Technology Guide 3

Data and Databases

T3.1 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 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 emerging: *multidimensional, object-oriented, small-footprint,* and *hypermedia.* (The latter three of these emerging models are explained in Section T3.5). 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 will present the three basic models. (Others are described in Chapter 3.)

THE HIERARCHICAL DATABASE MODEL

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 T3.1.

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 defines 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 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 T3.1, half 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,

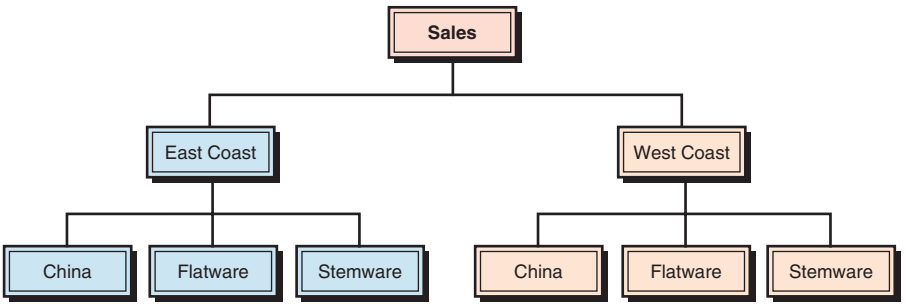

Figure T3.1 Hierarchical database model.

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 problem because real-world data do 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 T3.1, 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 T3.2). In this way, many-to-many relationships are possible with a network database model—a significant advantage of the network model over the hierarchical model.

Compare Figure T3.2 with Figure T3.1. In Figure T3.2, 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.

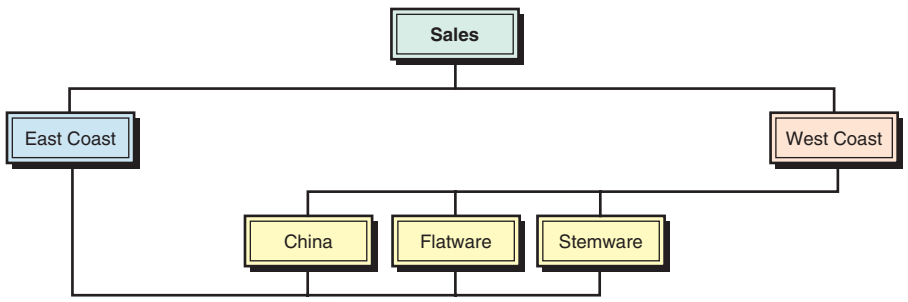

Figure T3.2 Network database model.

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 information required.

A major advantages 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 East Coast managers shown in Figure T3.3. The table contains data about the entity called East Coast managers. Attributes or characteristics about the entity are name, title, age, and division. The tuples, or occurrences of the entity, are the two records on A. Smith and W. Jones. The links among the data, and among tables, are implicit, as they are not necessarily physically linked in a storage device but are implicitly linked by the design of the tables into rows and columns.

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 the data

Division

Code	*Name*
01	Stemware

Title

Code	*Description*
01	Director

Employee

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

Figure T3.3 Relational database model tables.

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 led to an initial lack of acceptance of the relational model. These 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, it must be enforced with good design principles.

Object-Relational Database Systems. *Object-relational database* products are replacing purely relational databases. Object-relational database management systems (ORDBMSs) have some of the capabilities of object-oriented database systems as well as additional unique capabilities.

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 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 T3.1 summarizes the advantages and disadvantages of the three common database models.

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. A visual comparison of the three models is shown in Figure T3.4. The lines with arrows in the relational models show the duplication of information.

TABLE T3.1 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 are 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.

