Technology Guide

3

Data and Databases

TG3.1 Logical Data Organization

Just as there are many ways to structure business organizations, there are many ways to structure the data those organizations need. A manager's ability to use a database is highly dependent on how the database is structured logically and physically. The database management system (DBMS) separates the logical and physical views of the data, meaning that the programmer and end user do not have to know where and how the data are actually stored. In logically structuring a database, businesses need to consider the characteristics of the data and how the data will be accessed.

There are three basic models for logically structuring databases: hierarchical, network, and relational. Four additional models are multidimensional, object-oriented, small-footprint, and hypermedia. (The latter three of these models are explained in Section TG3.5.)

Using these various models, database designers can build logical or conceptual views of data that can then be physically implemented into virtually any database with any DBMS. Hierarchical, network, and object-oriented DBMSs usually tie related data together through linked lists. Relational and multidimensional DBMSs relate data through information contained in the data. In this section, we present the three basic models.

The hierarchical structure was developed because hierarchical relationships are commonly found in many traditional business organizations and processes. An example of a hierarchical database is shown in Figure TG3.1.

As illustrated, the hierarchical database model relates data by structuring data into an inverted "tree" in which records contain two elements:

1. A single root or master field, often called a key, which identifies the type, location, or ordering of the records
2. A variable number of subordinate fields that define the rest of the data within a record

As a rule, while all fields have only one "parent," each parent may have many "children."

The key advantages of the hierarchical approach are the speed and efficiency with which it can be searched for data. This speed is possible because so much of the database is eliminated in the search, with each "turn" going down the tree. As shown in Figure TG3.1, half of the records in the database (East Coast Sales) are eliminated once the search turns toward West Coast Sales, and two-thirds of the West Coast Sales are eliminated once the search turns toward stemware.

Finally, the explicit child/parent relationships in a hierarchical model mean that the integrity of the database is strictly maintained. Every child in a hierarchical database must belong to a parent, and if a parent is eliminated from the database, all its children automatically become children of the parent's parent. But the hierarchical approach does have some deficiencies.

In the hierarchical model, each relationship must be explicitly defined when the database is created. Each record in a hierarchical database can contain only one key field, and only one relationship is allowed between any two fields. This can create a

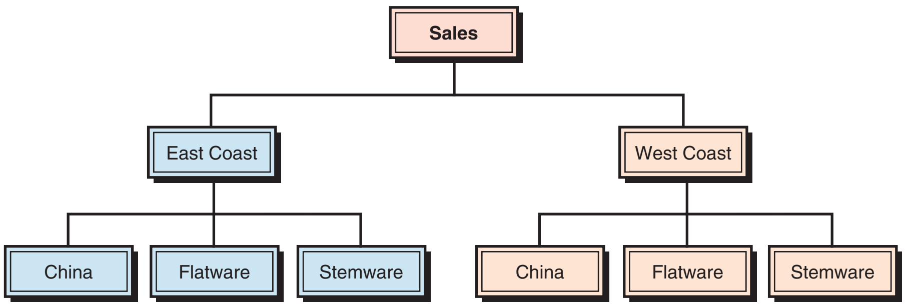


Figure TG3.1 Hierarchical database model.

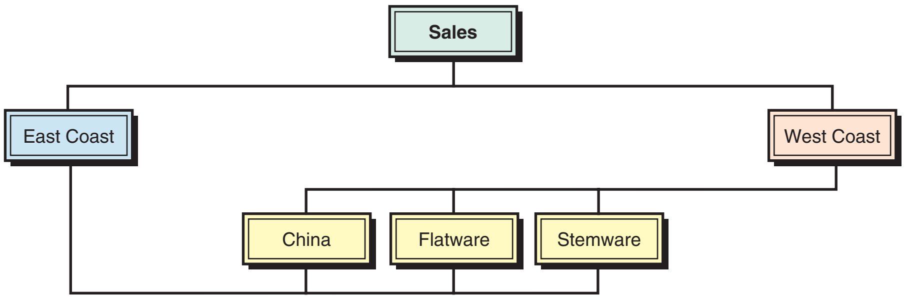


Figure TG3.2 Network database model.

problem because real-world data does not always conform to such a strict hierarchy. For example, in a matrix organization, an employee might report to more than one manager, a situation that would be awkward for a hierarchical structure to handle. Moreover, all data searches must originate at the top, or root, of the tree and work downward.

