allocation, consider the total memory available on the PowerCenter Server system.

If you do not see a significant increase in performance, DTM buffer memory allocation is not a factor in session performance.

Note: Reducing the DTM buffer allocation can cause the session to fail early in the process because the PowerCenter Server is unable to allocate memory to the required processes.

To increase DTM buffer size:

1. Go to the Performance settings of the Properties tab.

2. Increase the setting for DTM Buffer Size, and click OK.

The default for DTM Buffer Size is 12,000,000 bytes. Increase the setting by increments of multiples of the buffer block size, then run and time the session after each increase.

Optimizing the Buffer Block Size

Depending on the session source data, you might need to increase or decrease the buffer block size.

If the session mapping contains a large number of sources or targets, you might need to decrease the buffer block size.

If you are manipulating unusually large rows of data, you can increase the buffer block size to improve performance. If you do not know the approximate size of your rows, you can determine the configured row size by following the steps below.

To evaluate needed buffer block size:

1. In the Mapping Designer, open the mapping for the session.

2. Open the target instance.

3. Click the Ports tab.

4. Add the precisions for all the columns in the target.

5. If you have more than one target in the mapping, repeat steps 2-4 for each additional target to calculate the precision for each target.

6. Repeat steps 2-5 for each source definition in your mapping.

7. Choose the largest precision of all the source and target precisions for the total precision in your buffer block size calculation.

The total precision represents the total bytes needed to move the largest row of data. For example, if the total precision equals 33,000, then the PowerCenter Server requires 33,000 bytes in the buffers to move that row. If the buffer block size is 64,000 bytes, the PowerCenter Server can move only one row at a time.

Ideally, a buffer should accommodate at least 20 rows at a time. So if the total precision is greater than 32,000, increase the size of the buffers to improve performance.

To increase buffer block size:

1. Go to the Advanced settings on the Config Object tab.

2. Increase the setting for Default Buffer Block Size, and click OK.

The default for this setting is 64,000 bytes. Increase this setting in relation to the size of the rows. As with DTM buffer memory allocation, increasing buffer block size should improve performance. If you do not see an increase, buffer block size is not a factor in session performance.

Increasing the Cache Sizes

The PowerCenter Server uses the index and data caches for Aggregator, Rank, Lookup, and Joiner transformation. The PowerCenter Server stores transformed data from Aggregator, Rank, Lookup, and Joiner transformations in the data cache before returning it to the data flow. It stores group information for those transformations in the index cache. If the allocated data or index cache is not large enough to store the data, the PowerCenter Server stores the data in a temporary disk file as it processes the session data. Each time the PowerCenter Server pages to the temporary file, performance slows.

You can see when the PowerCenter Server pages to the temporary file by examining the performance details. The *Transformation*_readfromdisk or *Transformation*_writetodisk counters for any Aggregator, Rank, Lookup, or Joiner transformation indicate the number of times the PowerCenter Server must page to disk to process the transformation. Since the data cache is typically larger than the index cache, you should increase the data cache more than the index cache.

Increasing the Commit Interval

The Commit Interval setting determines the point at which the PowerCenter Server commits data to the target tables. Each time the PowerCenter Server commits, performance slows. Therefore, the smaller the commit interval, the more often the PowerCenter Server writes to the target database, and the slower the overall performance.

If you increase the commit interval, the number of times the PowerCenter Server commits decreases and performance improves.

When you increase the commit interval, consider the log file limits in the target database. If the commit interval is too high, the PowerCenter Server may fill the database log file and cause the session to fail.

Therefore, weigh the benefit of increasing the commit interval against the additional time you would spend recovering a failed session.

Click the General Options settings of the Properties tab to review and adjust the commit interval.

Disabling High Precision

If a session runs with high precision enabled, disabling high precision might improve session performance.

The Decimal datatype is a numeric datatype with a maximum precision of 28. To use a high precision Decimal datatype in a session, configure the PowerCenter Server to recognize this datatype by selecting Enable High Precision in the session properties. However, since reading and manipulating the high precision datatype slows the PowerCenter Server, you can improve session performance by disabling high precision.

When you disable high precision, the PowerCenter Server converts data to a double. The PowerCenter Server reads the Decimal row 3900058411382035317455530282 as 390005841138203 x 10^{13}.

Click the Performance settings on the Properties tab to enable high precision.

Reducing Error Tracing

If a session contains a large number of transformation errors that you have no time to correct, you can improve performance by reducing the amount of data the PowerCenter Server writes to the session log.

To reduce the amount of time spent writing to the session log file, set the tracing level to Terse. You specify Terse tracing if your sessions run without problems and you don't need session details. At this tracing level, the PowerCenter Server does not write error messages or row-level information for reject data.

To debug your mapping, set the tracing level to Verbose. However, it can significantly impact the session performance. Do not use Verbose tracing when you tune performance.

The session tracing level overrides any transformation-specific tracing levels within the mapping. This is not recommended as a long-term response to high levels of transformation errors.

Removing Staging Areas

When you use a staging area, the PowerCenter Server performs multiple passes on your data. Where possible, remove staging areas to improve performance. The PowerCenter Server can read multiple sources with a single pass, which may alleviate your need for staging areas. For details on single-pass reading.

Optimizing the System

Often performance slows because your session relies on inefficient connections or an overloaded PowerCenter Server system. System delays can also be caused by routers, switches, network protocols, and usage by many users. After you determine from the system monitoring tools that you have a system bottleneck, you can make the following global changes to improve the performance of all your sessions:

- **Improve network speed.** Slow network connections can slow session performance. Have your system administrator determine if your network runs at an optimal speed. Decrease the number of network hops between the PowerCenter Server and databases.
- **Use multiple PowerCenter Servers.** Using multiple PowerCenter Servers on separate systems might double or triple session performance.
- **Use a server grid.** Use a collection of PowerCenter Servers to distribute and process the workload of a workflow. For information on server grids.
- **Improve CPU performance.** Run the PowerCenter Server and related machines on high performance CPUs, or configure your system to use additional CPUs.
- **Configure the PowerCenter Server for ASCII data movement mode.** When all character data processed by the PowerCenter Server is 7-bit ASCII or EBCDIC, configure the PowerCenter Server for ASCII data movement mode.
- **Check hard disks on related machines.** Slow disk access on source and target databases, source and target file systems, as well as the PowerCenter Server and repository machines can slow session performance. Have your system administrator evaluate the hard disks on your machines.
- **Reduce paging.** When an operating system runs out of physical memory, it starts paging to disk to free physical memory. Configure the physical memory for the PowerCenter Server machine to minimize paging to disk.
- **Use processor binding.** In a multi-processor UNIX environment, the PowerCenter Server may use a large amount of system resources. Use processor binding to control processor usage by the PowerCenter Server.

