Database Access via JDBC

- The Java Database Connectivity (JDBC) interface enables any Java program to send SQL queries to any database, and receive back result tables with the desired data.

- Similar to the basic idea of Java in writing a program that will run on any hardware platform, JDBC enables the development of programs which function with nearly all commercially available DBMSs. Apart from the general popularity of Java, this is the fundamental reason for the widespread acceptance of JDBC.

- In order to guarantee the general database access, JDBC defines a certain core functionality supported by all DBMSs, This common denominator can be implemented by JDBC.
  - This implies that different product characteristics and manufacturer-specific optimizations are ignored by the JDBC standard.
Database Access via JDBC (cont.)

- One prerequisite for the use of JDBC is the availability of a JDBC driver for the database being utilized.

- The JDBC driver translates the JDBC queries of the Java database client into the respective supplier-specific calls.

- The simplest version on the Windows platform is the Open Database Connectivity (ODBC) interface. ODBC also enables different databases to function via a uniform interface.

- JDBC and ODBC are both based on the same idea. Using the JDBC-ODBC bridge, it is possible to access an ODBC data source via JDBC.
Different Methods for Database Access

- The figure on the next slide summarizes the various options available for accessing a database.

- The client software typically communicates with the server via a proprietary interface. The drivers translate JDBC or ODBC commands into the respective database specific calls.

- The user can also access the database using an SQL specific tool.

- One disadvantage of the ODBC solution is that every computer on which a Java database application is to run, the ODBC connection must be configured. This contradicts the Java principle “write once, run anywhere”. The way around this is through the use of servlets in which the only computer on which the ODBC must be configured is the one on which the servlet engine will run. If a database application is installed on several computers or distributed as an applet, the JDBC-ODBC bridge is not an option as the ODBC connection would have to be configured on every computer.
Summary of Database Access Methods

- SQL Tool
- JDBC
- ODBC
- Database
- DB-specific API
- JDBC Database Driver
- ODBC Database Driver
- JDBC-ODBC Bridge
- Java Database Client
- JDBC API
- ODBC API
For the time being we will focus on the JDBC API for database access and not concern ourselves with ODBC (we’ll look in more detail at ODBC later).

JDBC is almost always used with a RDBMS. However, it can be used with any table-based data source. This means that it also works with applications like Excel.

The separation of the JDBC API from the particular database drivers enables the application developer to change the underlying database without modifying the Java code that accesses the database.

Most commercially available RDBMSs provide JDBC drivers and there are many third-party JDBC drivers available.

We will focus on the JDBC and use it to manipulate a MySQL database. We’ll discuss JDBC in more detail later.
MySQL RDBMS

- MySQL is a **database server** (although it does come with a set of simple client programs). The current version is 4.1.9 and can be downloaded from [www.mysql.com](http://www.mysql.com).

- It is typically used in **thin client** environments. In other words, it is used in client-server systems where the bulk of the processing and storage takes place on the server, and the client is little more than a dumb terminal.

- MySQL performs multithreaded processing, which means that multiple clients are allowed to connect to it and run queries simultaneously. This makes MySQL extremely fast and well suited to client-server environments such as Web sites and other environments that process numerous transactions for multiple users.
MySQL features a user permissions system, which allows control over user’s access to the databases under MySQL control.

There are very few competitors of MySQL (Oracle, Sybase, DB2, and SQL Server) that can match the level of sophistication provided by MySQL’s permissions system in terms of granularity and level of security provided.

Note that I did not include Microsoft Access in the list above. There are a couple of reasons for this; Access concentrates on the client front-end, although available in shareable versions, it lacks the management system that is a key part of any RDBMS. Access provides virtually no user authentication capabilities nor does it have multithreading processing capabilities, in its normal form.
Starting MySQL Server

• On Windows, MySQL runs as a server program, which means that it is a background process that sits patiently waiting for client connections.

• To start the MySQL server, open a command window, switch to the bin directory of your MySQL directory and enter `mysqld --console`. (Omitting `--console` sends error messages to .err file in data directory.)

InnoDB is MySQL’s ACID compliant storage engine. Server starts.

Specifying `--console` prevents error message from going to .err file.

Server finishes its startup sequence.
Starting MySQL Server (cont.)

• Once the MySQL server is started, open another command window in which to run the client process. The command window in which the MySQL server is executing will continue to display error messages, if any, but otherwise will not return a user prompt until the server exits.

• You can stop the MySQL server by executing the following command from the client window:

  `mysqladmin -u root shutdown`

• The next page illustrates the execution of this command from the client window and its effect on the MySQL server.
Stopping MySQL Server

Client Window

A user must have the privilege of stopping the server. The root user has this permission by default.

Server Window

MySQL server responds with normal shutdown sequence and responses.
Running MySQL Client Programs Under Windows

- You can test whether the MySQL server is working by executing any of the following commands:
  - `C:\mysql\bin\mysqlshow`
  - `C:\mysql\bin\mysqlshow -u root`
  - `C:\mysql\bin\mysqlshow -u root mysql`
  - `C:\mysql\bin\mysqladmin version status proc`
  - `C:\mysql\bin\mysql test`

- An example of the first two formats is shown on the next slide.
Note: The MySQL root user is not necessarily the same as the root user in Linux/Unix systems (it might be, but it doesn’t have to be). All MySQL requires is that you have the necessary permission to execute the command entered. For right now we’ll assume that the only user is the root user. We’ll set permissions later.
This command allows you to see the current status of the MySQL server, in terms of the work being done.
Starting A MySQL Client

- Once the MySQL server is started, open another command window in which to run the client process.

