Dr. Nguyen Thi Oanh

DATABASE

Lesson 4. Structured Query Language – part 1

LEARNING POINTS

1. Introduction to SQL
2. Definition a Relation schema
3. Data Manipulation

LEARNING OBJECTIVES

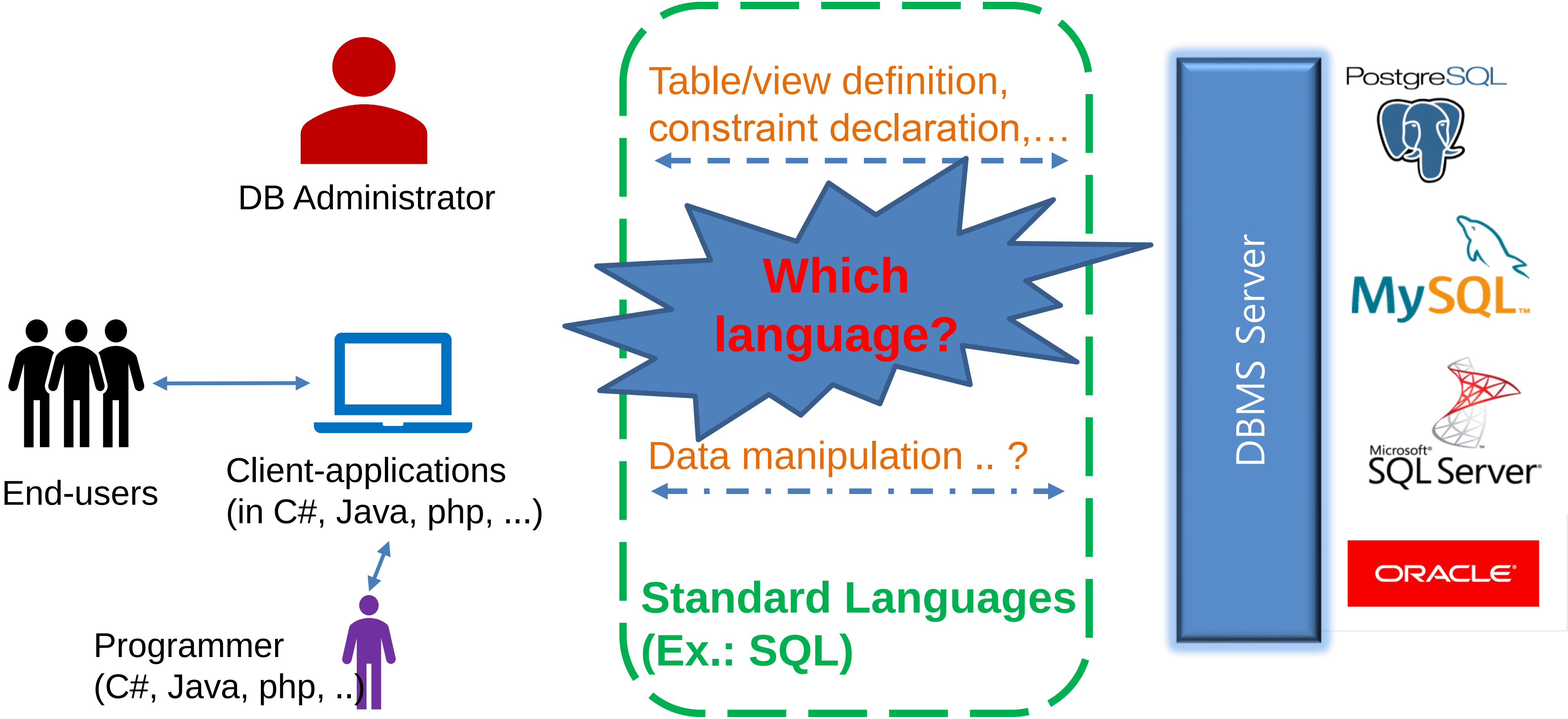
Upon completion of this lesson, students will be able to:

1. Have notions about the SQL language
2. Use SQL to define a relation schema in a database
3. Use SQL to populate a table with rows, update / delete data and to retrieve data from a table

Keywords

Keyword	Description
DBMS	Database Management System: system software for creating and managin g databases. The DBMS provides users and programmers with a systematic way to create, retrieve, update and manage data
CREATE TABLE	SQL statement to define a table into a database
ALTER TABLE	SQL statement to modify table structure if needed (add /delete/modify column(s), add/remove constraint(s))
INSERT/UPDATE/ DELETE	SQL statements to add new record to a table; to change the data of one or more records in a table; to remove single record or multiple records from a table
SELECT	SQL statement to retrieve data from a database

1. Introduction to SQL



DB Administrator

End-users

Client-applications
(in C#, Java, php, ...)