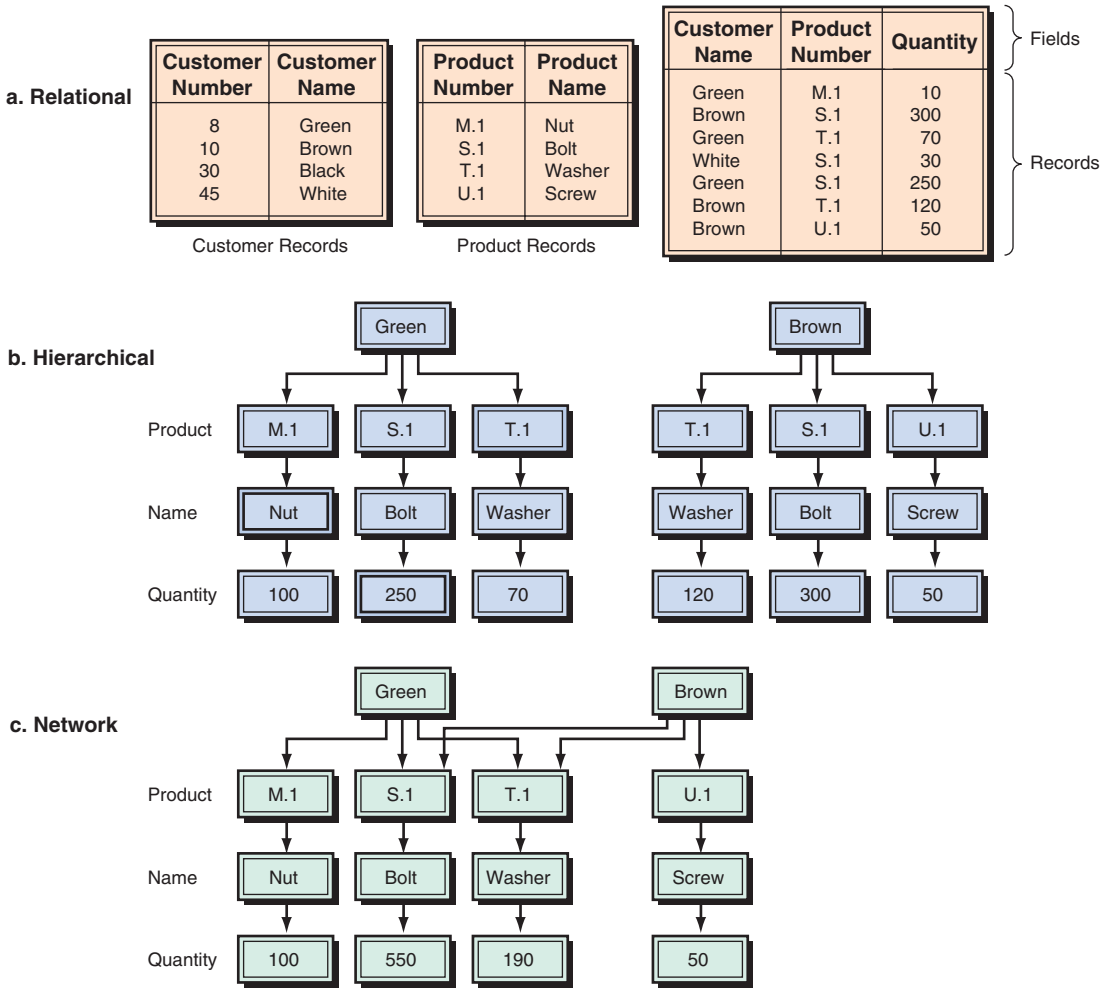

Figure T3.4 Database structures.

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 filled, subject, or keywords in the document.

T3.2 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." These features include advanced database storage, data management tools, information replication, and tools to back up corporate information.

During the past ten years, the open source community has improved the quality of its software, making it more enterprise worthy. As a result, enterprises have shown an interest in migrating from proprietary, commercial software to open-source software in recent years, for example, MySQL.

ORACLE 10g

Oracle Database 10g Enterprise Edition 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.

Oracle 10g is the first database designed for *grid computing* to reduce infrastructure costs through the use of shared hardware pools and self-management capabilities. Oracle 10g runs on all operating systems, including Windows, Linux, and UNIX, and supports the smallest to largest processors.