Improving Network Speed

The performance of the PowerCenter Server is related to network connections. A local disk can move data five to twenty times faster than a network. Consider the following options to minimize network activity and to improve PowerCenter Server performance.

If you use flat file as a source or target in your session, you can move the files onto the PowerCenter Server system to improve performance. When you store flat files on a machine other than the PowerCenter Server, session performance becomes dependent on the performance of your network connections. Moving the files onto the PowerCenter Server system and adding disk space might improve performance.

If you use relational source or target databases, try to minimize the number of network hops between the source and target databases and the PowerCenter Server. Moving the target database onto a server system might improve PowerCenter Server performance.

When you run sessions that contain multiple partitions, have your network administrator analyze the network and make sure it has enough bandwidth to handle the data moving across the network from all partitions.

Using Multiple PowerCenter Servers

You can run multiple PowerCenter Servers on separate systems against the same repository. Distributing the session load to separate PowerCenter Server systems increases performance. For details on using multiple PowerCenter Servers.

Using Server Grids A server grid allows you to use the combined processing power of multiple PowerCenter Servers to balance the workload of workflows.

In a server grid, a PowerCenter Server distributes sessions across the network of available PowerCenter Servers. You can further improve performance by assigning a more powerful server to run a complicated mapping.

Running the PowerCenter Server in ASCII Data Movement Mode

When all character data processed by the PowerCenter Server is 7-bit ASCII or EBCDIC, configure the PowerCenter Server to run in the ASCII data movement mode. In ASCII mode, the PowerCenter Server uses one byte to store each character. When you run the PowerCenter Server in Unicode mode, it uses two bytes for each character, which can slow session performance.

Using Additional CPUs

Configure your system to use additional CPUs to improve performance. Additional CPUs allows the system to run multiple sessions in parallel as well as multiple pipeline partitions in parallel.

However, additional CPUs might cause disk bottlenecks. To prevent disk bottlenecks, minimize the number of processes accessing the disk. Processes that access the disk include database functions and operating system functions. Parallel sessions or pipeline partitions also require disk access.

Reducing Paging

Paging occurs when the PowerCenter Server operating system runs out of memory for a particular operation and uses the local disk for memory. You can free up more memory or increase physical memory to reduce paging and the slow performance that results from paging. Monitor paging activity using system tools.

You might want to increase system memory in the following circumstances:

- You run a session that uses large cached lookups.
- You run a session with many partitions.

If you cannot free up memory, you might want to add memory to the system.

Using Processor Binding

In a multi-processor UNIX environment, the PowerCenter Server may use a large amount of system resources if you run a large number of sessions. As a result, other applications on the machine may not have enough system resources available. You can use processor binding to control processor usage by the PowerCenter Server. In a Sun Solaris environment, the system administrator can create and manage a processor set using the psrset command. The system administrator can then use the pbind command to bind the PowerCenter Server to a processor set so the processor set only runs the PowerCenter Server. The Sun Solaris environment also provides the psrinfo command to display details about each configured processor, and the psradm command to change the operational status of processors. In an HP-UX environment, the system administrator can use the Process Resource Manager utility to control CPU usage in the system. The Process Resource Manager allocates minimum system resources and uses a maximum cap of resources.

Pipeline Partitioning

Once you have tuned the application, databases, and system for maximum single-partition performance, you may find that your system is under-utilized. At this point, you can reconfigure your session to have two or more partitions. Adding partitions may improve performance by utilizing more of the hardware while processing the session.

Use the following tips when you add partitions to a session:

- **Add one partition at a time.** To best monitor performance, add one partition at a time, and note your session settings before you add each partition.
- **Set DTM Buffer Memory.** For a session with *n* partitions, this value should be at least *n* times the value for the session with one partition.
- **Set cached values for Sequence Generator.** For a session with *n* partitions, there should be no need to use the "Number of Cached Values" property of the Sequence Generator transformation. If you must set this value to a value greater than zero, make sure it is at least *n* times the original value for the session with one partition.
- **Partition the source data evenly.** Configure each partition to extract the same number of rows.
- **Monitor the system while running the session.** If there are CPU cycles available (twenty percent or more idle time) then this session might see a performance improvement by adding a partition.
- **Monitor the system after adding a partition.** If the CPU utilization does not go up, the wait for I/O time goes up, or the total data transformation rate goes down, then there is probably a hardware or software bottleneck. If the wait for I/O time goes up a significant amount, then check the system for hardware bottlenecks. Otherwise, check the database configuration.
- **Tune databases and system.** Make sure that your databases are tuned properly for parallel ETL and that your system has no bottlenecks.

. Optimizing the Source Database for Partitioning

Usually, each partition on the reader side represents a subset of the data to be processed. But if the database is not tuned properly, the results may not make your session any quicker. This is fairly easy to test. Create a pipeline with one partition. Measure the reader throughput in the Workflow Manager. After you do this, add partitions. Is the throughput scaling linearly? In other words, if you have two partitions, is your reader throughput twice as fast? If this is not true, you probably need to tune your database.

Some databases may have specific options that must be set to enable parallel queries. You should check your individual database manual for these options. If these options are off, the PowerCenter Server runs multiple partition SELECT statements serially.

You can also consider adding partitions to increase the speed of your query. Each database provides an option to separate the data into different tablespaces. If your database allows it, you can use the SQL override feature to provide a query that extracts data from a single partition.

To maximize a single-sorted query on your database, you need to look at options that enable parallelization. There are many options in each database that may increase the speed of your query.

Here are some configuration options to look for in your source database:

- **Check for configuration parameters that perform automatic tuning.** For example, Oracle has a parameter called parallel_automatic_tuning.
- **Make sure intra-parallelism (the ability to run multiple threads on a single query) is enabled.** For example, on Oracle you should look at parallel_adaptive_multi_user. On DB2, you should look at intra_parallel.
- **Maximum number of parallel processes that are available for parallel executions.** For example, on Oracle, you should look at parallel_max_servers. On DB2, you should look at max_agents.
- **Size for various resources used in parallelization.** For example, Oracle has parameters such as large_pool_size, shared_pool_size, hash_area_size, parallel_execution_message_size, and optimizer_percent_parallel. DB2 has configuration parameters such as dft_fetch_size, fcm_num_buffers, and sort_heap.
- **Degrees of parallelism (may occur as either a database configuration parameter or an option on the table or query).** For example, Oracle has parameters parallel_threads_per_cpu and optimizer_percent_parallel. DB2 has configuration parameters such as dft_prefetch_size, dft_degree, and max_query_degree.
- **Turn off options that may affect your database scalability.** For example, disable archive logging and timed statistics on Oracle.