Programmer
(C#, Java, php, ..)

Table/view definition,
constraint declaration,…

Which language?

Data manipulation .. ?

Standard Languages
(Ex.: SQL)

DBMS Server

1. Introduction to SQL

1.1. Brief history of SQL

- 1975: SEQUEL: System-R
- 1976: SEQUEL 2
- 1978/79: SQL (Structured Query Language) (used in System-R)
- **SQL1**: The first standard for SQL defined in 1986; adopted as an international by Standards Organisation (ISO) in 1987.
- **1992: SQL2:** revised version of the processor (also called SQL 92); adopted as the formal standard language for defining and manipulating relational database.
- 1999: **SQL3**: extension with additional features such as user-defined data types, triggers, user-defined functions and other Object Oriented features.
- **New versions** of the standard were published in 2003, 2006, 2008, 2011, 2016: more additional features: XML-based features, instead of triggers, ...

1. Introduction to SQL

1.2. Languages

- Data Definition Language (DDL)
  - define the logical schema (relations, views…) and storage schema stored in a Data Dictionary
- Data Manipulation Language (DML)
  - Manipulative populate schema, update database
  - Retrieval querying content of a database
- Data Control Language (DCL)
  - permissions, access control…

2. Definition a Relation Schema

- Example: Education database

  student(**student_id**, first_name, last_name, dob, gender, address, note, *clazz_id*)

  subject(**subject_id**, name, credit, percentage_final_exam)

  lecturer(**lecturer_id**, first_name, last_name, dob, gender, address, email)

  teaching(***subject_id, lecturer_id***)

  grade(**code**, fromScore, toScore)

  clazz(**clazz_id**, name, *lecturer_id*, *monitor_id*)

  enrollment(***student_id, subject_id,* semester,** midterm_score, final_score)

- Detailed description for relation/table enrollment

Attribute name	**Type**	**NOT NULL**	**Description**
student_id	CHAR(8)	Yes	Student identification code. FOREIGN KEY references to Student(student_id)
subject_id	CHAR(6)	Yes	Subject code. FOREIGN KEY references to Subject(subject_id)
semester	CHAR(5)	Yes	Annual semester: '20171', '20172', '20173', …
midterm_score	Float	No	Score of mid-term exam. DOM = [0,10] and (midtermScore mod 0.5) must be 0
final_score	Float	No	Score of final exam. DOM= [0,10] (finalScore mod 0.5) must be 0
PRIMARY KEY = {student_id, subject_id, semester}			

2. Definition a Relation Schema

2.1. Creating a Simple Table

- Syntax:

```
CREATE TABLE <table_name>(
     <col1> <type1>(<size1>)[NOT NULL] [DEFAULT <value>],
     <col2> <type2>(<size2>)[NOT NULL],
     ...,
 [[CONSTRAINT <constraint_name>] <constraint_type> clause], …);
```

- Example:

```
CREATE TABLE student(
     student_id CHAR(8) NOT NULL,
     first_name VARCHAR(20) NOT NULL,
     last_name VARCHAR(20) NOT NULL,
     dob DATE NOT NULL,
     gender CHAR(1), address VARCHAR(30),
     note TEXT, class_id CHAR(8) );
```

2. Definition a Relation Schema

2.1. Creating a Simple Table

- ***Naming conventions***
  - Ordinary identifiers:
    - Must begin with a letter
    - Contain only: letters (a…z), underscore (_), and digits (0…9)
    - No longer than 32 characters
  - Delimited identifiers:
    - Identifiers surrounded by double quotation marks (")
    - Can contain any characters

2. Definition a Relation Schema

2.1. Creating a Simple Table

● *Naming conventions*

- Have meaning, not so long, use common abbreviations if needed:
  - use student, firstname;
  - not table1, abc, fw12re, student_of_the_school…
- Avoid quotes: student ; not "Student" or "All Students"
- Use lowercase, underscores separate words:
  - Use firstname / first_name;
  - not "firstName"
- Avoid reserved words (keywords):
  - data types are not object names : not use text, integer, … as object names
  - Not use table, user, … as object names
- Tables/ Views should have singular names, not plural: student; not students

