Structured Query Language (SQL) – part 1

Learning Maps

Sequence	Title
1	Introduction to databases
2	Relational Databases
3	Relational Algebra
4	Structured Query Language – Part 1
5	Structured Query Language – Part 2
6	Constraints and Triggers
7	Entity Relationship Model
8	Functional Dependency
9	Normalization
10	Storage - Indexing
11	Query Processing
12	Transaction Management – Part 1
13	Transaction Management – Part 2

Intro > Overview

☐ A : Voice and PPT Overview
☐ B : Text-based Overview
■ C : Video and PPT Overview

Opening Message	→ In this lesson, we will study.
Lesson topic	1. Introduction to Relational Database languages 2. Definition a Relation schema 3. Data Manipulation
Learning Goals	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

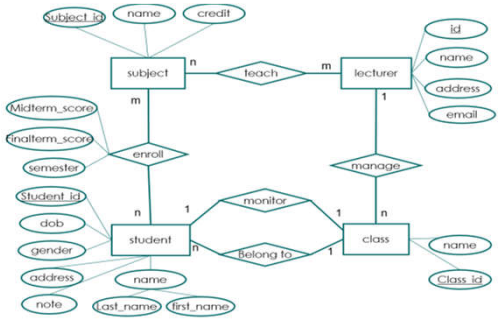
Intro > Keywords

Keyword	Description
DBMS	Database Management System : system software for creating and managing 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

Lesson > Topic 1: Introduction to SQL

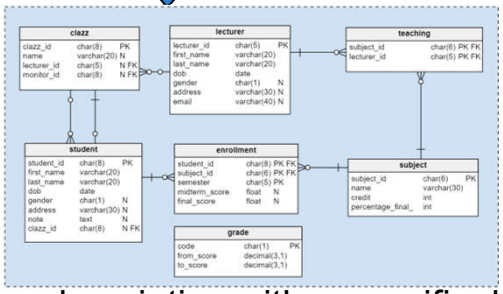
Business description

1: Data analyse



Data modeling (ex.: ER schema)