Another significant disadvantage of the hierarchical model is the fact that it is difficult to relate cousins in the tree. In the example shown in Figure TG3.1, there is no direct relationship between china sales on the East Coast and china sales on the West Coast. A comparison of company-wide china sales would entail two separate searches and then another step to combine the search results.

THE NETWORK DATABASE MODEL

The network database model creates relationships among data through a linked-list structure in which subordinated records (called *members*, not *children*) can be linked to more than one parent (called an owner). Similar to the hierarchical model, the network model uses explicit links, called pointers, to link subordinates and parents. That relationship is called a *set*.

Physically, pointers are storage addresses that contain the location of a related record. With the network approach, a member record can be linked to an owner record and, at the same time, can itself be an owner record linked to other sets of members (see Figure TG3.2). In this way, many-to-many relationships are possible with a network database model's significant advantage of the network model over the hierarchical model.

Compare Figure TG3.2 with Figure TG3.1. In Figure TG3.2, sales information about china, flatware, and stemware is in one subordinate, or member, location. Information about each has two parents or owners, East Coast and West Coast. The problem of getting a complete picture of nationwide china sales that exists with the hierarchical model does not occur with the network model. Moreover, searches for data do not have to start at a root—there may not even be a single root to a network—which gives much greater flexibility for data searches.

The network model essentially places no restrictions on the number of relationships or sets in which a field can be involved. This makes the model more consistent with real-world business relationships, where, for example, vendors have many customers and customers have many vendors. However, network databases are very complex. For every set, a pair of pointers must be maintained. As the number of sets or relationships increases, the overhead for processing queries becomes substantial. The network model is by far the most complicated type of database to design and implement.

THE RELATIONAL DATABASE MODEL

While most business organizations have been organized in a hierarchical fashion, most business data, especially accounting and financial data, have traditionally been organized into tables of columns and rows. Tables allow quick comparisons by row or column, and items are easy to retrieve by finding the point of intersection of a particular row and column. The relational database model is based on this simple concept of tables in order to capitalize on characteristics of rows and columns of data, which is consistent with real-world business situations.

In a relational database, the tables are called relations, and the model is based on the mathematical theory of sets and relations. In this model, each row of data is equivalent to a record, and each column of data is equivalent to a field. In the relational model terminology, a row is called a *tuple*, and a column is called an *attribute*. However, a relational database is not always one big table (usually called a flat file) consisting of all attributes and all tuples. That design would likely entail far too much data redundancy. Instead, a database is usually designed as a collection of several related tables.

There are some basic principles involved in creating a relational database. First, the order of tuples or attributes in a table is irrelevant, because their position relative to other tuples and attributes is irrelevant in finding data based on specific tuples and attributes. Second, each tuple must be uniquely identifiable by the data within the tuple—some sort of primary key data (for example, a Social Security number or employee number). Third, each table must have a unique identifier—the name of the relation. Fourth, there can be no duplicate attributes or tuples. Finally, there can be only one value in each row-column *cell* in a table.

In a relational database, three basic operations are used to develop useful sets of data: select, join, and project. The select operation creates a subset consisting of all records in the file that meet stated criteria. "Select" creates, in other words, a subset of rows that meet certain criteria. The join operation combines relational tables to provide the user with more information than is available in individual tables. The project operation creates a subset consisting of columns in a table, permitting the user to create new tables that contain only the information required.

A major advantage of the relational model is its conceptual simplicity and the ability to link records in a way that is not predefined (that is, they are not explicit as in the hierarchical and network models). This ability provides great flexibility, particularly for end users.