Note: The above examples are not a comprehensive list of all the tuning options available to you on the databases. Check your individual database documentation for all performance tuning configuration parameters available.

Optimizing the Target Database for Partitioning

If you have a mapping with multiple partitions, you want the throughput for each partition to be the same as the throughput for a single partition session. If you do not see this correlation, then your database is probably inserting rows into the database serially.

To make sure that your database inserts rows in parallel, check the following configuration options in your target database:

- **Look for a configuration option that needs to be set explicitly to enable parallel inserts.** For example, Oracle has db_writer_processes, and DB2 has max_agents (some databases may have this enabled by default).
- **Consider partitioning your target table.** If it is possible, try to have each partition write to a single database partition. You can use the Router transformation to do this. Also, look into having the database partitions on separate disks to prevent I/O contention among the pipeline partitions.
- **Turn off options that may affect your database scalability.** For example, disable archive logging and timed statistics on Oracle.

Session Recovery

If you stop a session or if an error causes a session to stop unexpectedly, refer to the session logs to determine the cause of the failure. Correct the errors, and then complete the session. The method you use to complete the session depends on the configuration of the mapping and the session, the specific failure, and how much progress the session made before it failed. If the PowerCenter Server did not commit any data, run the session again. If the session issued at least one commit and is recoverable, consider running the session in recovery mode.

Recovery allows you to restart a failed session and complete it as if the session had run without pause. When the PowerCenter Server runs in recovery mode, it continues to commit data from the point of the last successful commit. For more information on PowerCenter Server processing during recovery.

All recovery sessions run as part of a workflow. When you recover a session, you also have the option to run part of the workflow. Consider the configuration and design of the workflow and the status of other tasks in the workflow before you choose a method of recovery. Depending on the configuration and status of the workflow and session, you can choose one or more of the following recovery methods:

- **Recover a suspended workflow.** If the workflow suspends due to session failure, you can recover the failed session and resume the workflow**Recover a failed workflow.** If the workflow fails as a result of session failure, you can recover the session and run the rest of the workflow.
- **Recover a session task.** If the workflow completes, but a session fails, you can recover the session alone without running the rest of the workflow. You can also use this method to recover multiple failed sessions in a branched workflow.

Preparing for Recovery

Before you perform recovery, you must configure the mapping, session, workflow, and target database to ensure that the recovery session will consistently read, transform, and write data as though the session had not failed.

Under certain circumstances, you cannot recover the session and must run it again. For more information on completing unrecoverable sessions.

Configuring the Mapping

When you design a mapping, consider requirements for session recovery. Configure the mapping so that the PowerCenter Server can extract, transform, and load data with the same results each time it runs the session.

Use the following guidelines when you configure the mapping:

- **Sort the data from the source.** This guarantees that the PowerCenter Server always receives source rows in the same order. You can do this by configuring the Sorted Ports option in the Source Qualifier or Application Source Qualifier transformation or by adding a Sorter transformation configured for distinct output rows to the mapping after the source qualifier.
- **Verify all targets receive data from transformations that produce repeatable data.** Some transformations produce repeatable data. You can enable a session for recovery in the Workflow Manager when all targets in the mapping receive data from transformations that produce repeatable data. For more information on repeatable data.

Also, to perform consistent data recovery, the source, target, and transformation properties for the recovery session must be the same as those for the failed session. Do not change the properties of objects in the mapping before you run the recovery session.

Configuring the Session

To perform recovery on a failed session, the session must meet the following criteria:

- The session is enabled for recovery.
- The previous session run failed and the recovery information is accessible.

To enable recovery, select the Enable Recovery option in the Error Handling settings of the Configuration tab in the session properties.

If you enable recovery and also choose to truncate the target for a relational normal load session, the PowerCenter Server does not truncate the target when you run the session in recovery mode.

Use the following guidelines when you enable recovery for a partitioned session:

- The Workflow Manager configures all partition points to use the default partitioning scheme for each transformation when you enable recovery.
- The Workflow Manager sets the partition type to pass-through unless the transformation receiving the data is either an Aggregator transformation, a Rank transformation, or a sorted Joiner transformation.
- You can only enable recovery for unsorted Joiner transformations with one partition.

- For Custom transformations, you can enable recovery only for transformations with one input group.

The PowerCenter Server disables test load when you enable the session for recovery.

To perform consistent data recovery, the session properties for the recovery session must be the same as the session properties for the failed session. This includes the partitioning configuration and the session sort order.

Configuring the Workflow

The recovery method you choose for the workflow depends on the design and configuration of the workflow. As with sessions, you can configure a workflow so that you can correct errors and complete the workflow as though it ran without error.

If other tasks or workflows in your environment depend on the successful completion of a session, configure the workflow containing the session to suspend on error. This is useful for sequential and concurrent sessions because it prevents the PowerCenter Server from continuing the workflow after the session fails. This is also useful if multiple concurrent sessions fail or if other workflows depend on the successful completion of the workflowIf you do not want to configure the workflow to suspend on error, you can configure recoverable sessions to fail the workflow if the session fails. This prevents the PowerCenter Server from continuing to run the workflow after the session fails. In this case, you may want to perform recovery by running the part of the workflow that did not yet run. You can also allow the workflow to complete even if sessions or other tasks fail. You can then choose to recover only the failed session tasks. This allows you to recover the sessions without running previously successful tasks.

Configuring the Target Database

When the PowerCenter Server runs a session in recovery mode, it uses information in recovery tables that it creates on the target database system. The PowerCenter Server creates the recovery tables when it runs a session enabled for recovery. If the tables already exist, the PowerCenter Server writes information to them.

The PowerCenter Server creates the following recovery tables in the target database:

- **PM_RECOVERY.** This table records target load information during the session run. The PowerCenter Server removes the information from this table after each successful session and initializes the information at the beginning of subsequent sessions.
- **PM_TGT_RUN_ID.** This table records information the PowerCenter Server uses to identify each target on the database. The information remains in the table between session runs.

If you want the PowerCenter Server to create the recovery tables, you must grant table creation privileges to the database user name for the target database connection. If you do not want the PowerCenter Server to create the recovery tables, you must create the recovery tables manually.