Access to data can be made via standard interfaces such as SQL, JDBC, SQLJ, ODBC, OLE DB, ODP.NET, and SQL/XML. Oracle 10g can store up to 8 exabytes of data in a single database. Data stored in an existing Oracle database can be transported or uploaded to/from all other commercial databases.

Oracle 10i runs existing packaged or custom applications faster than the fastest mainframe in a cluster of low-cost servers. It extends database manageability via automated diagnostics, optimization, change management, and configuration management. The 10g enables tables and indexes to be split into smaller, more manageable components for improved performance and availability and content management capabilities for enterprise deployment.

Oracle 10g provides enterprisewide records management solution by transforming raw data it into high-quality information optimized for business reporting and analytics and then extracts information efficiently from very large databases for integrated business intelligence applications. Additionally, Oracle 10g has an industrial-strength calculation engine to support the entire spectrum of advanced analytical applications in the Oracle data warehouse. 10g also manages geographic and location data as native types in your database.

TABLE T3.2 SQL Server Capabilities—Enterprise Data Management

Technology	Capabilities
High Availability	Failover clustering and database mirroring technology in SQL Server 2005 will enable enterprises to deliver highly reliable, available applications to employees, customers, and partners.
Management Tools	SQL Server 2005 introduces an integrated suite of management tools and management application programming interfaces (APIs) to provide ease of use, manageability, and support for operating large-scale SQL Server deployments.
Security Enhancements	SQL Server 2005 has been designed to help provide the highest level of security for enterprise data through features such as database encryption, more secure default settings, password policy enforcement, granular permissions control, and an enhanced security model.
Scalability	Scalability advancements in SQL Server 2005 include table partitioning, replication enhancements, and 64-bit support.

MICROSOFT SQL SERVER 2005

Microsoft's SQL Server 2005 is a highly scalable database that runs on the Microsoft operation system server platform (MS Server 2003). SQL Server 2005 is a comprehensive database platform providing enterprise-class data management with integrated business intelligence tools. The SQL Server 2005 database engine provides more secure, reliable storage for both relational and structured data, enabling the building and management of data warehouse applications.

SQL Server 2005 provides the closest integration with Microsoft Visual Studio, Microsoft Office System, and a suite of new SQL Server development tools, including the Business Intelligence Development Studio. Table T3.2 shows the capabilities of the SQL Server 2005 product to include high availability, management tools, security enhancements and scalability.

Winter Corporation's 2005 TopTen Survey solidifies the growing position of SQL Server as the data management platform of the world's largest and most heavily used databases. SQL Server customers showed up in greater numbers with increases in database volumes and record counts, over 2003 data. Forty-three survey entries, or 25 percent of all survey entries, of more than 1 terabyte use SQL Server 2005. Table T3.3 shows the results of the SQL Server 2000 and 2005 products.

TABLE T3.3 Comparison of SQLServer 2000 and SQLServer 2005 Capabilities

Comparing Winter Corporation's 2003 and 2005 TopTen Surveys

	2005 Results	2003 Results
Number of terabytes + entries (>1 terabyte)	43	15
Number of terabytes − size entries (>0.5 terabyte)	50	19
Number of data warehousing entries	21	4
Number of online transaction processing (OLTP) entries	22	13
Number of OLTP entries in TopTen categories for all platforms	6	1
Number of data warehousing entries in TopTen categories for all platforms	2	0
Number of data warehousing entries in database size TopTen for Windows	7	1
SQL Server is largest entry in database size TopTen for Windows	Yes	No

TABLE T3.4 DB2 Enterprise 9 Capabilities

Reduces storage needs by up to 80% using industry unique data compression capabilities
Powers the next generation of agile SOA applications with pureXML™
Reduces risk of unauthorized access with an innovative label-based security model
Avoids unexpected performance problems with proactive query management and predictive analysis
Minimizes costs with adaptive memory allocation, automatic storage management and more
Scales with your preferred architecture: single-server, clusters of servers, or both with database partitioning and table partitioning
Maximizes data availability by reducing planned and unplanned downtime

IBM DB2 UDB 9

DB2 Universal Database is IBM's database management system that 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 like J2EE and Microsoft .NET. The DB2 UDB family also includes solutions tailored for specific needs like business intelligence and advanced tooling. Table T3.4 shows the capabilities of DB2 Enterprise 9.

