CS 252: Fundamentals of Relational Databases: SQL1
Dr. Alexandra I. Cristea
http://www.dcs.warwick.ac.uk/~acristea/

Lecturers & Practical Work

- Hugh Darwen
- Alexandra I. Cristea
- Zabin Visram
- Adrian Hudnott

- Other invited talks?: TBA

Schedule

- Usual:
  - Mon 14:00-15:00, 15:00-16:00 seminars (Z. Visram)
  - Tue 14:00-15:00 (Fundamentals, H. Darwen)
  - Thu 13:00-14:00 (SQL, A. Cristea)
  - Fri 13:00-14:00 (H. Darwen)

- Exceptions:
  - Others: TBA; check forum, website, course

Slides, acknowledgements and thanks

- Thanks to:
  - Mr. Tim Heron
  - Dr. Richard Cartwright
    (former undergraduate, doctoral student and lecturer at Warwick)
  - Dr. Paul Goldberg
    http://www.dcs.warwick.ac.uk/people/academic/Paul.Goldberg/cs190cs319index.html
  - Dr. Meurig Beynon
    http://www.dcs.warwick.ac.uk/people/staff/Meurig.Beynon
  - Prof. Dr. Paul De Bra:
    http://wwwis.win.tue.nl/~debra/
  - Others: mentioned directly

Contact

- Forum:
  http://forums.warwick.ac.uk/wf/browse/category.jsp?cat=24
- IF (and ONLY IF) a question is personal, you might address send personal email to HD or AIC
  - FORMAT: subject of email should contain 'CS252' and topic of the email (otherwise it will be filtered out)

Course site(s):

- Current:
  - http://www.dcs.warwick.ac.uk/~acristea/courses/CS252/
  - Will contain current slides, as taught at the course
  - Will contain notifications: check BEFORE & AFTER the course
- Official:
  - http://www.dcs.warwick.ac.uk/undergraduate/modules/cs252.html
**Books**

- C. Date, "An Introduction to Database Systems", Addison-Wesley Longman, 2003 (8th ed.)
- C. Date, "Database in Depth: The Relational Model for Practitioners", O'Reilly, 2005
- C. Date, "The Relational Database Dictionary", O'Reilly, 2006
- Oracle and JDBC references:
- Links to Oracle documentation are on the course website.

**Purpose of this course**

- To show how the theory of relational algebra serves as a framework and a foundation for the efficient organisation and retrieval of large amounts of data.
- To introduce students to some standard notations (for example, SQL and Tutorial D) that implement important parts of relational algebra.
- To give students practical experience of the use and limitations of some database notations (such as SQL) that are widely used in industry and business.

**Overlaps and sequencing**

- Prerequisite of
  - CS253: Topics in Database Systems

**Organization of the course**

- 7.5 CATS
- CS, CSE, CBS, Mathematics
- 14 one-hour lectures and 5 one-hour seminars
- Exam 1.5 hours (70%) coursework (30%)
- Rules of the game:
  - Read also comments on the slides.
  - Presence is optional, but beware: slides-only are NOT ENOUGH to learn from for the exam; you need to participate, take your own notes, read more: so self-study!
  - Go to seminars to practice! Do problems on worksheets at home. Participate in the forum.

**Goal SQL Lectures:**

Lectures with a practical emphasis:
- Using SQL

**Contents:**

- Introduction, getting started with Oracle, simple queries
- Formulating queries, inserting – deleting – modifying rows
- Creating and altering tables, relational algebra, constraints
- Views, functions, dates & times
This lecture

- Introduction
- Module outline
- History of SQL
- Accessing Oracle
- Some SQL!

Worksheets

Two worksheets (numbered 2 and 5) will be given at this lecture and in the lecture in week 5. Working through each worksheet and using a database between each lecture is an important part of the course. Each week's lectures build on the week before plus the worksheet. The questions in the assignment will take a form similar to the worksheets. This is an introductory course.

