ICDT1202Y Database Systems

Lecture 4. SQL DML (Part 1)

Anisah Ghoorah
a.ghoorah@uom.ac.mu
Dept. Digital Technologies
Faculty of ICDT
University of Mauritius

Acknowledgment and Reading

- **Chapter 4** in Elmasri, R. and Navathe, S. B. (2010) Fundamental of Database Systems, **6th edition**, Pearson.

- **Chapter 6** in Connolly, T. and Begg, C. (2014) Database Systems – A practical approach to design, implementation, and management, **6th edition**, Addison-Wesley.

Learning Outcomes

- Manipulate data from a database using SQL
  - Use DML statements to retrieve data from database tables
  - Use DML statements to summarize data from database tables
  - Use DML statements to modify data from database table
  - Use DML statements to query from multiple databases (using inner join or outer join)
    - To be covered in the next lecture

Content

- Use DML statements (INSERT, UPDATE, DELETE) to modify data from database tables
- Use DML statements (SELECT, WHERE, ORDER BY) to retrieve data from database tables
- Use DML statements (SUM, AVG, MIN, MAX) to summarize data from database tables

How to update databases

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

INSERT – Inserting a single row

```
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: INSERT ... VALUES

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

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

Example: 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

- Assume there exists a table
  - StaffPropCount(staffNo, fName, lName, propCount)
- Populate the StaffPropCount table using details from the Staff and PropertyForRent tables.

```
INSERT INTO StaffPropCount
( SELECT s.staffNo, fName, lName, COUNT(*)
  FROM   Staff s, PropertyForRent p
  WHERE  s.staffNo = p.staffNo
  GROUP BY s.staffNo, fName, lName
) ;
```

We get only a list of those staff who currently manage at least one property!

INSERT...SELECT

Correct SQL :

```
INSERT INTO StaffPropCount
( SELECT s.staffNo, fName, lName, COUNT(*)
  FROM   Staff s, PropertyForRent p
  WHERE  s.staffNo = p.staffNo
  GROUP BY s.staffNo, fName, lName
)
UNION
( SELECT staffNo, fName, lName, 0
  FROM   Staff s
  WHERE NOT EXISTS (
      SELECT *
      FROM   PropertyForRent p
      WHERE  p.staffNo = s.staffNo
  )
);
```

List of those staff who currently manage at least one property

Staff who do not manage any properties

INSERT...SELECT

Populate the StaffPropCount table using details from the Staff and PropertyForRent tables.

staffNo	fName	lName	propCount
SG14	David	Ford	1
SL21	John	White	0
SG37	Ann	Beech	2
SA9	Mary	Howe	1
SG5	Susan	Brand	0
SL41	Julie	Lee	1

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: UPDATE

Update all rows

E.g. Give all staff a 3% pay increase

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

Update specific rows

Give all Managers a 5% pay increase

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

Example: UPDATE Multiple Columns

Promote David Ford (staffNo='SG14') to Manager and change his salary to £18,000

```
UPDATE Staff
SET    position = 'Manager',
       salary = 18000
WHERE  staffNo = 'SG14';
```

DELETE

```
DELETE FROM TableName
[WHERE searchCondition]
```

- *TableName* can be name of a base table or an updatable 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

Example: DELETE Specific Rows

Delete all viewings that relate to property PG4

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

Delete all records from the Viewing table

This removes all rows from the table leaving only the table definition, so that we are still able to insert data into the table at a later stage.

```
DELETE FROM Viewing;
```

SELECT Statement

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

SELECT	Specifies which columns are to appear in output
FROM	Specifies table(s) to be used
WHERE	Filters rows
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** must be enclosed in single quotes (e.g. ‘London’)
- All **numeric literals** must not be enclosed in quotes (e.g. 650.00)

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 * as an abbreviation for 'all columns'

```
SELECT *
FROM   Staff;
```

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000.00	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000.00	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000.00	B005

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 (1)

List the property numbers of all properties that have been viewed

```
SELECT propertyNo
FROM   Viewing;
```

propertyNo
PA14
PG4
PG4
PA14
PG36

DUPLICATE ROWS !

Example 3: Use of DISTINCT (2)

Use DISTINCT to eliminate duplicates

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

propertyNo
PA14
PG4
PG36

Review Questions

Write SQL statements for each of the following queries:

1. List full details of all branch offices.
2. List the propertyNo of all properties together with their address.
3. List the cities in which there is a branch office.
4. List the cities in which there is a property available for rent.

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: Name a Column

To name column, use AS clause:

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

staffNo	fName	lName	MonthlySalary
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 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

Simple Comparison Operators

=	equals
<>	is not equal to (ISO standard)
!=	is not equal to (allowed in some dialects)
<	is less than
>	is greater than
<=	is less than or equal to
>=	is greater than or equal to

Rules for Evaluating Conditional Expressions

- An expression is evaluated left to right
- Subexpressions in brackets are evaluated first;
- NOTs are evaluated before ANDs and ORs
- ANDs are evaluated before Ors

- The use of parentheses is always recommended in order to remove any possible ambiguities.

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

- List all staff with a salary between 20,000 and 30,000
- BETWEEN test includes the endpoints of range
- Also a negated version NOT BETWEEN

