Data warehouses are databases that store and maintain analytical data separately from transaction-oriented databases for the purpose of decision support.

On the other hand, data warehouses tend to keep years' worth of data in order to enable analysis of historical data. They provide storage, functionality, and responsiveness to queries beyond the capabilities of transaction-oriented databases.

Presently there is a great need to provide decision makers from middle management upward with information at the correct level of detail to support decision making. *Data warehousing, online analytical processing* (OLAP).

29.1 Introduction, Definitions, and Terminology

Data warehouses are distinct from traditional databases in their structure, functioning, performance, and purpose. W. H. Inmon1 characterized a data warehouse as a *subject-oriented, integrated, nonvolatile, time-variant collection of data in support of management's decisions.* Data warehouses provide access to data for complex analysis, knowledge discovery, and decision making through ad hoc and canned queries. Canned queries refer to a-priori defined queries with parameters that may recur with high frequency. They support high-performance demands on an organization's data and information.

Traditional databases support **online transaction processing (OLTP)**, which includes insertions, updates, and deletions while also supporting information query requirements. Traditional relational databases are optimized to process queries that may touch a small part of the database and transactions that deal with insertions or updates of a few tuples per relation to process. By contrast, data warehouses are designed precisely to support efficient extraction, processing, and presentation for analytic and decisionmaking purposes. In comparison to traditional databases, data warehouses generally contain very large amounts of data from multiple sources that may include databases from different data models and sometimes files acquired from independent systems and platforms.

29.2 Characteristics of Data Warehouses

Warehouse insertions are handled by the warehouse's **ETL (extract, transform, load)** process, which does a large amount of preprocessing and which is shown in Figure 29.1.
extract
filter
transform
classify
integrate
aggregate
summarize

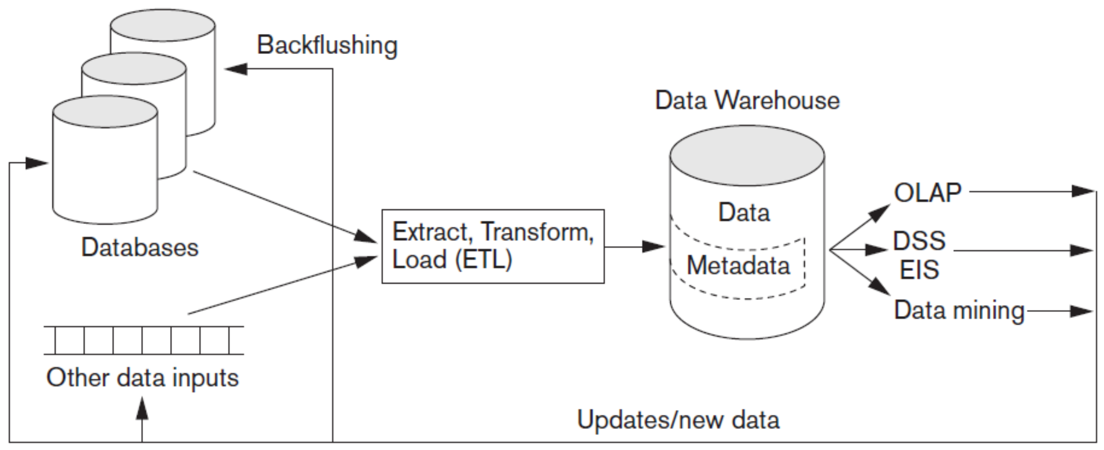


Figure 29.1 Overview of the general architecture of a data warehouse.

1

Data Warehousing Concepts

This chapter provides an overview of the Oracle data warehousing implementation. It includes:

- What is a Data Warehouse?
- Data Warehouse Architectures
- Extracting Information from a Data Warehouse

Note that this book is meant as a supplement to standard texts about data warehousing. This book focuses on Oracle-specific material and does not reproduce in detail material of a general nature. Two standard texts are:

- *The Data Warehouse Toolkit* by Ralph Kimball (John Wiley and Sons, 1996)
- *Building the Data Warehouse* by William Inmon (John Wiley and Sons, 1996)

What is a Data Warehouse?

