Designer/2000 for Oracle Warehouse

Mark Herring
Oracle Corporation

Introduction

Designer/2000 can be used to assist in the design, creation, population and extraction of information from a data warehouse. This paper will look at the basic components and steps that are required in building and maintaining a data warehouse, and will concentrate on practical areas where Designer/2000 can be used to facilitate this process.

Data Warehousing

Overview

Corporate applications are essentially optimized for OLTP (On-line Transaction Processing) applications. These transactions are characterized by fast access times, no end user customization, and multiple simultaneous user access. The majority of legacy systems are OLTP based and consequently the data resides in a variety of data sources. The data sources can be in flat files, hierarchical databases (like IMS), network structured databases (like IDMS), inverted list databases (like DATACOM/ DB), relational systems (like Oracle) or more commonly a combination of the above.

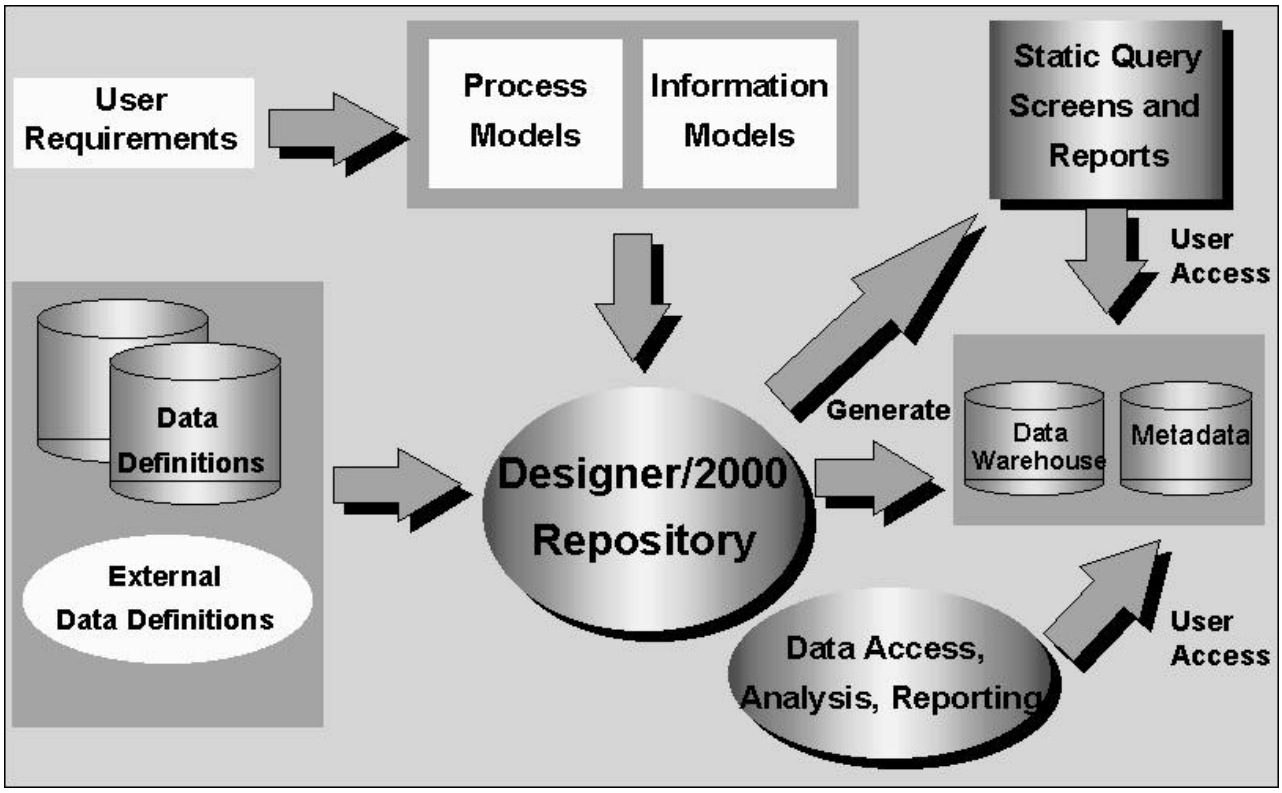
Businesses need to access the wealth of information that is stored in these data sources, collate the data and determine trends. Organizations that have been able to obtain such data are naturally better able to base decisions and strategy on past historical trends, and in doing so, gain competitive advantage.

The problem is data stored across multiple databases and repositories is inconsistent, incomplete and optimized for OLTP transactions. The data structure itself, does not lend itself to analysis and data query. In addition businesses cannot allow the production systems to be exposed to massive queries and statistical computations that might reduce their performance.