Worksheet 2: Getting Started with Oracle

Databases Available at Warwick

IT Services Oracle Database

All examples from these lectures and the worksheets are "verified" with the Oracle 9i database on mimosa. Some information on basic Oracle use is available via the module webpage.

Databases available at home

- Personal Oracle 9i (downloadable from http://www.oracle.com/technology/software/products/oracle9i/)
- Open source includes MySQL and PostgreSQL
  - MySQL (http://www.mysql.com)
  - PostgreSQL (http://www.postgresql.org/)
- Microsoft SQL Server 2005 (a free version called SQL Server Express is available: http://www.microsoft.com/sql/editions/express/default.mspx)
- Microsoft Access (a long way away from ISO SQL standard)

Background to SQL

- SQL abbreviation for Structured Query Language.
- Original name was SEQUEL, correct modern pronunciation is S-Q-L.
- Used for relational databases; where relations are tables, attributes are columns, tuples are rows …
- Chronology:
  - 1970s IBM - first relational database System R, then DB2. Others include: Ingres Database - query language QUEL
  - Digital - Relational Database Operator
  - ISBL - relational algebra DML
dbase family of products for PCs
- 1979 Oracle
- 1980s First standardisation efforts.

- 1984 ISO SQL standard - many flaws but universally adopted.
- 1992 Update to standard called SQL92 - The basic standard for any modern database
- 1999 Update to standard called SQL99 - Oracle database conforms to SQL99.
- 2003/6/7 Current standard SQL2006 - Not many databases fully support this standard yet.
- Major benefit: Virtually all relational databases can be manipulated using the same language.
- SQL combines:
  - Data Description Language (DDL) - how the tables represent the data
  - Query and data manipulation (DML)
  - 2008? next
- However, there are many non-standard extensions to SQL:
  - PL/SQL - Oracle Procedural SQL
  - DB2 Procedural SQL - IBM DB2
  - Transact-SQL - Microsoft SQL Server
  - To write portable SQL stick to standard SQL.

Database M. Systems (DBMS)
Data is persistent - each user has their own database space
- stored tables and data will remain unless modified or dropped.
Oracle uses an SQL interpreter called sqlplus as the main interface to the DBMS:
  - Standard SQL operations
  - Report generation

Oracle at Warwick
Located on server mimosa (mimosa.csv.warwick.ac.uk).
1. Login to mimosa using ssh. ITS usercodes apply.
   - Only use mimosa for database access.
2. Type "orasetup". This is a script that modifies your .bash profile or .profiles by adding information required for running Oracle.
3. Log out of mimosa and log back in again.
4. To run the text SQL interpreter, type "sqlplus /". Remember the "/" as it represents your personal username and password for Oracle.

Creating a Table
From the depths of a CD collection:

<table>
<thead>
<tr>
<th>Collection</th>
<th>artist</th>
<th>album</th>
<th>tracks</th>
<th>company</th>
<th>year</th>
</tr>
</thead>
<tbody>
<tr>
<td>U2</td>
<td>The Unforgettable Fire</td>
<td>10</td>
<td>Island</td>
<td>1984</td>
<td></td>
</tr>
<tr>
<td>U2</td>
<td>Rattle and Hum</td>
<td>17</td>
<td>Island</td>
<td>1988</td>
<td></td>
</tr>
<tr>
<td>U2</td>
<td>Achtung Baby</td>
<td>12</td>
<td>Island</td>
<td>1991</td>
<td></td>
</tr>
<tr>
<td>Underworld</td>
<td>Second Toughest in the Infants</td>
<td>8</td>
<td>Junior</td>
<td>1996</td>
<td></td>
</tr>
<tr>
<td>The Verve</td>
<td>Urban Hymns</td>
<td>13</td>
<td>Virgin</td>
<td>1997</td>
<td></td>
</tr>
<tr>
<td>Foo Fighters</td>
<td>The Colour and the Shape</td>
<td>13</td>
<td>Capital</td>
<td>1997</td>
<td></td>
</tr>
</tbody>
</table>

Defining the table in SQL:
```
CREATE TABLE Collection
(artist CHAR(16),
album CHAR(40),
tracks INTEGER,
company CHAR(16),
year INTEGER);
```

Note:
1. Convention to write SQL keywords in CAPITALS.
2. Oracle matches lower/upper case the same in table and column names.
3. ISO Standard - all table names and column names in capitals + numbers + underscore "_".
4. Semicolon ";" terminates every input (usually).