2. Definition a Relation Schema

2.1. Creating a Simple Table

- ***Data Types (SQL 92)***

boolean	logical boolean (true/false)
character(n)	fixed-length character string
varchar(n)	variable-length character string
smallint	signed two-byte integer
int, integer	signed 4-byte integer
float(p)	floating-point number with precision p
real, double precision	double-precision floating-point number
decimal(p,s), numeric(p,s)	user-specified precision, exact; recommended for storing monetary amounts p: number of digits in the whole number, s: number of digits after the decimal point.
date	calendar date without time of day
time	time of day
timestamp with time zone	date/time

2. Definition a Relation Schema

2.1. Creating a Simple Table

● *NULL, NOT NULL, Default value*

- NULL:
  - Attribute does not have a known value
  - NULL value means "I don't known"
- NOT NULL:
  - Attribute must have a known value
- Default value:
  - the value appears by default in a column if no other value is known

2. Definition a Relation Schema

2.2. Constraints

- Entity Integrity:
  - No duplicate tuples: PRIMARY KEY constraint
  - Valide values on a attribute or between attributes in a tuple: CHECK constraint
- Referential Integrity:
  - Make sure that values of some attributes must make sens: FOREIGN KEY constraint

2. Definition a Relation Schema

2.2. Constraints

● *PRIMARY KEY*

- Syntax:

```
[CONSTRAINT <constraint_name>]PRIMARY KEY (<fk1>,<fk2>,…)
```

- A relation may have only one primary key

```
Table: Clazz(clazz_id, name, lecturer_id, monitor_id)
SQL:
    CREATE TABLE clazz (
            clazz_id CHAR(8) NOT NULL,
            name VARCHAR(20),
            lecturer_id CHAR(5),
            monitor_id CHAR(8),
            CONSTRAINT clazz_pk PRIMARY KEY (clazz_id));
```

2. Definition a Relation Schema

2.2. Constraints

- ***PRIMARY KEY***

Table: Clazz(**clazz_id**, name, lecturer_id, monitor_id)

SQL:

```
CREATE TABLE clazz (
   clazz_id CHAR(8) NOT NULL,
   name VARCHAR(20),
   lecturer_id CHAR(5),
   monitor_id CHAR(8),
   PRIMARY KEY (clazz_id) );
```

```
CREATE TABLE clazz (
  clazz_id CHAR(8) NOT NULL PRIMARY KEY,
  name VARCHAR(20),
  lecturer_id CHAR(5),
  monitor_id CHAR(8) );
```

If primary key has only one attribute

2. Definition a Relation Schema

2.2. Constraints

● *CHECK*

- Syntax:

```
[CONSTRAINT <constraint_name>] CHECK <condition>
```

- Declaring check constraint when defining table

```
Table: student(student_id, first_name, last_name, dob, gende, address,
note, clazz_id)
SQL: CREATE TABLE student (
        student_id CHAR(8) NOT NULL,
        first_name VARCHAR(20) NOT NULL, last_name VARCHAR(20) NOT NULL,
        dob DATE NOT NULL, gender CHAR(1), address VARCHAR(30),
        note TEXT, clazz_id CHAR(8),
        CONSTRAINT student_pk PRIMARY KEY (student_id),
        CONSTRAINT student_chk_dob CHECK (gender='F' OR gender='M'));
```

2. Definition a Relation Schema

2.2. Constraints

FOREIGN KEY

- Syntax:

```
[CONSTRAINT <constraint_name>] FOREIGN KEY (<fk1>,<fk2>,…)
                     REFERENCES <tab>(<k1>,<k2>, …)
                     [ON UPDATE <option>][ON DELETE <option>]
```

- Options:
  - – CASCADE
    - Delete/update all matching foreign key tuples
  - – NO ACTION / RESTRICT
    - can’t delete primary key tuple whilst a foreign key tuple matches
    - default action
  - – SET NULL

2. Definition a Relation Schema

2.2. Constraints

● *FOREIGN KEY*

- Declaring check constraint when defining table

Table: Clazz(**clazz_id**, name, *lecturer_id*, *monitor_id*)

SQL:

```
      CREATE TABLE clazz (
        clazz_id CHAR(8) NOT NULL,
        name VARCHAR(20), lecturer_id CHAR(5),
        monitor_id CHAR(8),
        CONSTRAINT clazz_pk PRIMARY KEY (clazz_id),
        CONSTRAINT clazz_fk_student FOREIGN KEY (monitor_id) REFERENCES
student(student_id));
```

2. Definition a Relation Schema

2.3. Modifying Relation Schema

- ***Columns***

  - Add column(s)