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 pureXML 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.

An enhanced set of application development tools simplify database application development and ease application deployment, including a new Developer Workbench, enhanced functionality for Visual Studio 2005, and XML and XQuery support. DB2 V9.1 also introduced a unified debugger, rapid application deployment with a lightweight runtime client that works across different DB2 data servers.

DB2 V9.1 improves data availability with online integrity processing and improved recovery capabilities. Large database management is further improved with table partitioning that allows for larger tables, facilitates fast roll-in and roll-out of data in a warehouse, improves query performance, and reduces the administration time by allowing administrative tasks on individual data partitions. Performance is further improved with statistical views, faster data loading capabilities, and Materialized Query Table (MQT) enhancements.

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 they also dual-license it 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 decided to write their own front-end instead. Thus began the life of MySQL as the product.

MySQL works on many different platforms—including AIX, BSDi, FreeBSD, HP-UX, GNU/Linux, Mac OS X, NetBSD, Novell NetWare, OpenBSD, OS/2 Warp, QNX, SGI IRIX, Solaris, SunOS, SCO OpenServer, SCO UnixWare, Tru64, Windows 95, Windows 98, Windows ME, Windows NT, Windows 2000, Windows XP,

TABLE T3.5 MySQL 5.0 Capabilities
A broad subset of ANSI SQL 99, as well as extensions
Cross-platform support
Stored procedures
Triggers
Cursors
Updatable views
True VARCHAR support
INFORMATION_SCHEMA
Strict mode
X/Open XA distributed transaction processing (DTP) support; two-phase commit as part of this, using Oracle's InnoDB engine
Independent storage engines (MyISAM for read speed, InnoDB for transactions and referential integrity, Archive for storing historical data in little space)
Transactions with the InnoDB, BDB, and Cluster storage engines; savepoints with InnoDB
SSL support
Query caching
Sub-SELECTs (i.e., nested SELECTs)
Replication with one master per slave, many slaves per master, no automatic support for multiple masters per slave.
Full-text indexing and searching using MyISAM engine
Embedded database library
Full Unicode support
ACID compliance using the InnoDB, BDB, and Cluster engines
Shared-nothing clustering through MySQL Cluster

and more recent versions of Windows. A port of MySQL to OpenVMS is also available. Table T3.5 shows the capabilities of MySQL 5.0.

The following features are implemented by MySQL, but not by some other RDBMSs:

- Multiple storage engines (MyISAM, Merge, InnoDB, BDB, Memory/heap, Cluster, Federated, Archive, CSV, Blackhole and Example in 5.x), letting you choose the one which is most effective for each table in the application
- Commit grouping, gathering multiple transactions from multiple connections together to increase the number of commits per second

T3.3 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 SQL is defined by both ANSI and ISO, 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 = 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..

T3.4 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.

TABLE T3.6 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 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 the requirements of the second normal form. Data stored in a table must be dependent only on the primary key, and 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.

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 are 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 T3.6 shows the different normal forms and their characteristics.

T3.5 Emerging Database Models

Many of today's applications require database capabilities that can store, retrieve, and process diverse media and not just text and numbers. Full-motion video, voice, photos, and drawings cannot be handled effectively or efficiently by

either 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:

- **Multidimensional database.** This is an additional database that enables end users to quickly retrieve and present complex data that involve many dimensions (see Chapter 3).
- **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 (like 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 (see Chapter 3).
- **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 smart phones 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 bar-code scanner or medical tool).
- **Object-oriented databases.** In order to work in an object-oriented environment, it is necessary to use OO programming and OO databases. This topic is presented next. (Also see the description in Chapter 11.) If we know the value of IDnumber, we can find the student's major. Therefore, we say that a student's major is functionally dependent on the student's identification number, and that the student's identification number is a **determinant** of the student's major.