The relational or tabular model of data can be used in a variety of applications. Most people can easily visualize the relational model as a table, but the model does use some unfamiliar terminology. Consider the relational database example on Division, Title, and Employee shown in Figure TG3.3. The tables contain data about those three entities. Attributes or characteristics are code, name, description, title, age, and division code. The links among the data, and among tables, are implicit. They are not necessarily physically linked in a storage device but are implicitly linked by the Division Code (01).

This property of implicit links provides perhaps the strongest benefit of the relational model—flexibility in relating data. Unlike the hierarchical and network models, where the only links are those rigidly built into the design, all of the data within a table and between tables can be linked, related, and compared. This ability gives the relational model much more data independence than the hierarchical and network models. That is, the logical design of data into tables can be more independent of the physical implementation. This independence allows much more flexibility in implementing and modifying the logical design. Of course, as with all tables, an end user needs to know only two things: the identifier(s) of the tuple(s) to be searched and the desired attribute(s).

The relational model does have some disadvantages: Because large-scale databases may be composed of many interrelated tables, the overall design may be complex and therefore have slower search and access times (as compared to the hierarchical and network models). The slower search and access time may result in processing inefficiencies, which lead to an initial lack of acceptance of the relational model. These

Division

Code	*Name*
01	Stemware

Title

Code	*Description*
01	Director

Employee

Name	*Title Code*	*Division Code*	*Age*
Smith, A.	01	01	42

Figure TG3.3 Relational database model tables.

processing inefficiencies, however, are continually being reduced through improved database design and programming. Second, data integrity is not inherently a part of this model as with hierarchical and network models. Therefore, data integrity must be enforced with good design principles.

COMPARING THE DATABASE MODELS

The main advantage of the hierarchical and network database models is processing efficiency. The hierarchical and network structures are relatively easy for users to understand because they reflect the pattern of real-world business relationships. In addition, the hierarchical structure allows for data integrity to be easily maintained.

Hierarchical and network structures have several disadvantages, though. All of the access paths, directories, and indices must be specified in advance. Once specified, they are not easily changed without a major programming effort. Therefore, these designs have low flexibility. Hierarchical and network structures are programming intensive, time-consuming, difficult to install, and difficult to remedy if design errors occur. The two structures do not support ad hoc, English-language-like inquiries for information.

The advantages of relational DBMSs include high flexibility due to ad hoc queries, power to combine information from different sources, simplicity of design and maintenance, and the ability to add new data and records without disturbing existing applications.

