3ISY402 – DATABASE SYSTEMS

Lecture 3 –

SQL: Simple Data Manipulation

Material from essential text:

T CONNOLLY & C BEGG. Database Systems – A Practical Approach to Design, Implementation and Management, 4th Edition. Addison-Wesley, 2005.

Leena Gulabivala

Lecture - Objectives

- Purpose and importance of SQL.
- How to retrieve data from database using:
  - SELECT statement.
  - Use compound WHERE conditions.
  - Sort query results using ORDER BY.
  - Format column headings.
- How to update database using
  - INSERT,
  - UPDATE, and
  - DELETE.
- How to use Oracle SQL-Developer.

Objectives of SQL

- Ideally, a database language should allow a user to:
  - create the database and relation structures;
  - perform insertion, modification, deletion of data from relations;
  - perform simple and complex queries.
- Must perform these tasks with minimal user effort and command structure/syntax must be easy to learn.
- It must be portable.

SQL Statements

- SQL is a transform-oriented language with 2 major components:
  - A Data Definition Language (DDL) for defining database structure.
    - CREATE, ALTER, DROP, RENAME
  - A Data Manipulation Language (DML) for retrieving and updating data.
    - SELECT
    - INSERT
    - UPDATE
    - DELETE

SQL is relatively easy to learn

- SQL is relatively easy to learn:
  - It is non-procedural - you specify *what* information you require, rather than *how* to get it;
  - it is essentially free-format.
  - consists of standard English words (eg CREATE, SELECT, FROM, WHERE)
- Can be used by range of users including DBAs, management, application developers, and other types of end-users.
- An ISO standard now exists for SQL, making it both the formal and *de facto* standard language for relational databases.

Writing SQL Commands

- SQL statement consists of *reserved words* and *user-defined words*.
  - *Reserved words* are a fixed part of SQL and must be spelt exactly as required and cannot be split across lines.
  - *User-defined words* are made up by user and represent names of various database objects such as relations, columns, views.
- Most components of an SQL statement are case insensitive, except for literal character data.
- *** Note that all SQL statements are terminated by a semi-colon (;).*

Writing SQL Commands

- SQL commands are more readable with *indentation* and *lineation*:
  - Each clause should begin on a new line.
  - Start of a clause should line up with the start of other clauses.
  - If clause has several parts, they should each appear on a separate line and be indented under the start of the clause.

Writing SQL Commands

- Use extended form of BNF (Bachus-Naur Form) notation to define the syntax:

  - **Upper-case letters** represent reserved words.
  - **Lower-case letters** represent user-defined words.
  - **Square brackets** indicate an *optional element*.
  - A **vertical bar** (|) indicates a *choice* among alternatives.
  - **Curly braces** ({ })indicate a *required element*.
  - An **ellipsis** (…) indicates *optional repetition* (0 or more times).

SELECT Statement

SELECT [**DISTINCT** | **ALL**]
  {* | [columnExpression [**AS** newName]]
  [,...] }
FROM TableName [alias] [, ...]
[**WHERE** condition]
[**GROUP BY** columnList]
[**HAVING** condition]
[**ORDER BY** columnList]

SELECT Statement

SELECT	Specifies which columns are to appear in the output.
FROM	Specifies table(s) to be used.
WHERE	Filters rows subject to some condition.
GROUP BY	Forms groups of rows with same column value.
HAVING	Filters groups subject to some condition.
ORDER BY	Specifies the order of the output.

- Order of the clauses cannot be changed.
- Only SELECT and FROM are mandatory.

Literals

- Literals are constants used in SQL statements.
- All non-numeric literals (character strings and date values) must be enclosed in single quotes.
  - E.g - 'London', '12-Nov-1999'
- Character values are case-sensitive and date values are format-sensitive.
- Default date format is 'DD-MON-YY'.
- All numeric literals must NOT be enclosed in quotes.
  - E.g - 650.00

Arithmetic Expressions

- Use arithmetic operators to create expressions on NUMBER and DATE data types.
- Multiplication and division take priority over addition and subtraction.
- Operators of same priority are evaluated from left to right.
- Parentheses (brackets) are used to force prioritized evaluation and to clarify statements.

OPERATOR	DESCRIPTION
*	Multiply
/	Divide
+	Add
-	Subtract

Comparison Operators

OPERATOR	MEANING
=	Equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
<>	Not equal to

Other Comparison Operators

OPERATOR	MEANING
BETWEEN ... AND ...	Between two values (inclusive)
IN (list)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

- Use the LIKE operator to perform wildcard searches of valid search string values.
- A null is a value that is unavailable, unassigned, unknown, or inapplicable.