Do not edit or drop the recovery tables while recovery is enabled. If you want to disable recovery, the PowerCenter Server does not remove the recovery tables from the target database. You must manually remove the recovery tables.

Table 11-1 describes the format of PM_RECOVERY:

Table 11-1. PM_RECOVERY Table Definition

Column Name	Datatype
REP_GID	VARCHAR(240)
WFLOW_ID	NUMBER
SUBJ_ID	NUMBER
TASK_INST_ID	NUMBER
TGT_INST_ID	NUMBER
PARTITION_ID	NUMBER
TGT_RUN_ID	NUMBER
RECOVERY_VER	NUMBER
CHECK_POINT	NUMBER
ROW_COUNT	NUMBER

Table 11-2 describes the format of PM_TGT_RUN_ID:

Table 11-2. PM_TGT_RUN_ID Table Definition

Column Name	Datatype
LAST_TGT_RUN_ID	NUMBER

Note: If you manually create the PM_TGT_RUN_ID table, you must specify a value other than zero in the LAST_TGT_RUN_ID column to ensure that the session runs successfully in recovery mode.

Creating pmcmd Scripts

You can use *pmcmd* to perform recovery from the command line or in a script. When you use *pmcmd* commands in a script, *pmcmd* indicates the success or failure of the command with a return code. The following return codes apply to recovery sessions.

Table 11-3 describes the return codes for *pmcmd* that relate to recovery:

Table 11-3. pmcmd Return Codes for Recovery

Code	Description
12	The PowerCenter Server cannot start recovery because the session or workflow is scheduled, suspending, waiting for an event, waiting, initializing, aborting, stopping, disabled, or running.
19	The PowerCenter Server cannot start the session in recovery mode because the workflow is configured to run continuously.

Working with Repeatable Data

You can enable a session for recovery in the Workflow Manager when all targets in the mapping receive data from transformations that produce repeatable data. All transformations have a property that determines when the transformation produces repeatable data. For most transformations, this property is hidden. However, you can write the Custom transformation procedure to output repeatable data, and then configure the Custom transformation Output Is Repeatable property to match the procedure behavior.

Transformations can produce repeatable data under the following circumstances:

- **Never.** The order of the output data is inconsistent between session runs. This is the default for active Custom transformations.
- **Based on input order.** The output order is consistent between session runs when the input data order for all input groups is consistent between session runs. This is the default for passive Custom transformations.
- **Always.** The order of the output data is consistent between session runs even if the order of the input data is inconsistent between session runs.
- **Based on transformation configuration.** The transformation produces repeatable data depending on how you configure the transformation. You can always enable the session for recovery, but you may get inconsistent results depending on how you configure the transformation.

Table 11-4 lists which transformations produce repeatable data:

Table 11-4. Transformations that Output Repeatable Data

Transformation	Output is Repeatable
Source Qualifier (relational)	Based on transformation configuration. Use sorted ports to produce repeatable data. Or, add a transformation that produces repeatable data immediately after the Source Qualifier transformation. If you do not do either of these options, you might get inconsistent results.
Source Qualifier (flat file)	Always.
Application Source Qualifier	Based on transformation configuration. Use sorted ports for relational sources, such as Siebel sources, to produce repeatable data.

	Or, add a transformation that produces repeatable data immediately after the Application Source Qualifier transformation. If you do not do either of these options, you might get inconsistent results.
MQ Source Qualifier	Always.
XML Source Qualifier	Always.
Aggregator	Always.
Custom	Based on transformation configuration. Configure the Output is Repeatable property according to the Custom transformation procedure behavior.
Expression	Based on input order.
External Procedure	Based on input order.
Filter	Based on input order.
Joiner	Based on input order.
Lookup	Based on input order.
Normalizer (VSAM)	Always. You can enable the session for recovery, however, you might get inconsistent results if you run the session in recovery mode. The Normalizer transformation generates source data in the form of primary keys. Recovering a session might generate different values than if the session completed successfully. However, the PowerCenter Server continues to produce unique key values.
Normalizer (pipeline)	Based on input order.
Rank	Always.
Router	Based on input order.
Sequence Generator	Based on transformation configuration. You must reset the sequence value to the value set in the failed session run. If you do not, you might get inconsistent results.
Sorter, configured for distinct output rows	Always.
Sorter, not configured for distinct output rows	Based on input order.
Stored Procedure	Based on input order.
Transaction Control	Based on input order.

Union	Never.
Update Strategy	Based on input order.
XML Generator	Always.
XML Parser	Always.

To run a session in recovery mode, you must first enable the failed session for recovery. To enable a session for recovery, the Workflow Manager verifies all targets in the mapping receive data from transformations that produce repeatable data. The Workflow Manager uses the values in the Table 11-4 to determine whether or not you can enable a session for recovery.

However, the Workflow Manager cannot verify whether or not you configure some transformations, such as the Sequence Generator transformation, correctly and always allows you to enable these sessions for recovery. You may get inconsistent results if you do not configure these transformations correctly.

You cannot enable a session for recovery in the Workflow Manager under the following circumstances:

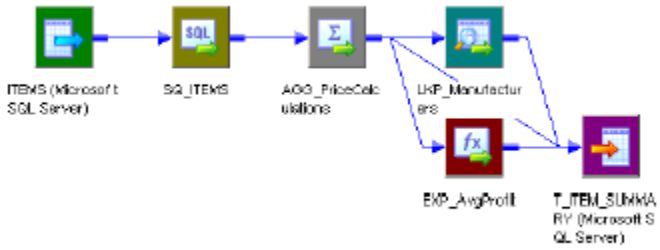
- **You connect a transformation that never produces repeatable data directly to a target.** To enable this session for recovery, you can add a transformation that always produces repeatable data between the transformation that never produces repeatable data and the target.
- **You connect a transformation that never produces repeatable data directly to a transformation that produces repeatable data based on input order.** To enable this session for recovery, you can add a transformation that always produces repeatable data immediately after the transformation that never produces repeatable data.

When a mapping contains a transformation that never produces repeatable data, you can add a transformation that always produces repeatable data immediately after it.

Note: In some cases, you might get inconsistent data if you run some sessions in recovery mode. For a description of circumstances that might lead to inconsistent data.