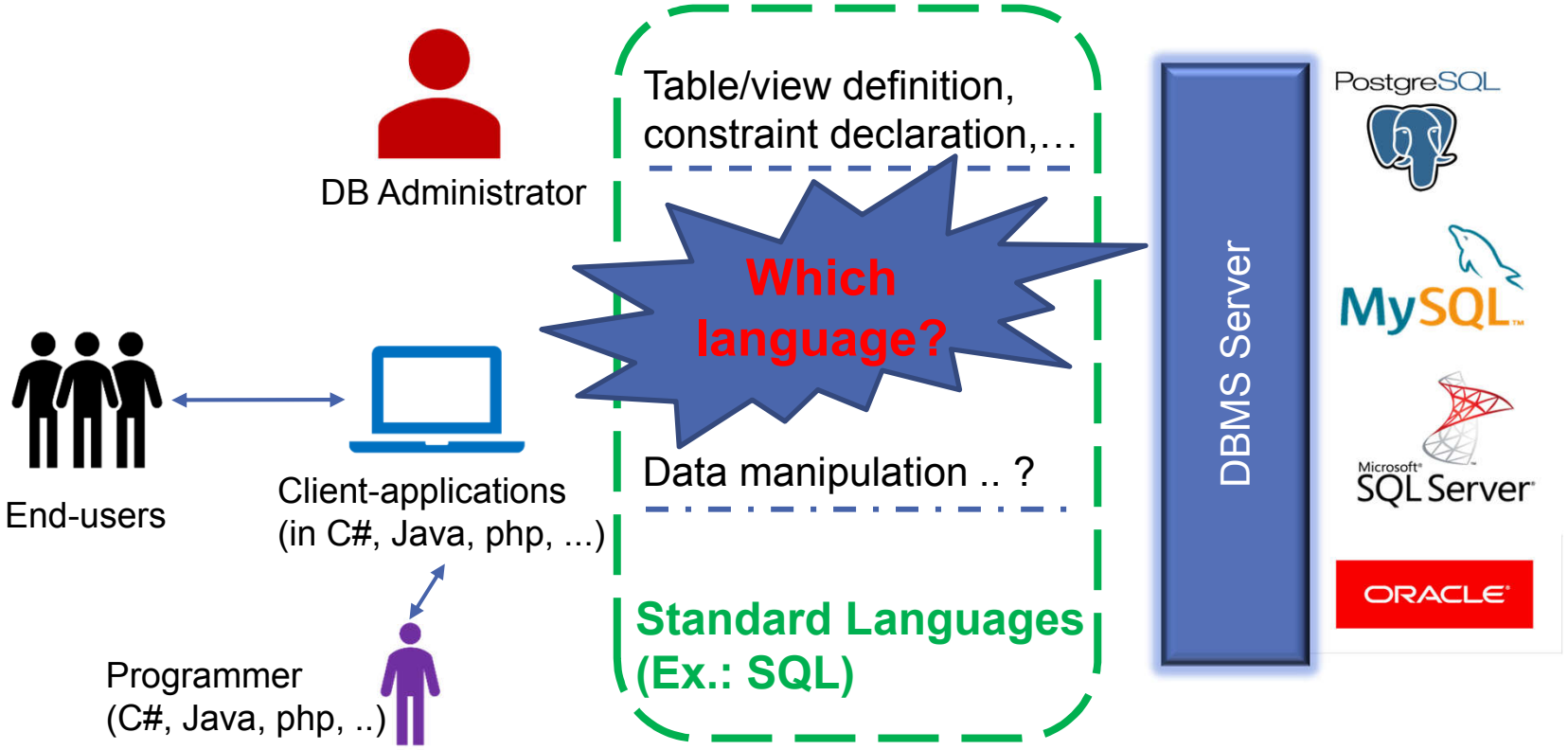
2: database design



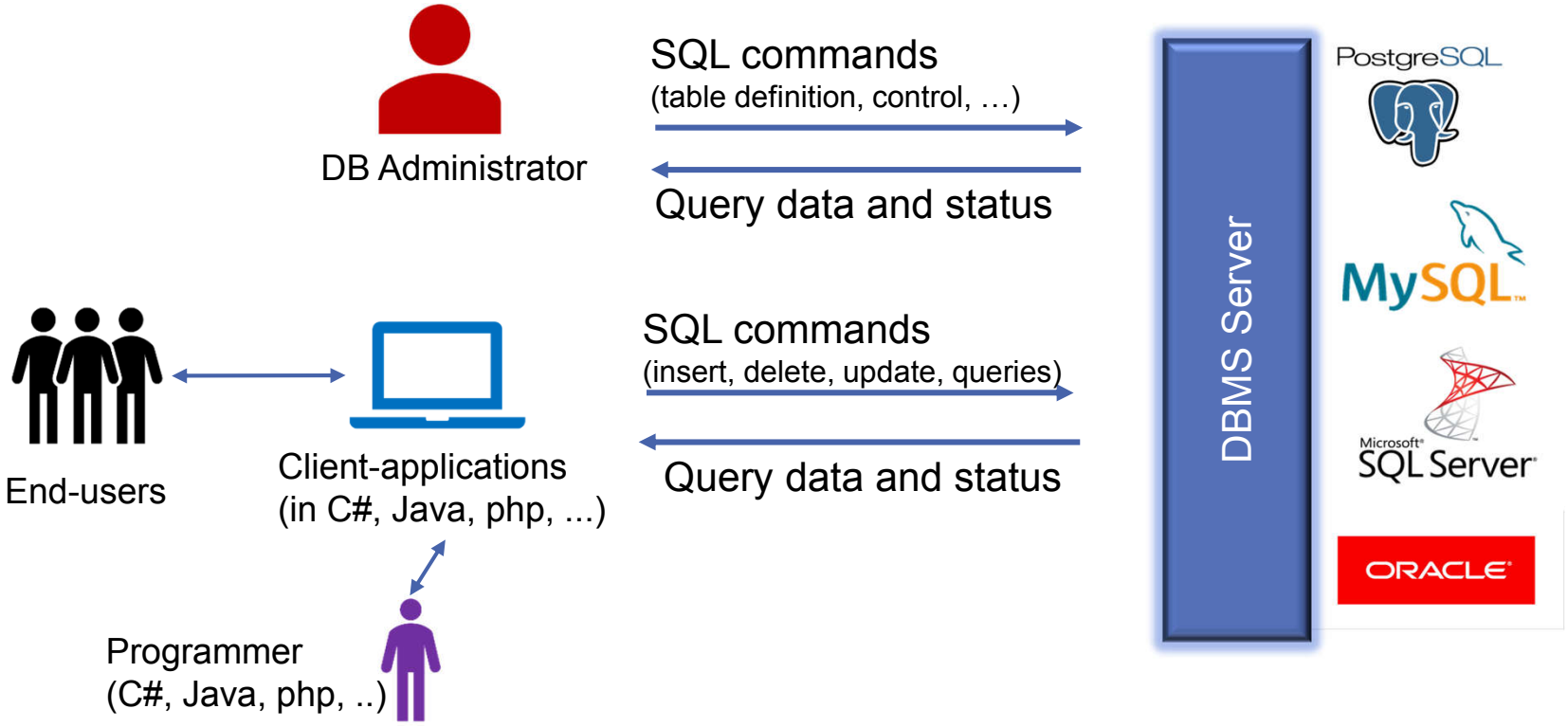
Database description with a specific data model
(Ex.: relational database model)

3: Implementation

Lesson > Topic 1: Introduction to SQL



Lesson > Topic 1: Introduction to SQL



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, fetch clause, row pattern matching, JSON, ….