The requirement is simple: collate data from these multiple sources into one logical database that is optimized for OLAP (On-line Analytical Processing) transactions and allow knowledgeable data users to access this data. This is what is referred to as a data warehouse.

The notion of data warehousing is not a new idea but is rather the formalization of lessons learned in Decision Support Systems (DSS) and the growing demand for historical data to be analyzed and manipulated by end users with little or no IS involvement.

Designer/2000 adds repository support in the following stages of the data warehouse creation, usage and maintenance cycles.



Stage 1: Analyze Warehouse Requirements

For any successful warehouse it is imperative to understand not only the shortcomings of the existing systems but requirements for the new system. At this stage it is important to ignore the physical implementation and the source of all the data entries for the warehouse. Joint Application Development (JAD) or traditional analysis design meetings can be used here to drive out the requirements. A good starting point is usually the master information systems plan, if one exists.

Designer/2000 provides the ideal medium to capture this information. It has many rich analysis and design tools which will aid in this requirement realization and exploration phase.

The data warehousing application should be created as a new application within the Designer/2000 repository. With the aid of the Function Hierarchy diagrammer determine the high level functions that the warehouse needs to support. Use the Entity Relationship diagrammer (ERD) to determine the logical data entities that are required as well as any relationships between these entities. Examine existing reports and data the user currently uses as well as additional requirements.