THE OBJECT-ORIENTED DATABASE MODEL

Although there is no common definition for *object-oriented (OO) database*, there is agreement as to some of its features. Terminology in the object-oriented model, similar to object-oriented programming languages, consists of objects, attributes, classes, methods, and messages (see Technology Guide 2).

OO databases store both data and procedures acting on the data, as objects. These objects can be automatically retrieved and processed. Therefore, the OO database can be particularly helpful in multimedia environments, such as in manufacturing sites using CAD/CAM. Data from design blueprints, photographic images of parts, operational acoustic signatures, and test or quality control data can all be combined into one object, itself consisting of structures and operations. For companies with widely distributed offices, an object-oriented database can provide users with a transparent view of data throughout the entire system.

OO databases can be particularly useful in supporting temporal and spatial dimensions. All things change; sometimes keeping track of temporal and spatial changes, rather than just the latest version, is important. Related but slightly different versions of an object can easily be maintained in an OO database. Object-oriented databases allow firms to structure their data and use them in ways that would be impossible, or at least very difficult, with other database models. An OO database is slow and therefore cannot be used efficiently for transaction-processing-type data. Therefore, as indicated earlier, it is sometimes combined with a relational database.

THE HYPERMEDIA DATABASE MODEL

The **hypermedia database model** stores chunks of information in the form of nodes connected by links established by the user. The nodes can contain text, graphics, sound, full-motion video, or executable computer programs. Searching for information does not have to follow a predetermined organizational scheme. Instead, users can branch to related information in any kind of relationship. The relationship between nodes is less structured than in a traditional DBMS. In most systems, each node can be displayed on a screen. The screen also displays the links between the node depicted and other nodes in the database. Like OO databases, this database model is slow.

T3.6 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, which optimizes both processing efficiency and user effectiveness. The process usually starts by analyzing what information each user (or group of users) needs and then producing 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 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 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 are 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.

Currently, there is a trend for users to back up data using a hard disk—what is called the *D2D (disk-to-disk) backup* method. This method is actually a disk duplication using another hard disk as the backup medium. It is a relatively easier operation than using tapes or diskettes and is now possible because the price of hard disks has fallen a lot lately. A report from QualStar said that 58 percent of respondents had already implemented the D2D backup method, and 25 percent were planning to implement the D2D backup method within the next 24 months (*bitpipe.com/detail/RES/1089741706_761.html&src%3Dzdib;* see QualStar story).

Nowadays, it is imperative to have the computer systems online in $24 \times 7 \times 365$, that is, in nonstop operation, partly because of e-business requirements. A survey done by Ziff-Davis Media in March 2003 found that lost employee productivity, lost revenue, and damaged company image were the top three consequences of a disruption of service.

HP's Business Continuity Storage Solutions are products to meet the need for continuous backup as a result of nonstop operations. HP's OpenView Continuous Access Storage Appliance (CASA) is a replication applicance for SANs (see section T3.8), in which a new SAN fabric is used to connect different SANs into a single logical pool of storage. HP's OpenView Storage Data Protector can help companies recover terabytes of data in minutes rather than in hours. HP's StorageWorks Enterprise Virtual Array (EVA), used mainly by data centers, can support heterogeneous environments (e.g., HP OpenVMS, IBM AIX, Microsoft Windows, Novell Netware, and Sun Solaris).

T3.7 Emerging Technologies: 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, etc.

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.

IP or Ethernet networks enable cost-effective SANs to be deployed by a broad market. Since IP storage runs over the existing Ethernet infrastructure, it retains all of the existing networking. This offers interoperability, manageability, compatibility, and cost advantages. People can use inexpensive, readily available Ethernet switches, hubs, and cables to implement low-cost, low-risk IP storage-based SANs. The advantages and disadvantages of SANs and NAS are shown in Table T3.7.