Figure 11-1 illustrates a mapping you can enable for recovery:

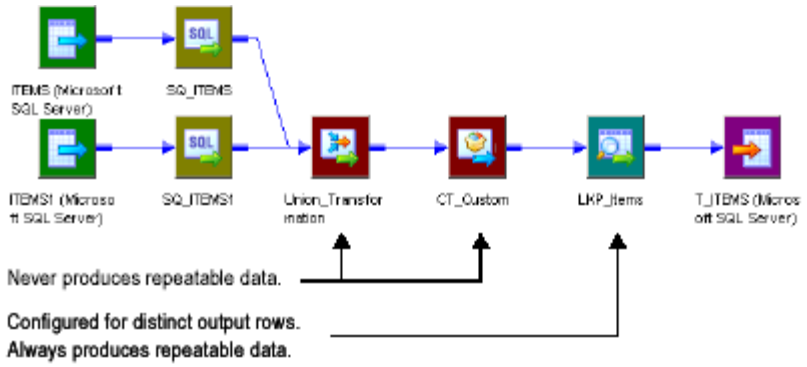
Figure 11-1. Mapping You Can Enable for Recovery



The mapping contains an Aggregator transformation that always produces repeatable data. The Aggregator transformation provides data for the Lookup and Expression transformations. Lookup and Expression transformations produce repeatable data if they receive repeatable data. Therefore, the target receives repeatable data, and you can enable this session for recovery.

Figure 11-2 illustrates a mapping you cannot enable for recovery:

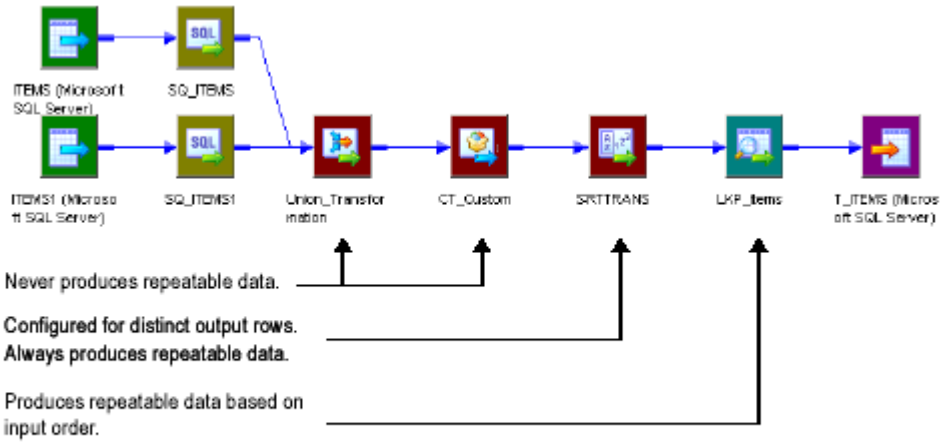
Figure 11-2. Mapping You Cannot Enable for Recovery



The mapping contains two Source Qualifier transformations that produce repeatable data. However, the mapping contains a Union and Custom transformation downstream that never produce repeatable data. The Lookup transformation only produces repeatable data if it receives repeatable data. Therefore, the target does not receive repeatable data, and you cannot enable this session for recovery.

You can modify this mapping to enable the session for recovery by adding a Sorter transformation configured for distinct output rows immediately after transformations that never output repeatable data. Since the Union transformation is connected directly to another transformation that never produces repeatable data, you only need to add a Sorter transformation after the Custom transformation, as shown in the mapping in Figure 11-3:

Figure 11-3. Modified Mapping You Can Enable for Recovery



Recovering a Suspended Workflow

You can configure the workflow to suspend if a task fails. If a session that is enabled for recovery fails, you can correct the error that caused the session to fail and resume the suspended workflow in recovery mode. When the PowerCenter Server resumes the workflow, it runs the failed session in recovery mode. If the recovery session succeeds, the PowerCenter Server runs the rest of the workflow.

You can recover a suspended workflow with sequential or concurrent sessions. For workflows with either sequential or concurrent sessions, suspending the workflow on error is useful if successive tasks in the workflow depend on the success of the previous sessions. For a workflow with concurrent sessions, resuming a suspended workflow in recovery mode also allows you to simultaneously recover concurrent failed sessions.

You can only resume a suspended workflow in recovery mode if a session that is enabled for recovery fails. If a session fails that is not enabled for recovery, you can resume the workflow normally. When you resume the workflow, the PowerCenter Server restarts the session. If the session succeeds, the PowerCenter Server runs the rest of the workflow.

To configure the workflow to suspend on error, enable the Suspend On Error option on the General tab of the workflow properties.

Recovering a Suspended Workflow with Sequential Sessions

When a sequential session enabled for recovery fails, the PowerCenter Server places the workflow in a suspended state. While the workflow is suspended, you can correct the error that caused the session to fail.

After you correct the error, you can resume the workflow in recovery mode. When it resumes the workflow, the PowerCenter Server starts the failed session in recovery mode.

If the recovery session succeeds, the PowerCenter Server runs the rest of the workflow. If the recovery session fails, the PowerCenter Server suspends the workflow again.

Example

Suppose the workflow w_ItemOrders contains two sequential sessions. In this workflow, s_ItemSales is enabled for recovery, and the workflow is configured to suspend on error.

Figure 11-4 illustrates w_ItemOrders:

Figure 11-4. Resuming a Suspended Workflow with Sequential Sessions

Suppose s_ItemSales fails, and the PowerCenter Server suspends the workflow. You correct the error and resume the workflow in recovery mode. The PowerCenter Server recovers the session successfully, and then runs s_UpdateOrders.

If s_UpdateOrders also fails, the PowerCenter Server suspends the workflow again. You correct the error, but you cannot resume the workflow in recovery mode because you did not enable the session for recovery. Instead, you resume the workflow. The PowerCenter Server starts s_UpdateOrders from the beginning, completes the session successfully, and then runs the StopWorkflow control task.

Recovering a Suspended Workflow with Concurrent Sessions

When a concurrent session enabled for recovery fails, the PowerCenter Server places the workflow in a suspending state while it completes any other concurrently running tasks. After concurrent tasks succeed or fail, the PowerCenter Server places the workflow in a suspended state. While the workflow is suspended, you can correct the error that caused the session to fail. If concurrent tasks failed, you can also correct those errors.

After you correct the error, you can resume the workflow in recovery mode. The PowerCenter Server runs the failed session in recovery mode. If multiple concurrent sessions failed, the PowerCenter Server starts all failed sessions enabled for recovery in recovery mode, and restarts other concurrent tasks or sessions not enabled for recovery.