    ```
    ALTER TABLE <table_name>  ADD COLUMN
    <column_name> <datatype> [NOT NULL] [DEFAULT <default_value>];
    ```

  - Delete column(s)

    ```
    ALTER TABLE <table_name> DROP COLUMN <column_name>;
    ```

  - Modify column(s)

    ```
    ALTER TABLE <table_name> CHANGE COLUMN <column_name> <datatype>;
    ```

  - Examples:

    ```
        ALTER TABLE student ADD COLUMN
                urgence_contact CHAR(15) DEFAULT '(+84)000-000-000';
        ALTER TABLE student DROP COLUMN urgence_contact;
    ```

2. Definition a Relation Schema

2.3. Modifying Relation Schema

- ***Constraints***

  - Add new constraint(s)

    ```
    ALTER TABLE <table_name>
    ADD CONSTRAINT <constraint_name> <constraint_type> clause;
    ```

    ```
      Example:
      ALTER TABLE student ADD CONSTRAINT student_fk_clazz
          FOREIGN KEY (clazz_id) REFERENCES clazz(clazz_id);
    ```

  - Delete existing constraints

    ```
    ALTER TABLE <table_name>  DROP CONSTRAINT <constraint_name>;
    ```

    ```
      Example:
      ALTER TABLE student DROP CONSTRAINT student_fk_clazz;
    ```

2. Definition a Relation Schema

2.4. Drop a Relation from Database

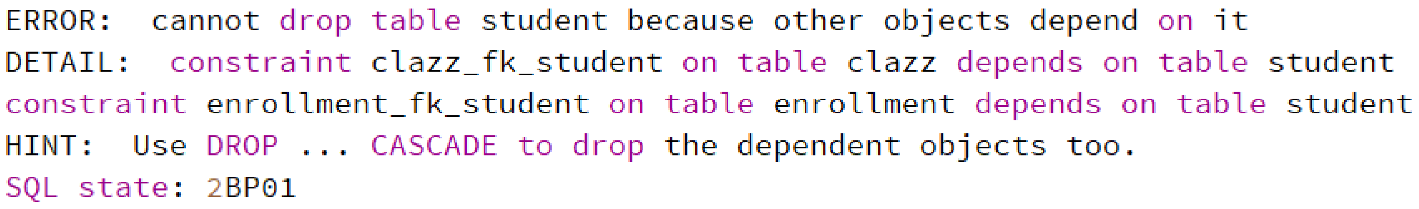
- Syntax: `DROP TABLE <table_name> [CASCADE | RESTRICT];`
  - CASCADE: allows to remove all dependent objects together with the table automatically
  - RESTRICT: refuses to drop table if there is any object depends on it; default value

2. Definition a Relation Schema

2.4. Drop a Relation from Database

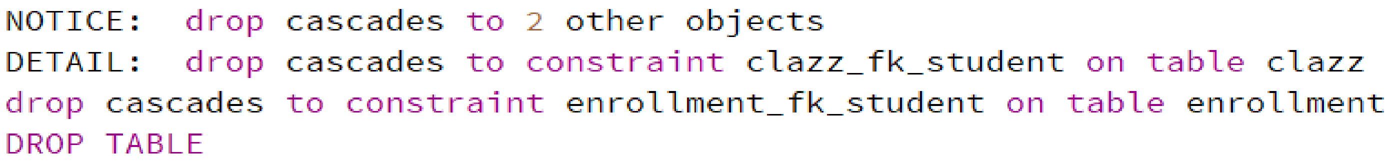
- Example:

```
DROP TABLE student;
```



```
ERROR:  cannot drop table student because other objects depend on it
DETAIL:  constraint clazz_fk_student on table clazz depends on table student
constraint enrollment_fk_student on table enrollment depends on table student
HINT:  Use DROP ... CASCADE to drop the dependent objects too.
SQL state: 2BP01
```

```
DROP TABLE student CASCADE;
```



```
NOTICE:  drop cascades to 2 other objects
DETAIL:  drop cascades to constraint clazz_fk_student on table clazz
drop cascades to constraint enrollment_fk_student on table enrollment
DROP TABLE
```