Logical Operators and Precedence Rules

Operator	Order Evaluated	Meaning
	1	All comparison operators.
NOT	2	Returns TRUE if the following condition is FALSE.
AND	3	Returns TRUE if both component conditions are TRUE.
OR	4	Returns TRUE if either component condition is TRUE.

- Override rules of precedence by using parentheses.

Tables – Branch and Staff

- Branch table

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

- Staff table

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Tables – Viewing and PrivateOwner

- Viewing table

clientNo	propertyNo	viewDate	comment
CR56	PA14	24-May-04	Too small
CR76	PG4	20-Apr-04	Too remote
CR56	PG4	26-May-04	
CR62	PA14	14-May-04	No dining room
CR56	PG36	28-Apr-04	

- PrivateOwner table

ownerNo	fName	lName	address	telNo
CO46	Joe	Keogh	2 Fergus Dr, Aberdeen AB2 7SX	01224-861212
CO87	Carol	Farrel	6 Achray St, Glasgow G32 8DX	0141-357-7419
CO40	Tina	Murphy	63, Well St, Glasgow G42	0141-943-1728
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025

Example 1 - All Columns, All Rows

- List full details of all staff.

```
SELECT  staffNo, fName, lName, position,
        sex, DOB, salary, branchNo
FROM    Staff;
```

- Can use * (asterisk) as an abbreviation for all columns:

```
SELECT  *
FROM    Staff;
```

- Both of the above queries will give all the rows and columns from the Staff table.

Example 2 - Specific Columns, All Rows

- Produce a list of salaries for all staff, showing only staff number, first and last names, and salary.

```
SELECT staffNo, fName, lName, salary
FROM   Staff;
```

staffNo	fName	lName	salary
SL21	John	White	30000.00
SG37	Ann	Beech	12000.00
SG14	David	Ford	18000.00
SA9	Mary	Howe	9000.00
SG5	Susan	Brand	24000.00
SL41	Julie	Lee	9000.00

Example 3 - Use of DISTINCT

- List the property numbers of all properties that have been viewed.

```
SELECT propertyNo
FROM   Viewing;
```

propertyNo
PA14
PG4
PG4
PA14
PG36

Example 3 - Use of DISTINCT

- Use DISTINCT to eliminate duplicates:

```
SELECT DISTINCT propertyNo
FROM       Viewing;
```

propertyNo
PA14
PG4
PG36

Example 4 - Calculated Fields

- Produce list of monthly salaries for all staff, showing staff number, first/last name, and salary.

```
SELECT  staffNo, fName, lName, salary/12
FROM    Staff;
```

staffNo	fName	lName	col4
SL21	John	White	2500.00
SG37	Ann	Beech	1000.00
SG14	David	Ford	1500.00
SA9	Mary	Howe	750.00
SG5	Susan	Brand	2000.00
SL41	Julie	Lee	750.00

Example 4 - Calculated Fields – Rename a Column (AS)

- To re-name a column, use AS clause:

```
SELECT staffNo, fName, lName,
       salary/12  AS monthlySalary
FROM   Staff;
```

Example 5 - Comparison Search Condition

- List all staff with a salary greater than 10,000.

```
SELECT  staffNo, fName, lName, position, salary
FROM    Staff
WHERE   salary > 10000;
```

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG37	Ann	Beech	Assistant	12000.00
SG14	David	Ford	Supervisor	18000.00
SG5	Susan	Brand	Manager	24000.00

Example 6 - Compound Comparison Search Condition

- List addresses of all branch offices in London or Glasgow.

```
SELECT  *
FROM    Branch
WHERE  city = 'London'  OR  city = 'Glasgow';
```

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B003	163 Main St	Glasgow	G11 9QX
B002	56 Clover Dr	London	NW10 6EU

Example 7 - Range Search Condition (BETWEEN/NOT BETWEEN)

- List all staff with a salary between 20,000 and 30,000.

```
SELECT   staffNo, fName, lName, position, salary
FROM     Staff
WHERE    salary BETWEEN 20000 AND 30000;
```

staffNo	fName	lName	position	salary
SL21	John	White	Manager	30000.00
SG5	Susan	Brand	Manager	24000.00

- BETWEEN test includes the endpoints of the range.
- Also a negated version NOT BETWEEN.