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, ...

Lesson > Topic 2: Definition a Relation Schema

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)

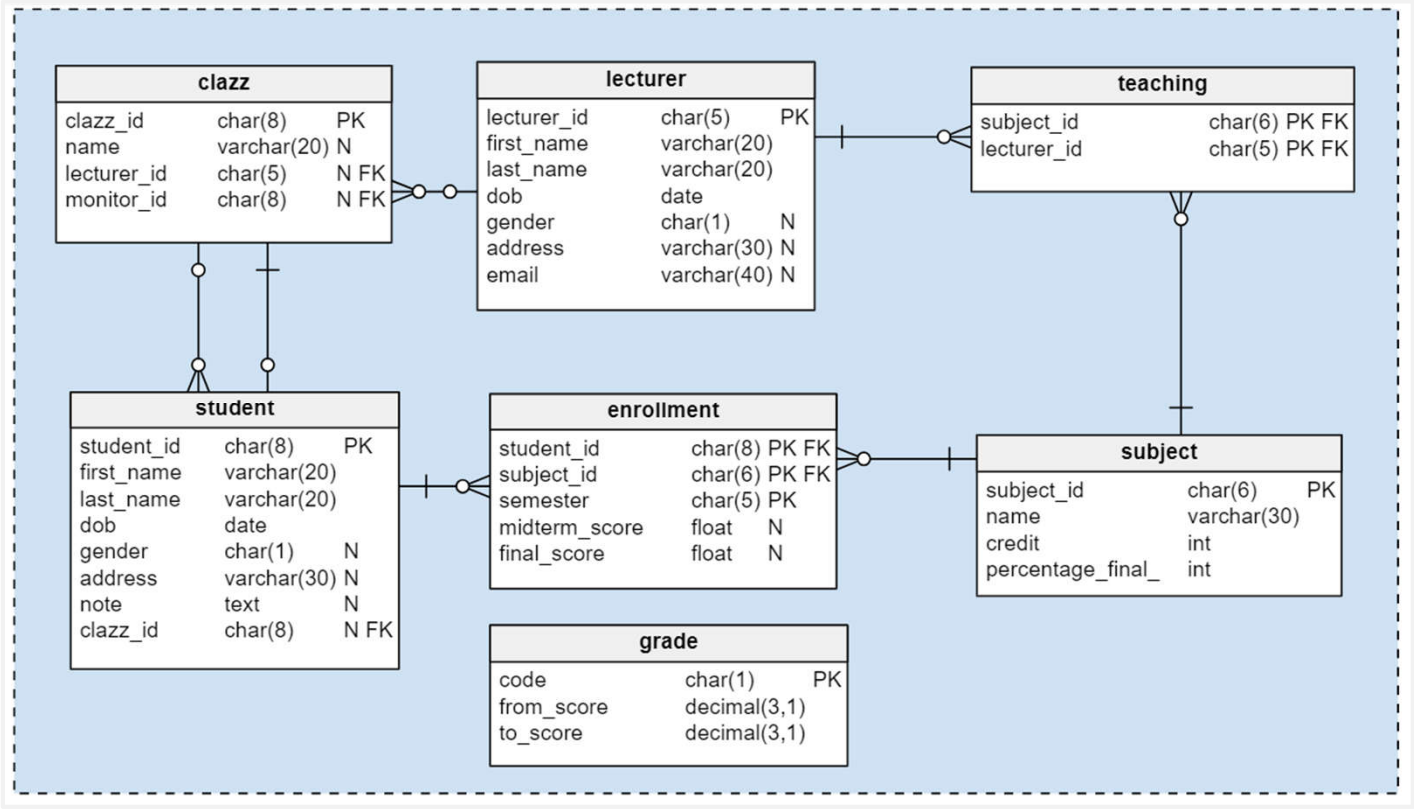
  clazz(**clazz_id**, name, *lecturer_id*, *monitor_id*)

  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. Definition a Relation Schema

2.1. Simple Relation Definition

2.2. Constraints

2.3. Modifying Relation Schema

2.4. Drop a Relation from Database

2. Definition a Relation Schema

2.1 Simple Relation Definition

- 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 Simple Relation Definition

Naming conventions

- Ordinary identifiers:
  - Must begin with a letter
  - Contain only: letters (a..z), underscore (_), and digits (0..9)
  - Non longer than 32 characters
- Delimited identifiers:
  - Identifiers surrounded by double quotation marks (")
  - Can contain any characters

2. Definition a Relation Schema

► 2.1 Simple Relation Definition

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 Simple Relation Definition

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
date	calendar date without time of day
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 Simple Relation Definition

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

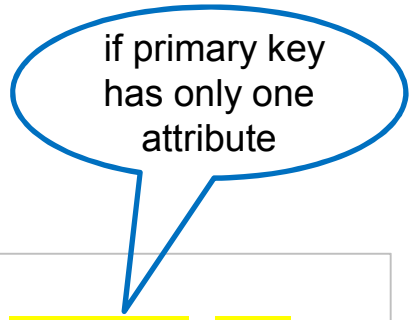
PRIMARY KEY

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) );
```



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 studen
t(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> <new_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**
- 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
```

Lesson > Topic 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 student list?