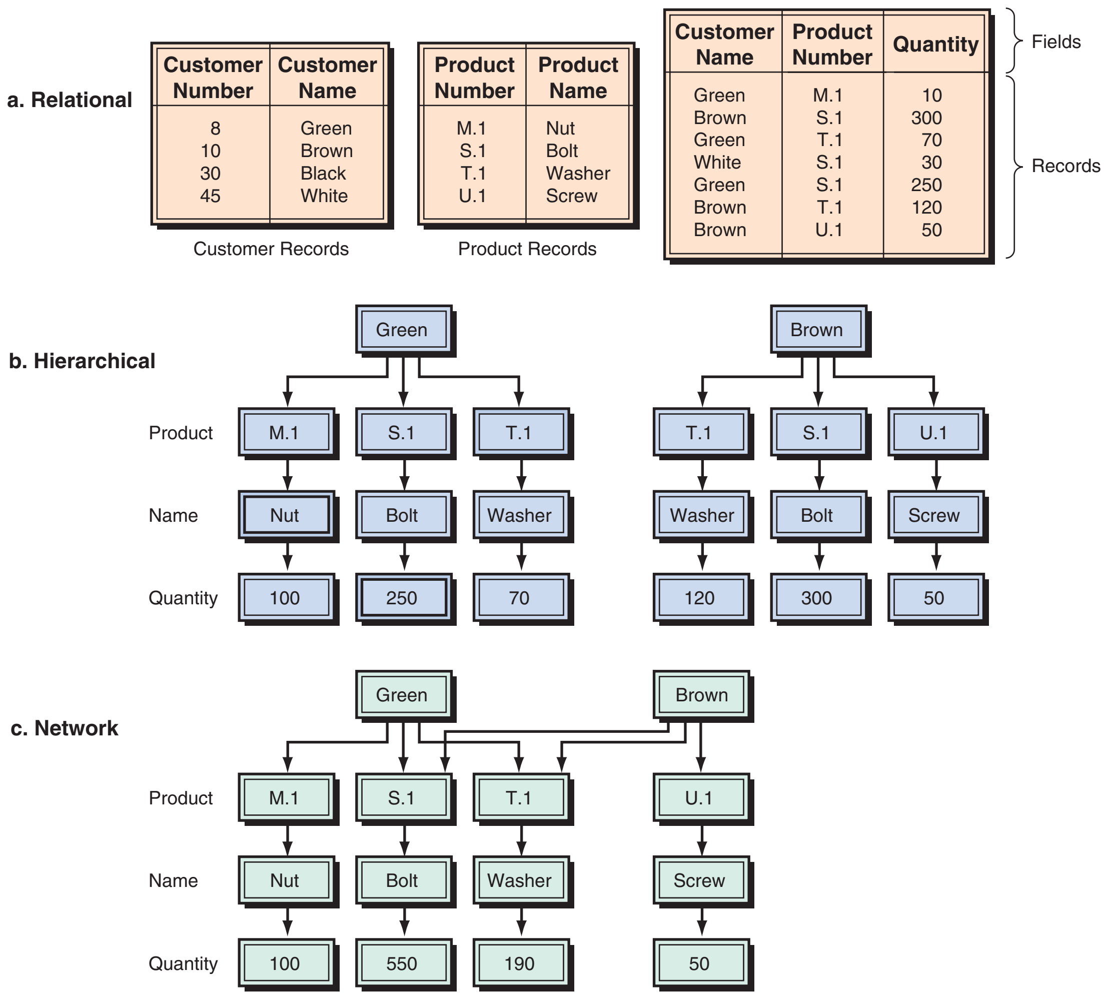
The disadvantages of relational DBMSs include their relatively low processing efficiency. These systems are somewhat slower because they typically require many accesses to the data stored on disk to carry out the select, join, and project commands. Relational systems do not have the large number of pointers carried by hierarchical systems, which speed search and retrieval. Further, large relational databases may be designed to have some data redundancy in order to make retrieval of data more efficient. The same data element may be stored in multiple tables. Special arrangements are necessary to ensure that all copies of the same data element are updated together. Table TG3.1 summarizes the advantages and disadvantages of the three common database models. Figure TG3.4 is a comparison of the three data models.

XML DATABASES

Extensible Markup Language (XML) databases can store whole documents in their native XML format. Such a database makes an archive easier to search by title, author, keywords, or other attributes. Relational databases, in contrast, either convert documents into relational data (stored in tables) or treat them as indiscriminate binary large objects (BLOBs), but it is difficult to find and to retrieve the part of the BLOB that you want. XML database products include Software AG's Tamino XML Database and Ipedo's XML Database System. X Query is the XML query language used in these products, which can query a large set of documents based on the name of an author, date filed, subject, or keywords in the document.

TABLE TG3.1 Advantages and Disadvantages of Logical Data Models

Model	Advantages	Disadvantages
Hierarchical database	Searching is fast and efficient.	Access to data is predefined by exclusively hierarchical relationships, predetermined by administrator. Limited search/query flexibility. Not all data is naturally hierarchical.
Network database	Many more relationships can be defined. There is greater speed and efficiency than with relational database models.	This is the most complicated model to design, implement, and maintain. Greater query flexibility than with hierarchical model, but less than with relational mode.
Relational database	Conceptual simplicity; there are no predefined relationships among data. High flexibility in ad hoc querying. New data and records can be added easily.	Processing efficiency and speed are lower. Data redundancy is common, requiring additional maintenance.



a. Relational

Customer Number	Customer Name
8	Green
10	Brown
30	Black
45	White

Customer Records

Product Number	Product Name
M.1	Nut
S.1	Bolt
T.1	Washer
U.1	Screw

Product Records

Customer Name	Product Number	Quantity
Green	M.1	10
Brown	S.1	300
Green	T.1	70
White	S.1	30
Green	S.1	250
Brown	T.1	120
Brown	U.1	50

Figure TG3.4 Database structures.

