CSCI3170 Introduction to Database Systems

Tutorial 6 – Introduction to SQL

Write SQL statements

- SQL statements are **CASE INSENSITIVE**
- SQL statements can be on one or more lines
- Place a **semicolon** (;) at the end of the statement
- SQL statements in different databases may have slightly **different syntax**.

SQL statements shown in lecture notes are based on SQL-92. They may not work when you are using Oracle database.

CREATE

```
CREATE TABLE student (
        student_id INT(10) PRIMARY KEY,
        name Char(30) NOT NULL,
        study_year INT(1) DEFAULT 1,
        FOREIGN KEY(study_year)
        REFERENCES admission (study_year)
                                ON DELETE CASCADE);
```

Table names and column names (restrictions on Oracle):

- Must begin with a letter
- Must be 1–30 characters long
- Must contain only A–Z, a–z, 0–9, _, $, and #
- Must not duplicate the name of another object
- Must not be an reserved word

ALTER

- Add a new column to an existing table

```
ALTER TABLE student ADD date_of_birth DATE
NOT NULL;
```

- Modify an existing column

```
ALTER TABLE student MODIFY name CHAR(40)
NOT NULL;
```

- Rename an existing column

```
ALTER TABLE student RENAME COLUMN
student_id TO sid;
```

- Remove an existing column

```
ALTER TABLE student DROP name;
```

DROP / TRUNCATE

- Delete a table

```
DROP TABLE student;
```

- Remove all the rows within a table

```
TRUNCATE TABLE student;
```

The operations cannot be reversed, please make sure all the data are of no used before performing the operations

INSERT

- Insert a row with specified values

```
INSERT INTO student (student_id, name)
VALUES (1155123456, 'Chan Tai Man');
```

- Copy rows from another table

```
INSERT INTO student (student_id, name)
SELECT sid, name
FROM new_student;
```

UPDATE

- Update the values of a row

```
UPDATE student
SET name = 'David Chan', study_year = 2
WHERE student_id = 1155123456;
```

- Update all the rows within the table

```
UPDATE student SET study_year = 3;
```

DELETE

- Delete a row

```
DELETE FROM student WHERE student_id =
1155123456;
```

- Delete all the rows within the table

```
DELETE FROM student;
```

Similar to TRUNCATE but require longer time

- Delete rows using sub-query

```
DELETE FROM student
WHERE student_id = ANY(SELECT student_id
                       FROM graduate_list);
```

Tables involved in the sub-query cannot be the one performing DELETE

SELECT - Basic

- Select all columns

```
SELECT * FROM student;
```

- Select specific columns

```
SELECT student_id, name FROM student;
```

- Rename the column

```
SELECT student_id AS SID,
       study_year AS "Year of Study"
FROM student;
```

Requires double quotation marks if it contains spaces or special characters

SELECT – Basic (2)

- Eliminate the duplicate rows

```
SELECT DISTINCT study_year FROM student;
```

- Select rows based on some conditions

```
SELECT student_id, name
FROM student
WHERE study_year > 1 AND
      name LIKE 'Chan%';
```

The value is CASE SENSITIVE

SELECT - Sorting

- Sort the result set in ascending order

```
SELECT * FROM student ORDER BY student_id ASC;
```

- Sort the result set in descending order

```
SELECT * FROM student ORDER BY student_id DESC;
```

SELECT – Sorting (2)

- Sort the result set by column alias

```
SELECT student_id AS ID FROM student
ORDER BY ID ASC;
```

- Sort the result by 2 columns (Sorted by name followed by study_year)

```
SELECT name, study_year FROM student
ORDER BY name ASC, study_year ASC;
```

SELECT - Join

- Display data from multiple table

Specific which table the column belongs to if there is name collision

```
SELECT programme.name, student_id
FROM student, programme
WHERE prog_code = programme_id
```

Schema

student (student_id, name, study_year, prog_code)
programme (programme_id, name)

SELECT – Join (2)

- Use table alias

```
SELECT P.name, student_id
FROM student, programme P
WHERE prog_code = programme_id;
```

Schema

student (student_id, name, study_year, prog_code)
programme (programme_id, name)

SELECT - Aggregate

- Operators

Operator	Meaning
COUNT([DISTINCT] A)	The number of (unique) value in the A column
SUM ([DISTINCT] A)	The sum of all (unique) values in the A column
AVG ([DISTINCT A)	The average of all (unique) values in the A column
MAX (A)	The maximum value in the A column
MIN (A)	The minimum value in the A column

study_year
1
1
2
2
3

COUNT(*)
5

Only one row will be returned

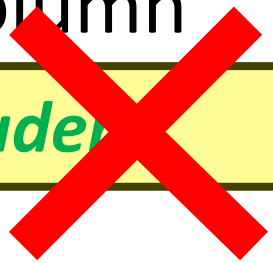
SELECT - Aggregate (2)

