1 Introduction to Relational Databases and SQL

Chapter Overview

This chapter defines the basic terms of relational databases and the various kinds of datatypes available in popular database-management systems. The chapter also discusses the history and significance of Structured Query Language (SQL), the universal language for reading and writing data from relational databases.

Chapter Objectives

In this chapter, we will:

- Study the terms and concepts of relational databases
- Study the basic concepts of datatypes
- Learn about the history and importance of SQL as a database language
- Learn how to issue SQL commands using common database engines

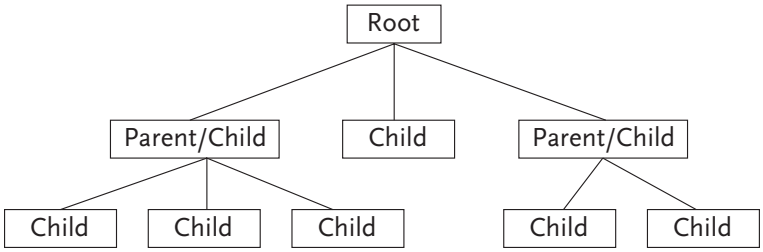
Database Concepts

Relational databases have been around for 30 years, but they were not the original kind of database, nor are they the newest kind of database. XML and object-oriented data structures have evolved in recent years. But relational databases are still by far the most popular kind of database available and will be for some time to come.

Before we get into the details of using ***SQL (Structured Query Language)***, it is important to understand some of the history and terminology associated with relational databases. The first computerized database-management systems were very different from the relational database systems we see today in products such as Oracle, SQL Server, and MySQL. In the 1970s, two data models were commonly used: hierarchical and network.

Hierarchical Databases

The hierarchical model dates back to the early 1960s and is the oldest of the database models. The most popular hierarchical database management system was IBM's IMS, which is still in use today. In the hierarchical model, the data is structured in what is referred to as parent-child relationships. Visualize this relationship as an upside down tree. An example is shown below.



An important concept to understand about this model is that a child element can have only one parent, but a parent element can have many children. This relationship in database design terms this is known as a one-to-many relationship. For example, one department can have many employees. If you are having problems visualizing this structure, use Windows Explorer to look at the file structure of your PC using. You see a file structure organized in a hierarchical fashion. To access any part of this structure you must navigate your way through the tree structure until your destination is reached. The hierarchical model is not without its problems; a major drawback is data redundancy. This problem occurs because the hierarchical model handles the one-to-many relationship very well, but is unable to deal with a many-to-many relationship, causing data to be duplicated. For example, a hierarchical chart showing an employee who reports to more than one supervisor would require duplicating that employee's information.

Network Databases

The network model was developed in response to the problems encountered with the hierarchical model. Representing data using set theory instead of a hierarchy eliminates the major problem of redundancy. The network model is very similar to the hierarchical model, except in this structure a child element is allowed to have more than one parent, so the underlying concept behind the network model is that a child can have many parents and parents can have many children. This allows many-to-many relationships to be represented. For example, each student can take many courses and many students can take each course. Another difference between the two models is that the network model allows the structure to be navigated without having to start at the root element.

Both the hierarchical and the network models have limitations. One limitation is that their indexing scheme is tied to the sector scheme of the hard drive on which they reside, which creates a mess if ever moved. Furthermore, it is difficult to query data from multiple tables.

Querying a database to retrieve data involves creating a program to navigate the database structure to retrieve the requested data. These programs use procedural or proprietary languages that require a great deal of knowledge and skill. Describing these two models in any more detail here is beyond the scope of this book.

Relational Database Concepts

In 1970 an IBM researcher, ***Dr. E. F. Codd***, came up with a better way—the relational data model. In this model, the database management system (***DBMS***) itself keeps track of all table relationships independent of hardware or outside programming languages. In the relational model, the user only needs to understand the logical structure of data, not how it is physically stored. In this model data is represented in simple two-dimensional ***tables*** (relations), which consist of rows (tuples) and columns (attributes). A ***relational database*** is simply a collection of tables.

As we discuss these relational concepts, we will use a portion of the Lyric Music database shown here. A full description of the Lyric Music database can be found in Appendix A.

