Chapter 1
Open It Up

A database system is simply a computerized record keeping system, which is used to manage databases, or structured data.

Once such a database is set up, a user can apply various operations on them, such as add new tables into, and/or remove an existing table from, a database; insert a record into, modify an existing record within, and delete, or retrieve, an existing record from, (an) existing table(s) in a database.

Database related application is used everywhere all the time these days. For example, over 100 million credit card transactions are processed each day from over 10 million merchants through more than 20,000 banks.
A general query process

A *relational database* (*RDB*) is just a collection of tables: rows of columns. Below is an instance of the *Student* table.

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Address</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>Jane Doe</td>
<td>123 Main St.</td>
<td>Freshman</td>
</tr>
<tr>
<td>2222</td>
<td>Mary Smith</td>
<td>1 Lake St.</td>
<td>Freshman</td>
</tr>
<tr>
<td>1234</td>
<td>Joe Blow</td>
<td>6 Yard Ct.</td>
<td>Junior</td>
</tr>
</tbody>
</table>

When getting information for a query, “Who are freshman?” we a) identify the table (*From*); b) choose the rows (*Where*); and c) pick the columns (*Select*).

```sql
mysql> Select Name
       -> From Student
       -> Where Status="Freshman";
```

```
+------------+
| Name       |
+------------+
| Jane Doe   |
| Mary Smith |
+------------+
```
Step up with something tasty...

Below is a sample database with only one table, cellar, consisting of some basic facts about various wines.

<table>
<thead>
<tr>
<th>BINNO</th>
<th>WINE</th>
<th>PRODUCER</th>
<th>YEAR</th>
<th>BOTTLES</th>
<th>READY</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Chardonnay</td>
<td>Buena Vista</td>
<td>1997</td>
<td>1</td>
<td>1999</td>
</tr>
<tr>
<td>3</td>
<td>Chardonnay</td>
<td>Geyser Peak</td>
<td>1997</td>
<td>5</td>
<td>1999</td>
</tr>
<tr>
<td>12</td>
<td>Joh. Riesling</td>
<td>Jekel</td>
<td>1998</td>
<td>1</td>
<td>1999</td>
</tr>
<tr>
<td>21</td>
<td>Fum Blanc</td>
<td>ch. St. Jean</td>
<td>1997</td>
<td>4</td>
<td>1999</td>
</tr>
<tr>
<td>30</td>
<td>gewurztraminer</td>
<td>Ch. St. Jean</td>
<td>1998</td>
<td>3</td>
<td>1999</td>
</tr>
<tr>
<td>50</td>
<td>Pinot Noir</td>
<td>Gary Farrell</td>
<td>1996</td>
<td>3</td>
<td>1999</td>
</tr>
</tbody>
</table>

This table holds eight rows, each keeps information for one particular wine with six columns. Each such column reflects one named attribute of that kind of wine.
How much space to kick in?

To help the DBMS to set aside certain space to keep such information, we have to specify a data type for each attribute.

mysql> desc cellar;

<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Null</th>
<th>Key</th>
<th>Default</th>
<th>Extra</th>
</tr>
</thead>
<tbody>
<tr>
<td>binno</td>
<td>int(11)</td>
<td>NO</td>
<td>PRI</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>wine</td>
<td>varchar(20)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>producer</td>
<td>varchar(20)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>year</td>
<td>int(11)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>bottles</td>
<td>int(11)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
<tr>
<td>ready</td>
<td>int(11)</td>
<td>YES</td>
<td></td>
<td>NULL</td>
<td></td>
</tr>
</tbody>
</table>

Moreover, BINNO is the primary key of CELLAR, in the sense that this field uniquely identifies any specific row: no two rows share the same BINNO value. The rows in such a table is usually ordered by its primary key.

**Note:** You have to use cellar, but you can use, e.g., BINNO, but not binno.
Play with it *first*

The only reason to keep data somewhere is to use them later, e.g., get them out, or more formally, data retrieval.