To insert the data into the table Collection use the INSERT INTO statement:
```
INSERT INTO Collection
VALUES ('U2', 'The Unforgettable Fire', 10, 'Island', 1984);
```
For every successful insertion, Oracle reports:
1 row created.
Otherwise an error message of the form:
*the line that caused the error*

ERROR at line 1: ORA-1438: value larger than specified precision allows for this column.

---

### Simple Queries

Retrieval from tables uses the SELECT clause.

**Order of statement:**
1. Attribute list
2. target list of tables
3. condition

**To view an entire base table:**

```sql
SELECT *
FROM Collection;
```

---

#### Single Column Selection

To view one column of a table:

```sql
SELECT artist
FROM Collection;
```

---

Output not necessarily unique. Add qualifier DISTINCT to achieve uniqueness:

```sql
SELECT DISTINCT artist
FROM Collection;
```

---

#### Multi-column Selection

To select more than one column from one table:

```sql
SELECT artist, album, year
FROM Collection;
```

This restricts the output to columns artist, album and year.

---

Results from a SELECT statement do not have a specified order. To sort the output into alphabetical order by artist name, use the **ORDER BY statement**:

```sql
SELECT artist, album, year
FROM Collection
ORDER BY artist;
```
Simple Restriction Conditions
Predicate = logical expression that must be satisfied (evaluate to true) for a row to be selected.
Select all albums from 1997 using the WHERE statement and "=" (equals):
\[
\text{SELECT artist, album, year} \\
\text{FROM Collection} \\
\text{WHERE year = 1997;}
\]
All data in left-hand side column must exactly match right-hand side expression.

<table>
<thead>
<tr>
<th>ARTIST</th>
<th>ALBUM</th>
<th>YEAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Verve</td>
<td>Urban Hymns</td>
<td>1997</td>
</tr>
<tr>
<td>Foo Fighters</td>
<td>The Colour and the Shape</td>
<td>1997</td>
</tr>
</tbody>
</table>

String Matching
Matching an identical string with "=":
\[
\text{SELECT artist, album} \\
\text{FROM Collection} \\
\text{WHERE artist = 'U2';}
\]
Select all artists beginning with the letter "U" using the LIKE statement and wildcard ":%":
\[
\text{SELECT artist, album} \\
\text{FROM Collection} \\
\text{WHERE artist LIKE 'U%';}
\]
Wildcard " " matches single characters. To escape wildcards use the @ symbol. So to match "10%" use "10@%". Can also use NOT LIKE.

<table>
<thead>
<tr>
<th>ARTIST</th>
<th>ALBUM</th>
<th>TRACKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>U2</td>
<td>The Unforgettable Fire</td>
<td>10</td>
</tr>
<tr>
<td>U2</td>
<td>Rattle and Hum</td>
<td>11</td>
</tr>
<tr>
<td>U2</td>
<td>Achtung Baby</td>
<td></td>
</tr>
<tr>
<td>Underworld</td>
<td>Second Toughest in the Infants</td>
<td>8</td>
</tr>
</tbody>
</table>

Additional Comparison Operators
To select everything below a certain numerical value, use ":<":
\[
\text{SELECT artist, album, tracks} \\
\text{FROM Collection} \\
\text{WHERE tracks < 11;}
\]
Similarly for:
• greater-than ":>"
• less-than-or-equal-to ":<="
• greater-than-or-equal-to ":>="
• two forms of not-equal-to "!=" and ":<>"

<table>
<thead>
<tr>
<th>ARTIST</th>
<th>ALBUM</th>
<th>TRACKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>U2</td>
<td>The Unforgettable Fire</td>
<td>10</td>
</tr>
<tr>
<td>Underworld</td>
<td>Second Toughest in the Infants</td>
<td>8</td>
</tr>
</tbody>
</table>