A data warehouse is a relational database that is designed for query and analysis rather than for transaction processing. It usually contains historical data derived from transaction data, but can include data from other sources. Data warehouses separate analysis workload from transaction workload and enable an organization to consolidate data from several sources. This helps in:

- Maintaining historical records
- Analyzing the data to gain a better understanding of the business and to improve the business.

In addition to a relational database, a data warehouse environment can include an extraction, transportation, transformation, and loading (ETL) solution, statistical analysis, reporting, data mining capabilities, client analysis tools, and other applications that manage the process of gathering data, transforming it into useful,
actionable information, and delivering it to business users.

Common way of introducing data warehousing is to refer to the characteristics of a data warehouse as set forth by William Inmon:

- Subject Oriented
- Integrated
- Nonvolatile
- Time Variant

Subject Oriented

Data warehouses are designed to help you analyze data. For example, to learn more about your company's sales data, you can build a data warehouse that concentrates on sales. Using this data warehouse, you can answer questions such as "Who was our best customer for this item last year?" or "Who is likely to be our best customer next year?" This ability to define a data warehouse by subject matter, sales in this case, makes the data warehouse subject oriented.

Subject-oriented – the warehouse is organized around the major subjects of the enterprise (such as customers, products, and sales) rather than the major application areas (such as customer invoicing, stock control, and product sales). This is reflected in the need to store decision-support data rather than application-oriented data.

Integrated

Integration is closely related to subject orientation. Data warehouses must put data from disparate sources into a consistent format. They must resolve such problems as naming conflicts and inconsistencies among units of measure. When they achieve this,
they are said to be integrated.

Integrated – the warehouse houses data from various enterprise-wide sources. The source data is often inconsistent using, for example, different formats. The integrated data source must be made consistent in order to present a unified view of the data to the users.

Nonvolatile

Non-volatile – the data in the warehouse is not updated in real-time but is refreshed from operational systems on a regular basis. New data is always added as a supplement to the database, rather than as a replacement. The database continuously absorbs this new data, incrementally integrating it with the previous data.

Time Variant

A data warehouse's focus on change over time is what is meant by the term time variant. In order to discover trends and identify hidden patterns and relationships in business, analysts need large amounts of data. This is very much in contrast to **online transaction processing (OLTP)** systems, where performance requirements demand that historical data be moved to an archive.

Contrasting OLTP and Data Warehousing Environments

Figure 1–1 illustrates key differences between an OLTP system and a data warehouse.

Figure 1–1 Contrasting OLTP and Data Warehousing Environments

OLTP		DATA WAREHOUSING
Complex Data Structures (3NF Databases)		Multidimensional Data Structures+NF
Few	Indexes	Many
Many	Joins	Some
Normalized DBMS	Duplicated Data	Denormalized DBMS
Rare	Derived Data And Aggregates	Common

One major difference between the types of system is that data warehouses are not usually in **third normal form (3NF)**, a type of data normalization common in OLTP environments.
Data warehouses and OLTP systems have very different requirements. Here are some examples of differences between typical data warehouses and OLTP systems:

- Workload

Data warehouses are designed to accommodate *ad hoc* queries and data analysis. You might not know the workload of your data warehouse in advance, so a data warehouse should be optimized to perform well for a wide variety of possible query and analytical operations.
OLTP systems support only predefined operations. Your applications might be specifically tuned or designed to support only these operations.

- Data modifications

A data warehouse is updated on a regular basis by the ETL process (run nightly or weekly) using bulk data modification techniques. The end users of a data warehouse do not directly update the data warehouse except when using analytical tools, such as data mining, to make predictions with associated probabilities, assign customers to market segments, and develop customer profiles.

In OLTP systems, end users routinely issue individual data modification statements to the database. The OLTP database is always up to date, and reflects the current state of each business transaction.

- Schema design

Data warehouses often use denormalized or partially denormalized schemas (such as a star schema) to optimize query and analytical performance.
OLTP systems often use fully normalized schemas to optimize update/insert/delete performance, and to guarantee data consistency.