**Question:** Which wines are ready in 2000?

```sql
mysql> Select WINE, BINNO, PRODUCER
    -> From cellar
    -> Where READY=2000;
```

```plaintext
+----------------+-------+--------------+
| WINE | BINNO | PRODUCER |
|----------------+-------+--------------+
| Cab. Sauvignon | 43    | Windsor     |
| Pinot Noir     | 51    | Fetzer      |
| Merlot         | 58    | Clos du Bois|
+----------------+-------+--------------+
3 rows in set (0.00 sec)
```
More queries

Other *update* operations, such as insertion, deletion, and revision, can also be applied. The basic syntactic structure stays the same.

For example, the following adds in another row.

Insert into cellar
(BINNO,WINE,PRODUCER,YEAR,BOTTLES,READY)
Values (53,’Pinot Noir’,’Saintsbury’,1997,6,2001);

We now do it with MySQL:

```
mysql> Insert into cellar
     -> (BINNO,WINE,PRODUCER,YEAR,BOTTLES,READY)
    -> Values (53,’Pinot Noir’,’Saintsbury’,1997,6,2001);
Query OK, 1 row affected (0.08 sec)
```

We can repeatedly add in more rows...
... until it looks like the following:

```sql
mysql> select * from cellar;
```

<table>
<thead>
<tr>
<th>binno</th>
<th>wine</th>
<th>producer</th>
<th>year</th>
<th>bottles</th>
<th>ready</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Chardonnay</td>
<td>Buena Vista</td>
<td>1997</td>
<td>1</td>
<td>1999</td>
</tr>
<tr>
<td>3</td>
<td>Chardonnay</td>
<td>Geyser Peak</td>
<td>1997</td>
<td>5</td>
<td>1999</td>
</tr>
<tr>
<td>6</td>
<td>Chardonnay</td>
<td>Simi</td>
<td>1996</td>
<td>4</td>
<td>1998</td>
</tr>
<tr>
<td>12</td>
<td>Joh. Riesling</td>
<td>Jekel</td>
<td>1998</td>
<td>1</td>
<td>1999</td>
</tr>
<tr>
<td>21</td>
<td>Fum Blanc</td>
<td>ch. St. Jean</td>
<td>1997</td>
<td>4</td>
<td>1999</td>
</tr>
<tr>
<td>30</td>
<td>gewurztraminer</td>
<td>Ch. St. Jean</td>
<td>1998</td>
<td>3</td>
<td>1999</td>
</tr>
<tr>
<td>45</td>
<td>Cab. Sauvignon</td>
<td>Geyser Peak</td>
<td>1994</td>
<td>12</td>
<td>2002</td>
</tr>
<tr>
<td>48</td>
<td>Cab. Sauvignon</td>
<td>Robt. Mondavi</td>
<td>1993</td>
<td>12</td>
<td>2004</td>
</tr>
<tr>
<td>50</td>
<td>Pinot Noir</td>
<td>Gary Farrell</td>
<td>1996</td>
<td>3</td>
<td>1999</td>
</tr>
<tr>
<td>51</td>
<td>Pinot Noir</td>
<td>Fetzer</td>
<td>1993</td>
<td>5</td>
<td>2000</td>
</tr>
<tr>
<td>52</td>
<td>Pinot Noir</td>
<td>Dehlinger</td>
<td>1995</td>
<td>2</td>
<td>1998</td>
</tr>
<tr>
<td>53</td>
<td>Pinot Noir</td>
<td>Saintsbury</td>
<td>1997</td>
<td>6</td>
<td>2001</td>
</tr>
<tr>
<td>58</td>
<td>Merlot</td>
<td>Clos du Bois</td>
<td>1994</td>
<td>9</td>
<td>2000</td>
</tr>
<tr>
<td>64</td>
<td>zinfandel</td>
<td>Cline</td>
<td>1994</td>
<td>9</td>
<td>2003</td>
</tr>
<tr>
<td>72</td>
<td>zinfandel</td>
<td>Rafanelli</td>
<td>1995</td>
<td>2</td>
<td>2003</td>
</tr>
</tbody>
</table>