Artists

ArtistID	ArtistName	City	Region	Country	WebAddress	EntryDate	LeadSource
1	The Neurotics	Peterson	NC	USA	www.theneurotics.com	5/14/2003	Directmail
2	Louis Holiday	Clinton	IL	USA		6/3/2003	Directmail
3	Word	Anderson	IN	USA		6/8/2003	Email
5	Sonata	Alexandria	VA	USA	www.classical.com/sonata	6/8/2003	Ad
10	The Bullets	Alverez	TX	USA		8/10/2003	Email
14	Jose MacArthur	Santa Rosa	CA	USA	www.josemacarthur.com	8/17/2003	Ad
15	Confused	Tybee Island	GA	USA		9/14/2003	Directmail
17	The Kicks	New Rochelle	NY	USA		12/3/2003	Ad
16	Today	London	ONT	Canada	www.today.com	10/7/2003	Email
18	21 West Elm	Alabama	VT	USA	www.21westelm.com	2/5/2003	Ad
11	Highlander	Columbus	OH	USA		8/10/2002	Email

Titles

TitleID	ArtistID	Title	StudioID	UPC	Genre
1	1	Meet the Neurotics	1	2727366627	alternative
3	15	Smell the Glove	2	1283772282	metal
4	10	Time Flies	3	1882344222	alternative
5	1	Neurotic Sequel	1	2828830202	alternative
6	5	Sonatas	2	3999320021	classical
7	2	Louis at the Keys	3	3838227111	jazz

Table

The basic unit of a relational database is the table. You can think of a table as rows and columns of information, as in a spreadsheet. A relational database is a collection of at least one—and generally, several—tables. Each table tracks data about something different: employees, products for sale, orders, etc. In the Lyric Music database, we have tables for artists and titles, among other tables.

The "relational" part of relational databases is how these multiple tables relate to each other, allowing a user to pull information from a combination of tables. For instance, in the Lyric Music database a relationship exists between artists (the Artists table) and titles (the CD Titles table). Using the ArtistID field in Titles, one could look up the artist's name and other information for each CD title.

Record/Row

A record holds all the information about one item or subject. Conceptually, if you collected business cards from 50 people, all 50 cards would represent a table and the information on any one business card would represent one record. In a database table, each row is a record. In the same way, the seven rows in the Titles table represent seven different CD titles available through Lyric Music. Some database experts prefer to just use the word row, since record sometimes has other connotations outside of relational databases.

Field/Column

A field holds one piece of information about an item or subject. Since a field is a column in a database table, some database experts prefer to just use the word *column*. In the Artists table there are fields for City, WebAddress, LeadSource, etc. In a relational database, the relationships are maintained by having matching fields in two tables. For instance, the Artists table and the Titles table share the ArtistID field, thus allowing the two tables to be linked and data to be pulled from the two tables together.

Datatype

Every field in a database table is assigned a ***datatype***, which describes the kind of data that can be stored in the field. You need to be familiar with datatypes because how you handle the data in SQL depends a lot upon the datatype.

Generic Datatype	Access	SQL Server	Oracle	MySQL	Description
Text or Char	Text Memo	Char VarChar NChar NVarChar	Char NChar VarChar2 Long CLob	Char VarChar TinyText Text MediumText LongText Set	Holds alphanumeric data (letters and numbers). Used for all fields that will hold at least one letter. Also, generally used for numeric fields that don't involve calculations, such as phone number and postal code. Some variations of these datatypes hold either 8-bit or 16-bit numbers, pad with spaces or not, or have other characteristics.
Numeric	Byte Integer Long Integer Single Double Decimal	TinyInt SmallInt Integer Decimal	Byte SmallInt Integer Number Float Real	TinyInt SmallInt MediumInt Int BigInt Float Real	Holds numeric data. Some variations of these datatypes hold larger or smaller numbers and either whole numbers or fractions.
Currency	Currency	SmallMoney Money	Money	Decimal	Used for money fields.
Date	DateTime	SmallDateTime DateTime	Date	Date Time DateTime	Holds date and/or time data.
Boolean or True/False	Yes/No	Bit	Bit	TinyInt Enum	Holds only two values: On/Off, True/False, Yes/No. This datatype often stands behind checkboxes. Generally, Yes/No fields are stored with a 0 for No/False and a 1 for Yes/True (-1 in the case of Microsoft Access). See the mp3 and RealAud fields of the Tracks table for an example.
Special	OLE Object Hyperlink	Image Binary	Blob Raw Long Raw	TinyBlob Blob MediumBlob LongBlob	Used to hold graphics, sounds, hyperlinks, and other special kinds of data.