- Typical operations

A typical data warehouse query scans thousands or millions of rows. For example, "Find the total sales for all customers last month."
A typical OLTP operation accesses only a handful of records. For example, "Retrieve the current order for this customer."

- Historical data

Data warehouses usually store many months or years of data. This is to support historical analysis and reporting.
OLTP systems usually store data from only a few weeks or months. The OLTP system stores only historical data as needed to successfully meet the requirements of the current transaction.

Data Warehouse Architectures

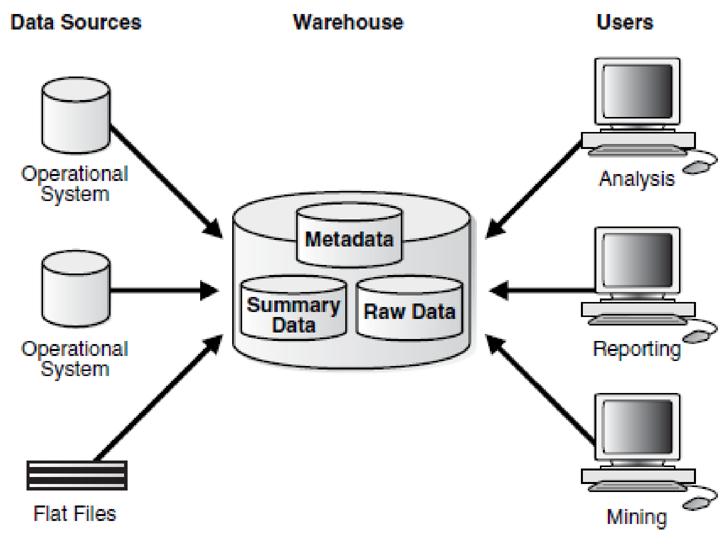
Data warehouses and their architectures vary depending upon the specifics of an organization's situation. Three common architectures are:

- Data Warehouse Architecture: Basic
- Data Warehouse Architecture: with a Staging Area
- Data Warehouse Architecture: with a Staging Area and Data Marts

Data Warehouse Architecture: Basic

Figure 1–2 shows a simple architecture for a data warehouse. End users directly access data derived from several source systems through the data warehouse.

Figure 1–2 Architecture of a Data Warehouse

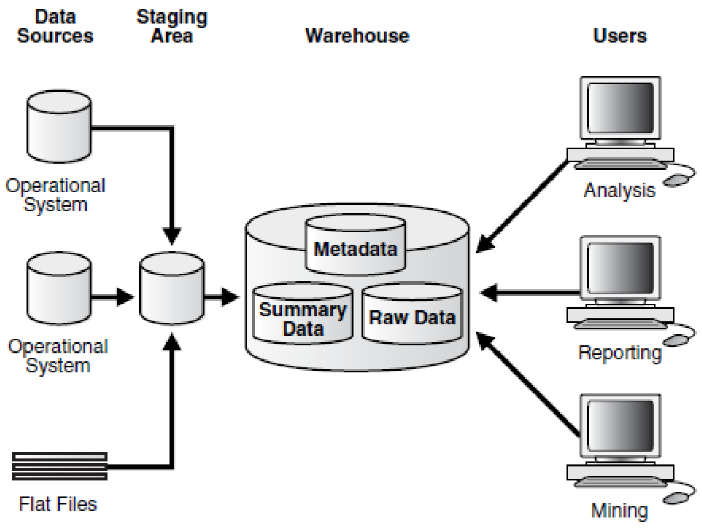


In Figure 1–2, the metadata and raw data of a traditional OLTP system is present, as is an additional type of data, summary data. Summaries are very valuable in data warehouses because they pre-compute long operations in advance. For example, a typical data warehouse query is to retrieve something such as August sales. A summary in an Oracle database is called a **materialized view**.

Data Warehouse Architecture: with a Staging Area

You must clean and process your operational data before putting it into the warehouse, as shown in Figure 1–3. You can do this programmatically, although most data warehouses use a **staging area** instead. A staging area simplifies building summaries and general warehouse management. Figure 1–3 illustrates this typical architecture.