17 rows in set (0.00 sec)

Notice that the row with its binno being 53 has been added into the table.
Update: Just got another Chardonnay?

mysql> Update cellar
    -> Set BOTTLES=4
    -> Where BINNO=3;
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0
Let's check it out.

mysql> select binno, wine, bottles
    -> from cellar where binno=3;
+-----------------+-----------------+-------+
<table>
<thead>
<tr>
<th>binno</th>
<th>wine</th>
<th>bottles</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>Chardonnay</td>
<td>4</td>
</tr>
</tbody>
</table>
+-----------------+-----------------+-------+
1 row in set (0.00 sec)
A more realistic approach

To build meaningful database applications, particularly, transaction processing applications, the power of a full-fledged high-level programming language, such as Java, C++, or PHP, is needed. Moreover, in today’s WEB age, database programming is almost always done over the Internet, it thus makes sense to use WEB as the platform for data presentation.

We will discuss, in the lab part, PHP, a script programming language, and its relationship with MySQL, a “free” DBMS, and their connection with HTML.

The three musketeers are often used together because of they are free, reasonably good, and easy to mingle together.
An example

We will set up a newsletter on line and ask our guests to sign up with their email addresses. Thus, later on, we can send our guests some relevant information.

One way to do it is to manually enter those email addresses.

mysql> insert into mailinglist
   -> (ID, Email, Source)
   -> Values (26, 'zshen@goofy.com', 'www.example.com/');
Query OK, 1 row affected (0.01 sec)

mysql> select * from mailinglist;
+----+--------------------+----------------------------------------+
| ID | Email | Source |
+----+--------------------+----------------------------------------+
| 26 | zshen@goofy.com | www.example.com/ |
| 70 | zshen@plymouth.edu | www.example.com/newsletter_signup.html |
+----+--------------------+----------------------------------------+
2 rows in set (0.00 sec)

This is just a bit too bulky. Let’s try the following: http://turing.plymouth.edu/~zshen/PhPFiles/newsletterSignup.html.

The associated code can be found in §5.2 of the *PhP Notes*. 
What is a DBMS?

Here it goes again: A Database Management System (DBMS) is a collection of programs that manage databases.

It supports a high-level access language (e.g. SQL), which a programmer uses to describe the structure of (DDL), and gets access to (DML), a database.

The description part is usually easy (?), while the DML part is often put into two classes: query and update, as we have seen.

When a user submits a query to a DBMS, it will interpret such a query, perform requested database access, and send the results back to the user.
Time is the essence....

Some of the databases are timing sensitive, since the real world that those data models change constantly, e.g., *how much money do you have in your account today?* Hence, a database often stores information about the *current state* of such a world, e.g., the current balance of your bank account.

When an event happens in the real world that changes such a state, a corresponding change must be made to the relevant information kept in the database to reflect such a change.

For example, when you make a deposit or withdraw, your balance will be changed as well.

Particularly, the change corresponding to a withdraw operation needs to be done immediately, i.e., in real time.
What is a transaction?

A transaction is a sequence of those changes made in real time. Besides changing the state of the database, a transaction might trigger other events, such as sending out cash via an ATM, or dial 1-800-847-2911 when it identifies a stolen Visa card.

As we will see later, a transaction is atomic, in the sense that either everything in the sequence will be done, or nothing is.

For example, when you withdraw some money from the bank, you will fill a form, give it to the teller, who will then give you the money.

If you change your mind during the process, not only will you not get the money, the slip should (will) be shredded, as well.
Transaction processing system

Such a system contains a few relevant databases that store the state of the enterprise, the software that manages the transactions that manipulate that state, and the transactions themselves.

Because of long term and wide scope of database applications, it could be the case that several DBMS are involved. Then a transaction execution is controlled by a TP monitor that is to coordinate transactions across multiple sites and systems.