3. Data Manipulation

student

student_id	first_name	last_name	dob	gender	address	note	clazz_id
20160001	Ngọc An	Bùi	3/18/1987	M	15 Lương Định Của,Đ. Đa, HN		20162101
~~20160002~~	~~Anh~~	~~Hoàng~~	~~5/20/1987~~	~~M~~	~~513 B8 KTX BKHN~~		~~20162101~~
20160003	Thu Hồng	Trần	6/6/1987	F	15 Trần Đại Nghĩa, HBT, Hà nội		20162101
20160004	Minh Anh	Nguyễn	5/20/1987	F	513 TT Phương Mai, Đ. Đa, HN		20162101
20170001	Nhật Ánh	Nguyễn	5/15/1988	F	214 B6 KTX BKHN		20172201
20170002	Nhật Cường	Nguyễn	10/24/1988	M	214 B5 KTX BKHN		20172201
20170003	Nhật Cường	Nguyễn	1/24/1988	M	214 B5 KTX BKHN		20172201
20170004	Minh Đức	Bùi	1/25/1988	M	214 B5 KTX BKHN		20172201

Modifying address?

Adding new student / new class?

Deleting student data?

Retrieving list of all students?

clazz

clazz_id	name	lecturer_id	monitor_id
20162101	CNTT1.01-K61	02001	20160003
20162102	CNTT1.02-K61		
20172201	CNTT2.01-K62	02002	20170001
20172202	CNTT2.02-K62		

3. Data Manipulation

3.1. Insertion

- Syntax:

```
INSERT INTO <table1>[(<col1>,<col2>,…)] VALUES(<exp1>,<exp2>,…);

INSERT INTO <table1>[(<col1>,<col2>,…)]
      SELECT  <col1>, <col2>, …
      FROM    <tab1>, <tab2>, …
      WHERE   <condition>;
```

- Examples:

```
INSERT INTO clazz(clazz_id, name) VALUES ('20162101', 'CNTT1.01-K61');
INSERT INTO clazz(name, clazz_id) VALUES ('CNTT2.02-K62', '20172202');
INSERT INTO clazz VALUES ('20172201', 'CNTT2.01-K62', NULL, NULL);
```

3. Data Manipulation

3.2. Deletion, Update

- Deletion:

```
DELETE FROM <table_name> [WHERE <condition>];

DELETE FROM student WHERE student_id  = '20160002';
```

- Update:

```
UPDATE   <table_name>
SET      <col1> = <exp1>,
         <col2> = <exp2>,...
[WHERE   <condition>];

UPDATE   student
SET      address = '179 Le Thanh Nghi, HBT, HN'
WHERE    student_id = '20170003';
```

3. Data Manipulation

3.3. Examples

```sql
INSERT INTO clazz VALUES ('20172201', 'CNTT3.01-K62', NULL, NULL);
```

```
ERROR: duplicate key value violates unique constraint "clazz_pk"
DETAIL: Key (clazz_id)=(20172201) already exists. SQL state: 23505
```

```sql
UPDATE clazz SET monitor_id = '20160022' WHERE clazz_id = '20162102';
```

```
ERROR: insert or update on table "clazz" violates foreign key constraint "clazz_fk_student"
DETAIL: Key (monitor_id)=(20160022) is not present in table "student". SQL state: 23503
```

```sql
DELETE FROM clazz WHERE clazz_id = '20162101';
```

```
ERROR: update or delete on table "clazz" violates foreign key constraint "student_fk_clazz" on table
"student" DETAIL: Key (clazz_id)=(20162101) is still referenced from table "student". SQL state: 23503
```

```sql
UPDATE student SET gender ='N' WHERE student_id = '20160003';
```

```
ERROR: new row for relation "student" violates check constraint "student_chk_gender"
DETAIL: Failing row contains (20160003, Thu Hồng, Trần, 1987-06-06, N, 15 Trần Đại Nghĩa, HBT, Hà
nội, null, 20162101). SQL state: 23514
```

3. Data Manipulation

3.4. Querying data from a table

Retrieving column(s)

- Syntax:

```
SELECT <col_1>, <col_2>,… ,<col_n> | *
FROM <table_name>;
```

- Example:

clazz

clazz_id	name	lecturer_id	monitor_id
20162101	CNTT1.01-K61	02001	20160003
20162102	CNTT1.02-K61		
20172201	CNTT2.01-K62	02002	20170001
20172202	CNTT2.02-K62		

```
SELECT name, monitor_id
FROM clazz;
```

Result

name	monitor_id
CNTT1.01-K61	20160003
CNTT1.02-K61	
CNTT2.01-K62	20170001
CNTT2.02-K62	

3.4. Querying data from a table

Retrieving row(s)

- Syntax:

```
SELECT <col_1>, <col_2>,… ,<col_n> | *
FROM <table_name>
WHERE <condition_expression>;
```