Figure 1–3 Architecture of a Data Warehouse with a Staging Area

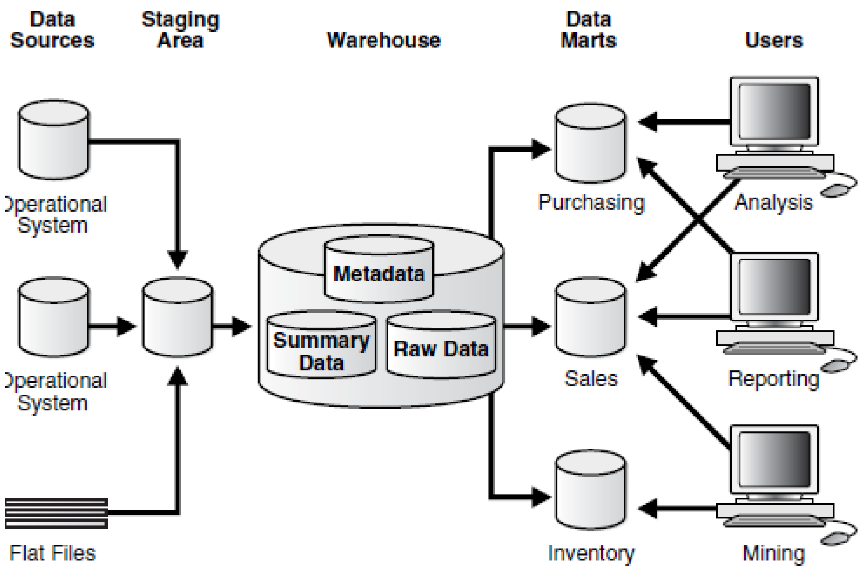


Data Warehouse Architecture: with a Staging Area and Data Marts

Although the architecture in Figure 1–3 is quite common, you may want to customize your warehouse's architecture for different groups within your organization. You can do this by adding **data marts**, which are systems designed for a particular line of business. Figure 1–4 illustrates an example where purchasing, sales, and inventories are separated. In this example, a financial analyst might want to analyze historical data for purchases and sales or mine historical data to make predictions about customer behavior.

OLTP systems (OnLine Transaction Processing)	Data warehousing systems
Holds current data	Holds historical data
Stores detailed data	Stores detailed, lightly, and highly summarized data
Data is dynamic	Data is largely static
Repetitive processing	Ad hoc, unstructured, and heuristic processing
High level of transaction throughput	Medium to low level of transaction throughput
Predictable pattern of usage	Unpredictable pattern of usage
Transaction-driven	Analysis driven
Application-oriented	Subject-oriented
Supports day-to-day decisions	Supports strategic decisions
Serves large number of clerical/operational users	Serves relatively low number of managerial users

Figure 1–4 Architecture of a Data Warehouse with a Staging Area and Data Marts



Note: Data marts are an important part of many data warehouses, but they are not the focus of this book.

20 Schema Modeling Techniques

The following topics provide information about schemas in a data warehouse:

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. The sh sample schema (the basis for most of the examples in this book) uses a star schema. However, there are other schema models that are commonly used for data warehouses. The most prevalent of these schema models is the **third normal form (3NF)** schema. Additionally, some data warehouse schemas are neither star schemas nor 3NF schemas, but instead share characteristics of both schemas; these are referred to as hybrid schema models.

The Oracle Database is designed to support all data warehouse schemas. Some features may be specific to one schema model (such as the star transformation feature, described in "Using Star Transformation" on page 20-4, which is specific to star schemas). However, the vast majority of Oracle's data warehousing features are equally applicable to star schemas, 3NF schemas, and hybrid schemas. Key data warehousing capabilities such as partitioning (including the rolling window load technique), parallelism, materialized views, and analytic SQL are implemented in all schema models.

The determination of which schema model should be used for a data warehouse should be based upon the requirements and preferences of the data warehouse project team. Comparing the merits of the alternative schema models is outside of the scope of this book; instead, this chapter briefly introduces each schema model and suggest how Oracle Database can be optimized for those environments.

