SQL DML STATEMENT

CT230 Database Systems

QUESTIONS?

Recall: SQL:

- Structured
  - Query
    - Language

A special purpose programming language for relational database systems

Recall: ANSI/ISO SQL

Standardised SQL which comprises three components:

DDL – data definition language

DCL – data control language

DML – data manipulation language

DML: DATA MANIPULATION LANGUAGE

4 DML statements:

`INSERT`	insert data
`SELECT`	query data
`UPDATE`	update data
`DELETE`	delete data

DML SUPPORTS CRUD OPERATIONS

CRUD operations are the four basic functions we wish to perform on persistent data:

Create: insert a new tuple (INSERT)

Read: retrieve some data (SELECT)

Update: modify some data (UPDATE)

Delete: delete some data or a tuple (DELETE)

* we have already seen examples of INSERT, UPDATE and DELETE

SELECT

Basic syntax for an SQL select query to READ data consists of 6 clauses:

```
SELECT [DISTINCT] <attribute list>
FROM <table list>
WHERE <condition>
GROUP BY <group attributes>
HAVING <group condition>
ORDER BY <attribute list>
```

Notes:

- The order of the clauses cannot be changed
- SELECT and FROM are **always** required, other clauses are optional

NOTES ON SQL CLASS WORK:

For SQL SELECT work all examples with have a unique (!) number to ease cross-reference between lecture notes, your own attempts, and examples on Blackboard.

SELECT FROM WHERE

```
SELECT [DISTINCT] <attribute list>
FROM <table list>
WHERE <condition>
```

<attribute list> list of attribute (column) names (separated by commas) whose values will be retrieved by the query

<table list> list of table names (separated by commas) containing the attributes

<condition> Boolean expression that identifies the tuples to be retrieved by the query

WHERE clause: Boolean condition

For each tuple (row) in the table(s) which are part of query:

- tuple is checked to see if condition is true for this tuple
  - If true, tuple is part of the output
  - If not true, tuple is not part of the output

COMPARISON OPERATORS:

The comparison operators are:

= <= < > >= !=

Conditions can be compounded by used of Boolean

AND, OR

Conditions can be negated with NOT

(Note: In some versions of SQL (e.g. in MS) the != operator is written as <>)

RECALL: SQL is case insensitive …

But linux **is** case sensitive …. and web1.cs.nuigalway.ie is a linux server

Therefore need to be careful with table names in particular as

```
EMPLOYEE != employee
```

First SELECT Examples

Using the COMPANY relational database instance of the COMPANY SCHEMA develop SQL queries for the following:

attribute

table

1. List the names of all employees who earn more than 45000

condition

employee(fname, minit, lname, ssn, bdate, address, gender, salary, superssn, dno)

```
SELECT    fname, minit, lname
FROM      employee
WHERE     salary > 55000;
```

What is output? … how many employees? … menti.com

mySQL ...

Adminer 4.8.1

mydb2974

SQL command | Import

Export | Create table

- department
- dependent
- dept_locations
- employee
- project
- works_on

SQL command

```
SELECT fname, minit, lname
FROM   employee
WHERE  salary > 45000
```

fname	minit	lname
John	B	Smith
Franklin	T	Wong
Ramesh	K	Narayan
James	E	Borg
Jennifer	S	Wallace

5 rows (0.002 s) Edit, Explain, Export

```
SELECT fname, minit, lname
FROM   employee
WHERE  salary > 45000;
```

- project
- works_on

+ Options

fname	minit	lname
John	B	Smith
Franklin	T	Wong
Ramesh	K	Narayan
James	E	Borg
Jennifer	S	Wallace

NOTE:

** Attribute names are separated by commas
** Numbers are **NOT** enclosed in quotes
** Strings are enclosed in quotes

SQL command

```
SELECT fname, minit, lname
FROM   employee
WHERE  salary > 45000
```

Using AND and OR … SEE menti.com

What is the difference in output between these two versions of the query:

```
employee(fname, minit, lname, ssn, bdate, address, gender,
salary, superssn, dno)
```

```
SELECT    fname, minit, lname
FROM      employee
WHERE     dno != 5 AND salary > 45000;
```

```
SELECT    fname, minit, lname
FROM      employee
WHERE     dno != 5 OR salary > 45000;
```

Recall: BOOLEAN ALGEBRA:

In order for the Boolean AND of three conditions to be true, each individual condition (a, b, c) must be true.

Evaluation usually proceeds from Left to Right evaluating the TRUTH or each condition before returning True or False.

CODING STYLE

- Complying with coding style rules is crucial for a career in computing.
- Clean code is focused and understandable.
- Usually SQL keywords are capitalised and table and column names are mostly kept in lowercase unless combining words and not using an underscore
- Code should be organised *horizontally* and *vertically* (and not all written on one line).
- Code blocks are separated by a semi-colon.
- Use comments (#, --, /* and */) to explain code.