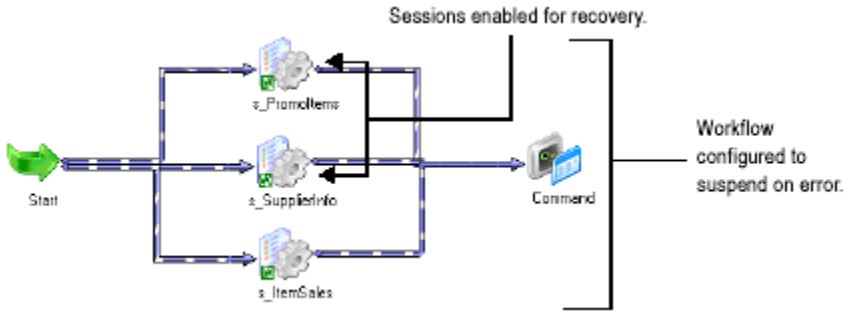
After successful recovery or completion of all failed sessions and tasks, the PowerCenter Server completes the rest of the workflow. If a recovery session or task fails again, the PowerCenter Server suspends the workflow.

Example

Suppose you have the workflow w_ItemsDaily, containing three concurrent sessions, s_SupplierInfo, s_PromoItems, and s_ItemSales. In this workflow, s_SupplierInfo and s_PromoItems are enabled for recovery, and the workflow is configured to suspend on error.

Figure 11-5 illustrates w_ItemsDaily:

Figure 11-5. Resuming a Suspended Workflow with Concurrent Sessions



Suppose s_SupplierInfo fails while the PowerCenter Server is running the three sessions. The PowerCenter Server places the workflow in a suspending state and continues running the other two sessions. s_PromoItems and s_ItemSales also fail, and the PowerCenter Server then places the workflow in a suspended state.

You correct the errors that caused each session to fail and then resume the workflow in recovery mode. The PowerCenter Server starts s_SupplierInfo and s_PromoItems in recovery mode. Since s_ItemSales is not enabled for recovery, it restarts the session from the beginning. The PowerCenter Server runs the three sessions concurrently.

After all sessions succeed, the PowerCenter Server runs the Command task.

Steps for Recovering a Suspended Workflow

You can use the Workflow Monitor to resume a workflow in recovery mode. If the workflow or session is currently scheduled, waiting, or disabled, the PowerCenter Server cannot run the session in recovery mode. You must stop or unschedule the workflow or stop the session.

To resume a workflow or worklet in recovery mode:

1. In the Navigator, select the suspended workflow you want to resume.
2. Choose Task-Resume/Recover.

   The PowerCenter Server resumes the workflow.

Recovering a Failed Workflow

You can configure a session to fail the workflow if the session fails. If the session is also enabled for recovery, you can correct the error that caused the session to fail and recover the workflow from the failed session. When the PowerCenter Server recovers the workflow from the failed session, it runs the failed session in recovery mode. If the recovery session succeeds, the PowerCenter Server runs the rest of the workflow.

You can recover a workflow from a failed sequential or concurrent session. You might want to fail a workflow as a result of session failure if successive tasks in the workflow depend on the success of the previous sessions.

To configure a session to fail the workflow if the session fails, enable the Fail Parent If This Task Fails option on the General tab of the session properties..

Recovering a Failed Workflow with Sequential Sessions

When a sequential session fails that is enabled for recovery and configured to fail the workflow, the PowerCenter Server fails the workflow. You can correct the error that caused the session to fail and recover the workflow from the failed session. When the PowerCenter Server recovers the workflow from the session, it runs the session in recovery mode.

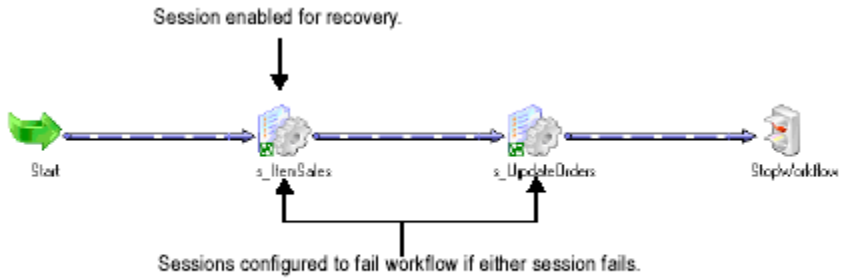
If the recovery session succeeds, the PowerCenter Server runs the rest of the workflow. If the recovery session fails, the PowerCenter Server fails the workflow again.

Example

Suppose the workflow w_ItemOrders contains two sequential sessions. s_ItemSales is enabled for recovery and also configured to fail the parent workflow if it fails.

Figure 11-6 illustrates w_ItemOrders:

Figure 11-6. Recovering Part of a Workflow With Sequential Sessions



Suppose s_ItemSales fails, and the PowerCenter Server fails the workflow. You correct the error and recover the workflow from s_ItemSales. The PowerCenter Server successfully recovers the session, and then runs the next task in the workflow, s_UpdateOrders.

Suppose s_UpdateOrders also fails, and the PowerCenter Server fails the workflow again. You correct the error, but you cannot recover the workflow from the session. Instead, you start the workflow from the session. The PowerCenter Server starts s_UpdateOrders from the beginning, completes the session successfully, and then runs the StopWorkflow control task.

Recovering a Failed Workflow with Concurrent Sessions

When a concurrent session fails that is enabled for recovery and configured to fail the workflow, the PowerCenter Server fails the workflow. You can then correct the error that caused the session to fail and recover the workflow from the failed session. When the PowerCenter Server recovers the workflow, it runs the session in recovery mode. If the recovery session succeeds, the PowerCenter Server runs successive tasks in the workflow in the same path as the session. The PowerCenter Server does not recover or restart concurrent tasks when you recover a workflow from a failed session.

If multiple concurrent sessions fail that are enabled for recovery and configured to fail the workflow, the Informatica fails the workflow when the first session fails. Concurrent sessions continue to run until they succeed or fail. After all concurrent sessions complete, you can correct the errors that caused failures.

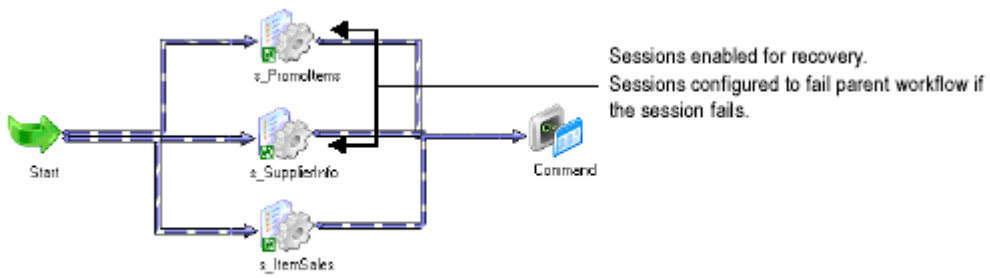
After you correct the errors, you can recover the workflow. If multiple sessions enabled for recovery fail, individually recover all but one failed session. You can then recover the workflow from the remaining failed session. This ensures that the Informatica recovers all concurrent failed sessions before it runs the rest of the workflow.