TG3.2 Commercial Databases

There are a number of commercial databases. The common commercial databases, such as Oracle, Microsoft's SQL Server, and IBM's DB2 server, include many features that people have come to rely on to make their database servers "enterprise worthy."

ORACLE 11G

Oracle Database 11g Enterprise Edition (*oracle.com/database/*) is the best-selling commercial database used by organizations to support high volume for transaction processing, query intensive data, and data warehouse operations.

Oracle is a scalable database running on all hardware configurations that can manage large amounts of information with high levels of security. Benefits are:

- Maximum performance and availability—automatically monitors the entire database environment and proactively resolves issues before they turn into emergencies.
- Elevated administrator productivity—gives administrators the tools they need to manage more databases.
- Failures from human error eliminated—increases control of IT environments by addressing the #1 cause of unplanned downtime through extensive out-of-box automation, configuration, and change management capabilities.

Oracle 11g is designed for grid computing to reduce infrastructure costs through the use of shared hardware pools and self-management capabilities. Oracle 11g runs on all operating systems, including Windows, Linux, and UNIX, and supports the smallest to largest processors.

MICROSOFT SQL SERVER

Microsoft's SQL Server is a highly scalable database that runs on the Microsoft OS server platform. SQL Server is a comprehensive database platform providing enterprise-class data management with integrated business intelligence tools. It provides the highest levels of security, reliability, and scalability for business-critical applications.

SQL Server delivers on Microsoft's Data Platform vision by helping enterprises manage any data, any place, any time. It enables companies to store data from structured, semistructured, and unstructured documents, such as images and music, directly within the database. SQL Server delivers a rich set of integrated services that enable users to do more with their data, such as query, search, synchronize, report, and analyze. Data can be stored and accessed in the largest servers within the data center all the way down to desktops and mobile devices, enabling users to have control over their data no matter where the data is stored.

IBM'S DB2 UNIVERSAL DATABASE

DB2 Universal Database (UDB) is IBM's database management system, which delivers a flexible and cost-effective database platform to build robust on-demand business applications. DB2 UDB leverages resources with broad support for open standards and popular development platforms such as J2EE and Microsoft .NET. The DB2 UDB family also includes solutions tailored for specific needs such as business intelligence and advanced tooling.

DB2 has functionality for the XML format, or directly storable as XML format, rather than in relational data tables. DB2 V9.1 introduces the first hybrid data server for the industry, serving data from both pure relational and pure XML structures. This technology delivers unprecedented application performance and development time/cost savings that makes XML data cost effective for the first time, enabling greater business insight faster at lower cost.

MYSQL

MySQL is a multithreaded, multi-user, SQL Database Management System (DBMS) with more than six million installations. MySQL AB makes MySQL available as free software under the GNU General Public License (GPL), but it is also dual-licensed under traditional proprietary licensing arrangements for cases where the intended use is incompatible with the GPL. Prior to creating MySQL, the developers used mSQL to connect to their own low-level data structure and discovered that it lacked the features and speed they wanted, and so they decided to write their own front-end instead. Thus began the life of MySQL as the product. MySQL works on many different platforms.

TG3.3 Database Queries

Query languages are computer languages used to make queries into databases and information systems. Query languages can be classified according to whether they are database query languages or information retrieval query languages.

SQL (commonly expanded to Structured Query Language) is the most popular computer language used to create, modify, retrieve, and manipulate data from relational database management systems. The language has evolved beyond its original purpose to support object-relational database management systems. It is an ANSI/ ISO standard.

LACK OF SQL STANDARD

Although both ANSI and ISO define SQL, there are many extensions to and variations on the version of the language defined by these standards bodies. SQL code can rarely be ported between database systems without major modifications. There are several reasons for this lack of portability between database systems:

- Most databases do not implement the entire SQL standard.
- SQL standard does not specify database behavior in several important areas (e.g., indexes).
- SQL standard's specification of the semantics of language constructs is less well-defined, leading to areas of ambiguity.
- Lack of compatibility between database systems is intentional in order to ensure vendor lock-in.

