ID1354
Internet Applications
Relational Databases

Leif Lindbäck, Nima Dokoohaki

leifl@kth.se, nimad@kth.se

SCS/ICT/KTH

13.1 Relational Databases

- A database is a collection of data organized to allow access for retrievals, additions, and deletions.
- A relational database is a collection of tables of data, each of which has one or more columns. One of the columns stores the primary keys of the table.
- The primary key is identifies a row. It must be unique within a table.

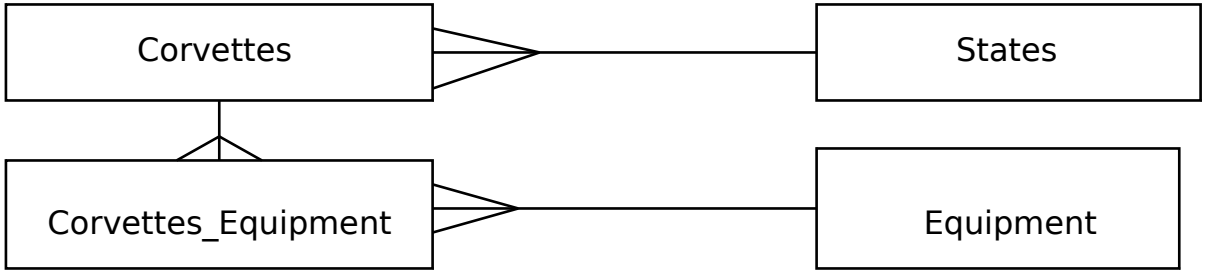
13.1 The *Used Corvette* Sample Database

- Could just put all data in a single table, whose key would be a simple sequence number

- The table could have information about various equipment the cars could have

- Better to put the equipment in a different table and use a *cross-reference table* to relate cars to equipment

13.1 The *Used Corvette* Sample Database



The `Corvettes` table

Vette_id	*Body_style*	*Miles*	*Year*	*State*
1	coupe	18.0	1997	4
2	hatchback	58.0	1996	7
3	convertible	13.5	2001	1
4	hatchback	19.0	1995	2
5	hatchback	25.0	1991	5
6	hardtop	15.0	2000	2
7	coupe	55.0	1979	8
8	convertible	17.0	1999	5
9	hardtop	17.0	2000	5

The `States` table

State_id	*State*
1	Alabama
2	Alaska
3	Arizona
4	Arkansas
5	California
6	Colorado
7	Connecticut
8	Delaware
9	Florida

The `Equipment` table

Equip_id	*Equipment*
1	Automatic
2	4-speed
3	5-speed
4	6-speed
5	CD
6	Leather

`Corvettes-Equipment` cross-reference table

Vette_id	Equip
1	1
1	5
1	6
2	1
2	5
2	6
3	1
3	6
4	2
4	6
5	1
5	6
6	2
7	4
7	6
8	4
8	5
8	6
9	4
9	5
9	6

13.2 Intro to SQL

- The `SELECT` Command

  - Used to specify queries
  - Three clauses: `SELECT`, `FROM`, and `WHERE`

- General form:

  `SELECT` *column names*
  `FROM` *table names*
  `WHERE` *condition*

```
SELECT Body_style FROM Corvettes WHERE Year > 1994
```

13.2 Intro to SQL (continued)

- *Joins*

  - If you want all cars that have CD players, you need information from two tables, `Corvettes` and `Equipment`

  - `SELECT` can build a *temporary table* with info from two tables, from which the desired results can be gotten - this is called a *join* of the two tables

  - A `SELECT` that does a join operation specifies two tables in its `FROM` clause and also has a compound `WHERE` clause

13.2 Intro to SQL (continued)

- For our example, we must have three `WHERE` conditions

1. The `Vette_id` column from the `Corvettes` table and the `Corvettes_Equipment` table must match

2. The `Equip` column from the `Corvettes_Equipment` table must match the `Equip_id` column from the `Equipment` table

3. The `Equip` column from the `Equipment` table must have the value `'CD'`

13.2 Intro to SQL (continued)

- *Joins* (continued)

```
SELECT Corvettes.Vette_id,
       Corvettes.Body_style,
       Corvettes.Miles,
       Corvettes.Year, Corvettes.State,
       Equipment.Equip
FROM Corvettes, Equipment,
     Corvettes_Equipment
WHERE Corvettes.Vette_id =
                Corvettes_Equipment.Vette_id
   AND Corvettes_Equipment.Equip =
                Equipment.Equip_id
   AND Equipment.Equip = 'CD'
```

13.2 Intro to SQL (continued)

This query produces:

VETTE_ID	BODY_STYLE	MILES	YEAR	STATE	EQUIP.
1	coupe	18.0	1997	4	CD
2	hatchback	58.0	1996	7	CD
8	convertible	17.0	1999	5	CD
9	hardtop	17.0	2000	5	CD
10	hatchback	50.0	1995	7	CD

- To get the state's names:
  1. Replace `Corvettes.State` with `States.State` in the `SELECT` clause
  2. Add `States` to the `FROM` clause
  3. Add `AND Corvettes.State_id = States.State_id` to the `WHERE` clause

13.2 Intro to SQL (continued)

- The **INSERT** Command

INSERT INTO *table_name* (*col_name*$_1$, … *col_name*$_n$)

VALUES (*value*$_1$, …, *value*$_n$)