Third Normal Form

Although this guide primarily uses star schemas in its examples, you can also use the third normal form for your data warehouse implementation.

Third normal form modeling is a classical relational-database modeling technique that minimizes data redundancy through normalization. When compared to a star schema, a 3NF schema typically has a larger number of tables due to this normalization process. For example, in Figure 20–1, orders and order items tables contain similar information as sales table in the star schema in Figure 20–2.

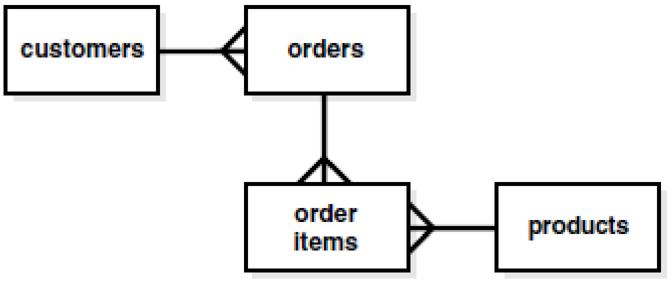
3NF schemas are typically chosen for large data warehouses, especially environments with significant data-loading requirements that are used to feed data marts and execute long-running queries.

The main advantages of 3NF schemas are that they:

- Provide a neutral schema design, independent of any application or data-usage considerations
- May require less data-transformation than more normalized schemas such as star schemas

Figure 20–1 presents a graphical representation of a third normal form schema.

Figure 20–1 Third Normal Form Schema



Optimizing Third Normal Form Queries

Queries on 3NF schemas are often very complex and involve a large number of tables. The performance of joins between large tables is thus a primary consideration when using 3NF schemas.

One particularly important feature for 3NF schemas is partition-wise joins. The largest tables in a 3NF schema should be partitioned to enable partition-wise joins. The most common partitioning technique in these environments is composite range-hash partitioning for the largest tables, with the most-common join key chosen as the hash-partitioning key.

Parallelism is often heavily utilized in 3NF environments, and parallelism should typically be enabled in these environments.

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. It is not mandatory to have any foreign keys on the fact table for star transformation to take effect.

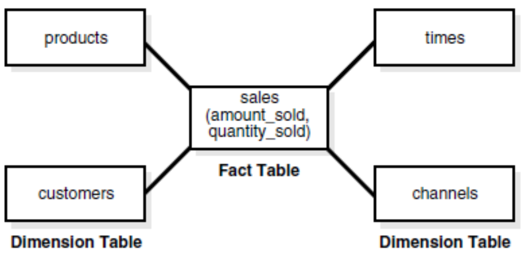
A typical fact table contains keys and measures. For example, in the sh sample schema, the fact table, sales, contain the measures quantity_sold, amount, and cost, and the keys cust_id, time_id, prod_id, channel_id, and promo_id. The dimension tables are customers, times, products, channels, and promotions.

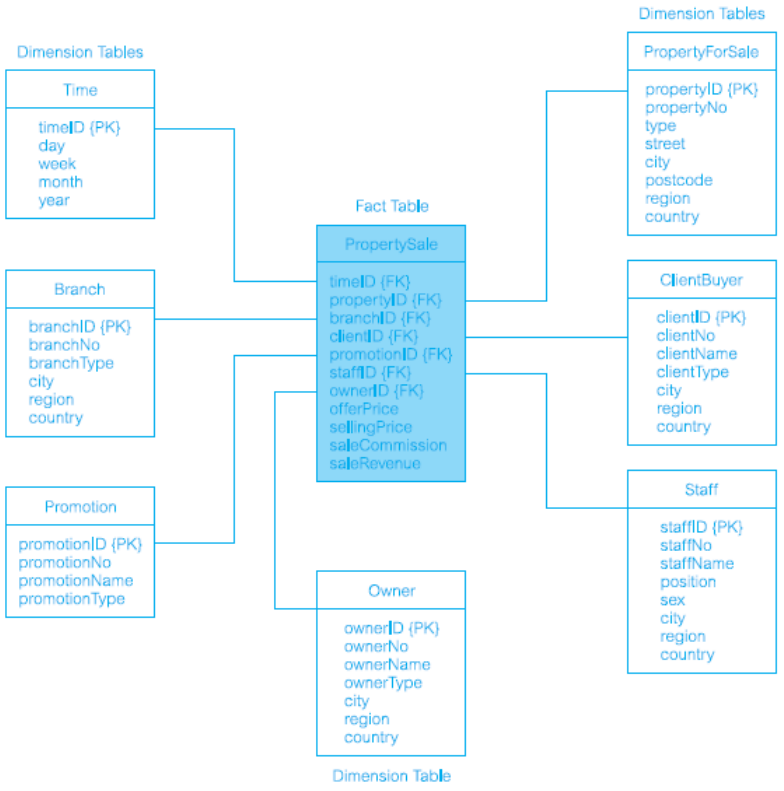
The products dimension table, for example, contains information about each product number that appears in the fact table. A star join is a primary key to foreign key join of the dimension tables to a fact table.
The main advantages of star schemas are that they:

- 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 20–2 presents a graphical representation of a star schema.
Figure 20–2 Star Schema

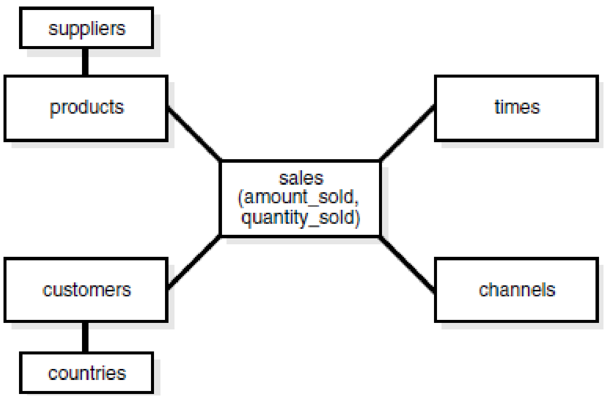


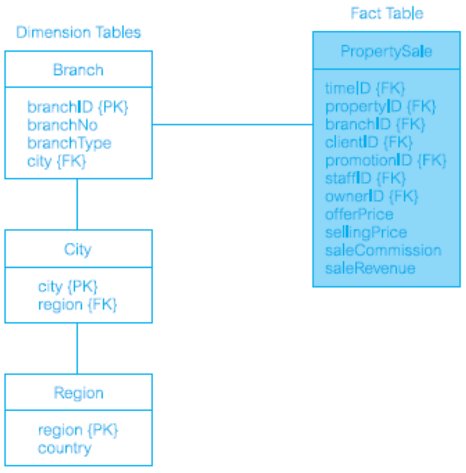


Snowflake Schemas

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 20–3 presents a graphical representation of a snowflake schema.

Figure 20–3 Snowflake Schema





Note: Oracle recommends you choose a star schema over a snowflake schema unless you have a clear reason not to.

The **multidimensional model** (also called the **dimensional model**)-involves two types of tables: dimension tables and fact tables. A **dimension table** consists of tuples of attributes of the dimension. A **fact table** can be thought of as having tuples, one per a recorded fact. This fact contains some measured or observed variable(s) and identifies it (them) with pointers to dimension tables. The fact table contains the data, and the dimensions identify each tuple in that data. Another way to look at a fact table is as an agglomerated view of the transaction data whereas each dimension table represents so-called "master data" that those transactions belonged to. In multidimensional database systems, the multidimensional model has been implemented as specialized software system known as a *multidimensional database*.

Figure 29.7 shows an example of a fact table that can be viewed from the perspective of multi-dimension tables. Two common multidimensional schemas are the star schema and the snowflake schema. The **star schema** consists of a fact table with a single table for each dimension (Figure 29.7). The **snowflake schema** is a variation on the star schema in which the dimensional tables from a star schema are organized into a hierarchy by normalizing them (Figure 29.8). A **fact constellation** is a set of fact tables that share some dimension tables. Figure 29.9 shows a fact constellation with two fact tables, business results and business forecast. These share the dimension table called product. Fact constellations limit the possible queries for the warehouse.