SQL keywords fall into several groups: data retrieval, data manipulation, data transaction, data definition, and data control. Here we will look at the use of SQL for retrieving data from a database.

DATA RETRIEVAL

The most frequently used operation in transactional databases is the data retrieval operation. When restricted to data retrieval commands, SQL acts as a declarative language.

- SELECT is used to retrieve zero or more rows from one or more tables in a database.
- FROM is used to indicate from which tables the data are to be taken, as well as how the tables JOIN to each other.
- WHERE is used to identify which rows to be retrieved or applied to GROUP BY. WHERE is evaluated before the GROUP BY.
- GROUP BY is used to combine rows with related values into elements of a smaller set of rows.
- HAVING is used to identify which of the "combined rows" (combined rows are produced when the query has a GROUP BY keyword or when the SELECT part contains aggregates) are to be retrieved. HAVING acts much like a WHERE, but it operates on the results of the GROUP BY and hence can use aggregate functions.
- ORDER BY is used to identify which columns are used to sort the resulting data.

Example 1.

```
SELECT * FROM cds
WHERE price . $20
ORDER BY artist
```

This is an example that could be used to get a list of expensive compact disks. It retrieves the records from the cds table that have a price field which is greater than $20.00. The result is sorted alphabetically by artist. The asterisk (*) means to show all columns of the cds table. Alternatively, specific columns could be named.

Example 2.

```
SELECT cds.title, count(*) AS artists
FROM cds
INNER_JOIN artists ON cds.cdnumber 5 cdartist.cdnumber
GROUP BY cds.title
```

Example 2 shows both the use of multiple tables in a join and aggregation (grouping). This example shows how many artists there are per CD.

OTHER USES OF SQL

Data Manipulation. First there are the standard Data Manipulation Language (DML) elements. DML is the subset of the language used to add, update, and delete data.

- INSERT is used to add zero or more rows (formally tuples) to an existing table.
- UPDATE is used to modify the values of a set of existing table rows.

• MERGE is used to combine the data of multiple tables. It is something of a combination of the INSERT and UPDATE elements.

• TRUNCATE deletes all data from a table (nonstandard but common SQL command).

• DELETE removes zero or more existing rows from a table.

Data Transaction. Transaction, if available, can be used to wrap around the DML operations.

• START TRANSACTION (or BEGIN WORK, depending on SQL dialect) can be used to mark the start of a database transaction, which either completes completely or not at all.

• COMMIT causes all data changes in a transaction to be made permanent.

• ROLLBACK causes all data changes since the last COMMIT or ROLLBACK to be discarded, so that the state of the data is "rolled back" to the way it was prior to those changes being requested.

Data Definition. The second group of keywords is the Data Definition Language (DDL). DDL allows the user to define new tables and associated elements. Most commercial SQL databases have proprietary extensions in their DDL, which allow control over nonstandard features of the database system. The most basic items of DDL are the CREATE and DROP commands.

• CREATE causes an object (a table, for example) to be created within the database.

• DROP causes an existing object within the database to be deleted, usually irretrievably.

Data Control. The third group of SQL keywords is the Data Control Language (DCL). DCL handles the authorization aspects of data and permits the user to control who has access to see or manipulate data within the database.

• GRANT authorizes one or more users to perform an operation or a set of operations on an object.

• REVOKE removes or restricts the capability of a user to perform an operation or a set of operations.

TG3.4 Database Normalization

In relational databases, normalization is the restructuring of database fields and tables. It eliminates redundancy, organizes data efficiently, reduces the potential for anomalies during data operations, and improves data consistency.

A nonnormalized database can suffer from data anomalies:

• A nonnormalized database may store data representing a particular referent in multiple locations, called an update anomaly, yielding inconsistent data.