- The correspondence between column names and values is positional

```
INSERT INTO Corvettes(Vette_id, Body_style,
Miles, Year, State)
VALUES (37, 'convertible', 25.5, 1986, 17)
```

13.2 Intro to SQL (continued)

- The **UPDATE** Command
- To change one or more values of a row in a table

```
UPDATE table_name
   SET col_name1 = value1,
   …
            col_namen = valuen
 WHERE col_name = value
```

- The **WHERE** clause identifies the row to be updated

13.2 Intro to SQL (continued)

- The **UPDATE** Command (continued)

- Example:

```
UPDATE Corvettes
SET Year = 1996
WHERE Vette_id = 17
```

13.2 Intro to SQL (continued)

- The **DELETE** Command

- Example:

```
DELETE FROM Corvettes
WHERE Vette_id = 27
```

- The **WHERE** clause could specify more than one row of the table

13.2 Intro to SQL (continued)

- The **DROP** Command

  - To delete whole databases or complete tables

```
DROP (TABLE | DATABASE) [IF EXISTS] name
```

```
DROP TABLE IF EXISTS States
```

13.2 Intro to SQL (continued)

- The **`CREATE TABLE`** command:

`CREATE TABLE` ***`table_name`*** **`(`**

`column_name`$_1$ ***`data_type constraints,`***

…

`column_name`$_n$ ***`data_type constraints`***`)`

- There are many different data types (**`INTEGER`**, **`REAL`**, **`CHAR(length)`**, **`…`**)

13.2 Intro to SQL (continued)

- There are several constraints possible

  e.g., **NOT NULL, PRIMARY KEY**

```
CREATE TABLE States (
   State_id  INTEGER  PRIMARY KEY  NOT
    NULL,
  State CHAR(20))
```

13.3 Architectures for Database Access

- *PHP & Database Access*

- An API for each specific database system

- Convenient for Web access to databases.

13.4 The MySQL Database Syste

- A free, efficient, widely used SQL implementation

- Available from http://www.mysql.org

- Logging on to MySQL (starting it):

```
mysql [-h host] [-u username]
[database name] [-p password]
```

13.4 The MySQL Database System (continued)

- Host is the name of the MySQL server
  - Default is the user's machine
  - Username is that of the database
  - Default is the name used to log into the system
  - The given database name becomes the "focus" of MySQL

- If you want to access an existing database, but it was not named in the `mysql` command, you must choose it for focus `use cars;`

  - Response is: `Database changed`

13.4 The MySQL Database System (continued)

- To create a new database,

```
CREATE DATABASE cars;
```

- Response:

```
Query ok, 1 row affected (0.05 sec)
```

13.4 The MySQL Database System (continued)

- Example:

```
CREATE TABLE Equipment
    (Equip_id INT UNSIGNED NOT NULL
  AUTO_INCREMENT PRIMARY KEY,
     Equip INT UNSIGNED
     );
```

- To see the tables of a database:

```
SHOW TABLES;
```

- To see the description of a table (columns):

```
DESCRIBE Corvettes;
```

13.5 Database Access with PHP/MySQL

- When values from a DB are to be put in HTML, you must worry about HTML special characters, e.g., < and >

- To convert these characters to HTML entities, use the function, `htmlspecialchars($str)`.

- Returns the content of `$str`, with HTML special characters converted to entities.

- Another problem with PHP and HTML forms is the string special characters (*', ", \, and NULL*), which could come from `$_GET` and `$_POST`

13.5 Database Access with PHP/MySQL

- When using HTTP parameters in database calls, we might get problems with the characters (`'` `"` `\` and `NULL`).
- These charactares can come from `$_GET` and `$_POST` data.
- To escape these characters, use the function `mysql_real_escape_string($str)`

13.5 Database Access with PHP/MySQL (continued)

- To connect PHP to a database, use `mysql_connect`, which can have three parameters:

1. *host* (default is localhost)
2. *Username* (default is the username of the PHP script)
3. *Password* (default is blank, which works if the database does not require a password)

```
$db = mysql_connect();
```

- Close the connection to the database with `mysql_close`

13.5 Database Access with PHP/MySQL (continued)

- To focus MySQL,

```
mysql_select_db("cars");
```

- Call `mysql_query` with a string parameter, which is an SQL command

```
$query = "SELECT * from States";
$result = mysql_query($query);
```

13.5 Database Access with PHP/MySQL (continued)

- Dealing with the result:

  - Get the number of rows in the result

```
$num_rows = mysql_num_rows($result);
```

  - Get the number of fields in the result

```
$num_fields = mysql_num_fields($result);
```

  - Get a row of the result

```
$row = mysql_fetch_assoc($result);
```

13.5 Database Access with PHP/MySQL (continued)

- Display the column names

```
$keys = array_keys($row);
for ($index = 0; $index < $num_fields; $index++){
    print $keys[$index] . "   ";
}
```

13.5 Database Access with PHP/MySQL (continued)

- Display the values of the fields in the rows

```
$num_rows = mysql_num_rows($result);
$num_fields = mysql_num_fields($result);
$row = mysql_fetch_assoc($result);
for ($row_num = 0; $row_num < $num_rows; $row_num++) {
    $values = array_values($row);
    for ($index = 0; $index < $num_fields; $index++) {
        $value = htmlspecialchars($values[$index]);
        print $value . "  ";
    }
    print "<br />";
    $row = mysql_fetch_assoc($result);
}
```