Database Management System (DBMS)

L-4:

SQL (Structure Query Language)

Dr. Kamruddin Nur
Associate Professor, Computer Science
kamruddin.nur@gmail.com

Lecture Content

- DML- Data Manipulation Language
- DDL - Data Definition Language

Reading: Chapter - 5&6 Text Book

Data Manipulation Language (DML)

"A DML is used to manipulate data."

By using a DML we can do the followings:

- *Single Table Query*
- *Multiple Table Query*
- *Nested Query*
- *Aggregate Functions*
- *Database Updates*

SELECT

SELECT	*to specify which columns to appear in the output*
FROM	*to specify the table/s*
WHERE	*to specify some condition to filter rows*
GROUP BY	*to group rows with same column value*
HAVING	*to filter groups having some condition*
ORDER BY	*to specify the order of the output*

SELECT : Retrieve all columns

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

Or

SELECT ***
FROM *Staff;*

SELECT : Specific columns

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

SELECT: DISTINCT

```
SELECT   propertyNo
FROM Viewing;
```

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

SELECT : Calculated Fields

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

WHERE : Row Selection

WHERE clause is used for -

Comparison	*- to compare values*
Range	*- to set a range*
Set Membership	*- to test whether a value one of a set of values*
Pattern Match	*- to test if a string matches a specified pattern*
Null	*- to test a column has a null value*

Comparison Operators	***: =, <, >, !=, <=, >=***
Range Operators	*: AND, OR, BETWEEN, NOT BETWEEN*
Set Membership	*: IN, NOT IN*
Pattern Match	*: LIKE, NOT LIKE*
Null	*: IS NULL, IS NOT NULL*

WHERE : Row Selection

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

SELECT   staffNo, fName, lName, salary
FROM Staff
WHERE salary BETWEEN 10000 AND 20000;
```

GROUP BY : Group Rows With Same Value

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

HAVING : Filter Groups Having Some Condition

```
SELECT   staffNo, fName, lName, salary
FROM Staff
GROUP BY salary
HAVING salary < 25000;
```

Aggregate Functions

COUNT	*- returns the no. of values in a specified column*
SUM	*- returns the sum of values in a specified column*
AVG	*- returns the average of the values in a specified column*
MIN	*- returns the smallest value in a specified column*
MAX	*- returns the largest value in a specified column*

COUNT

```
SELECT   COUNT(staffNo)
FROM Staff;
```

SUM

SELECT SUM(*salary)*
FROM *Staff;*

AVG

SELECT AVG(*salary)*
FROM *Staff;*

MIN

SELECT MIN(*salary)*
FROM *Staff;*

MAX

SELECT MAX(*salary)*
FROM *Staff;*

Subqueries

```
SELECT   staffNo, fName, lName, position
FROM Staff
WHERE branchNo = (SELECT  branchNo
                  FROM  Branch
                  WHERE street = '163 Main St');
```

Nested Subqueries: Use of IN

```
SELECT   propertyNo, street, city, postcode, type, rooms, rent
FROM PropertyForRent
WHERE staffNo IN (SELECT  staffNo
                  FROM  Staff
                  WHERE branchNo = (SELECT branchNo
                                    FROM Branch
                                    WHERE street = '163 Main St'));
```

ANY and ALL

- *The word **ANY** and **ALL** may be used in subqueries.*
- *If the subquery is preceded by the keyword **ANY**, the condition will be true if it is satisfied by any values produced by the subquery.*
- *If the subquery is preceded by the keyword **ALL**, the condition will be true if it is satisfied by all values produced by the subquery.*
- *If the subquery is empty, **ALL** condition returns true and the **ANY** condition returns false.*

ANY/SOME

```
SELECT   staffNo, fName, lName, position
FROM Staff
WHERE salary > SOME (SELECT  salary
                     FROM  Staff
                     WHERE branchNo = 'B003');
```

ALL

```
SELECT   staffNo, fName, lName, position
FROM Staff
WHERE salary > ALL (SELECT  salary
                    FROM  Staff
                    WHERE branchNo = 'B003');
```

Join

```
SELECT   c.clientNo, fName, lName, propertyNo, comment
FROM Client c, Viewing v
WHERE c.clientNo = v.clientNo;
```