Example 7 - Range Search Condition (BETWEEN/NOT BETWEEN)

- BETWEEN does not add much to SQL's expressive power. Could also write:

```
SELECT  staffNo, fName, lName, position, salary
FROM    Staff
WHERE   salary >= 20000 AND salary <= 30000;
```

- List all staff whose date of birth is between 01/01/1955 and 31/12/1975.

```
SELECT   fName, lName, position, salary
FROM     Staff
WHERE    DOB BETWEEN '01-Jan-1955' AND
         '31-Dec-1975';
```

Example 8 - Set Membership Search Condition (IN/NOT IN)

- List all managers and supervisors.

```
SELECT  staffNo, fName, lName, position
FROM    Staff
WHERE   position IN ('Manager', 'Supervisor');
```

staffNo	fName	lName	position
SL21	John	White	Manager
SG14	David	Ford	Supervisor
SG5	Susan	Brand	Manager

Example 8 - Set Membership Search Condition (IN/NOT IN)

- There is a negated version (NOT IN).
- IN does not add much to SQL's expressive power. Could have expressed this as:

```
SELECT  staffNo, fName, lName, position
FROM    Staff
WHERE   position='Manager'
OR      position='Supervisor';
```

- IN is more efficient when the set contains many values.

Example 9 - Pattern Matching (LIKE/NOT LIKE)

- SQL has two special pattern matching symbols:
  - % (percentage): sequence of zero or more characters;
  - _ (underscore): any single character.
- LIKE '%Glasgow%' means a sequence of characters of any length containing '*Glasgow*'.
- LIKE 'A%' means a sequence of characters of any length starting with the letter A.
- LIKE '%son' means a sequence of characters of any length ending with the letters 'son'.

Example 9 - Pattern Matching (LIKE/NOT LIKE)

- Find all owners with the string 'Glasgow' in their address.

```
SELECT  ownerNo, fName, lName, address, telNo
FROM    PrivateOwner
WHERE   address LIKE '%Glasgow%';
```

ownerNo	fName	lName	address	telNo
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025

Example 10 - NULL Search Condition

- List details of all viewings on property PG4 where a comment has not been supplied.
- There are 2 viewings for property PG4, one with and one without a comment.
- Have to test for null explicitly using special keywords IS NULL:

```
SELECT  clientNo, viewDate
FROM    Viewing
WHERE   propertyNo = 'PG4'
AND     comment  IS NULL;
```

clientNo	viewDate
CR56	26-May-04

- Negated version (IS NOT NULL) can test for non-null values.

Example 11 - Single Column Ordering

- Sort rows with the ORDER BY clause
  - ASC ascending order (default).
  - DESC descending order.
- List salaries for all staff, arranged in *ascending* order of salary.

```
SELECT staffNo, fName, lName, salary
FROM   Staff
ORDER BY salary ASC;
```

- List salaries for all staff, arranged in *descending* order of salary.

```
SELECT staffNo, fName, lName, salary
FROM   Staff
ORDER BY salary DESC;
```

Example 11 - Single Column Ordering

- List salaries for all staff, arranged in *descending* order of salary.

staffNo	fName	lName	salary
SL21	John	White	30000.00
SG5	Susan	Brand	24000.00
SG14	David	Ford	18000.00
SG37	Ann	Beech	12000.00
SA9	Mary	Howe	9000.00
SL41	Julie	Lee	9000.00

Example 12 - Multiple Column Ordering

- Produce abbreviated list of properties in order of property type.

```
SELECT  propertyNo, type, rooms, rent
FROM    PropertyForRent
ORDER BY type;
```

- Result with one sort key:

propertyNo	type	rooms	rent
PL94	Flat	4	400
PG4	Flat	3	350
PG36	Flat	3	375
PG16	Flat	4	450
PA14	House	6	650
PG21	House	5	600

Example 12 - Multiple Column Ordering

- Four flats in this list - as no minor sort key specified, system arranges these rows in any order it chooses.
- To arrange in order of rent, specify minor order:

```
SELECT  propertyNo, type, rooms, rent
FROM    PropertyForRent
ORDER BY type, rent DESC;
```

Example 12 - Multiple Column Ordering

- Result with two sort keys:

propertyNo	type	rooms	rent
PG16	Flat	4	450
PL94	Flat	4	400
PG36	Flat	3	375
PG4	Flat	3	350
PA14	House	6	650
PG21	House	5	600

Defining a Column Alias (AS)

- Renames a column heading.
- Is useful with calculations.
- Immediately follows column name; optional AS keyword between column name and alias.
- Requires double quotation marks if it contains spaces or special characters or is case sensitive.