- Example:

```
SELECT * FROM clazz
WHERE lecture_id = '02001'
OR lecture_id = '02002';
```

clazz

clazz_id	name	lecturer_id	monitor_id
20162101	CNTT1.01-K61	02001	20160003
20162102	CNTT1.02-K61		
20172201	CNTT2.01-K62	02002	20170001
20172202	CNTT2.02-K62		

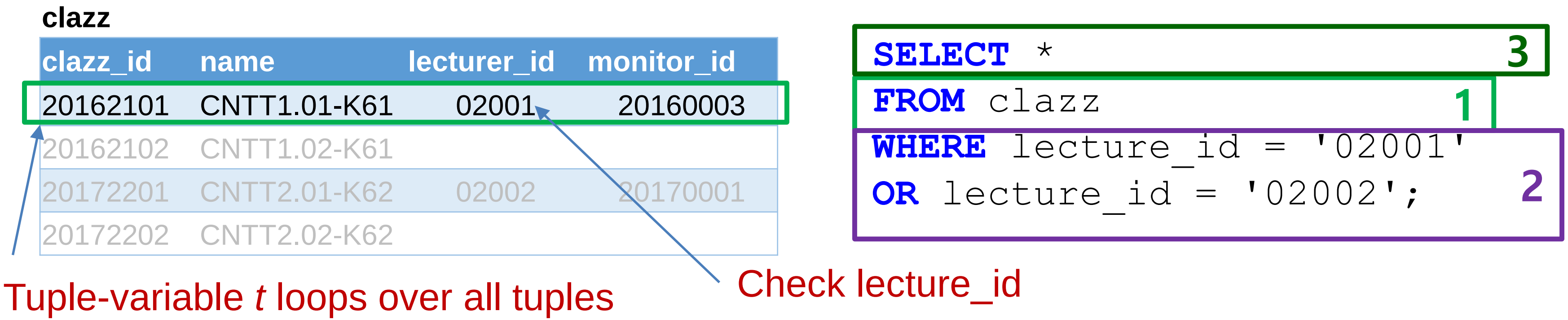
result

clazz_id	name	lecturer_id	monitor_id
20162101	CNTT1.01-K61	02001	20160003
20172201	CNTT2.01-K62	02002	20170001

3.4. Querying data from a table

Operational Semantics

- Think of a tuple variable visiting each tuple of the relation mentioned in FROM clause
- Check if the "current" tuple satisfies the WHERE clause
- If so, compute the attributes or expressions of the SELECT clause using the components of this tuple



clazz

clazz_id	name	lecturer_id	monitor_id
20162101	CNTT1.01-K61	02001	20160003
20162102	CNTT1.02-K61		
20172201	CNTT2.01-K62	02002	20170001
20172202	CNTT2.02-K62		

```
SELECT *
FROM clazz
WHERE lecture_id = '02001'
OR lecture_id = '02002';
```

Tuple-variable *t* loops over all tuples

Check lecture_id

3.4. Querying data from a table

Condition Expression

- Comparative operations: =, !=, <>, <, >, <=, >= , IS NULL, IS NOT NULL
- Logic operation: NOT, AND, OR
- Other operation: BETWEEN, IN, LIKE
  - Digital / string/ date data type
    - attr **BETWEEN** val1 **AND** val2(⇔ (attr>=val1) and (attr<=val2))
    - attr **IN** (val1, val2, ...) (⇔ (attr=val1) or (attr=val2) or ...)
  - String data type
    - **LIKE**: _ instead of one character

      % instead of any characters (string)

      attr **LIKE** '_IT%'

      attr **LIKE** 'IT%'

3. Data Manipulation

3.4. Querying data from a table

- **Examples**

student

student_id	first_name	last_name	dob	gender	address	note	clazz_id
20160001	Ngọc An	Bùi	3/18/1987	M	15 Lương Định Của,Đ. Đa, HN		20162101
20160002	Anh	Hoàng	5/20/1987	M	513 B8 KTX BKHN		20162101
20160003	Thu Hồng	Trần	6/6/1987	F	15 Trần Đại Nghĩa, HBT, Hà nội		20162101
20160004	Minh Anh	Nguyễn	5/20/1987	F	513 TT Phương Mai, Đ. Đa, HN		20162101
20170001	Nhật Ánh	Nguyễn	5/15/1988	F	214 B6 KTX BKHN		20172201
20170002	Nhật Cường	Nguyễn	10/24/1988	M	214 B5 KTX BKHN		20172201
20170003	Nhật Cường	Nguyễn	1/24/1988	M	214 B5 KTX BKHN		20172201
20170004	Minh Đức	Bùi	1/25/1988	M	214 B5 KTX BKHN		20172201

```sql
SELECT student_id, first_name, dob, address FROM student
WHERE address LIKE '%KTX%' AND gender = 'F';
```

result

student_id	first_name	last_name	dob	address
20170001	Nhật Ánh	Nguyễn	5/15/1988	214 B6 KTX BKHN

