COMP 3311
Database Management Systems

Lab 3

Basic SQL Statements

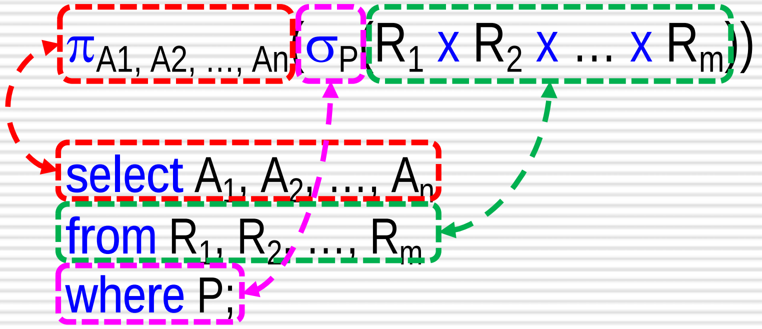
Lab Topics

- The SELECT-FROM-WHERE SQL clauses.
- The ORDER BY clause.
- Simple Join operations.

To see the result of the example queries in these lab notes, run them in SQL Developer against the database created by the Lab3DB.sql script file.

SELECT Statement

- Recall the correspondence between relational algebra and the SELECT statement.



$$\pi_{A1, A2, \ldots, An}(\sigma_P(R_1 \times R_2 \times \ldots \times R_m))$$

```
select A1, A2, ..., An
from R1, R2, ..., Rm
where P;
```

- Basic SELECT statement syntax

```
select * { [distinct] column | expression [alias], ...}
from table, ...
where condition
order by column [asc | desc], ...;
```

Example SELECT Statements

- Select all table columns.

```
select *
from Department;
```

- Select specified table columns.

```
select departmentId, departmentName
from Department;
```

- Select non-duplicate values from a table column.

```
select distinct departmentId
from Student;
```

- Concatenate results (|| operator), rename columns (as) / incorporate arithmetic expressions.

```
select firstName || ' ' || lastName as "Full Name”, cga+2.0
from Student;
```

Removing Duplicate Results

- The default setting for the SELECT statement is to return all the qualifying records – including duplicate ones.
- For example, the following statement will return all the department ids from the Student table:

```
select departmentId
from Student;
```

- To remove duplicates, the DISTINCT keyword can be added to the SELECT statement:

```
select distinct departmentId
from Student;
```

Incorporating Arithmetic Operations

- Arithmetic operations like * , / , + , - can be included in a SELECT statement.

```
select lastName, cga, cga+2.0
from Student;

select lastName, cga, cga/2.0
from Student;
```

Note: cga/2.0 will return the same result as cga/2 in SQL, this is different from some higher-level languages like C++.

Renaming A Column In The Query Result

- A column name in the query result can be renamed by using the AS keyword.

```
select lastName as Familyname
from Student;
```

- The SELECT statement can be used to output a column named Quarter CGA which displays the result cga/4.

```
select lastName, cga/4 as "Quarter CGA”
from Student;
```

Note: Double quotes are required around an alias only if it has an embedded space as in the example above.

Concatenating Query Results

- The || operator can be used to concatenate two columns in a select statement.

```
select firstName || ' ' || lastName as "Full Name"
from Student;
```

- The || operator can be used to add a string to the result.

```
select firstName || ' ' || lastName || ' studies in ' || departmentId as "Description”
from Student;
```

Note: If double quotes are placed around a single word alias such as Description, then it is displayed as typed; otherwise the alias name will be displayed in all capital letters.

Example of Concatenations

- Using concatenation, a query result can be expressed in a more easily comprehensible form.
- An example output from the table Student can be:

  Ariana Grande (13456789) from the COMP department has CGA 2.83. His/Her email is csgrande@connect.ust.hk.

- The corresponding SELECT statement is:

```
select firstName ||' '|| lastName || '(' || studentId || ') ' || 'from the ' || departmentId ||
   ' department has CGA ' || CGA ||'.' ||' His/Her email is ' || email || '@connect.ust.hk.'
   as Lab2
from Student;
```

SELECT Statement With WHERE Clause

- Select specified rows from a table.

```
select *
from Department
where departmentId='COMP';
```

The WHERE clause restricts the SELECT statement so that only the COMP department information is retrieved. The string 'COMP' in the condition clause is case sensitive.