Datatypes vary depending on the database system you are working in. Below are generic datatypes that are found in most database systems and the equivalent specific datatypes in each of four popular database systems. This list is not exhaustive, as datatypes vary from one version of a database system to another. For more information see Appendix B.

Primary Key

A primary key is a field (or possibly multiple fields used together) that uniquely identifies each record in the table from every other record in the table. In our Artists table it is ArtistID. Each artist has an ArtistID that is different from every other artist in the table. In our Tracks table, the primary key is composed of TitleID and TrackNum. TitleID is not unique because a given title will have multiple tracks. TrackNum is not unique because every title will have a track 1, track 2, etc. But the combination of TitleID and TrackNum is unique for every record.

Primary keys are essential in a relational database. Suppose you are working by the hour at some job. Your hours have to be recorded somehow. It may be with a pencil entry in a logbook. It may be with a mechanical punch clock. It may be with a magnetic employee card that you swipe through a reader. But no matter how the hours are recorded, getting paid the correct amount depends on the fact that your hours will not be confused with someone else's hours. Your hours must be recorded with some identifier that attributes them to you and to no one else. That is a primary key.

Foreign Key

Earlier we discussed how tables in relational databases relate to each other through matching fields. The primary key of one of the tables is almost always involved in the relationship. The field in the other table on the other end of that relationship is called the ***foreign key***. The term simply refers to the fact that this field is key to relating to a foreign (or other) table. In the Lyric Music database there is a relationship between artists and titles. The ArtistID field is the primary key in the Artists table. Therefore, the ArtistID field in the Titles table is a foreign key. It relates the Titles table to the primary key in the Artists table.

Most table relationships can be described as one-to-many. In a one-to-many relationship, a single record in the first table can be related to many records in the second table. However, each record in the second table relates to only one record in the first table. For instance, ArtistID 1, The Neurotics, has just one record in the Artists table (it can have only one record since ArtistID is the primary key in that table). But there are two records in the Titles table for ArtistID1 (the foreign key), Meet the Neurotics and Neurotic Sequel. So artists and titles have a one-to-many relationship. The table on the "one" side of the relationship is also called the "parent" table, while the table on the "many" side of the relationship is called the "child" table. A parent can have many children. In addition to one-to-many relationship, tables can have one-to-one relationships. But these are much less common.

Query/View

A *query* is used to display information from a database. Queries can be designed using either Query By Example (QBE), which is a GUI interface for query building, or Structured Query Language (SQL), which is a query language. Of the two, SQL is the more universal and more powerful tool. The end of this chapter will discuss SQL in detail.

Queries can report any columns from a table, any set of rows from a table, and even information from multiple tables using table relationships. Queries can summarize and calculate information. In addition to selecting information for display, queries can also insert, update, and delete data.

In some databases you can save queries for reuse. In high-end databases, a saved query is called a view. The creation of views or queries is a very powerful tool in Internet programming. A view can transform a very complicated multi-table query into something a very simple, which in turn makes the programming code much simpler.

Stored Procedure

A stored procedure is a high-end database tool that adds programming power right into the database. Database administrators will often create stored procedures to handle inserts, edits, and updates of records. The front-end programmer, then, only needs to call the stored procedure to accomplish these functions. It makes the programming code simpler and helps protect the database from problems caused by program bugs.

Overview of SQL

What is SQL?

SQL, or Structured Query Language, is a language used to create relational databases and manipulate data in them. It is pronounced either by its letters (Ess-Que-Ell) or as a word (See-quel), depending on whom you talk to.

SQL is an open standard database language, supported by ANSI (American National Standards Institute). SQL has become the language of choice for designing, querying, and updating relational databases. Oracle, Sybase, Microsoft SQL Server, MySQL, IBM DB2, Microsoft Access, and Lotus Approach, and many others all support SQL. Indeed over 100 database management products, running on everything from PCs to mainframes, support SQL. Most systems also have their own visual Query by Example (QBE) interface. But while QBE varies widely among database systems, SQL is an ANSI standard and varies only slightly. Also, some kinds of queries cannot be done in QBE on many database systems.

For both database administrators and programmers, knowledge of SQL is essential and transportable. SQL is often used in client-server programming, web programming, and many other environments.