In many cases, the transaction processing system has become the life blood of a company. The credit card business is certainly one of the first examples we can think of.

Did you read the scary Equifax story yet?
We are mainly interested in the basics of database design and manipulation, but not the algorithms and data structures behind the transaction processing systems.

These topics are often the subjects of a second course on database systems, which you often take when you start with a grad school.
Some of the DBMS features

Technological progress has led to significant advances in terms of architecture, design and use of databases and transaction processing systems. They are much faster, can handle much more data, much easier to use and implement, thus providing much more business opportunities. As a price to pay, it also leads to more requirement. For example,

*High availability:* On-line systems must be constantly operational 😊. Visa reported back in 2002 that their system is down for a total of 8 minutes for five years.

*High reliability:* The system must be accurate, correctly tracks state, does not lose data, with controlled concurrency.
High throughput: With many users, the system must have the ability to process as many transactions per second.

Low response Time: Users are waiting 😞.

Long lifetime: These systems are not pieces of cake. We have to make lots of efforts and investment. We certainly want them to stay there for a while.

Security: Lots of data are either mission critical or private, thus have to be protected all the time. (Could we?)

To keep or not to keep: Even though they are protected, sometimes, they could be subpoenaed or simply handed over when national security is involved. It is certainly not a technical issue.
Major players

A transaction processing system and the involved databases are highly complicated stuff that requires the participation of many people for its success.

*System analysts* to find out what the user really wants and comes up with a specification for the system. *CS3720* comes to my mind....

*Database designers* further specify the database structures that keep the state and support the needed queries and updates and *application programmers* implement the GUI part, and all the appropriate transactions in the system, topics of this and some subsequent courses.

*Project managers* are responsible for the overall success of such a project. *CS 4140 SE?*
Any such a system must be accepted by its *Users*. User interface is the key, which must be appropriate for their capabilities. You might learn something from *CS3820 HCI*, recently becoming required for IT majors; maybe desirable for CS majors, too.

A *database administrator* maintains database once system is operational in terms of space allocation, performance optimization, database security, etc. Check out the relevant link in the course page on how to become an Oracle DBA.

A *system administrator* maintains transaction processing system by monitoring interconnection of hardware and software modules, deals with failures and congestion.

Are you taking *CS4230 System Administration* this semester? Now, you have to learn both *linux* and *Windows.* 😊
Database applications

While a transaction system is to use database to maintain an accurate model of some real world situation; a decision support system is to use a database to guide management decisions.

For example, in a supermarket company, each local supermarket maintains a database that keeps the price and inventory of all the items it sells.

On the other hand, the managers of the whole chain will have to analyze the local databases to help them make the decisions for the chain as a whole, e.g., what are selling, and where, so that more, or less, can be sent there.

In general, we use an on-line analytical processing system (OLAP) to make analysis of information in a database for the purpose of making management decisions.
Data warehouse

Such a system can analyze a(n) (h)uge quantity of historical data using complex queries with lots of statistical analysis.

Due to volume of data and complexity of queries, OLAP often uses a data warehouse, e.g., sales and inventories from various stores in the last 10 years, perhaps updated once a day.

With such data, a manager can enter a complicate query, e.g., “During the winter months of the last five years, what is the percentage of customers in the Northeast urban supermarkets who bought crackers while buying soup?”
Big data and data mining

A manager can also use warehouse data to discover relationships that might influence enterprise strategy. For example, “Are there any interesting purchasing patterns of our customers?”

Instead of asking for specific facts, data mining tries to discover some knowledge or a pattern in a general nature, based on an analysis of the collected data. It is complicated, but sometimes leads to interesting and inspiring findings.

This has recently been picking up in terms of “big data” since we have got a (h)uge amount of data in our finger tips. Why don’t we make a good use of them?

We are offering a big data course right now, which will be required for IT majors one year from now.
An interesting pattern: A high percentage of male customers who bought diapers also bought beer.

Question: What to do?

What else?

Frequently Bought Together

Check out the stuff in the course page.