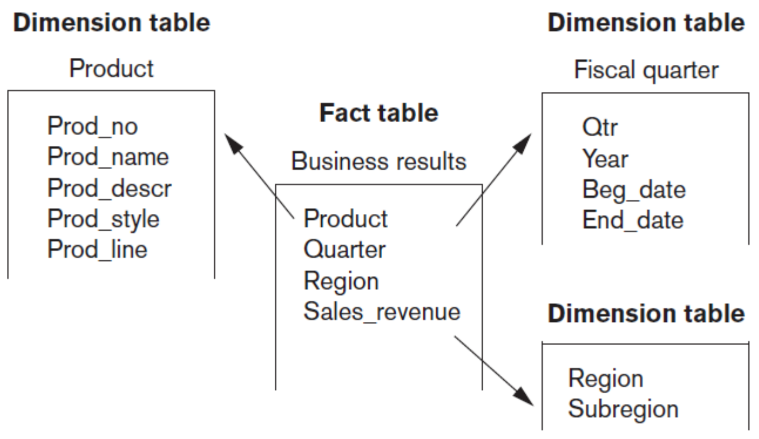


Figure 29.7 A star schema with fact and dimension tables.

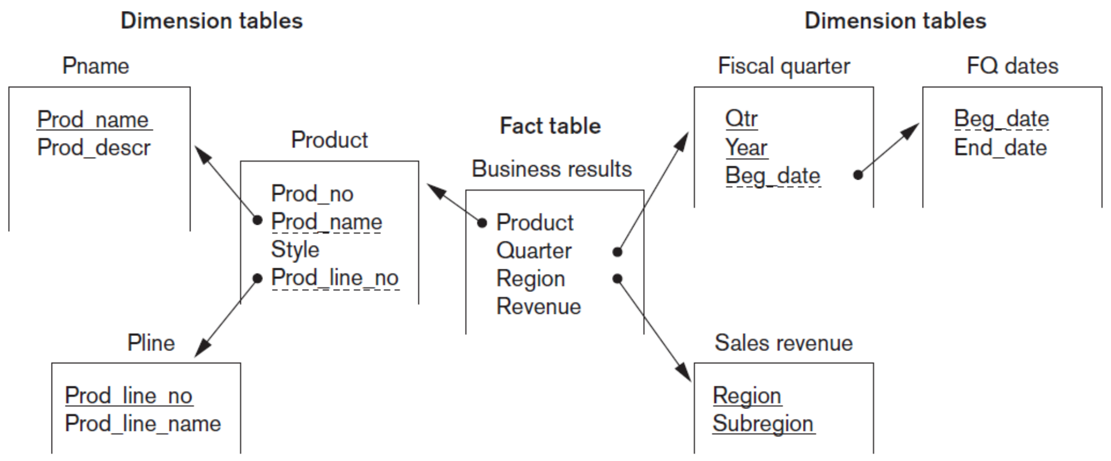


Figure 29.8 A snowflake schema.