• A nonnormalized database may have inappropriate dependencies (i.e., relationships between data with no functional dependencies). Adding data to such a database may require first adding the unrelated dependency, called insertion anomalies.

• Similarly, such dependencies in nonnormalized databases can hinder deletion, called deletion anomalies.

• A nonnormalized database can suffer from inferential integrity problems where data cannot be added into multiple tables simultaneously.

Normalized databases have a design that reflects the true dependencies between tracked quantities, allowing quick updates to data with little risk of introducing inconsistencies. Instead of attempting to lump all information into one table, data

TABLE TG3.2 Normal Forms

Normal Form	Characteristics
First normal form (1NF)—all fields must contain single values only, derived from the data model.	Ensures that each table has a primary key—a minimal set of attributes that can uniquely identify a record. Each attribute must contain a single value, not a set of values. Eliminate repeating groups (categories of data that would seem to be required a different number of times on different records) by defining key and non-key attributes.
Second normal form (2NF) requires that part of the primary key may not determine a non-key field.	The database must meet all of the requirements of the first normal form. Data stored in a table with a composite primary key must not be dependent on only part of the table's primary key. Data that are redundantly duplicated across multiple rows of a table are moved out to a separate table.
Third normal form (3NF) requires that a non-key field may not determine another non-key field.	The database must meet all of the requirements of the second normal form. Data stored in a table must be dependent only on the primary key, not on any other field in the table. Any field that is dependent not only on the primary key but also on another field is moved out to a separate table.

is spread out logically into many tables. Normalizing the data is decomposing a single relation into a set of smaller relations which satisfy the constraints of the original relation. Redundancy can be solved by decomposing the tables. However, certain new problems are caused by decomposition. Normalization helps us to make a conscious decision to avoid redundancy, keeping the pros and cons in mind.

Several levels of normalization exist, which build upon each other, addressing increasingly specialized and complex normalization problems. Table TG3.2 shows the different normal forms and their characteristics.

TG3.5 Database Models

Many of today's applications require database capabilities that can store, retrieve, and process diverse media, not just text and numbers. Full-motion video, voice, photos, and drawings cannot be handled effectively or efficiently by hierarchical, network, or relational databases and the DBMS. For multimedia and other complex data we use special data models.

The most common database models are the following:

- Multidimensional database. This is an additional database that enables end users to quickly retrieve and present complex data that involve many dimensions.
- Deductive databases. Hierarchical, network, and relational DBMSs have been used for decades to facilitate data access by users. Users, of course, must understand what they are looking for, the database they are looking at, and at least something about the information sought (such as a key and some field or attribute about a record). This approach, however, may not be adequate for some knowledge-based applications that require deductive reasoning for searches. As a result, there is interest in what is called deductive database systems.
- Multimedia and hypermedia databases. These are analogous to contemporary databases for textual and numeric data; however, they have been tailored to meet the special requirements of dealing with different types of media materials.
- Small-footprint databases. Small-footprint databases enable organizations to put certain types of data in the field where the workers are located. Whereas laptops were

once the only portable machines capable of running a database, advances in technology (e.g., more powerful CPUs and increased memory at lower cost) are enabling handheld devices and smartphones to run some form of an SQL database and to synchronize that mobile database with a central database at headquarters. The name comes from the fact that the engines running these databases (e.g., Access) typically are small, and thus the databases do not use a lot of space in memory.

Small-footprint databases have replication mechanisms that take into account the occasionally connected nature of laptops and handhelds that are programmed to resolve replication conflicts among mobile users and that ensure that data synchronization will survive a low-quality wireless or modem connection. Small-footprint database technology also runs on PDAs (such as those from Palm or Psion) and can be embedded in specialty devices and appliances—like a barcode scanner or medical tool.

TG3.6 Database Management

Database management, outside of purely technical hardware and software considerations, consists primarily of two functions: database design and implementation, and database administration.

In designing and implementing databases, specialists should carefully consider the individual needs of all existing and potential users in order to design and implement a database that optimizes both processing efficiency and user effectiveness. The process usually starts by analysis of what information each user (or group of users) needs and then production of logical views for each. These logical views are analyzed as a whole for similarities that can lead to simplification, and then are related so that a single, cohesive logical database can be formed from all of the parts. This logical database is implemented with a particular DBMS in a specific hardware system.