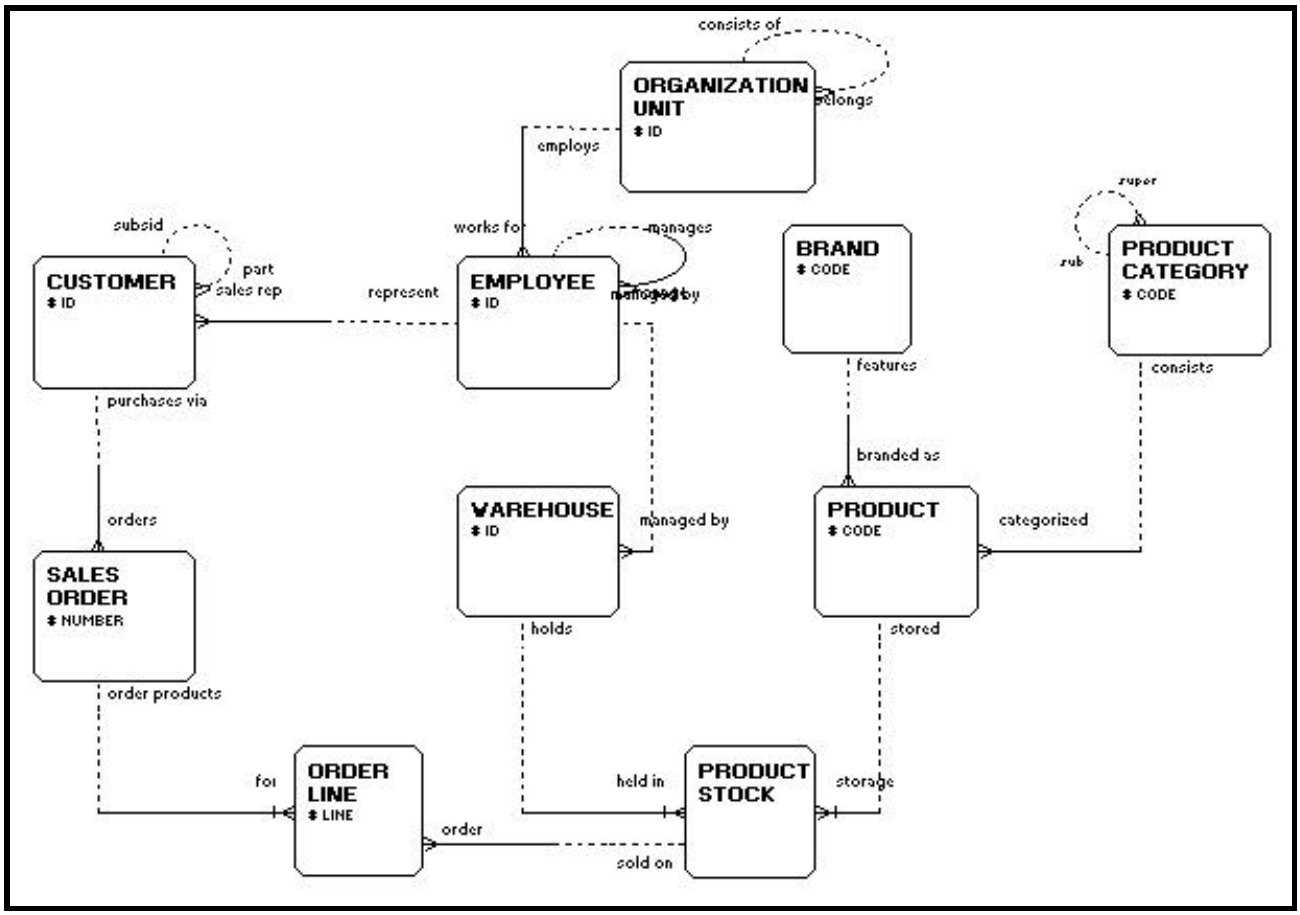


Figure 1: Logical Data Requirements

Data warehouses are usually developed in subject areas. Depending on the business requirements and accessibility of the data, one subject area might be developed before another. Each subject area should be depicted as a separate diagram within the Designer/2000 repository.

In the paper we will consider the creation of a data warehouse to support the sales subject area for a simple product distribution system. Figure 1 represents the logical data requirements for the system.

Each query or set of data the user requires capture as a process in Designer/2000 and set the flag that the process is also a business function. Use either the Data Flow diagrammer or the Process Modeller to confirm that the system can logically deliver the data the users require. Capture the following minimum set of information per process:

- How often the user needs the data (periodicity), or how often this query will be executed.
- Allowed "staleness" of data. Since the warehouse requires periodic updates, and some processes will not require "live" data this is an important consideration. For example, will the sales analysis query yield sufficient trend analysis if the data is 1 week old?
- What are the primary ways the user will want to drill down or up on this data? These are called **Data Dimensions** in warehousing terms. For example, sales analysis might be shown per week, drill up to by month, drill up to financial quarter, finally drill up to financial year. Reflect any new data elements on the ERD.
- What is acceptable to the user for the query response? For example, can the process be automated to produce results every day, week or month or must the process be run in real-time?

Figure 2 and 3 is an example of the documentation captured for a process, “ANALYZE PRODUCT SALES”. Figure 4 shows the data flow diagram which ensures that there is sufficient data to be able to process the queries.