Although SANs and NAS have distinct profiles and different environments in which they work best, several companies including DataCore Software (*datacore.com/products/prod_home.asp*), Nishan Systems (*mcdata.com/splash/nishan*), Pirus Networks (*storagesearch.com/pirusnetw.html*), and Vicom Systems (*vicom.com/library/cs_BCBSofTennesseeCompletes.shtml*) are taking the advantages of both SANs and NAS by producing software and devices that work with both. New standards are being developed for both types of networked storage. The SCSI-over-IP protocol, called iSCSI, is a new Internet Engineering Task Force (IETF) specification that will let storage systems using SANs data transfer method send SCSI-style blocks of data over an IP network. In addition, a protocol, developed by IETF, called Fiber Channel-over-IP, would allow an enterprise to connect a SAN at one location with a SAN at another over an IP network

TABLE T3.7 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
	Equipment is designed to be highly scalable	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

T3.8 Data Storage Infrastructure and Management

DATA STORAGE INFRASTRUCTURE

The **Direct Access File System (DAFS)** protocol is one of the important technologies in data center storage infrastructure. It is a collaborative effort among dozens of vendors that will enable databases, Web servers, e-mail backends, and a host of other server-resident applications to achieve performance levels that are simply unattainable in the pre-DAFS world.

Another important technology in data center storage is IBM's Storage Tank storage management system, which combines storage virtualization, enterprise performance, policy-based storage management, and data sharing across heterogeneous storage systems at a greatly reduced total cost of ownership (TCO) due to more simplified management. IP storage protocols like iSCSI can simplify the complexity of SANs while allowing many customers to use a networking infrastructure with which they are comfortable or at least have already deployed for other uses.

Analysts and consultants estimate that from 50 percent to 70 percent of most companies' capital technology budget is spent on storage. An analyst at GartnerGroup (May 21, 2003) reports that worldwide storage capacity will skyrocket from 283,000 Tb in 2000 to more than 5 million Tb by 2005 (*hi.is/~joner/eaps/wwwgrow4.htm*).

In the long term, as part of the data center's "rearchitecting," the storage infrastructure will be transformed to provide storage automation for resource pooling, provisioning, and policy-driven management. Two strategic storage trends will continue in order to help IT departments: expansion of storage networking, and the continued splitting of the storage pyramid into more categories.

Ongoing storage networking trends will include: storage networking intelligence, IP-based storage networking, NAS-SAN convergence, and single-image file systems. Ongoing storage pyramid trends will focus on lifecycle content management to determine where in the storage pyramid content in each lifecycle stage should reside. Lifecycle content management builds around the concept of *data temperature*—that is, hot to cold. Hot data are accessible immediately whenever needed; cold data require some arrangement before they can be used. The trade-offs between hot and cold data are value/cost versus the need for responsive access.

STORAGE RESOURCE MANAGEMENT

Storage resource management (SRM) and storage virtualization are pieces of software that help manage storage as a *whole entity* rather than the disparate bits of technology you actually own. It works much like network management devices on corporate networks: The idea is to be able to have a bird's-eye view of everything on the storage networks and allocate storage resources as needed.

Fujitsu's Softek Storage Manager Software (*az.softek.com/en/press/20020219-001.html*) is designed specifically to meet the complex storage management requirements. Organizations are creating more information than they can manage, often doubling storage data each year. On the other hand, many current storage resources are not effectively utilized; only 40% of storage capacity is utilized. Softek Storage Manager has the following features highlights: centralized management—view and manage storage resources from a single console; meaningful reporting—assess how storage resources are used and identify capacity and performance trends with views at both the physical and logical layer; operation across heterogeneous environments—monitor and manage storage resources across hardware vendors, platforms and operating systems as well as disparate storage topologies; automation of routine tasks—schedule actions based on predefined criteria; business-process views—define and view storage as it applies to the business model; management of storage related

costs—assess, utilize, and where possible, reduce storage costs; management of service-level requirements—proactively manage service levels as required by the business.

Iomega is one of the mobile storage device manufacturers. Its Predator comes with Hotburn mastering software for Windows XP and Mac, MusicMatch Jukebox (*musicmatch.com*), and Adobe ActiveShare (*adobe.com/products/activeshare/main.html*) to organize digital photos. Also, it has the buffer underrun protection.