Example

Suppose the workflow w_ItemsDaily contains three concurrent sessions, s_SupplierInfo, s_PromoItems, and s_ItemSales. In this workflow, each session is enabled for recovery and configured to fail the parent workflow if the session fails.

Figure 11-7 illustrates w_ItemsDaily:

Figure 11-7. Recovering Part of a Workflow with Concurrent Sessions



Suppose s_SupplierInfo fails while the three concurrent sessions are running, and the PowerCenter Server fails the workflow. s_PromoItems and s_ItemSales also fail. You correct the errors that caused each session to fail.

In this case, you must combine two recovery methods to run all sessions before completing the workflow. You recover s_PromoItems individually. You cannot recover s_ItemSales because it is not enabled for recovery, but you start the session from the beginning. After the PowerCenter Server successfully completes s_PromoItems and s_ItemSales, you recover the workflow from s_SupplierInfo. The PowerCenter Server runs the session in recovery mode, and then runs the Command task.

Steps for Recovering a Failed Workflow

You can use the Workflow Manager or Workflow Monitor to recover a failed workflow. If the workflow or session is currently scheduled, waiting, or disabled, the PowerCenter Server cannot run the session in recovery mode. You must stop or unschedule the workflow or stop the session.

To recover a failed workflow using the Workflow Manager:

1. Select the failed session in the Navigator or in the Workflow Designer workspace.
2. Right-click the failed session and choose Recover Workflow from Task.

   The PowerCenter Server runs the failed session in recovery mode, and then runs the rest of the workflow.

To recover a failed workflow using the Workflow Monitor:

1. Select the failed session in the Navigator.
2. Right-click the session and choose Recover Workflow From Task.

   or

   Choose Task-Recover Workflow From Task.

The PowerCenter Server runs the session in recovery mode.

Recovering a Session Task

If you do not configure the workflow to suspend on error, and you do not configure the workflow to fail if sessions or tasks fail, the PowerCenter Server completes the workflow even if it encounters errors. If a session fails, but other tasks in the workflow complete successfully, you may want to recover only the failed session. When the PowerCenter Server recovers a session, it runs the session in recovery mode.

You can recover sequential or concurrent sessions. For workflows with sequential sessions, individually recovering a session is useful if the rest of the workflow succeeded and you need to recover the failed session. This allows you to recover the session without restarting successful tasks.

For workflows with concurrent sessions, this method is useful if multiple concurrent sessions fail and also cause the workflow to fail. You can individually recover concurrent sessions and individually start subsequent tasks in the workflow paths until the paths converge at a single task.

In other complex, branched workflows, individually recovering multiple failed sessions allows you to specify the order in which the sessions run.

Recovering Sequential Sessions

When a sequential session enabled for recovery fails, and the workflow is not configured to suspend or fail on error, the PowerCenter Server continues to run the workflow. You can correct the error that caused the session to fail.

After you correct the error, you can individually recover the failed session. When the PowerCenter Server individually recovers a session, it runs the session in recovery mode. It does not run other tasks in the workflow.

Recovering Concurrent Sessions

When a concurrent session enabled for recovery fails, the PowerCenter Server continues to run the workflow. Other tasks and the workflow may succeed. You can correct the error that caused the session to fail. If concurrent tasks failed, you can also correct those errors. After you correct the errors, you can individually recover each session without running the rest of the workflow.

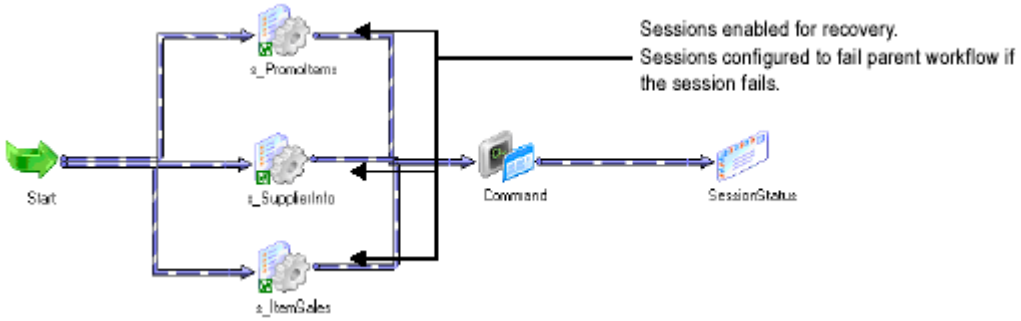
If multiple concurrent sessions fail that are enabled for recovery and configured to fail the workflow on session failure, the PowerCenter Server fails the workflow. You can correct the errors that caused the sessions to fail. After you correct the errors, you can individually recover each session. Once all concurrent tasks are recovered or complete, you can start the session from a task where the concurrent paths converge.

Example

Suppose the workflow w_ItemsDaily contains three concurrently running sessions. Each session is enabled for recovery and configured to fail the workflow if the session fails.

Figure 11-8 illustrates w_ItemsDaily:

Figure 11-8. Recovering Concurrent Sessions Individually



Suppose s_ItemSales fails and the PowerCenter Server fails the workflow. s_PromoItems and s_SupplierInfo also fail. You correct the errors that caused the sessions to fail.

After you correct the errors, you individually recover each failed session. The PowerCenter Server successfully recovers the sessions. The workflow paths after the sessions converge at the Command task, allowing you to start the workflow from the Command task and complete the workflow.

Alternatively, after you correct the errors, you could also individually recover two of the three failed sessions. After the PowerCenter Server successfully recovers the sessions, you can recover the workflow from the third session. The PowerCenter Server then recovers the third session and, on successful recovery, runs the rest of the workflow.

Steps for Recovering a Session Task

You can use the Workflow Manager or Workflow Monitor to recover a failed session in a workflow. If the workflow or session is currently scheduled, waiting, or disabled, the PowerCenter Server cannot run the session in recovery mode. You must stop or unschedule the workflow or stop the session.

To recover a failed session using the Workflow Manager:

1. Select the failed session in the Navigator or in the Workflow Designer workspace.

2. Right-click the failed session and choose Recover Task.

   The PowerCenter Server runs the session in recovery mode.