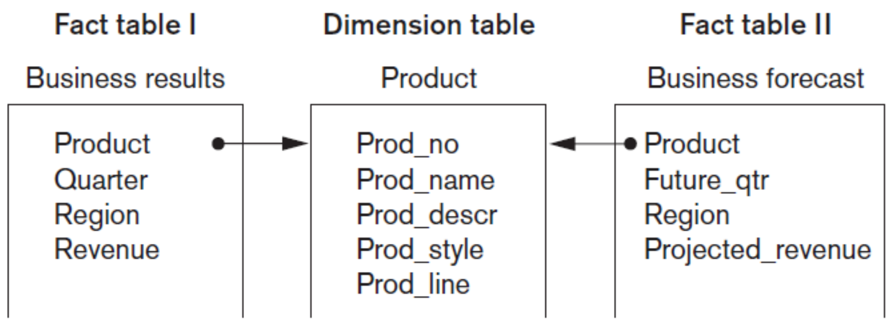
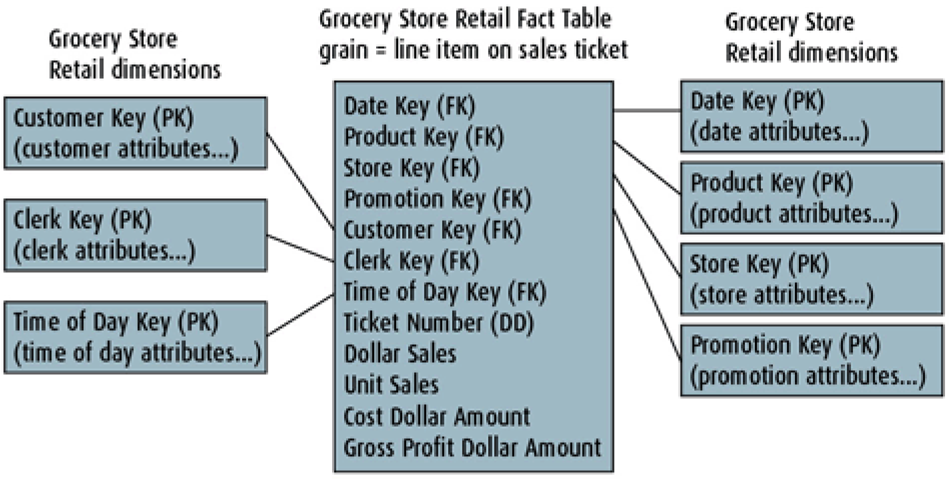
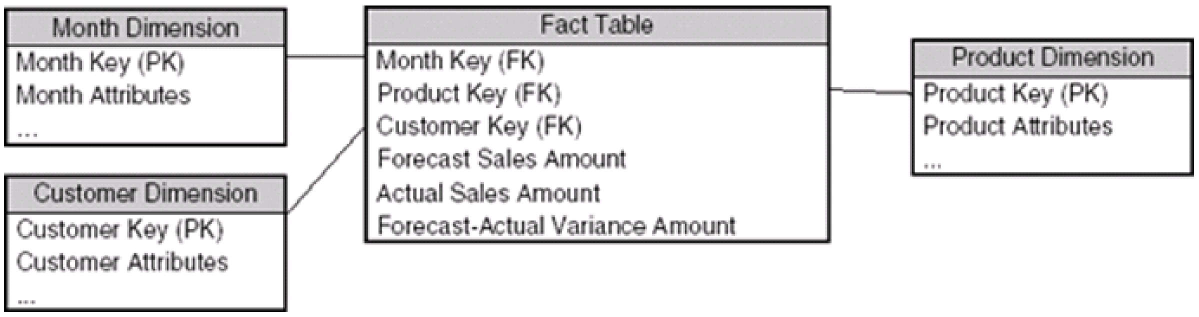


Figure 29.9 A fact constellation (takım yıldızı).


Grocery Store
Retail dimensions
Customer Key (PK)
(customer attributes...)
Clerk Key (PK)
(clerk attributes...)
Time of Day Key (PK)
(time of day attributes...)
Grocery Store Retail Fact Table
grain = line item on sales ticket
Date Key (FK)
Product Key (FK)
Store Key (FK)
Promotion Key (FK)
Customer Key (FK)
Clerk Key (FK)
Time of Day Key (FK)
Ticket Number (DD)
Dollar Sales
Unit Sales
Cost Dollar Amount
Gross Profit Dollar Amount
Grocery Store
Retail dimensions
Date Key (PK)
(date attributes...)
Product Key (PK)
(product attributes...)
Store Key (PK)
(store attributes...)
Promotion Key (PK)
(promotion attributes...)


Month Dimension
Month Key (PK)
Month Attributes
...
Customer Dimension
Customer Key (PK)
Customer Attributes
...
Fact Table
Month Key (FK)
Product Key (FK)
Customer Key (FK)
Forecast Sales Amount
Actual Sales Amount
Forecast-Actual Variance Amount
Product Dimension
Product Key (PK)
Product Attributes
...

.