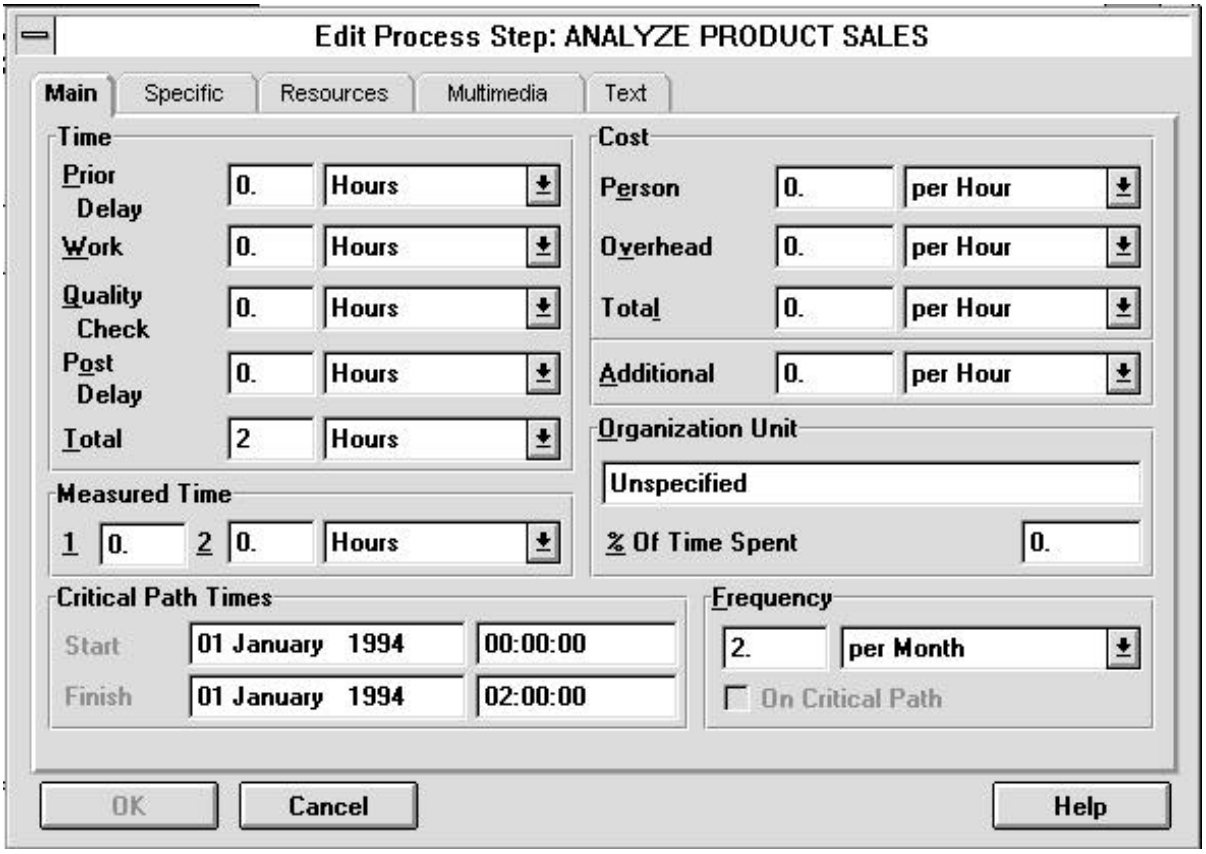


Figure 2: Process Information

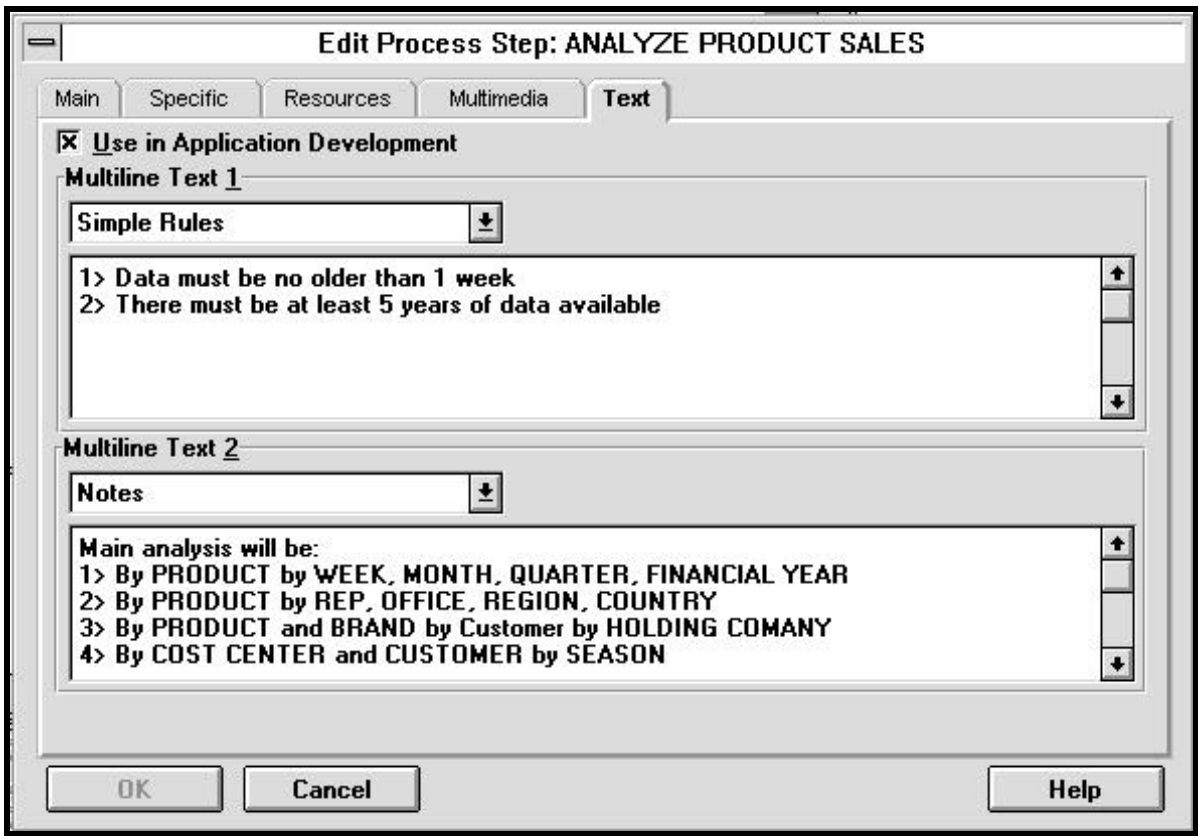


Figure 3: Minimum Process Information

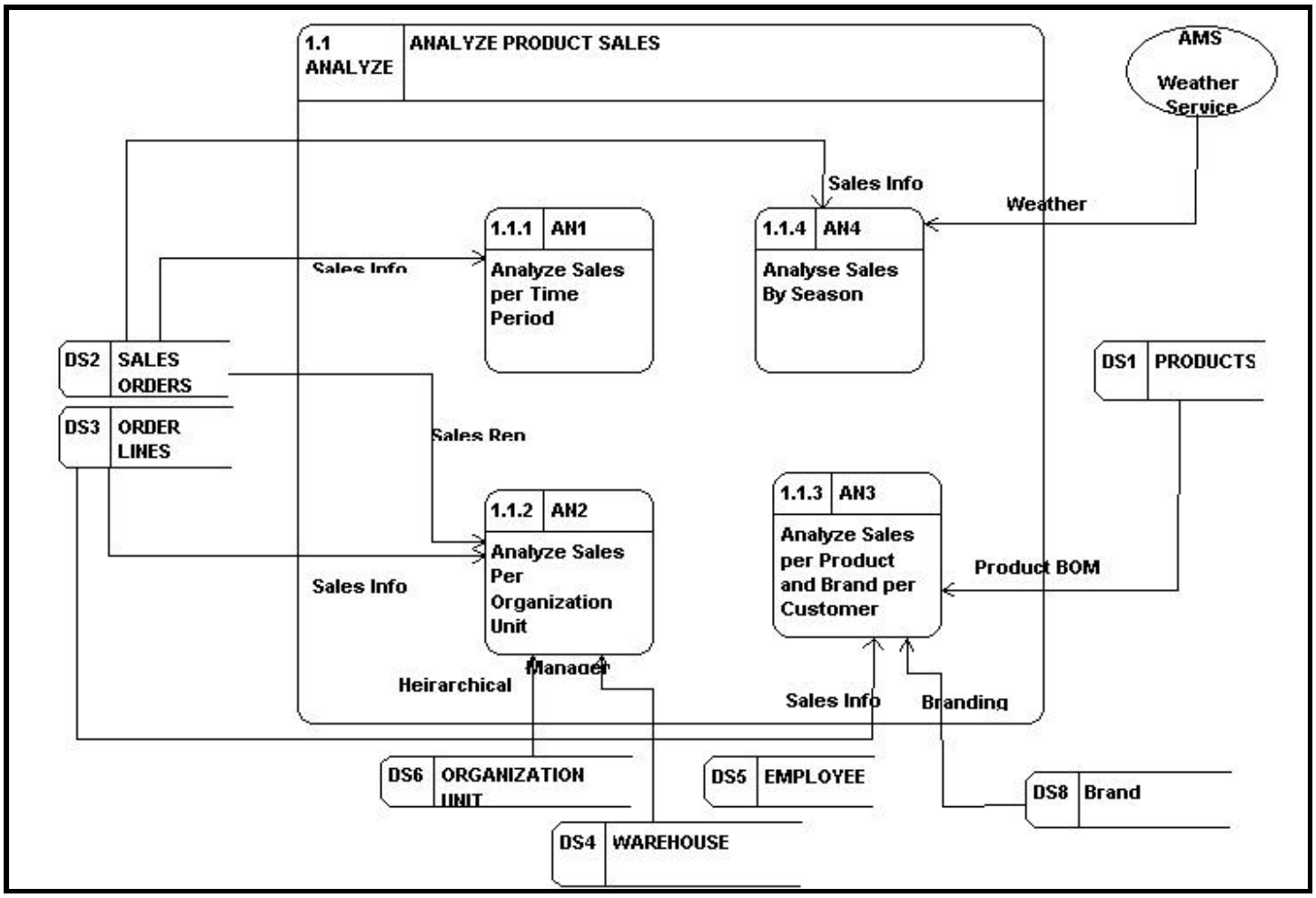


Figure 4: Data Flow Diagram for Analyze Sales

Designer/2000's impact analysis and quality reports, and the matrix diagram should be used to verify the completeness of the requirements.

Stage 2: Design the Physical Data Warehouse

Once the logical data and processes have been captured it is necessary to define the physical constructs of the warehouse. From the ERD any entity which has no foreign keys from it will be the basis for a reference table. Based on the analysis of the processes, data is denormalized into either reference tables or fact tables. The reference tables will be the basis for any dimensional analysis that must be done for each process. Reference tables, in contrast to fact tables, consist of static data that is seldom updated or changed. Fact data is usually lightly summarized data based on live OLTP systems. The degree of summarization will impact the usability and performance of the warehouse. Highly summarized data will allow better performance but might not give the users the ability to determine sales per product per hour for example.

The goal of this phase is to denormalize and summarize data into a few fact tables with relationships to single-key reference tables. The resulting database schema is termed a *Star Schema*, see Figure 5.