To recover a failed session using the Workflow Monitor:

1. Select the failed session in the Navigator.
2. Right-click the session and choose Recover Task.

   or

   Choose Task-Recover Task.

   The PowerCenter Server runs the session in recovery mode.

Server Handling for Recovery

The PowerCenter Server writes recovery data to relational target databases when you run a session enabled for recovery. If the session fails, the PowerCenter Server uses the recovery data to determine the point at which it continues to commit data during the recovery session.

Verifying Recovery Tables

The PowerCenter Server creates recovery information in cache files for all sessions enabled for recovery. It also creates recovery tables on the target database for relational targets during the initial session run.

If the session is enabled for recovery, the PowerCenter Server creates recovery information in cache files during the normal session run. The PowerCenter Server stores the cache files in the directory specified for $PMCacheDir. The PowerCenter Server generates file names in the format PMGMD_METADATA_*.dat. Do not alter these files or remove them from the PowerCenter Server cache directory. The PowerCenter Server cannot run the recovery session if you delete the recovery cache files.

If the session writes to a relational database and is enabled for recovery, the PowerCenter Server also verifies the recovery tables on the target database for all relational targets at the beginning of a normal session run. If the tables do not exist, the PowerCenter Server creates them. If the database user name the PowerCenter Server uses to connect to the target database does not have permission to create the recovery tables, you must manually create them.

During the session run, the PowerCenter Server writes target load information for normal load targets into the recovery tables. If the session fails, the PowerCenter Server uses this information to complete the session in recovery mode. If the session is configured to write to relational targets in bulk mode, the PowerCenter Server does not write recovery information to the recovery tables.

If the session completes successfully, the PowerCenter Server deletes all recovery cache files and removes recovery table entries that are related to the session. The PowerCenter Server initializes the information in the recovery tables at the beginning of the next session run.

The PowerCenter Server also uses the recovery cache files to store messages from real-time sources.

Running Recovery

If a session enabled for recovery fails, you can run the session in recovery mode. The PowerCenter Server moves a recovery session through the states of a normal session: scheduled, waiting, running, succeeded, and failed. When the PowerCenter Server starts the recovery session, it runs all pre-session tasks.

For relational normal load targets, the PowerCenter Server performs incremental load recovery. It uses the recovery information created during the normal session run to determine the point at which the session stopped committing data to the target. It then continues writing data to the target. On successful recovery, the PowerCenter Server removes the recovery information from the tables.

For example, if the PowerCenter Server commits 10,000 rows before the session fails, when you run the session in recovery mode, the PowerCenter Server bypasses the rows up to 10,000 and starts loading with row 10,001.

If the session writes to a relational target in bulk mode, the PowerCenter Server performs the entire writer run. If the Truncate Target Table option is enabled in the session properties, the PowerCenter Server truncates the target before loading data.

If the session writes to a flat file or XML file, the PowerCenter Server performs full load recovery. It overwrites the existing output file and performs the entire writer run. If the session writes to heterogeneous targets, the PowerCenter Server performs incremental load recovery for all relational normal load targets and full load recovery for all other target types.

On successful recovery, the PowerCenter Server deletes recovery cache files associated with the session. It also performs all post-session tasks.

Completing Unrecoverable Sessions

In some cases, you cannot perform recovery for a session. There may also be circumstances that cause a recovery session to fail or produce inconsistent data. If you cannot recover a session, you can run the session again.

You cannot run sessions in recovery mode under the following circumstances:

- **You change the number of partitions.** If you change the number of partitions after the session fails, the recovery session fails.
- **Recovery table is empty or missing from the target database.** The PowerCenter Server fails the recovery session under the following circumstances:
    - You deleted the table after the PowerCenter Server created it.
    - The session enabled for recovery succeeded, and the PowerCenter Server removed the recovery information from the table.

- **Recovery cache file is missing.** The PowerCenter Server fails the recovery session if the recovery cache file is missing from the PowerCenter Server cache directory.
- **The PowerCenter Server performing recovery is on a different operating system.** The operating system of the PowerCenter Server that runs the recovery session must be the same as the operating system of the PowerCenter Server that ran the failed session.

You might get inconsistent data if you perform recovery under the following circumstances:

- **You change the partitioning configuration.** If you change any partitioning options after the session fails, you may get inconsistent data.
- **Source data is not sorted.** To perform a successful recovery, the PowerCenter Server must process source rows during recovery in the same order it processes them during the initial session. Use the Sorted Ports option in the Source Qualifier transformation or add a Sorter transformation directly after the Source Qualifier transformation.
- **The sources or targets change after the initial session failure.** If you drop or create indexes, or edit data in the source or target tables before recovering a session, the PowerCenter Server may return missing or repeat rows.
- **The session writes to a relational target in bulk mode, but the session is not configured to truncate the target table.** The PowerCenter Server may load duplicate rows to the during the recovery session.
- **The mapping uses a Normalizer transformation.** The Normalizer transformation generates source data in the form of primary keys. Recovering a session might generate different values than if the session completed successfully. However, the PowerCenter Server will continue to produce unique key values.
- **The mapping uses a Sequence Generator transformation.** The Sequence Generator transformation generates source data in the form of sequence values. Recovering a session might generate different values than if the session completed successfully.

  If you want to ensure the same sequence data is generated during the recovery session, you can reset the value specified as the Current Value in the Sequence Generator transformation properties to the same value used when you ran the failed session. If you do not reset the Current Value, the PowerCenter Server will continue to generate unique Sequence values.

- **The session performs incremental aggregation and the PowerCenter Server stops unexpectedly.** If the PowerCenter Server stops unexpectedly while running an incremental aggregation session, the recovery session cannot use the incremental aggregation cache files. Rename the backup cache files for the session from PMAGG*.idx.bak and PMAGG*.dat.bak to PMAGG*.idx and PMAGG*.dat before you perform recovery.
- **The PowerCenter Server data movement mode changes after the initial session failure.** If you change the data movement mode before recovering the session, the PowerCenter Server might return incorrect data.
- **The PowerCenter Server code page or source and target code pages change after the initial session failure.** If you change the source, target, or PowerCenter Server code pages, the PowerCenter Server might

return incorrect data. You can perform recovery if the new code pages are two-way compatible with the original code pages.

- **The PowerCenter Server runs in Unicode mode and you change the session sort order.** When the PowerCenter Server runs in Unicode mode, it sorts character data based on the sort order selected for the session. Do not perform recovery if you change the session sort order after the session fails.

Name : K.Ramesh Babu

Mail : RameshNani08@yahoo.com

Ph : 8978992991