3. Data Manipulation

3.4. Querying data from a table

Pattern Matching

- Special character in the pattern: single quote ('), %, _
  - Single code (') ➔ use double single quote: `title LIKE '%''%'`

```
SELECT * FROM subject
WHERE name LIKE '%''%';
```

result

subject_id	name	credit	….
LI0001	life's happy song	5	
LI0002	%life's happy song 2	5	

  - Symbol %, _ ➔ use escape characters: `title LIKE 'x%%x_' ESCAPE 'x'`

```
SELECT * FROM subject
WHERE name LIKE 'x%%' ESCAPE 'x';
```

result

subject_id	name	credit	….
LI0002	%life's happy song 2	5	

3. Data Manipulation

● NULL value

- Arithmetic operators :

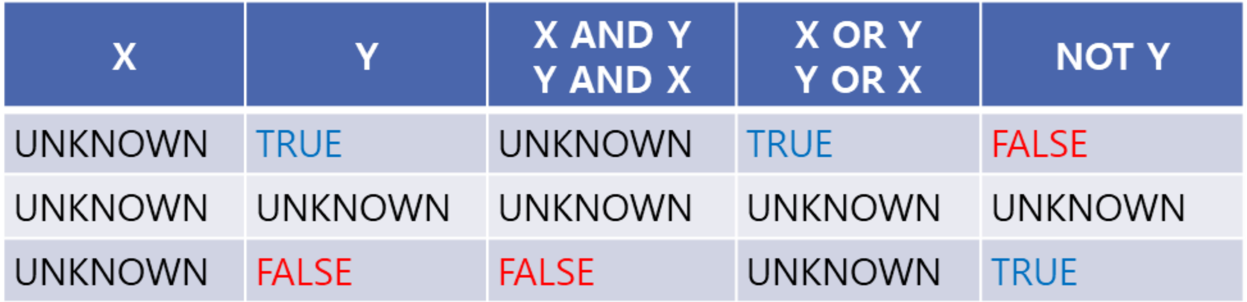
  NULL +-/x any value ➔ NULL
- Comparative operations:

  =, !=, <>, <, >, <=, >= with a NULL ➔ UNKNOWN

  (UNKNOWN: a truth-value as TRUE, FALSE)
- Check if an attribute has NULL value: IS NULL, IS NOT NULL
- Remark: NULL is not a constant
  - If x is NULL then **x + 3 results NULL**
  - **NULL + 3** : not a legal SQL expression

3. Data Manipulation

- **Truth-values: UNKNOWN (1/2), TRUE (1), FALSE (0)**
  - Comparative operations: with a NULL → UNKNOWN
  - Logic operation: AND ~MIN, OR ~MAX, NOT(x) ~ 1-x



X	Y	X AND Y Y AND X	X OR Y Y OR X	NOT Y
UNKNOWN	TRUE	UNKNOWN	TRUE	FALSE
UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN
UNKNOWN	FALSE	FALSE	UNKNOWN	TRUE

- Conditions in WHERE clauses apply on each tuples of some relation

  → Only the tuples for which the condition has the **TRUE** value become part of the answer

3. Data Manipulation

- **Examples**

subject

subject_id	name	credit	per..
IT1110	Tin học đại cương	4	60
IT3080	Mạng máy tính	3	70
IT3090	Cơ sở dữ liệu	3	70
IT4857	Thị giác máy tính	3	60
IT4866	Học máy	2	70
LI0001	life's happy song	5	
LI0002	%life's happy song 2	5	

```
SELECT * FROM subject
WHERE credit >= 4 AND
percentage_final_exam <= 60;
```

result

subject_id	name	credit	per..
IT1110	Tin học đại cương	4	60

```
SELECT * FROM subject
WHERE percentage_final_exam = NULL;
```

result

subject_id	name	credit	....

```
SELECT * FROM subject
WHERE percentage_final_exam IS NULL;
```

result

subject_id	name	credit	per..
LI0001	life's happy song	5	
LI0002	%life's happy song 2	5	

3. Data Manipulation

● Renaming output attributes

- Syntax:

  SELECT <col_name> **AS** <alias_name>, <expr> **AS** <alias_name>…

  FROM … **WHERE** ...