Database administrators are IT specialists responsible for the data as well as for ensuring that the database fulfills the users' business needs, in terms of functionality. User needs, like business in general, do not remain constant. As the business environment changes, and organizational goals and structures react, the database that the firm depends on must also change to remain effective. The computer hardware on which the DBMS software is installed must change to meet changing environments or to take advantage of new technology. This brings accompanying constraints and/or new opportunities for the DBMS processing performance.

Further, database administrators need to ensure the reliability of databases under their care by managing daily operations, including planning for emergency contingencies by providing backup data and systems to ensure minimal loss of data in the event of a disaster. Security is always a data administration concern when there are multiple accesses to databases that contain all of the corporate data. Administrators must balance the business advantages of widespread access with the threat of corporate espionage, sabotage by disgruntled employees, and database damage due to negligence.

Database administrators also play a significant role in training users about what data are available and how to access them. Finally, administrators are responsible for ensuring that the data contained in the database is accurate, reliable, verifiable, complete, timely, and relevant—a daunting task at best. Otherwise-brilliant business decisions based on wrong information can be disastrous in a highly competitive market.

One of the elements of contingency planning is data backup, which is of critical importance to any IT users. Tapes and diskettes are popular data backup media because they are relatively cheap. There are two main methods of backup, full backup and incremental backup. Full backup involves keeping a duplicate of the entire database; incremental backup involves keeping just the additional or updated data each time the database is backed up. The incremental backup method is more efficient because of shorter backup time, but it is not as safe as full backup since loss of any one medium may make recovery impossible.

TG3.7 IP-Based Storage, SANs, and NAS

Storage connected to servers over IP (Internet Protocol) networks, also known as IP storage, enables servers to connect to SCSI (small computer system interface) storage devices and treat them as if they were directly attached to the server, regardless of the location. IP storage is a transport mechanism that seeks to solve the problem of sending storage data over a regular network in the block format it prefers rather than the file format generally used. IP storage can save money by allowing a company to use its existing network infrastructure for storage. We need to describe what IP storage attempts to replace in order to understand why it is an improvement.

Traditionally, data management tasks are handled by tape and disk devices directly attached to each server in the network, called direct attached storage (DAS). Network storage devices are optimized for data management tasks.

These devices are attached to the corporate network and can be accessed from networked applications throughout the enterprise. However, sending storage data over the company network can seriously slow network speeds, which will affect applications such as e-mail and Internet access. Enterprises have transitioned much of their direct attached storage (DAS) to networked storage.

Network attached storage (NAS) is an IP-based and Ethernet-based network storage architecture replacing the general-purpose file server with a server running a custom operating system that is optimized for data processing and management. The optimized operating system improves file server performance and supports features of RAID, caching, clustering, and so forth.

A storage area network (SAN) can solve problems associated with sending storage data over regular networks by building a separate, dedicated, high-speed network just for storage devices, servers, and backup systems. It can handle the heavy bandwidth demands of storage data, and segregates storage traffic to a network built specifically for storage needs. Communication between the application server and the storage devices is done using a low-level block-based SCSI-3 protocol. SAN technology is implemented using either a direct point-to-point connection or a network switch to a data storage farm. It is both expensive and complicated to construct. A SAN requires specially trained management personnel and uses relatively expensive hardware that may be incompatible among vendors. See Table TG3.3.

TABLE TG3.3 Pros and Cons of SANs and NAS

	Pros	Cons
SANs	Off-loads storage traffic from existing network	Expensive—requires new subnetwork
	Flexible design improves reliability	Manages data in blocks, not files, so it requires specialized software
	Highly scalable equipment design	Requires fiber channel networking skills
NAS	Uses existing network infrastructure	Slower—network protocols are not streamlined for storage
	Manages data as files	Loads already burdened network with storage data, including backup
	Easy to install and use	Doesn't scale up easily