- Count the number of rows

```
SELECT COUNT(*) FROM student;
```

- Select aggregate value with other column

```
SELECT study_year, COUNT(*) FROM student
```



SELECT - Grouping

- Count the numbers of rows in different group

```
SELECT study_year, COUNT(*)
FROM student
GROUP BY study_year;
```

study_year
1
1
2
2
2
3
3

study_year	COUNT(*)
1	2
2	3
3	2

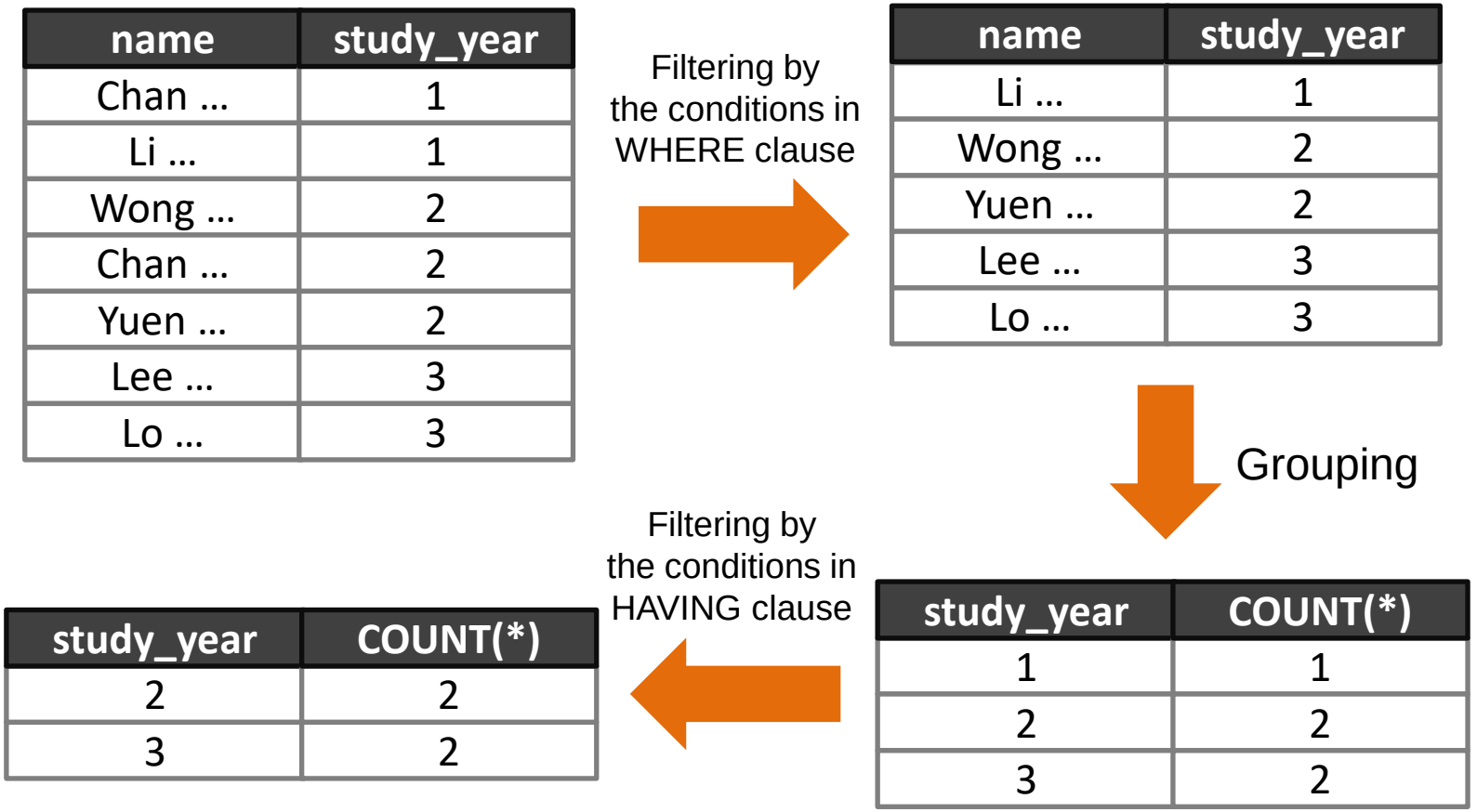
SELECT – Grouping (2)

- Use together with HAVING

```
SELECT study_year, COUNT(*)
FROM student
WHERE name NOT LIKE 'Chan%';
GROUP BY study_year
HAVING COUNT(*) > 1;
```

SELECT – Grouping (3)