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(clazz_id, name, lecturer_id, monitor_id)
     VALUES ('20162102', 'CNTT1.02-K61', NULL, NULL);
INSERT INTO clazz VALUES ('20172201', 'CNTT2.01-K62', NULL, NULL);
```

3. Data Manipulation

▶ 3.2 Deletion, Update

- Syntax:

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

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

- Examples:

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

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

3. Data Manipulation

▶ 3.2 Examples

```
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

```
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

```
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

```
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

▶ 3.3 Quering 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. Data Manipulation

▶ 3.3 Quering 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. Data Manipulation

▶ 3.3 Quering 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 *                          3
FROM clazz                        1
WHERE lecture_id = '02001'
OR lecture_id = '02002';          2
```

Tuple-variable *t* loops over all tuples

Check lecture_id

3. Data Manipulation

▶ 3.3 Quering 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. Data Manipulation

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

```
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. Data Manipulation

Pattern Matching

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

```
title LIKE '%''%': title contains a single quote
```

  - Symbol %, _ ➔ use escape characters:

```
title LIKE 'x%%x_' ESCAPE 'x':
title begins with the % character and ends with the _ character
```

- Example:

3. Data Manipulation

▶ 3. Data Manipulation

Pattern Matching - Example

subject

subject_id	name	credit	….
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 name LIKE '%''%';
```

result

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

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

result

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

3. Data Manipulation

► 3. Data Manipulation

Dates /Times

- Variety of format : 12/24/2018, 24 Dec 2018, 24/12/2018, ..
- SQL standard:
  - Date constant: DATE '2018-12-24'
  - Time constant: TIME '15:00:20.5'
  - Timestamp constant: TIMESTAMP '2018-12-24 15:00:20'

3. Data Manipulation

► 3. Data Manipulation

NULL value

- Arithmetic operator :

  NULL +-/x any value ➔ NULL

- Comparative operations:

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

(UNKNOWN: a truth-value as TRUE, FALSE)

- Check if a 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

▶ 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 value **TRUE** become part of the answer

3. Data Manipulation

▶ 3. Data Manipulation

Example

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	

Pratices

- 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 an d correct them
  - Checking if defined constraints work
- Avaiable 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

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) );
```

And actual data on the table :

subject

subject_id	name	credit	percentage_final_exam
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

Quiz

No	Question (Multiple Choice)	Answer (1,2,3,4)	Commentary
1	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? A. (6, 70) B. (6, NULL) C. (NULL 70) D. No new row inserted into the database	D	The check constraint `subject_chk_credit is violated`
2	Suppose that we execute this insert statement: `INSERT INTO subject(subject_id, name, credit) VALUES ('IT3090', 'Thực hành CSDL', 5);` What's happen? A. A row inserted successfully B. Error raised	B	ERROR: duplicate key value violates unique constraint "subject_pk"
3	Suppose that we execute this insert statement: `INSERT INTO subject(subject_id, name) VALUES ('IT1010', 'Tin học đại cương');` What's happen? A. A row inserted successfully B. Error raised	B	`Error: null value in column "credit" violates not-null constraint`

Quiz

No	Question (Multiple Choice)	Answer (1,2,3,4)	Commentary
4	For each table we must define a primary key ? A. True B. False	B	A table may have no primary key. If you do not define primary key for the table, DBMS can not help you to check duplicated values/ rows pr oblem. So, each table should have its primary key.
5	You must define all constraints when defining table? A. Yes B. No	B	Alter table can help you to add more constraints
6	How many attributes are there in a primary key, a foreign key? A. Primary key : only one ; foreign key: only one B. Primary key : only one ; foreign key: one or more C. Primary key : one or more ; foreign key: one or more D. Primary key : one or more ; foreign key: only one	C	
7	How many foreign keys and primary keys can we define for a table ? 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	B	A table has only one primary key, but it can have 0, 1 or many foreign key(s). A table can have also 0, 1 or many check constraint(s).

Outro > Summary

No	Topic	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 - Quering a table

You've just have an overview of ……

Next lesson:

Structured Query Language – part 2

1. Data Manipulation (part 2)
2. User Management
3. View definition