- Example:

```
SELECT subject_id AS id, name,
        credit "ETC"
FROM subject;
```

- Keyword **AS**: optional
  - <alias_name>: used in ORDER BY clause,
  - <alias_name>: not used in WHERE or HAVING clauses

result

id	name	ETC
IT1110	Tin học đại cương	4
IT3080	Mạng máy tính	3
IT3090	Cơ sở dữ liệu	3
IT4857	Thị giác máy tính	3
IT4866	Học máy	2
LI0001	life's happy song	5
LI0002	%life's happy song 2	5

Remark

- Each DBMS has its own implementation. So the syntax for each statement can vary from one database system to another:
  - Meaning of special characters used (%, _, *, ", '),
  - less or more options
  - standard part & extension part
- More options for each statement: see documentations of the DBMS used in your system

Practices

- Installing a DBMS
- Defining all relation schemas of Education database
- Do not forget constraints
- Inserting data into each table:
  - a lot of errors will be raised but it is good, try to understand these errors and correct them
  - Checking if defined constraints work
- Available documents:
  - detailed description for all tables the database
  - Tutorial of the installed DBMS
  - A demo sql script to define this database (avaiable before the next lession)

QUIZ (For Quiz 1, 2, 3)

Given table defined as follows:

```
CREATE TABLE subject (
    subject_id CHAR(6) NOT NULL,
    name VARCHAR(30) NOT NULL, credit INT NOT NULL,
    percentage_final_exam INT DEFAULT 70,
    CONSTRAINT subject_pk PRIMARY KEY (subject_id),
    CONSTRAINT subject_chk_credit CHECK (credit >=1 AND credit <=5),
    CONSTRAINT subject_chk_percentage CHECK percentage_final_exam
BETWEEN 0 AND 100) );
```

Quiz 1.

Quiz Number	1	Quiz Type	OX	Example Select
Question	Suppose that we execute this insert statement: `INSERT INTO subject(subject_id, name, credit) VALUES ('IT3091', 'Thực hành CSDL', 6);` What are values assigned to attribute credit and percentage_final_exam of new row inserted into database?			
Example	A. (6, 70) B. (6, NULL) C. (NULL 70) D. No new row inserted into the database			
Answer	[redacted]			
Feedback	[redacted]			

Quiz 2.

Quiz Number	1	Quiz Type	OX	Example Select
Question	Suppose that we execute this insert statement: `INSERT INTO subject(subject_id, name) VALUES ('IT1010', 'Tin học đại cương');` What's happen?			
Example	A. A row inserted successfully B. Error raised			
Answer	[redacted]			
Feedback	[redacted]			

Quiz 3.

Quiz Number	1	Quiz Type	OX	Example Select

Question	Given two queries, do they alway give the same output ?
	SELECT * FROM subject
	WHERE percentage_final_exam >= 60
	      OR percentage_final_exam < 60 ;
	SELECT * FROM subject;
	A.Yes
	B.No			

Example	A. True B. False			
Answer	[redacted]			
Feedback	[redacted]			

Quiz 4.

Quiz Number	1	Quiz Type	OX	Example Select
Question	For each table we must define a primary key ?			
Example	A. True B. False			
Answer	[redacted]			
Feedback	[redacted]			

Quiz 5.

Quiz Number	2	Quiz Type	OX	Example Select
Question	How many foreign keys and primary keys can we define for a table?			
Example	A. Primary key: zero or one; foreign key: zero or one B. Primary key: zero or one; foreign key: zero, one or more C. Primary key: zero, one or more; foreign key: zero, one or more D. Primary key: zero, one or more; foreign key: zero or one			
Answer	[redacted]			
Feedback	[redacted]			

Summary

1. **Introduction to SQL**
    - A brief history of SQL
    - SQL languages
2. **Definition a relation schema**
    - Creating a simple table
    - Defining constraints
    - Modifying relation schema: modifying data structure, modifying constraints
3. **Data manipulation**
    - Populating a table with rows
    - Removing row(s) from a table
    - Updating existing rows
    - Querying a table

Congratulation!
You’ve already completed the 1st lesson of Database

Next lesson guide…

Structured Query Language – part 2

reference

- Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom. Database Systems: The Complete Book. Pearson Prentice Hall. the 2nd edition. 2008: Chapter 6
- Nguyen Kim Anh, Nguyên lý các hệ cơ sở dữ liệu, NXB Giáo dục. 2004: Chương 3