- Difference between WHERE and HAVING



SELECT - Subquery

- Single-row subquery

```
SELECT name FROM student
WHERE student_id = (SELECT MAX(sid)
                    FROM graduate_list);
```

- Multiple-row subquery

```
SELECT name FROM student
WHERE student_id = ANY(SELECT sid
                       FROM graduate_list);
```

SELECT - Subquery (2)

- Pass value to the subquery

```
SELECT name FROM student S
WHERE EXISTS (SELECT *
                    FROM graduate_list
                    WHERE sid = S.student_id);
```

Become a condition of the subquery

SELECT - Set manipulation

- A and B

```
(SELECT …) INTERSECT (SELECT);
```

- A or B

```
(SELECT …) UNION (SELECT);
```

- A - (A and B)

```
(SELECT …) EXCEPT (SELECT);
```

Use **MINUS** when you are using Oracle database

Remarks:

A is the result set from the first SQL statement

B is the result set from the second SQL statement

VIEW

- Create a view (a temporary table)

```
CREATE VIEW temp AS SELECT *
                FROM student WHERE study_year = 3;
```

- Remove the view

```
DROP VIEW temp;
```

- Create or replace a view

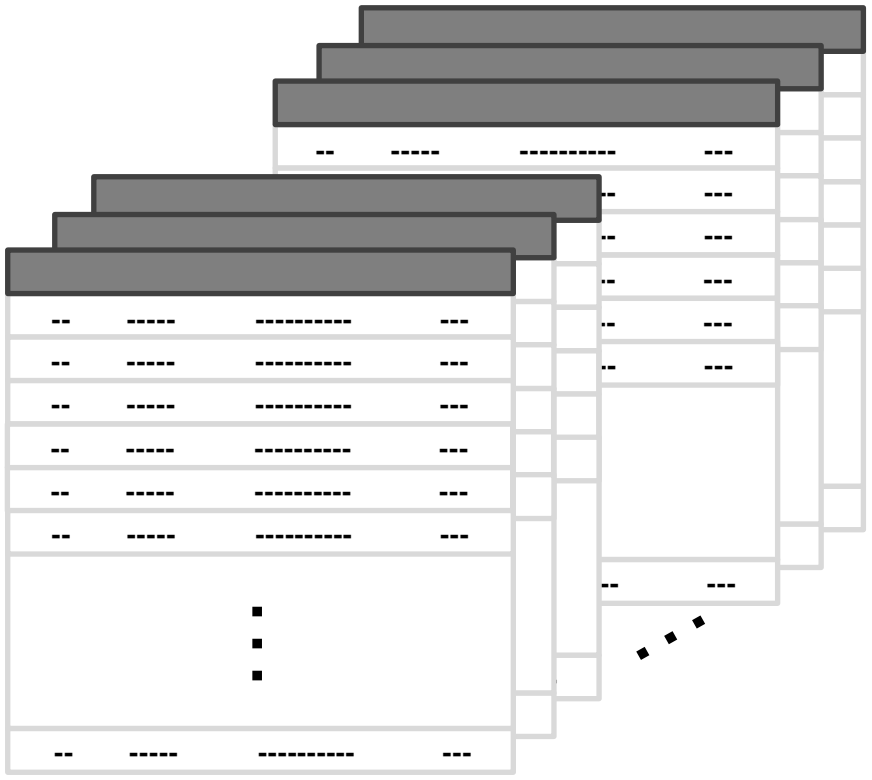
```
CREATE OR REPLACE VIEW temp AS
                SELECT * FROM student
                WHERE study_year = 3;
```

May not appear in the examination

EXTRA NOTES

Pagination

- In many applications, data is divided into discrete pages



Pagination (2)

```
SELECT student_id, name, study_year FROM
(SELECT ROWNUM AS RN,
                student_id, name, study_year
                FROM student)
WHERE RN BETWEEN 11 AND 20;
```

The above queries return the rows in the second page and each page contains ten rows

Date

- DATE is a data type for storing date in a database.
- A 'Date' string can be converted to a DATE by:

```
TO_DATE('Date', 'Format')
```

- Similarity, a DATE can be convert back to a string using:

```
TO_CHAR(DATE, 'Format')
```

Timestamp (2)

- Format of ‘Date&Time’ is defined by ‘Format’

Format Code	Meaning
YYYY	Year (Displayed in 4 digit)
MM	Month of a year(Displayed in 2 digit)
DD	Day of a month (Displayed in 2 digit)
HH24	Hour of a day (Displayed in 2 digit, 24 hour notation)
MI	Minute of a hour(Displayed in 2 digit)

- An example on how to define ‘Format’:

```
TO_TIMESTAMP('2016/01/01 10-31',
'YYYY/MM/DD HH24-MI')
```