Why do we have databases?
Four perspectives on the database

- Large data store
- Persistent data store
- Query service
- Transaction service
Programmer’s view

- Not
  - how it works
  - how to administer it
  - how to design a database

- Database services
  - persistent storage
  - sophisticated querying
  - transactions
JDBC

- Group of Java classes
- Correspond to basic database concepts
  - Connecting to the database
  - Issuing a query
  - Examining results
Higher-level option

- Enterprise JavaBeans
  - simply state that an object is going to be persistent
  - the EJB “container” uses JDBC to save the object in a database
  - SE 554
<table>
<thead>
<tr>
<th>Row #</th>
<th>ID</th>
<th>Last name</th>
<th>First name</th>
<th>Car type</th>
<th>Days</th>
<th>Fuel Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1234</td>
<td>Burke</td>
<td>Robin</td>
<td>5</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>5678</td>
<td>Q</td>
<td>Suzy</td>
<td>3</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>2020</td>
<td>Flintstone</td>
<td>Fred</td>
<td>5</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3232</td>
<td>Presley</td>
<td>Elvis</td>
<td>4</td>
<td>12</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>3434</td>
<td>Presley</td>
<td>Aaron</td>
<td>1</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>9999</td>
<td>Lennon</td>
<td>John</td>
<td>0</td>
<td>50</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>9998</td>
<td>McCartney</td>
<td>Paul</td>
<td>3</td>
<td>10</td>
<td>1</td>
</tr>
</tbody>
</table>
### Key relationships

<table>
<thead>
<tr>
<th>ID</th>
<th>Last name</th>
<th>First name</th>
<th>Car type</th>
<th>Days</th>
<th>Fuel Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>Burke</td>
<td>Robin</td>
<td>5</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>5678</td>
<td>Q</td>
<td>Suzy</td>
<td>3</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>2020</td>
<td>Flintstone</td>
<td>Fred</td>
<td>5</td>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Subcompact</td>
</tr>
<tr>
<td>1</td>
<td>Compact</td>
</tr>
<tr>
<td>2</td>
<td>Mid-size</td>
</tr>
<tr>
<td>3</td>
<td>Full size</td>
</tr>
<tr>
<td>4</td>
<td>Luxury</td>
</tr>
<tr>
<td>5</td>
<td>SUV</td>
</tr>
</tbody>
</table>
Basic terms

- **Table**
  - basic unit of organization

- **Row / Record**
  - single “entry”

- **Column**
  - named attributed of a row

- **View**
  - a dynamically created table

- **Query**
  - an operation on the database
    - typically retrieval of a view

- **Record set / result set / row set**
  - the information returned by a query
SQL

- Declarative language
  - not procedural
- Describe what you want
  - database figures out “how”
- In general
  - do as much in SQL as you can
Query

SELECT LastName, FirstName
FROM Reservations
Sorting

SELECT * FROM Reservations
ORDER BY Reservations.LastName;
Choosing

```sql
SELECT * FROM Reservations
WHERE (LastName = 'Presley')
AND (CarType = 1)
```
SQL Language

- Case insensitive
- Strings enclosed in single quotes
- Whitespace ignored
- Commands for
  - (creating and structuring databases)
  - retrieving data
  - inserting rows
  - deleting rows
- Different versions
  - SQL-92 most widely supported
SELECT statement

SELECT { columns } 
FROM { table(s) } 
WHERE { criteria } 
... other options ... ;
Whole table

SELECT *
FROM Reservations
SELECT * FROM Reservations
WHERE (LastName = 'Presley')
AND (CarType = 1)
Complex criteria

```sql
SELECT titles.type, titles.title
FROM titles t1
WHERE titles.price >
    (SELECT AVG(t2.price)
     FROM titles t2
     WHERE t1.type = t2.type)
```
Note

- Dot syntax for columns
  - `table.column`
  - `titles.type`

- Table aliases
  - `table alias`
  - `titles t1`
Join

SELECT Reservations.ID, Reservations.LastName, Reservations.FirstName, CarType.Description, Reservation.Days, FuelOption.Description
FROM Reservations, CarType, FuelOption
WHERE Reservations.CarType = CarType.ID
    AND Reservations.FuelOption = FuelOption.ID
## Join Example

<table>
<thead>
<tr>
<th>ID</th>
<th>Last name</th>
<th>First name</th>
<th>Car type</th>
<th>Days</th>
<th>Fuel Option</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>Burke</td>
<td>Robin</td>
<td>5</td>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>SUV</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pre-paid fuel</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R.ID</th>
<th>R.LastName</th>
<th>R.FirstName</th>
<th>C.Description</th>
<th>R.Days</th>
<th>F.Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1234</td>
<td>Burke</td>
<td>Robin</td>
<td>SUV</td>
<td>7</td>
<td>Pre-paid fuel</td>
</tr>
</tbody>
</table>
Join example cont’d

- What if?

<table>
<thead>
<tr>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>SUV</td>
</tr>
<tr>
<td>5</td>
<td>Sport Ut.</td>
</tr>
</tbody>
</table>
JDBC
JDBC

- DriverManager
  - knows how to connect to different DBs
- Connection
  - represents a connection to a particular DB
- Statement
  - a query or other request made to a DB
- ResultSet
  - results returned from a query
Loading the driver

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

- What is this?
Creating a connection

```java
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
String url = "jdbc:odbc:mydatabase";
Connection conn = DriverManager.getConnection(url,
    "UserName", "Password");

- Why not "new Connection ()"?
Property file

jdbc.drivers=sun.jdbc.odbc.JdbcOdbcDriver
jdbc.url=jdbc:odbc:rentalcars
jdbc.username=PUBLIC
jdbc.password=PUBLIC
Using properties

```java
public static Connection getConnection() throws SQLException, IOException {
    Properties props = new Properties();
    String fileName = "rentalcars.properties";
    FileInputStream in = new FileInputStream(fileName);
    props.load(in);

    String drivers = props.getProperty("jdbc.drivers");
    if (drivers != null)
        System.setProperty("jdbc.drivers", drivers);

    String url = props.getProperty("jdbc.url");
    String username = props.getProperty("jdbc.username");
    String password = props.getProperty("jdbc.password");
    return DriverManager.getConnection(url, username, password);
}
```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
String url = "jdbc:odbc:mydatabase";
Connection conn = DriverManager.getConnection(url,
    "UserName", "Password");
Statement stmt = conn.createStatement();
Executing a statement

```java
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
String url = "jdbc:odbc:mydatabase";
Connection conn = DriverManager.getConnection(url,
    "UserName", "Password");
Statement stmt = conn.createStatement();
String sql = "SELECT * FROM Reservations
    WHERE (LastName = 'Burke');";
stmt.execute (sql);
```
ResultSet

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
String url = "jdbc:odbc:mydatabase";
Connection conn = DriverManager.getConnection(url,
        "UserName", "Password");
Statement stmt = conn.createStatement();
String sql = "SELECT * FROM Reservations
       WHERE (LastName = 'Burke');";
ResultSet rs = stmt.executeQuery (sql);
rs.next();
int carType = rs.getInt ("CarType");
Iteration using ResultSet

```java
int totalDays = 0;
String sql = "SELECT * FROM Reservations;";
ResultSet rs = stmt.executeQuery(sql);
while (rs.next())
{
    totalDays += rs.getInt("Days");
}
```
Better solution

```java
int totalDays = 0;
String sql = "SELECT SUM(Days) FROM Reservations;";
ResultSet rs = stmt.executeQuery(sql);
rs.next();
totalDays = rs.getInt(1);
```
Example

- Connection
- Statement
- ResultSet
- Properties / property file