Concatenation Operator (||)

- Concatenates columns or character strings to other columns.
- Is represented by two vertical bars (||).
- Creates a resultant column that is a character expression.
- Literal character strings:
  - A literal is a character, expression, or number included in the SELECT list.
  - Date and character literal values must be enclosed within single quotation marks.
  - Each character string is output once for each row returned.

Literal Character Strings - Example

```
SELECT fName || ' is a ' || position
                AS "Employee Details"
FROM   Staff;
```

staffNo	fName	lName	position
SL21	John	White	Manager
SG37	Ann	Beech	Assistant
SG14	David	Ford	Supervisor
SA9	Mary	Howe	Assistant
SG5	Susan	Brand	Manager
SL41	Julie	Lee	Assistant

```
Employee Details
-----------------------
John is a Manager
Ann is a Assistant
David is a Supervisor
…
6 rows selected.
```

INSERT

INSERT INTO TableName [(columnList)]
VALUES (dataValueList)

- *columnList* is optional; if omitted, SQL assumes a list of all columns in their original CREATE TABLE order.
- Any columns omitted must have been declared as NULL when table was created, unless DEFAULT was specified when creating column.
- *dataValueList* must match *columnList* as follows:
  - number of items in each list must be same;
  - must be direct correspondence in position of items in two lists;
  - data type of each item in *dataValueList* must be compatible with data type of corresponding column.

Example 13 - INSERT … VALUES

- Insert a new row into Staff table supplying data for all columns.

```
INSERT INTO Staff
VALUES ('SG16', 'Alan', 'Brown', 'Assistant', 'M',
                Date'1957-05-25', 8300, 'B003');
```

Example 14 - INSERT using Defaults

- Insert a new row into Staff table supplying data for all mandatory columns.

```
INSERT INTO    Staff (staffNo, fName, lName,
                      position, salary, branchNo)
VALUES         ('SG44', 'Anne', 'Jones',
                      'Assistant', 8100, 'B003');
```

- ***Or***

```
INSERT INTO    Staff
VALUES         ('SG44', 'Anne', 'Jones', 'Assistant',
                      NULL, NULL, 8100, 'B003');
```

INSERT … SELECT

- Second form of INSERT allows multiple rows to be copied from one or more tables to another:

```
INSERT INTO TableName [ (columnList) ]
    SELECT ...
```

UPDATE

UPDATE TableName
SET columnName1 = dataValue1
[, columnName2 = dataValue2...]
[**WHERE** searchCondition]

- *TableName* can be name of a base table or an updatable view.
- SET clause specifies names of one or more columns that are to be updated.
- WHERE clause is optional:
  - if omitted, named columns are updated for all rows in table;
  - if specified, only those rows that satisfy *searchCondition* are updated.
- New *dataValue(s)* must be compatible with data type for corresponding column.

Example 15 - UPDATE Rows

- Give all staff a 3% pay increase. *(Updates ALL rows)*

```
UPDATE  Staff
SET     salary = salary*1.03;
```

- Give all Managers a 5% pay increase. *(Updates some rows)*

```
UPDATE  Staff
SET     salary = salary*1.05
WHERE   position = 'Manager';
```

- Promote David Ford (staffNo = 'SG14') to Manager and change his salary to £28,000. *(Updates multiple columns)*

```
UPDATE  Staff
SET     position = 'Manager',
        salary = 28000
WHERE   staffNo = 'SG14';
```

DELETE

DELETE FROM TableName
[**WHERE** searchCondition]

- *TableName* can be name of a base table or a view.
- *searchCondition* is optional; if omitted, all rows are deleted from table. This does not delete table. If *search_condition* is specified, only those rows that satisfy condition are deleted.

- Delete all viewings that relate to property PG4.

```
DELETE FROM Viewing
WHERE       propertyNo = 'PG4';
```

- Delete all records from the Viewing table.

```
DELETE FROM Viewing;
```

References and Further Reading:

- T CONNOLLY & C BEGG. *Database Systems – A Practical Approach to Design, Implementation and Management,* 4th Edition. Addison-Wesley, 2005.
  - Chapter 5 – SQL Data Manipulation.
- T CONNOLLY & C BEGG. *Database Solutions – A step-by-step guide to building databases,* 2nd Edition. Addison-Wesley, 2004.
  - Chapter on SQL Data Manipulation.
- Oracle Corporation – online teaching slides.