```
SELECT   b.branchNo, b.city,s.staffNo, fName, lName, propertyNo
FROM Branch b, Staff s, PropertyForRent p
WHERE b.branchNo = s.branchNo AND s.staffNo = p.staffNo
ORDER BY b.branchNo, s.staffNo, propertyNo;
```

Join

```
SELECT   s.branchNo, s.staffNo, COUNT(*) AS Count
FROM  Staff s, PropertyForRent p
WHERE  s.staffNo = p.staffNo
GROUP BY s.branchNo, s.staffNo
ORDER BY s.branchNo, s.staffNo;
```

Left Outer Join

```
SELECT   b.*, p.*
FROM  Branch1 b LEFT JOIN PropertyForRent1 p ON b.city = p.city;
```

Right Outer Join

```
SELECT   b.*, p.*
FROM  Branch1 b RIGHT JOIN PropertyForRent1 p ON b.city = p.city;
```

Full Outer Join

```
SELECT   b.*, p.*
FROM  Branch1 b FULL JOIN PropertyForRent1 p ON b.city = p.city;
```

EXISTS and NOT EXISTS

- ***EXISTS*** *and* ***NOT EXISTS*** *are designed for subqueries.*
- *EXISTS produces true if and only if there exists at least one row the result table returned by the subquery. It is false if the subquery returns an empty result table*
- *NOT EXISTS is the opposite of EXISTS.*

EXISTS

```
SELECT   staffNo, fName, lName, position
FROM Staff s
WHERE EXISTS (SELECT  *
                    FROM  Branch  b
                   WHERE s.branchNo = b.branchNo AND
                              city = 'London');
```

Database Updates

- ***INSERT*** – *adds rows of data to a table*
- ***UPDATE*** – *modifies existing data in a table*
- ***DELETE*** – *removes rows of data from a table*

INSERT

INTERT INTO TableName [(Column List)]
VALUES (Data Value List);

For example,

INSERT INTO *staff*
VALUES *('SG16', 'Alan', 'Brown', 'Assistant', 'M', DATE, '1979-01-15',*
'8300', 'B003');

INSERT INTO *staff(staffNo, fName, lName, position, salary, branchNo)*
VALUES *('SG44', 'Anne', 'Jones', 'Assistant', '8300', 'B003');*

UPDATE

UPDATE TableName
SET *column1 = datavalue1, column2 = datavalue2, ...*
WHERE *searchCondition;*

For example,

UPDATE *staff*
SET *salary = salary*1.1;*

UPDATE *staff*
SET *salary = salary*1.1*
WHERE *position = 'Manager';*

DELETE

DELETE FROM TableName
WHERE *searchCondition;*

For example,

DELETE FROM *viewing*
WHERE *propertyNo= 'PG4';*

DELETE FROM *viewing*

The ISO SQL Data Types

Data Type	Declarations
boolean	*BOOLEAN*
character	*CHAR* *VARCHAR*
bit	*BIT* *BIT VARYING*
exact numeric	*NUMERIC* *DECIMAL* *INTEGER* *SMALLINT*

The ISO SQL Data Types cont.

Data Type	Declarations
approximate numeric	*FLOAT* *REAL* *DOUBLE PRECISION*
datetime	*DATE* *TIME* *TIMESTAMP*
interval	*INTERVAL*
large objects	*CHARACTER LARGE OBJECTS* *BINARY LARGE OBJECTS*

DOMAIN

gender CHAR NOT NULL CHECK (gender IN ('M', 'F'))

or

```
CREATE DOMAIN gender AS CHAR
                        DEFAULT 'M'
                        CHECK (VALUE IN ('M', 'F'));
```

Primary Key

PRIMARY KEY(propertyNo)

PRIMARY KEY(clientNo, propertyNo)

UNIQUE

To ensure uniqueness of alternate keys, we could use UNIQUE -

```
clientNo        VARCHAR(5)        NOT NULL
propertyNo      VARCHAR(5)        NOT NULL
UNIQUE(clientNo,propertyNo)
```

Foreign Key

FOREIGN KEY(branchNo) REFERENCES Branch

FOREIGN KEY(staffNo) REFERENCES Staff ON DELETE SET NULL

FOREIGN KEY(ownerNo) REFERENCES PrivateOwner
ON UPDATE CASCADE

ON DELETE

Options are -

CASCADE:

- Delete the row in parent table

- Delete the matching rows in child table

SET NULL:

- Delete the row in parent table

- Set foreign key values in the child table to NULL

SET DEFAULT:

- Delete the row from parent table

- Set default value for each matching value in the child table

NO ACTION:

- Reject the delete operation from the parent table

- This is default if ON DELETE rule is omitted.

CREATE TABLE