- You start a direct MySQL client application by executing the following command from the client window:

  ```shell
c:\mysql\bin> mysql
  ```

  MySQL is awaiting commands.
Starting A MySQL Client (cont.)

Start sequence for a non-root user

Start sequence for root user
Specifying A Database Within MySQL

- Unless, it is specifically stated, in the following slides we’ll assume that the user has root-level privileges.

- To select a database for use in MySQL the `use` command must be issued. In the example below, we’ll select the `bikedb` database.

```
mysql> use bikedb;
Database changed
```

SQL acknowledges selection of `bikedb` database.
Viewing the Relations of a Database

• Once a database has been selected you can see the relations (tables) within that database with the `show tables` command as illustrated below.

Show tables command lists all the relations within a database visible to the user. There is only a single table in this database.
Viewing the Schema of a Relation

• To see the schema of a relation within a database, use the `describe < tablename >` command as illustrated below.

![Command Prompt (2) - mysql -u root -p](image)

Specify which table’s schema to describe. All information regarding the schema visible to the user is displayed.
Running a Simple Select Query in MySQL

• Within the MySQL monitor, running an SQL query is straightforward. The example below illustrates a simple selection query on the bikes table of the bikedb database.

```
mysql> describe bikes;
+-------+--------+------+-----+---------+-------+
| Field | Type   | Null | Key | Default | Extra |
+-------+--------+------+-----+---------+-------+
| bikename | varchar(30) | YES | PRI | NULL |       |
| size | int(2) | YES | | NULL |       |
| color | varchar(15) | YES | | NULL |       |
| cost | int(5) | YES | | NULL |       |
| purchased | date | YES | | NULL |       |
| mileage | int(6) | YES | | NULL |       |
+-------+--------+------+-----+---------+-------+
6 rows in set (0.00 sec)

mysql> select * from bikes;
<table>
<thead>
<tr>
<th>bikename</th>
<th>size</th>
<th>color</th>
<th>cost</th>
<th>purchased</th>
<th>mileage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Colnago Dream Rabobank</td>
<td>60</td>
<td>blue/orange</td>
<td>5500</td>
<td>2002-06-27</td>
<td>4300</td>
</tr>
<tr>
<td>Bianchi Evolution 3</td>
<td>58</td>
<td>celeste</td>
<td>4800</td>
<td>2003-11-16</td>
<td>2000</td>
</tr>
<tr>
<td>Eddy Merckx Molteni</td>
<td>58</td>
<td>orange</td>
<td>5100</td>
<td>2004-08-12</td>
<td>0</td>
</tr>
<tr>
<td>Eddy Merckx Domo</td>
<td>58</td>
<td>blue/black</td>
<td>5300</td>
<td>2004-02-02</td>
<td>0</td>
</tr>
<tr>
<td>Battaglin Carrera</td>
<td>60</td>
<td>red/white</td>
<td>4000</td>
<td>2001-03-14</td>
<td>11200</td>
</tr>
<tr>
<td>Gianni Motta Personal</td>
<td>59</td>
<td>red/green</td>
<td>4400</td>
<td>2000-05-01</td>
<td>8700</td>
</tr>
</tbody>
</table>
+------------------|------|-------------|-------|--------------|---------+
6 rows in set (0.00 sec)
```

The tuples within the bikes table are displayed as the result of the query.
Creating a Database in MySQL

• There are two ways to create a MySQL database.

  1. From a client command window execute the `mysqladmin` script `create` and specify the name of the database.

Specify creation of new database named `sample`

Subsequent listing of databases shows newly created `sample` database.
Creating a Database in MySQL (cont.)

2. From the MySQL monitor enter `create database <db name>`
Dropping a Database in MySQL

• There are two ways to destroy a MySQL database.

1. From a client command window execute the `mysqladmin script drop` and specify the name of the database.

   ```
   C:\mysql\bin>mysqladmin drop sample -u root
   Dropping the database is potentially a very bad thing to do. Any data stored in the database will be destroyed.
   Do you really want to drop the 'sample' database [y/N] y
   Database "sample" dropped
   C:\mysql\bin>
   ```

   MySQL gives you a warning and a bailout before the database is destroyed.

   Specify dropping the database named sample.
Dropping a Database in MySQL

2. From the MySQL monitor execute the `drop database <db name>` command.

From within the MySQL monitor, no warning is given when dropping a database. Be very sure that this is what you want to do before you do it.
Manipulating Tables in MySQL

- The creation of a database does not place into the database any relations. Relations must be separately created.

- To create a table within a database, first select the database (or create one if you haven’t already done so), then execute the `create table` command.
Manipulating Tables in MySQL (cont.)

Screen shot showing newly created table.
Manipulating Tables in MySQL (cont.)

- The `create table` command has the following general format:

  ```
  create [temporary] table
  [if not exists] tablename
  [(create_definition, ...)]
  [table_options] [select_statement];
  ```

- If the `[if not exists]` clause is present, MySQL will produce an error message if a table with the specified name already exists in the database, otherwise the table is created.
Manipulating Tables in MySQL (cont.)

- A temporary table exists only for the life of the current database connection. It is automatically destroyed when the connection is closed or dies.

- Two different connections can use the same name for a temporary table without conflicting with one another.

- Temporary tables are most useful when queries get complex and intermediate results become useful. Also, versions of MySQL earlier than version 4.1 do not have subselect capability and temporary tables are a convenient way to simulate subselect query results.

Note: Non-root users require special permission to be able to create temporary tables. These users must have the Create_tmp_tables privilege set in the user grant table. We'll see more on this later.
Creating A Temporary Table From A Select Query

A SELECT query produces a result set which has been extracted from one or more tables. A table can be created with the results of this data using the `create table` command.

Notice that temporary tables do not appear in a table listing.