- WHERE clause comparison operators:

=	equal	>	greater than	<	less than
>=	greater or equal	<=	less or equal	<>	not equal

Examples:

```
select *
from Student
where cga<>2.5;
```

```
select *
from Student
where cga<=1.9;
```

WHERE Clause Condition Operators

- Range of values: BETWEEN / NOT BETWEEN

```
select *
from Student
where cga between 2.8 and 3;
```

- Set membership: IN / NOT IN

```
select *
from Student
where departmentId in ('ELEC', 'MATH');
```

- Null value: IS NULL

```
select *
from Student
where cga is null;
```

NOTE: Cannot use where cga=null. This is illegal in SQL as null values are treated in a special way.

WHERE Clause Boolean Operators

- AND

```
select *
from Student
where cga>=2 and departmentId='MATH';
```

- OR

```
select *
from Student
where cga>=2 or departmentId='MATH';
```

- NOT

```
select *
from Student
where not departmentId='COMP';
```

WHERE Clause Logical Condition—Operator Precedence

- The AND condition has higher precedence than the OR condition.
  - Select students from the COMP department *plus* students from the MATH department with cga>=3:

    ```
    select *
    from Student
    where departmentId='COMP' or departmentId='MATH' and cga>=3;
    ```

  - To select students with cga>=3, from either the 'COMP' or the 'MATH' departments, add parentheses.

    ```
    select *
    from Student
    where (departmentId='COMP' or departmentId='MATH') and cga>=3;
    ```

WHERE Clause String Matching (1)

- LIKE (for matching patterns)

% can match zero or more characters.

Find students whose first name contains a "u" anywhere in the name.

What is the query result?

```
select *
from Student
where firstName like '%a%';
```

```
select *
from Student
where firstName like 'A%';
```

_ (underscore) matches exactly one character.

Find students whose first name contains a "u" as the second character.

What is the query result?

```
select *
from Student
where firstName like '_u%';
```

```
select *
from Student
where firstName like '%_u%';
```

WHERE Clause String Matching (2)

- REGEXP_LIKE function (for matching regular expressions)

  Syntax: REGEXP_LIKE(*attribute-name*, *regular-expression*, *match-parameter*)

  match-parameter: i → case insensitive; c → case sensitive.

  Find students with a double vowel in their last name.

```
select *
from Student
where regexp_like(lastName, '([aeiou])\1', 'i');
```

For information on the regular expressions supported by Oracle see https://docs.oracle.com/cd/B12037_01/server.101/b10759/ap_posix001.htm#i690819.

For examples of regular expression use in Oracle see https://www.salvis.com/blog/2018/09/28/regular-expressions-sql-examples/.

For testing your regular expressions see https://www.regextester.com/ (use the PCRE option).

ORDER BY Clause

- ASC ascending order (default)

```
select studentId, firstName, LastName, cga
from Student
where departmentId='COMP'
order by cga;
```

- DESC descending order

```
select studentId, firstName, LastName, cga
from Student
where departmentId='COMP'
order by cga desc;
```

- Sort on multiple columns

```
select *
from Student
order by departmentId asc, firstName desc;
```

Cartesian Product And Join Operation

- **Cartesian product**

```
select firstName, lastName, departmentName
from Student, Department;
```

Note: If the Student table has 20 entries and the Department table has 5 entries, then the query result has (20x5) 100 entries.

- **Join**

```
select firstName, lastName, departmentName
from Student, Department
where Student.departmentId=Department.departmentId;
```

Note: Attributes names need to be qualified with table names if they are ambiguous. For example, departmentId is an attribute of both the Student and Department tables in the above example.

If the Student table has 20 entries and the Department table has 5 entries, how many entries are there in the query result?

20 entries, one for each student.

Join Operation With Conditions

- A condition in the WHERE clause with a join condition further restricts the tuples selected.

```
select firstName, lastName, departmentName
from Student, Department
where Student.departmentId=Department.departmentId
      and Student.departmentId='COMP'
      and cga>2.5;
```

Note: Attributes names need to be qualified with table names if they are ambiguous. For example, departmentId is an attribute of both the Student and Department tables in the above example.

Natural Join Operation

- natural join

```
select firstName, lastName, departmentName
from Student natural join Department;
```

The rows of the tables are merged if the column(s) with identical name(s) match on their values.

For the tables Student and Department, only rows with identical values in the column departmentId will be merged, so students with departmentId ='COMP' will merge with the department with departmentId = 'COMP'.

The dual Table

- dual is an Oracle built-in table for SQL queries that do not logically have table names.

```
select 'The results of the queries are: '
from dual;
```

will output the string:

```
The results of the queries are:
```

Note: To suppress the output of table column headers in the Script Output pane of SQL Developer, place the SQL*Plus command “set heading off” in a script file before the SQL statement(s) whose result column headers you want to suppress. Use the command “set heading on” to again show the column headers for the result of SQL statements.