From Figure 5 it should be apparent that not all reference tables are derived from the logical model. The

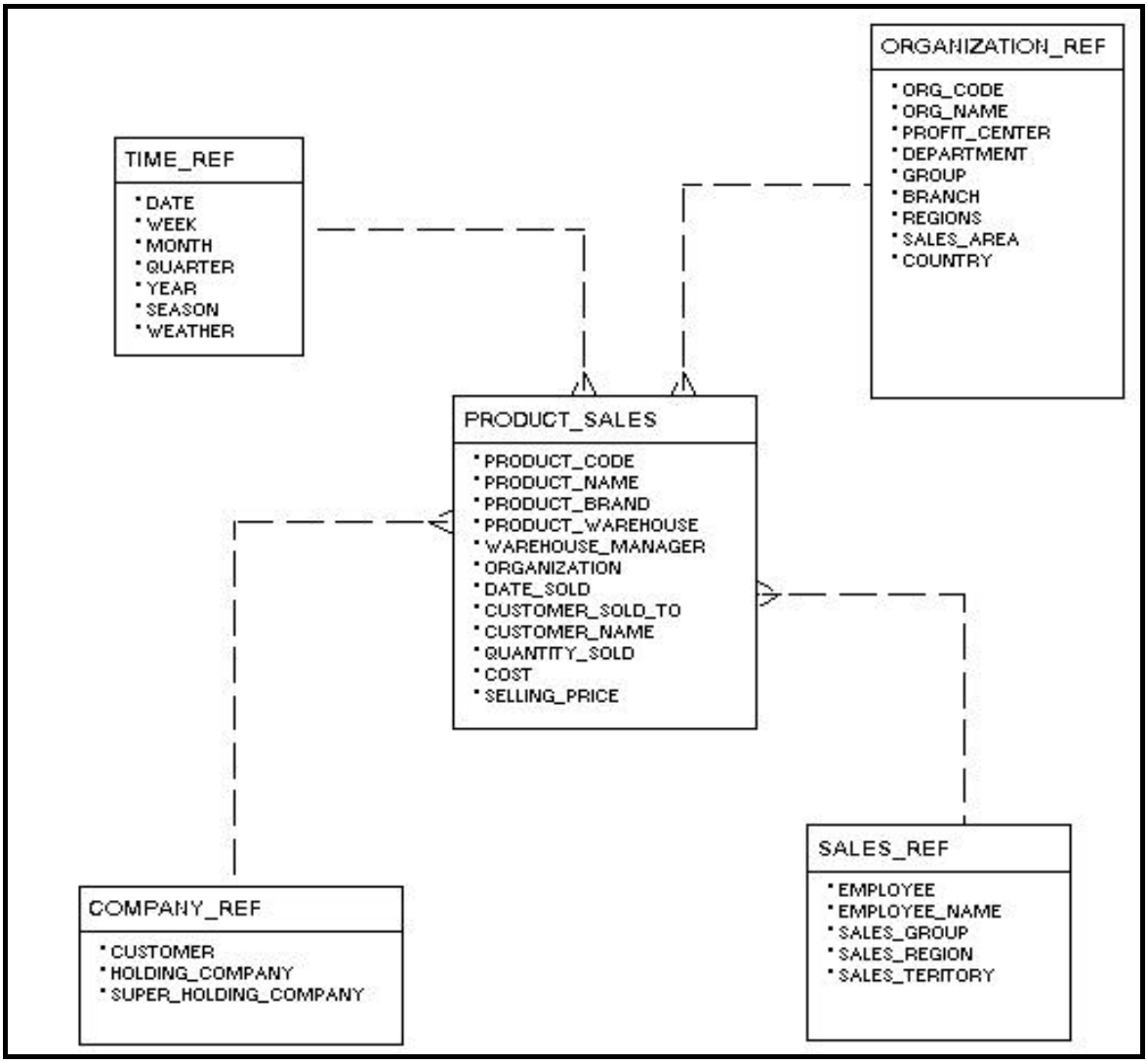


Figure 5: Star Schema Diagram

TIME_REF table was constructed to allow for the drilling up of data from days, into weeks, into financial years. In the logical model date was stored on the SALES ORDER entity and logically, or even technically it would be possible to create SQL to provide the roll-up analysis. From a user perspective this SQL manipulation is too complex and the TIME_REF reference table is created to allow for a much simpler query. The TIME_REF table holds details about the season and weather, this is something the current system does not track but experienced users have expressed the desire to analyze sales in this dimension as well.

It is important to document design considerations behind each reference and fact table as well as documenting the relationship these models have with the logical ERD. Designer/2000 allows the documentation of denormalization, validation and derivation logic whilst keeping track of source entity/attribute information.

The data warehouse must be optimized for queries so multiple indices must be defined and created based on the process analysis already performed. It is worth noting that since there will be no OLTP transactions to consider the warehouse application has many more indices than normally created for OLTP applications. Designer/2000 should be used to create each index definition and the reasons behind why this index was created.

Designer/2000 can generate the SQL to create the tables, views, snapshots and indexes defines but this will be modified once more knowledge is gained on the data warehouse's data sources.