```
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

Example 7: Range Search Condition

A range may also be expressed using >= and <=

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

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

Example 8: Set Membership

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

- There is a negated version (NOT IN)
- IN may be expressed using OR

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

- IN is more efficient when set contains many values

Review Questions

Write SQL statements for each of the following queries:

5. List the propertyNo and address of all properties together with their annual rent.
6. List full details of branch offices located in London.
7. List the propertyNo and address all properties with more than 3 rooms.
8. List the propertyNo and address of all houses.
9. List the propertyNo and address all flats with 3-4 rooms and rent no more than £400.
10. List full details of branch offices located in London, Glasgow and Bristol.
11. List full details of all branch offices except those located in London.

Example 9: Pattern Matching

- SQL has two special pattern matching symbols:
  - %: sequence of zero or more characters
  - _ (underscore): any single character
- LIKE ‘H%’ means the first character must be *H*, but the rest of the string can be anything.
- LIKE ‘H_ _ _’ means that there must be exactly four characters in the string, the first of which must be an *H*.
- LIKE ‘%e’ means any sequence of characters, of length at least 1, with the last character an *e*.
- LIKE ‘%Glasgow%’ means a sequence of characters of any length containing *Glasgow*.
- NOT LIKE ‘H%’ means the first character cannot be an *H*.

Example 9: Pattern Matching

- **ESCAPE**
- If the search string can include the pattern-matching character itself, we can use an **escape character** to represent the pattern-matching character.
- E.g. To check for the string ‘15%’
- **LIKE** ‘15#%’ **ESCAPE** ‘#’

Example 9: Pattern Matching

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

- To test for null explicitly using special keyword IS NULL
- Negated version (IS NOT NULL) can test for non-null values
- Example
  - 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

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

clientNo	viewDate
CR56	26-May-04

Review Questions

Write SQL statements for each of the following queries:

12. List the propertyNo and address of all properties having their postcode starting with G12.

13. List the propertyNo and address of all properties not yet assigned to a staff member.

Example 11: Single Column Ordering

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

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

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: Single Column Ordering

Produce abbreviated list of properties in order of property type

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

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;
```

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

Review Questions

Write SQL statements for each of the following queries:

14. List the names and address of all owners living in Glasgow, alphabetically ordered by lName.
15. List the propertyNo, address and rent of all houses with rent below £400.00 ordered by price.
16. List the details of staff ordered by their branchNo. Records of staff with same branchNo should be ordered by staffNo.

SELECT Statement - Aggregates

- Five aggregate functions:
  - COUNT returns number of values in specified column
  - SUM returns sum of values in specified column
  - AVG returns average of values in specified column
  - MIN returns smallest value in specified column
  - MAX returns largest value in specified column

SELECT Statement – Aggregates (2)

- Each operates on a single column of a table and returns a single value
- COUNT, MIN, and MAX apply to numeric and non-numeric fields, but SUM and AVG may be used on numeric fields only
- Apart from COUNT(*), each function eliminates nulls first and operates only on remaining non-null values
- COUNT(*) **counts all rows of a table**, regardless of whether nulls or duplicate values occur
- Can use DISTINCT before column name to eliminate duplicates
- DISTINCT has no effect with MIN/MAX, but may have with SUM/AVG

SELECT Statement – Aggregates (3)

- Aggregate functions can be used only in SELECT list and in HAVING clause
- If SELECT list includes an aggregate function and there is no GROUP BY clause (see Lecture 5), SELECT list cannot reference a column out with an aggregate function. E.g., the following is illegal:

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

ILLEGAL !

See next lecture

Example 13: Use of COUNT(*)

How many properties cost more than £350 per month to rent?

```
SELECT COUNT(*) AS myCount
FROM   PropertyForRent
WHERE  rent > 350;
```

myCount
5

Example 14: Use of COUNT(DISTINCT)

How many different properties viewed in May 2004?

```
SELECT COUNT(DISTINCT propertyNo) AS myCount
FROM   Viewing
WHERE  viewDate BETWEEN '1-May-04' AND '31-May-04';
```

myCount
2

Example 15: Use of COUNT and SUM

Find number of Managers and sum of their salaries

```
SELECT COUNT(staffNo) AS myCount,
       SUM(salary) AS mySum
FROM   Staff
WHERE  position = 'Manager';
```

myCount	mySum
2	54000.00

Example 16: Use of MIN, MAX, AVG

Find minimum, maximum, and average staff salary

```
SELECT MIN(salary) AS myMin,
       MAX(salary) AS myMax,
       AVG(salary) AS myAvg
FROM   Staff;
```

myMin	myMax	myAvg
9000.00	30000.00	17000.00

Review Questions

Write SQL statements for each of the following queries:

17. How many branch offices are there?
18. How many managers are there?
19. How many clients are looking for flats?
20. How many houses are available for rent?
21. What is the average rent of a property?
22. What is the max rent for a flat with 3 rooms?
23. In how many cities does Dreamhome have a branch office?
24. How many properties have been viewed?