Corporate storage networks often contain components built by different manufacturers, each of which speaks in a proprietary computer language. EMC's WideSky module translates these proprietary languages into a single data format. EMC's AutoIS (now owned by Imation) includes a set of control modules that lets systems administrators manage the storage network from a single console (see *netherlands.emc.com/about/auto_is.jsp?openfolder=all,* and *tech-fag.com/storage/widesky-emc.shtml*).

InPhase Technologies has developed a holographic storage system (*inphasetechnologies. com/technology*). It can have storage capacity of 100 Gb per disk. Tandberg Data's O-Mass optical tape storage can store 1.2 Tb per standard-size tape cartridge. In the future, O-Mass claims that they would have 20 Tb of data on a single O-Mass (version 5) tape (*tandberg.com/graphics/O-Mass/Documents/O-Mass_2004_VI_200104.pdf*).

References

Babcock, C., "Storage Stakes Rising," *Interactive Week,* May 28, 2001.

Babcock, C., "Storage Vendors Whip Up Faster Data Recipes," *Interactive Week,* May 21, 2001.

Babcock, C., "XML Databases Offer Greater Search Capabilities," *Interactive Week,* May 7, 2001.

Connor, D., "Start-up Plots Storage Over IP Coup," *Network World,* May 22, 2000, *crmandcontactcentre247.com/Customer_Database_Solutions/Article2197.aspx#.*

Eweek Labs, "What Technology Will Be the Most Important for Storage Managers to Follow Over the Next Few Years?" *Eweek,* November 26, 2001.

Frost, R., J. Day, and C. van Slyke, *Database Design and Development: A Visual Approach*. Prentice Hall, 2006.

Helft, D., "Power to Spare," *The Industry Standard,* May 21, 2001.

Hilderbrand, C., "Why Squirrels Manage Storage Better Than You Do," *darwinmag.com/read/040102/squirrels_content.html,* February 2002.

Holstein, W. J., "Save EMC?" *Business 2.0,* September 2002.

IBM, "Capabilities of DB2 9," *www-306.ibm.com/software/data/db2/9/features.html,* 2006.

Katz, M., ed., *Technology Forecast: 1998 (also 1999, 2000)*. Menlo Park, CA: Price Waterhouse World Technology Centre, 1998, 1999, 2000.

Kroenke, D., *Database Processing: Fundamentals, Design, and Implementation,* 10th ed. Upper Saddle River, NJ: Prentice-Hall, 2006.

McDonald, G., "More Data on Your Hard Drive," *Business2.com,* December 12, 2000.

Mearian, L., "Storage Conference: Long-term View Taking Precedence," *ComputerWorld,* April 10, 2001.

Metz, C., "Hot Spots Getting Hotter," *PCM,* August 2002.

Neel, D., and M. Apicella, "Tomorrow's Storage," *infoworld.com,* April 15, 2002.

Ong, C., "DVDs Face a Challenge from Out of the Blue," *Technology Post,* February 25, 2003.

Oracle Corp., "Oracle Capabilities," *oracle.com,* 2006.

Post, G. V., and A. Kagan, "Comparison of Database Management Systems," *Journal of Computer Information Systems,* Summer 2001.

Silberschatz, A., et al., *Database Systems Concepts,* 4th ed. New York: McGraw Hill, 2001.

"Storage Trends: What Will Be Hot in 2003—And Beyond," *aberdeen.com/2001/research/12020008.asp,* January 2003.

Toh, A., "The Storage Behemoth," *CIO,* December 2002.

"What's New in Platforms and Storage," *lw8fd.law8.hotmail.msn.com,* January 2003.

Wikipedia,"SQL," *en.wikipedia.org/wiki/SQL,* 2006.

Wikipedia, "MySQL," *en.wikipedia.org/wiki/Mysql#The_latest_production_version,* 2006.

Winter Corporation, "Spotlight on SQL Server," *microsoft.com/sql/prodinfo/compare/wintercorp-survey.mspx,* 2006.