Stage 3: Identify the data sources

Each logical piece of warehouse data must be derived from somewhere! Source data might be in existing systems, other portions of it may be derived or computed off existing systems, whilst other data might be imported from third parties. Consider the TIME_REF reference table in Figure 5. The weather information does not exist in the operational systems but can be obtained from a third party company as a flat file.

Create an Oracle database definition in Designer/2000 for each distinct Oracle data source, for non-Oracle data sources create an ANSI database definition. All data objects stored in an Oracle database can be reverse engineered into Designer/2000 and shown graphically on the Data Diagrammer. This is true for any data source that can be addressed through Oracle's gateway technology. Using these techniques most relational database models can be reverse engineered into Designer/2000. An alternative strategy would be to use some other third party tools to reverse engineer these data usages and either via the Designer/2000 API or CASE*Exchange populate the Designer/2000 repository.

For flat file data the data is usually composed of multiple records of fixed length. These records cannot be reverse engineered into the repository but can be documented in the repository as Files composed of records composed of fields. Figure 6 shows a sample of the flat file obtained for weather information, all this information can be stored in the Designer/2000 repository.

Figure 6 - Weather information Flat file

```
12345678901234567890123456789012345678901234
010194 SSW 20 20 080 020 PCD 23 67 67 161 16
020194 SW  22 22 082 016 SUN 22 78 56 121  8
```

Each record in the file is fixed length with the following columns:
Pos 1-6: Date in MMDDYY format
Pos 8-10: Wind direction
Pos 14-16: Wind Speed
Pos 18-20: Maximum Temperature
Pos 22-24: Minimum Temperature
Pos 26-28: Classification (SUN = Sunny etc.)
etc.

For each data element in the repository document the following minimum properties:

- Coding of data, e.g. Code A means Accepted etc. (use allowed values or domains in Designer/2000 to store this information, see figure 6)
- Number of records and growth
- Accuracy of the data, e.g. credit history information here is poor.
- Frequency of data update
- Synonyms, business terms, etc. that relate to this data.
- Derivation formulae and rules

Stage 4: Map the Data Sources to the Warehouse

At this stage the data warehouse tables have been designed and all data source members have been identified and documented. Designer/2000 will be used to document the transformations and mapping necessary from the data sources to the data in the warehouse.

In order to map source columns the repository can either be extended or an additional table can be created. The latter approach, is probably the simplest but this is not the most robust since it does not consider the impact of versioning, access control and data migration. To follow the simplest approach, build a form around the new table ADD_DATA. The SQL for this table is:

```
create table ADD_DATA_MAP (
      WHSE_COLUMN  number(38),
      TO_COLUMN    number(38),
      REASON       varchar2(2000));
```

Creating a simple form against this table and the API views ci_columns, ci_table_definitions, ci_oracle_databases and ci_ansi_databases it is possible to show the data source mapping, see Figure 7.

The most critical information is how the extraction and enrichment routines need to be run in order to populate the data warehouse. In data warehousing terms this is known as ***Data Scrubbing***.

For each data scrubbing routine document the actual transformation logic or pseudocode of the routine as a module in Designer/2000. Some modules might be of type PL/SQL and can be created as a stored procedure inside the Oracle 7 kernel, others might be 4GL or even assembly code routines. Since Designer/2000 supports the notion of user-defined module languages, all of these details can be captured in the tool. Multiple modules might support the update of one fact or reference table and these can be captured as module networks. For each module the following minimum properties should be captured:

- Any sequence or dependent module information
- Rules for data inclusion
- Data cleaning and enriching rules
- Periodicity of updates
- Data sources required for each module
- Language of the module

Determine how each module interacts with the data sources and the data warehouse, depict this graphically on the Matrix Diagrammer, showing module to column associations. This will provide invaluable impact analysis reports if the warehouse or source data changes.