```
CREATE TABLE TableName
{ (columnName dataType [NOT NULL] [UNIQUE]
    [DEFAULT defaultOption] [CHECK (SearchCondition)] [,...]}
[ PRIMARY KEY(listOfColumns),]
{ [UNIQUE (listOfColumns),] [....] }
{ [ FOREIGN KEY (listOfForeignKeyColumns)
REFERENCES parentTableName [(listOfCandidateKeyColumns)],
    [MATCH {PARTIAL | FULL}
    [ON UPDATE referentialAction]
    [ON DELETE referentialAction]] [,....]}
{[CHECK (SearchCondition)] [,....]})
```

CREATE TABLE

```
CREATE TABLE Student(
studId INT NOT NULL
name VARCHAR(100) NOT NULL
addr VARCHAR(200)
PRIMARY KEY(studId));
```

CREATE DOMAIN

```
CREATE DOMAIN city AS VARCHAR(25);

CREATE DOMAIN staffNo AS VARCHAR(5)
        CHECK (VALUE IN (SELECT staffNo FROM Staff));
```

ALTER TABLE

ALTER TABLE *TableName*
[ADD [COLUMN] *columnName* ***dataType [NOT NULL] [UNIQUE]***
[DEFAULT *defaultOption*] ***[CHECK*** *(SearchCondition)]]*
[DROP [COLUMN] *columnName* ***[RESTRICT | CASCADE]]***
ADD [CONSTRAINT [*constraintName]]* *tableConstraintDefinition]***
DROP CONSTRAINT *ConstraintName* ***[RESTRICT | CASCADE]]***
[ALTER [COLUMN] SET DEFAULT *defaultOption]*
[ALTER [COLUMN] DROP DEFAULT]

ALTER TABLE

***ALTER TABLE** Staff*
ALTER** position **DROP DEFAULT

***ALTER TABLE** Staff*
***ALTER** gender **SET DEFAULT** 'F';*

DROP TABLE

***DROP TABLE** Staff;*

DROP TABLE** Staff **[RESTRICT | CASCADE]

RESTRICT:
The drop operation is rejected if there are any other objects that depend for their existence upon the continued existence of the table to be dropped.

CASCADE:
The DROP operation proceeds and SQL automatically drops all dependent objects.

VIEWS

- *The dynamic result of one or more relational operations operating on the base relations to produce another relation.*

- *A view is a virtual relation that does not necessarily exist in the database but can be produced upon request by a particular user, at the time of request.*

CREATE VIEW

CREATE VIEW *viewName [(newColumnName [,...])]*
AS *subselect [**WITH [CASCADE | LOCAL] CHECK OPTION]***

CREATE VIEW

```
CREATE VIEW Manager
AS SELECT * FROM staff WHERE position = 'Manager';

CREATE VIEW Staff3
AS SELECT staffNo, fName, lName, position, gender
    FROM Manager;
```

	Saturday Jul 15, 2006	Sunday Jul 16, 2006	Monday Jul 17, 2006	Tuesday Jul 18, 2006	Wednesday Jul 19, 2006	Thursday Jul 20, 2006
8 am :30	**8:00am–9:20am** **CSI331 A505**					
9 am :30	*9:30am–10:30am* *Counselling - S*					*9:00am–5:00pm* *Marking / Official Work - D*
10 am :30		*10:00am–12:00pm* *Counselling - S*				
11 am :30	**11:00am–12:20pm** **CSI223 A503**		**11:00am–2:00pm** **CSI224 Lab-C**			
12 pm :30	*12:30pm–2:00pm* *Counselling - S*	**12:30pm–1:50pm** **CSI331 A505**		*12:00pm–6:00pm* *Counselling - D*	*12:00pm–5:00pm* *Counselling - S*	
1 pm :30		*2:00pm–4:00pm* *Marking / Official Work - S*	**2:00pm–3:20pm** **CSI223 A502**			
2 pm :30						
3 pm :30						
4 pm :30						
5 pm :30						
6 pm :30				**6:30pm–9:00pm** **MIS411 D-Lab**		
7 pm :30						
8 pm :30						

Summary

From this lecture we have learned the details of

- Data Manipulation Language (DML)
- Data Definition Language (DDL)