What Does SQL Do?

First, SQL is the premier tool for viewing information from a relational database. It doesn't just give you a data dump. SQL gives you sophisticated tools to summarize, consolidate, and calculate from the data. Using table relationships, data can be combined from multiple tables in a number of ways. With a properly designed database, SQL can answer practically any question about the data.

Second, SQL provides commands to manipulate the data in a relational database. Records can be updated and added to or deleted from a table. Here is SQL as a database language really shines. Procedural programming languages, such as BASIC, might require several lines of code to update a record in a database table. In addition, procedural programming languages would have to use some sort of looping structure to repeat this process on every record. SQL operates on an entire set of records all at the same time. SQL is like haiku for programmers; often a dozen words or fewer can delete or change thousands of records.

Finally, SQL is a complete data definition language (***DDL***). The database itself can be created along with all tables, fields, primary keys, and relationships. Add to that the record insert commands, and you can have a complete database and all its data expressed in programming code. This greatly enhances a database programmer's ability to work remotely or to port data enhancements among various installations.

History of SQL

OK. Not everyone likes history, so this will be short. But to understand the present impact of SQL, it helps to understand how it came about.

Remember, the idea behind relational databases was that the database management system (DBMS) would itself keep track of all table relationships independent of hardware or outside programming languages. Thus a language to query data from multiple tables could be much more high level and easy to use. Dr. E.F. Codd, the IBM researcher described such a language, calling it Structured English QUEry Language (or SEQUEL). Later the name was shortened to just Structured Query Language (SQL).

Codd's ideas quickly caught on. Several companies began working on DBMSs to implement the relational concept. The first systems to hit the market in the late 1970s were Oracle, Relational Technology's Ingres, and IBM's System/R. All of them used a version of SQL to manipulate the data and database structures. With IBM's market dominance, SQL quickly caught on and IBM's version of SQL became the de facto standard.

The computer industry saw a need to bring all the versions of SQL together into a standard that everyone could adopt. The American National Standards Institute (ANSI), one of the major standards bodies in the United States representing 1,000 companies, organizations, and government agencies, headed up this effort. The first SQL standard (called SQL89) was released in 1989. The SQL specification has since been twice revised with SQL92 (in 1992) and SQL99 (in 1999).

One advantage of SQL being an open standard is that SQL programmers can transport their skill to any DBMS. Also, it allows specific databases to be ported to other DBMSs with only minor changes to the SQL code.

How Standard Is SQL?

Most DBMSs meet SQL89 standards, and many meet SQL92 standards. So to a large extent, SQL written for one DBMS will work in another DBMS with little or no changes. However, every DBMS vendor wants to differentiate its product on the market with more powerful features. In addition, the SQL standard does not define some things that programmers often want to do with data, thus leaving a gap in functionality that each vendor must fill. Finally, for whatever reason, DBMS vendors often digress from the SQL standard in both minor and major ways.

Every implementation of SQL is at least a little different. Throughout this book, we will stick as close to ANSI-SQL as possible and note differences among the various DBMS products. Chapter 5 lists many of the SQL extensions added by various vendors. Many of these are quite useful. In addition, many of the SQL extensions provide faster performance than ANSI-SQL because they are optimized for a particular DBMS.

However, programmers should approach DBMS-specific extensions to SQL with caution. Suppose you are writing a database-driven web application. You shop around for a web-hosting company and find one you like that offers SQL Server as the back-end database. A year later that web-hosting company goes out of business and you switch to a hosting company that does not offer SQL Server but does offer MySQL for database connectivity. The more you have relied upon SQL Server–specific functions in your code, the more work you are going to have to do to get your application running on the new host.

On the other hand, suppose your company purchases Oracle to use as the back-end database for a company intranet. It is likely they will keep Oracle as their database forever, so if you can wring more performance from your code by using Oracle-specific functions, it may very well be worth it.

This book will stick as closely as possible to ANSI standard SQL, using database specific code only when necessary to accomplish common tasks. As will be evident as you read the book, one can probably never write SQL that is 100 percent portable among different DBMS products.

But be aware that the further you stray from ANSI-SQL, the more portability you lose.