Can also use operator BETWEEN-AND to test inclusion within a range of values:
\[
\text{SELECT artist, album, year} \\
\text{FROM Collection} \\
\text{WHERE year BETWEEN 1980 AND 1992;}
\]

<table>
<thead>
<tr>
<th>ARTIST</th>
<th>ALBUM</th>
<th>YEAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>U2</td>
<td>The Unforgettable Fire</td>
<td>1984</td>
</tr>
<tr>
<td>U2</td>
<td>Rattle and Hum</td>
<td>1988</td>
</tr>
<tr>
<td>U2</td>
<td>Achtung Baby</td>
<td>1991</td>
</tr>
</tbody>
</table>

BETWEEN-AND is inclusive.
To leave sqlplus type "exit".

List Tables (from the catalog db)
Non-standard SQL, specific to each database.
To list the tables in your area in an Oracle system use :
\[
\text{SELECT TABLE_NAME FROM USER_TABLES;}
\]
To view definition of a table use:
\[
\text{DESCRIBE Collection;}
\]
SQL operators covered so far

CREATE TABLE Create a new, empty table.
INSERT INTO VALUES Insert a row of related data.
SELECT FROM Retrieve data from a table.
DISTINCT Unique selection of data.
ORDER BY Sort data into a particular order before display on selection.
WHERE Select a row from a table only if the data in that row satisfies a predicate expression.
LIKE % Wildcard string matching.
< > <= >= != <> Logical selection operators.
BETWEEN-AND Select within a certain range.

NULL
What if not all the data values are known?
• insert dummy values and change them later
• insert a marker - NULL
Consider example question 4 of worksheet 2:
... a new 11 track album called Leftism by Leftfield.
We know artist, album and tracks but not company or year.
The following SQL will fail:
INSERT INTO Collection VALUES ('Leftfield', 'Leftism', 11);
In this form of INSERT, values for all columns must be given.

Partial Inserts
To insert the incomplete information about the Leftfield album, we can use:
INSERT INTO Collection
VALUES ('Leftfield', 'Leftism', 11,
        NULL, NULL);
Alternatively, we can specify what data is given and in what order:
INSERT INTO Collection
(album, artist, tracks)
VALUES ('Leftism', 'Leftfield', 11);
Missing fields will be set to "NULL".

Logical Connectives
The usual connectives, AND and OR, are available in SQL.
SELECT artist, album
FROM Collection
WHERE artist LIKE '%e%'
        OR artist LIKE '%a%'
        AND tracks >= 10;
This command lists just the artist and album for the Underworld CD.
How about all CDs by artists with an "a" or an "e" in their names with 10 tracks or more?
Using Parentheses

Note how "AND" takes precedence over "OR". Use parentheses to indicate a different priority:

```
SELECT album, tracks
FROM Collection
WHERE (album LIKE '%y%' OR album LIKE '%S%')
AND tracks >= 10;
```

<table>
<thead>
<tr>
<th>ALBUM</th>
<th>TRACKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Urban Hymns</td>
<td>13</td>
</tr>
<tr>
<td>The Colour and the Shape</td>
<td>13</td>
</tr>
</tbody>
</table>

Testing for presence in a List

A list is of the form:
- numbers – (1, 2, 3)
- strings – (“U2”, “Underworld”, “The Verve”)  
In SQL, use keyword "IN" to test for presence in a list of values:

```
SELECT album, year
FROM Collection
WHERE year IN (1984, 1996, 1997);
```

This selects all albums from years 1984, 1996 and 1997.

To select all items by artists U2, Underworld and Foo Fighters, use query:

```
SELECT album, year
FROM Collection
WHERE artist IN ('U2', 'Underworld', 'Foo Fighters');
```

Subqueries

A subquery in SQL is a query (i.e., a SELECT expression) that appears inside another query. It is specified by placing parentheses around a query. Subqueries can be used in several different ways:
1. To denote a table e.g., as the source for INSERT INTO or an operand of FROM.
2. To denote a "single value" when the query yields a table with one column and one row.
3. To denote a "list of values", for use with IN when the result has one column and any number of rows.

