

CS233: Database Systems

Lecture Slides 1

presented by
Timothy Heron*

September 30, 2005

*E-mail: theron@dcs.warwick.ac.uk

Declaration

These lecture slides are originally the work of Dr Richard Cartwright, former undergraduate, doctoral student and lecturer at Warwick. Dr Meurig Beynon made one or two essential but minor adaptations and updates.

They have been updated and extended by Timothy Heron.

Course Structure

Lectures with a practical emphasis:

- Using **SQL** — relational databases
- Programming with databases — the Java database connectivity (**JDBC**) API

Course assessment:

- 70% exam - Lectures
- 30% assessed - Coursework exercise

Content of lectures will be examined and assessed.

Timetable: Lectures - Friday 1pm, ACCR.

Forum & Newsgroup

WWW: <http://www.dcs.warwick.ac.uk/~theron/cs233/>

Lecture Plan

Contents of the lectures:

- Introduction, getting started with *Oracle*, simple queries.
- Formulating queries, inserting - deleting - modifying rows.
- Creating and altering tables, relational algebra, constraints.
- Views, functions, dates & times.
- Programming with databases — using the JDBC API.
- Database Security

This Lecture

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

Books and Resources

C. J. Date. *An Introduction to Database Systems*. Addison Wesley, 8th edition, 2003.

C. J. Date, *Database in Depth: The Relational Model for Practitioners*. O'Reilly, 2005.

C. Begg, T. Connolly, *Database Systems: A Practical Approach to Design, Implementation and Management (International Computer Science S.)*, Addison-Wesley Longman, 2004 (4th ed.)

Oracle and JDBC references:

G. Reese. *Database Programming with JDBC and Java*. O'Reilly and Associates, 1997.

Links to Oracle documentation are on the course website.

Worksheets

Every week you are expected to take away and work on a worksheet *before* the next week.

Working through each worksheet and using a database between each lecture is an important part of the course.

Each weeks 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 1: *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 webpage.

Databases available at home

At Home

- Personal Oracle 9i (downloadable from <http://www.oracle.com/technology/software/products/oracle9i/>)
- Open source includes mSQL, mySQL and PostgreSQL
- mySQL via www.mysql.com
- mSQL under Linux (<http://www.Hughes.com.au/>)
- PostgreSQL (<http://www.postgresql.org/>)
- Microsoft SQL Server 2000 (a free version called MSDE is available)
- Microsoft Access (a long way from ISO SQL standard)

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 Current standard SQL2003 - 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)

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

1980s First standardisation efforts.

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 Systems (DBS)

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

There is a more user-friendly interface (Java GUI based) called SQLPlus Worksheets, run `sqlworksheet` from `mimosa` on a graphical terminal.

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 `.profile` files 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 “/” — it represents your personal username and password for Oracle.
5. Graphical SQL interpreter, type “`sqlworksheet`”. Press OK on the login screen - you do not need to enter a username or password

Creating a Table

From the depths of a CD collection:

Collection

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

Defining the table in SQL:

```
CREATE TABLE Collection
    (artist CHAR(16),
     album CHAR(40),
     tracks INTEGER,
     company CHAR(16),
     year INTEGER);
```

Note:

- Convention to write SQL statements in **CAPITALS**.
- Oracle matches lower/upper case the same in table and column names.
- ISO Standard — all table names and column names in capitals + numbers + underscore “_”.
- Semicolon “;” terminates every input.

Inserting Values

To insert the data into the table Collection use the `INSERT INTO` statement:

```
INSERT INTO Collection
VALUES ('U2', 'The Unforgettable Fire',
       10, 'Island', 1984);
```

```
INSERT INTO Collection
VALUES ('U2', 'Rattle and Hum',
       17, 'Island', 1988);
:
```

For every successful insertion, Oracle reports:

1 record 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` statement.

Order of statement:

1. *range expression(s)*
2. *target list*
3. *predicate*

To view an entire table:

```
SELECT *
FROM Collection;
```

ARTIST	ALBUM	TRACKS	COMPANY	YEAR
U2	The Unforgettable Fire	10	Island	1984
U2	Rattle and Hum	17	Island	1988
U2	Achtung Baby	12	Island	1991
Underworld	Second Toughest in the Infants	8	Junior	1996
The Verve	Urban Hymns	13	Virgin	1997
Foo Fighters	The Colour and the Shape	13	Capital	1997

Single Column Selection

To view one column of a table :

```
SELECT artist
FROM Collection;
```

artist
U2
U2
U2
Underworld
The Verve
Foo Fighters

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

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

artist
Foo Fighters
The Verve
U2
Underworld

Multi-column Selection

To select more than one column from one table:

```
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:

```
SELECT artist, album, year
FROM Collection
ORDER BY artist;
```

ARTIST	ALBUM	YEAR
Foo Fighters	The Colour and the Shape	1997
The Verve	Urban Hymns	1997
U2	The Unforgettable Fire	1984
U2	Rattle and Hum	1988
U2	Achtung Baby	1991
Underworld	Second Toughest in the Infants	1996

Simple Predicates

Predicate \equiv 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):

```
SELECT artist, album, year
FROM Collection
WHERE year = 1997;
```

All data in left-hand side *column* must exactly match right-hand side expression.

ARTIST	ALBUM	YEAR
The Verve	Urban Hymns	1997
Foo Fighters	The Colour and the Shape	1997

String Matching

Matching an identical string with “=”:

```
SELECT artist, album
FROM Collection
WHERE artist = 'U2';
```

Select all artists beginning with the letter “U” using the LIKE statement and wildcard “%”:

```
SELECT artist, album
FROM Collection
WHERE artist LIKE 'U%';
```

ARTIST	ALBUM
U2	The Unforgettable Fire
U2	Rattle and Hum
U2	Achtung Baby
Underworld	Second Toughest in the Infants

Wildcard “_” matches single characters. To escape wildcards use the “@” symbol. So to match “10%” use “10@%”.

Can also use NOT LIKE.

Number Range Selection

To select everything below a certain numerical value, use “<”:

```
SELECT artist, album, tracks
FROM Collection
WHERE tracks < 11;
```

Similarly for:

- greater-than “>”
- less-than-or-equal-to “<=”
- greater-than-or-equal-to “>=”
- two forms of not-equal-to “!=” and “<>”

ARTIST	ALBUM	TRACKS
U2	The Unforgettable Fire	10
Underworld	Second Toughest in the Infants	8

Can also use operator BETWEEN-AND to select a range of values:

```
SELECT artist, album, year
FROM Collection
WHERE year BETWEEN 1980 AND 1992;
```

ARTIST	ALBUM	YEAR
U2	The Unforgettable Fire	1984
U2	Rattle and Hum	1988
U2	Achtung Baby	1991

BETWEEN-AND is *inclusive*.

To leave sqlplus type “exit”.

List Tables

Non-standard SQL, specific to each database.

To list the tables in your area in an Oracle system use :

```
SELECT TABLE_NAME FROM USER_TABLES;
```

To view definition of a table use :

```
DESCRIBE Collection;
```

SQL commands 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 values

What if not all the data values are known?

- insert dummy values and change them later
- insert a marker — NULL value

Consider example question 4 of worksheet 1:

... 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, 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".

Selecting NULL values using IS:

```
SELECT artist, album
FROM Collection
WHERE year IS NULL;
```

Can also use “IS NOT NULL”.

Logic in Selections

Can use logical expressions in predicates — “AND” and “OR”.

```
SELECT artist, album
FROM Collection
WHERE artist LIKE 'U%'
      AND company != 'Island';
```

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?

```
SELECT artist, album, tracks
FROM Collection
WHERE artist LIKE '%e%'
      OR artist LIKE '%a%'
      AND tracks >= 10;
```

ARTIST	ALBUM	TRACKS
Underworld	Second Toughest in the Infants	8
The Verve	Urban Hymns	13
Foo Fighters	The Colour and the Shape	13

Using Brackets

Note how “AND” takes precedence over “OR”.

Use brackets to indicate a different priority:

```
SELECT artist, album, tracks
FROM Collection
WHERE (artist LIKE '%e%'
      OR artist LIKE '%a%')
      AND tracks >= 10;
```

ARTIST	ALBUM	TRACKS
The Verve	Urban Hymns	13
Foo Fighters	The Colour and the Shape	13

Selection from a List

A list is of the form:

- numbers — “(1, 2, 3)”
- strings — “(‘U2’, ‘Underworld’, ‘The Verve’)”

In SQL, use keyword “IN” to select from a list:

```
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');
```

Data Directed Selection

One of the most powerful aspects of SQL — data from tables can be used to direct a query. These are known as *subqueries*.

Two kinds:

- Single values — can be used with logical operators (<, >, = etc.)
- Lists of values — used with keyword “IN”.

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:

ALBUM	YEAR
-----	-----
The Colour and the Shape	1997
The Unforgettable Fire	1984
Rattle and Hum	1988
Achtung Baby	1991

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:

```
show autocommit;

autocommit is OFF;
```

OFF is the default value.

To set automatic:

```
set autocommit ON;

show autocommit;

autocommit is IMMEDIATE;
```

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

To commit recent changes associated with INSERT, DELETE or UPDATE statements, type “COMMIT;”.

Commitment occurs implicitly after the commands:

- QUIT or EXIT
- CREATE TABLE or CREATE VIEW
- DROP TABLE or DROP VIEW
- GRANT, REVOKE, CONNECT, DISCONNECT, ALTER, AUDIT, NOAUDIT

To undo changes since the last commitment, use command ROLLBACK;. Returns the database to state after last explicit or implicit commitment.

If the plug gets pulled out, uncommitted changes are rolled back.

Deleting Rows

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

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

2 records deleted.

Deletes all CDs from the table *Collection* prior to 1990.

We can use any predicate after the WHERE statement, including subqueries.

Rollback any unwanted deletions.

Updating Values

It is 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

```
UPDATE Collection
SET company = 'Columbia',
    year = 1994
WHERE artist = 'Leftfield'
AND album = 'Leftism';
```

This SQL contains a mistake (the year is wrong) — it is also possible to increment values with update (and to fix the mistake):

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

Update with Select

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:

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

ARTIST	ALBUM	COMPANY
U2	The Unforgettable Fire	Columbia
U2	Rattle and Hum	Columbia
U2	Achtung Baby	Columbia
Leftfield	Leftism	Columbia

Interim Summary

Material covered so far:

- Everything for data stored in one table.
- Creating tables without constraints.
- Inserting rows and partial rows, `NULL` values.
- Selecting data from tables (`SELECT ... FROM`).
- Predicates (`WHERE`) and predicate combination (`AND` and `OR`).
- Commitment and rollback.
- Deleting (`DELETE`) and updating (`UPDATE`) rows and values.
- Subqueries and embedding `SELECT`.