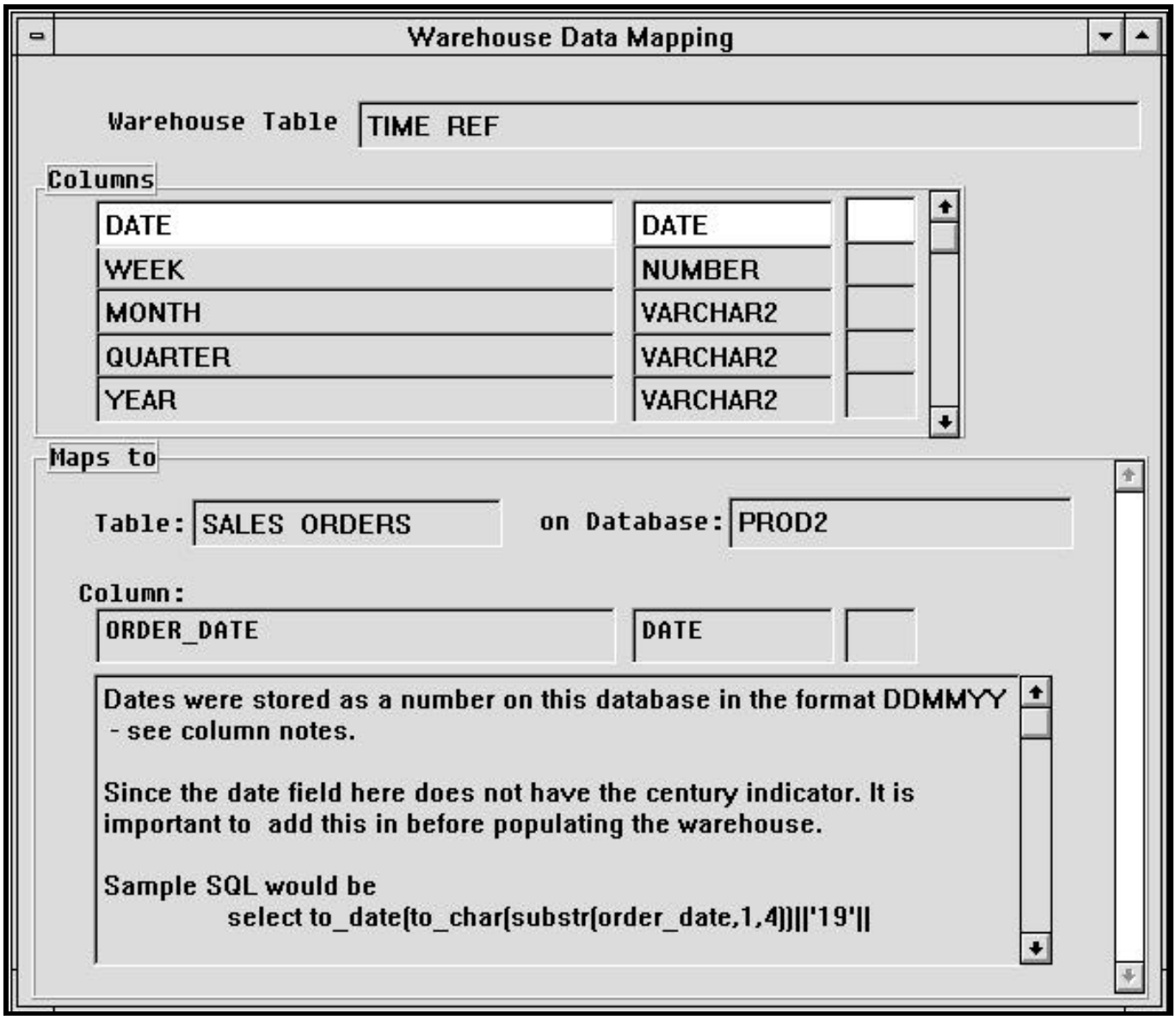


Figure 6: Data Source to Warehouse Mapping

Stage 5: Create the Data Warehouse

From the initial analysis and the determination of the data warehouse population schemes it should be possible to update Designer/2000 with accurate estimates of the warehouse data's number of rows, initial size and growth predictions. Designer/2000 will use this information to provide database sizing estimates and storage definitions.

At the end of this stage allow Designer/2000 to generate file specifications, tablespace definitions and the actual table, snapshot and view definitions. The data warehouse is now ready for population.

Stage 6: Populate the data warehouse

Since all the transformation, enrichment and refinement methods are stored in the Designer/2000 repository these can be extracted and run based on their logical precedence and language. Some of this might have to be scheduled and placed in an automated scheduler, some of the modules might make use of Oracle7's powerful replication feature to automatically keep data synchronized between data bases, whilst other data might be created on a recurrent basis via Oracle7's snapshot technology.

Stage 7: Enable the users to access the Data Warehouse

Designer/2000 can be used to generate very powerful screens and reports via the Developer/2000 generators. These programs will allow some user interaction and run time formatting.

Designer/2000 can generate directly into the Discoverer/2000 metamodels, enabling the end user to create custom queries with tools that were created for that purpose.

Data warehouses also need to be updated with budget and forecasting information. Designer/2000 and its powerful Developer/2000 generator can generate the applications to support this functionality.

Conclusion

Organizations need to exploit the information already stored in their legacy and other systems. Designer/2000 provides a strong repository to hold the requirements and specifications for the data warehouse. In addition it can hold and even run data scrubbing routines to ensure only "clean" data is allowed into the warehouse. Designer/2000 will support the extraction of the data from the warehouse via its strong integration with Developer/2000 and Discoverer/2000.