Importance of SQL Today and Tomorrow

While hierarchical and network database systems still exist and while new, non-relational data concepts (such as Extensible Markup Language, XML, and object-oriented programming, OOP) have arisen, the relational database is still king, and SQL is the language of choice for relational data. In fact, SQL is being enlarged to include XML. Microsoft's SQL Server 2000 includes extensions to SQL called SQLXML to create XML views of relational data. Oracle9i Release 2 has fully incorporated the W3C (World Wide Web Consortium) XML data model into its DBMS and provides access methods for navigating and querying XML. So there is every indication that relational databases and SQL will remain important tools in the future.

Database administrators and back-end developers use SQL as a stand-alone language to create and maintain databases. SQL is also used by front-end application programmers as a set of embedded commands for database connectivity. This is standard practice in both client-server programming and many kinds of web programming, including ASP, PHP, JSP, Java, and other environments. SQL is everywhere.

SQL is an important skill even for non-programmers. Many managers face the demands of querying a corporate database to get vital information. Though many kinds of user interfaces have been developed for that task, nearly all of them use either SQL concepts or SQL syntax.

How to Practice SQL Commands with this Book

You can't learn SQL without practicing SQL. So what software tools can you use to practice the SQL commands in this book? You have several options.

Microsoft Access

Microsoft Access is part of Microsoft Office. It is a competent database system for personal use and small workgroups of as many as 25 users. Access also works adequately as a database backend for low traffic database-driven web sites.

If you have Microsoft Access 97 or higher, you can enter SQL commands directly in Access. Copy lyric2k.mdb (Access 2000 or higher) or lyric97.mdb (Access 97) from the CD-ROM. You may need to un-check the file's read-only property. Launch it in Access, and convert it if necessary. Then do the following:

Access 97

- ❍ Click on Queries
- ❍ Click on New
- ❍ Click on Design View and OK
- ❍ In the Show Tables dialog, click Close
- ❍ Click on the SQL button in the toolbar

Access 2000+

- ❍ Click on Queries
- ❍ Double-click on "Create query in Design view"
- ❍ In the Show Tables dialog, click Close
- ❍ Click on the SQL button in the toolbar

Data description commands:

- ❍ In Access you do not use SQL commands to view the tables and table information. To view the tables, go to the Tables tab of the database window. To view the information on any table, highlight that table and click on the Design button.

SQL Server

SQL Server is Microsoft's high-end client-server database. To load the Lyric Music database into SQL Server, do the following:

- ❍ Launch the SQL Server Query Analyzer and log in using the master sa (system administrator) login or a login provided to you by your database administrator.
- ❍ From the menu select File | Open. With the book's CD-ROM in your CD-ROM drive, browse to your CD-ROM drive and select LyricSQLServer.sql.
- ❍ If you do not have rights to create a database, delete the first four lines of the script file that is loaded.
- ❍ From the menu select Query | Execute to run the script and create the Lyric Music database.

To practice your SQL, do the following:

- ❍ Launch the SQL Server Query Analyzer and login using the master sa login or a login provided to you by your database administrator.
- ❍ Type the SQL command in the upper pane.
- ❍ Run the SQL by selecting Query | Execute from the menu or by clicking the green triangle icon in the toolbar. The results will appear in the lower pane.

Data description commands:

- ❍ To list all user tables in your SQL Server database type: **select name from sysobjects where type = 'U'**
- ❍ To list all fields, data types, and related information for a particular table type: **exec sp_help *tablename*;**

MySQL

MySQL is a free client-server database that can be downloaded from http://www.mysql.com. MySQL is used in many websites, especially web sites hosted on Linux servers. However, it also runs under Windows. To load the Lyric Music database, do the following:

- Go to a command prompt (Start | All Programs | Accessories Command Prompt depending on your version of Windows).
- Once at a command prompt you will probably need to change to the directory in which your MySQL programs are installed. Under Windows that is normally done with: cd\mysql\bin
- Place the book's CD-ROM in your CD-ROM drive, then type: **mysql mysql < d:LyricMySQL.sql**. Replace the **d:** with the letter of your CD-ROM drive.

To practice your SQL, do the following:

- Launch MySQLManager (You will find this in the /mysql/bin folder. It would be a good idea to create a shortcut on your desktop or in your Start menu. This program is where you can enter SQL.)
- From the drop-down menu select Tools | SQL Query
- Click the yellow cylinder database icon to view a list of your databases. Select Lyric
- Click on the Query tab and type **Use** followed by the name of your Lyric Music database (i.e., **Use Lyric**).
- Click the green triangle run icon. The display will switch to the Results tab, but not show any results.
- Click back on the Query tab and type your SQL command. Use the green triangle run icon to view your results.