Single Values

Subquery should return only one value:

```
SELECT album, year
FROM Collection
WHERE year <
(SELECT year
FROM Collection
WHERE album = 'Urban Hymns');
```

Selects all albums in the table Collection in years prior to the year related to the album Urban Hymns.

“List” of Values

Subquery should return one column of values:

```
SELECT album, year
FROM Collection
WHERE artist IN
(SELECT artist
FROM Collection
WHERE album LIKE 'The%');
```

Subquery returns “list”:

- (‘U2’, ‘Foo Fighters’)
Oracle output from whole query is:

<table>
<thead>
<tr>
<th>ALBUM</th>
<th>YEAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Colour and the Shape</td>
<td>1997</td>
</tr>
<tr>
<td>The Unforgettable Fire</td>
<td>1984</td>
</tr>
<tr>
<td>Rattle and Hum</td>
<td>1988</td>
</tr>
<tr>
<td>Achtung Baby</td>
<td>1991</td>
</tr>
</tbody>
</table>

So we are selecting the title and year of all the albums written by bands who have written albums that start with 'The'!

Commitment and Rollback

Changes to the database need to be committed. This can be automatic.

Uncommitted values are not permanent in the tables and can be rolled back.

To find out whether this is currently automated, type:

```sql
show autocommit;
```

```
autocommit OFF
```

`OFF` is the default value.

To set automatic:

```sql
set autocommit ON;
```

```
show autocommit;
```

```
autocommit IMMEDIATE
```

If other users access this data they see the old data until the commit has been performed.

Deleting Rows

To delete rows requires a predicate. Using the DELETE statement:

```sql
DELETE FROM Collection
WHERE year < 1990;
```

2 rows deleted.

Deletes all CDs from the table Collection prior to 1990. The WHERE clause is as for SELECT expressions. Rollback any unwanted deletions.

Updating: modifying existing rows

It is theoretically possible to achieve all database alterations to data using INSERT and DELETE.

More efficient for simple changes to use UPDATE. Consider adding the missing values for the Leftfield album:

- company - Columbia
- year - 1995

```sql
UPDATE Collection
SET company = 'Columbia',
year = 1994
WHERE artist = 'Leftfield'
AND album = 'Leftism';
```
If the previous SQL contained a mistake (e.g., the year is wrong) - it is also possible to increment values with update (and to fix the mistake):

```sql
UPDATE Collection
SET year = year + 1
WHERE album = 'Leftism';
```

### Update with Scalar Subqueries

The value SET can be selected from a table. Imagine that Island merge with the record company related with the Leftism album. To update the database:

```sql
UPDATE Collection
SET company = (SELECT company
                FROM Collection
                WHERE album = 'Leftism')
WHERE company = 'Island';
```

<table>
<thead>
<tr>
<th>ARTIST</th>
<th>ALBUM</th>
<th>COMPANY</th>
</tr>
</thead>
<tbody>
<tr>
<td>U2</td>
<td>The Unforgettable Fire</td>
<td>Columbia</td>
</tr>
<tr>
<td>U2</td>
<td>Rattle and Hum</td>
<td>Columbia</td>
</tr>
<tr>
<td>U2</td>
<td>Achtung Baby</td>
<td>Columbia</td>
</tr>
<tr>
<td>Leftfield</td>
<td>Leftism</td>
<td>Columbia</td>
</tr>
</tbody>
</table>

### Interim Summary

Material covered so far:
- Everything for data stored in one base table.
- Creating base tables without constraints.
- Inserting rows and partial rows, NULL.
- Expressing queries using (SELECT ... FROM ... WHERE).
- Predicates (WHERE) and predicate combination (AND and OR).
- Commitment and rollback.
- Deleting (DELETE) and updating (UPDATE) rows.
- Subqueries.