2 EXAMPLES TO TRY … menti.com

employee(fname, minit, lname, ssn, bdate, address, gender, salary, superssn, dno)

department(dname, dnumber, mgrssn, mgrstartdate)

dept_locations(dnumber, dlocation)

project(pname, pnumber, plocation, dnum)

works_on(essn, pno, hours)

dependent(essn, dependent_name, gender, bdate, relationship)

Example 2: Write a query to list the names of all projects located in Stafford

Example 3: Write a query to list the address and birth date of the employee with name John B Smith

Note: strings MUST BE enclosed in single quotes

Are these solutions correct?

#3: Write a query to list the address and birth date of the employee with name John B Smith

```
SELECT   bdate, address
FROM     employee
WHERE    fname = 'John B Smith';
```

```
SELECT   bdate, address
FROM     employee
WHERE    ssn = 123456789;
```

Be VERY careful of getting the "right" result using the "wrong" query

CALCULATED OR DERIVED FIELDS

Can specify an SQL expression in the SELECT clause which can involve numerical operations on numeric fields and counting operations on non-numeric fields

Example 4: Produce a list of monthly salaries for staff, showing staff ID and the salary details

```
employee(fname, minit, lname, ssn, bdate, address, gender,
salary, superssn, dno)
```

WILL THIS WORK?

Example 4: produce a list of monthly salaries for staff, showing staff ID (ssn) and the monrthly salary details

employee(fname, minit, lname, ssn, bdate, address, gender, salary, superssn, dno)

```
SELECT    ssn, salary/12
FROM      employee;
```

```
SELECT    ssn, salary/12
FROM      employee
```

ssn	salary/12
123456789	4604.166666666667
333445555	5416.666666666667
453453453	3681.9166666666665
666884444	5000
888665555	7849.916666666667
987654321	5770
987987987	3681.9166666666665
999887777	3681.9166666666665

8 rows (0.002 s) Edit, Explain, Export

TIDYING UP THE OUTPUT ....

1. Using Keywords CAST, AS and DECIMAL(x, y) to specify the total number of digits (x) and number of digits (y) after the decimal point when working with real numbers :

```
SELECT    ssn, CAST(salary/12.0 AS DECIMAL(8, 2))
FROM      employee;
```

ssn	mthlySalary
123456789	4604.17
333445555	5416.67
453453453	3681.92
666884444	5000.00
888665555	7849.92
987654321	5770.00
987987987	3681.92
999887777	3681.92

2. Using Keyword AS to rename output:

```
SELECT    ssn, CAST(salary/12.0 AS DECIMAL(8, 2))
          AS mthlySalary
FROM      employee;
```

USING KEYWORD DISTINCT

Keyword DISTINCT automatically removes duplicates from the returned result set.

Should be careful of using with large result sets as can be an expensive operation to perform (not a problem for our small examples).

QUESTION … how do you think DISTINCT could be implemented?

EXAMPLE 5:
Produce a list of all salaries

```
SELECT salary
FROM employee;
```

salary
55250
65000
44183
60000
94199
69240
44183
44183

8 rows (0.002 s) E

EXAMPLE 6:
Produce a list of DISTINCT salaries

```
SELECT DISTINCT salary
FROM employee;
```

SQL command

```
SELECT    DISTINCT salary
FROM      employee
```

salary
55250
65000
44183
60000
94199
69240

6 rows (0.002 s) Edit, Explain, Ex

NOTE:

To retrieve all attribute values of selected tuples, you do not have to explicitly list all the attribute names …. Instead can use SELECT *

May need to be careful of using this when you begin to join multiple tables or in real-world applications

```
SELECT *
FROM employee;
```

MORE EXAMPLES TO TRY: SEE menti.com

#7: Retrieve the address of the employee whose SSN is 123456789

#8: Retrieve all details stored on all employees in the employee table who work in department 4.

#9. List all locations where departments are (no need to list the department as well)

#10. Retrieve the salary and name of all employees working in department 5

SOME NEW OPERATORS:

BETWEEN : range search, including endpoints of range

IN : tests if a data value matches one of a list of values

(NOT IN)

LIKE : allows string comparison, when equality is too strict

IS NULL : allow an explicit search for NULL

Set Operators:

UNION, INTERSECTION,

MINUS/DIFFERENCE

EXAMPLE 11: Retrieve names of all employees whose salary is between 50000 and 80000

```
-- option1:

SELECT fname, minit, lname

FROM  employee

WHERE  salary >= 50000 AND salary <= 80000;
```

```
SELECT fname, minit, lname
FROM  employee
WHERE  salary BETWEEN 50000 AND 80000
```

fname	minit	lname
John	B	Smith
Franklin	T	Wong
Ramesh	K	Narayan
Jennifer	S	Wallace

4 rows (0.002 s) Edit, Explain, Export

```
-- option 2:

SELECT fname, minit, lname

FROM  employee

WHERE  salary BETWEEN 50000 AND 80000;
```

```
SELECT fname, minit, lname
FROM  employee
WHERE  salary BETWEEN 50000 AND 80000;
```

end points included in range

SUMMARY

The 3 most important keywords in Database Programming:

SELECT

FROM

WHERE

Practice with your own company database until questions 1-11 make sense to you!