Data description commands:

- To list all user tables in your Oracle database type: **show tables**
- To list all fields and data types in a particular table type: **describe** ***tablename***

Oracle

Oracle is the most popular client-server database system in the current market. If you have access to an Oracle database, you can load the Lyric Music database with the following steps.

- Launch Oracle SQL Plus and log in using the master system user name or a login provided to you by your database administrator.
- Place the book's CD-ROM in your CD-ROM drive.
- At the SQL prompt type @ followed by the drive letter of your CD-ROM drive followed by a colon (:) and **LyricOracle.sql** (i.e. **d: LyricOracle.sql**)

To practice your SQL, do the following:

- Launch Oracle SQL Plus and login using the master system user name or a login provided to you by your database administrator.
- At the SQL prompt type your SQL command. You may use multiple lines for the command if you wish. At the end of the last line add a semi-colon (;) to execute the command.

Data description commands:

- To list all user tables in your Oracle database type: select table_name from user_tables;
- To list all fields and data types in a particular table type: describe *tablename*;

Chapter Summary

Though other kinds of data structures exist (including XML and object-oriented databases), relational databases are the backbone of most applications today. Relational databases keep data in various tables that relate to each other through primary key–foreign key relationships. A primary key is one field or a group of fields that uniquely identifies each row (or record) in a table. Each column (field) in a database table has a datatype that defines the type of data that can be stored in it. These datatypes vary from one database system to another, but fall into the general categories of text, numeric, currency, date, Boolean, and special datatypes.

Structured Query Language (SQL) is the language of choice for querying data from a relational database. In addition to being a sophisticated query language, SQL has commands to insert, update, and delete records and even to create a relational database completely from scratch. SQL is nearly universally implemented in relational database products. In addition, SQL is an ANSI standard, which means there is a high level of consistency in SQL between various database systems. Still, there are differences in each vendor's SQL implementation. Often, the database-specific extensions for SQL add needed features and welcomed speed. However, programmers who desire maximum portability will want to stick as close to ANSI standard SQL as possible.

Key Terms

ANSI (American National Standards Institute)
Dr. E.F. Codd
datatype
DBMS (database management system)
DDL (data definition language)field
foreign key
primary key
query
record
relational database
SQL (Structured Query Language)
stored procedure
table
view

Review Questions

1. How are relational databases an improvement over the databases that came before them?
2. What does a primary key do in a relational database table?
3. What does a foreign key do in a relational database table?
4. What does the term "one-to-many" mean in a relational database?
5. Why is it a good thing that SQL is an ANSI standard?
6. What kind of datatype would best be used to hold a street address such as "203 West 11th Street"?

Exercises

Using any of the SQL tools described above, issue the following SQL commands and answer the following questions:

1. SQL Command: Select * from Studios
   List the name of the fields (columns) in the Studios table. How many records (rows) are in the Studios table?
2. SQL Command: Select * from Titles
   What field (column) does the Titles table share with the Studios table? Judging from the data, in which table is it the primary key and in which is it the foreign key? How can you tell?
3. SQL Command: Select * from Artists
   List the names of each of the fields (columns) along with what is the likely generic datatype for each.
4. Use the appropriate data description command listed in the "How to Practice SQL Commands with This Book" section to view a list of the user tables. List their names.
5. Use the appropriate data description command listed in the "How to Practice SQL Commands with This Book" section to view information in the Artists table. What is the datatype of the ArtistID column?

6. Suppose we needed to add tables and fields to the Lyric Music database to sell CDs through e-commerce. Select the appropriate generic datatypes for each of the following new fields:
   Order Number
   Order Date
   Customer Name
   Customer Ship Address
   Customer Ship City
   Customer Ship State
   Customer Ship Zip
   Qty Ordered
   Shipping Charge

Additional References

A Brief History of SQL	**http://www.vbip.com/books/1861001800/chapter_1800_02.asp**
The Rise of Relational Databases	**http://www.nap.edu/readingroom/books/far/ch6.html**
A Brief History of Databases	**http://wwwinfo.cern.ch/db/aboutdbs/history/industry.html**
MySQL	**http://www.mysql.com/**