ECE30030/ITP30010 Database Systems

Structured Query Language

Reading: Chapter 3

Charmgil Hong

charmgil@handong.edu

Spring, 2023

Handong Global University

Agenda

- SQL data manipulation language (DML)

Running Examples

- Relations (tables): *instructor*, *teaches*

Instructor relation

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000.00
12121	Wu	Finance	90000.00
15151	Mozart	Music	40000.00
22222	Einstein	Physics	95000.00
32343	El Said	History	60000.00
33456	Gold	Physics	87000.00
45565	Katz	Comp. Sci.	75000.00
58583	Califieri	History	62000.00
76543	Singh	Finance	80000.00
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
98345	Kim	Elec. Eng.	80000.00

teaches relation

ID	course_id	sec_id	semester	year
76766	BIO-101	1	Summer	2017
76766	BIO-301	1	Summer	2018
10101	CS-101	1	Fall	2017
45565	CS-101	1	Spring	2018
83821	CS-190	1	Spring	2017
83821	CS-190	2	Spring	2017
10101	CS-315	1	Spring	2018
45565	CS-319	1	Spring	2018
83821	CS-319	2	Spring	2018
10101	CS-347	1	Fall	2017
98345	EE-181	1	Spring	2017
12121	FIN-201	1	Spring	2018
32343	HIS-351	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017

Running Examples

- Relations (tables): *course*, *takes*

course relation

course_id	title	dept_name	credits
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

takes relation

ID	course_id	sec_id	semester	year	grade
00128	CS-101	1	Fall	2017	A
00128	CS-347	1	Fall	2017	A-
12345	CS-101	1	Fall	2017	C
12345	CS-190	2	Spring	2017	A
12345	CS-315	1	Spring	2018	A
12345	CS-347	1	Fall	2017	A
19991	HIS-351	1	Spring	2018	B
23121	FIN-201	1	Spring	2018	C+
44553	PHY-101	1	Fall	2017	B-
45678	CS-101	1	Fall	2017	F
45678	CS-101	1	Spring	2018	B+
45678	CS-319	1	Spring	2018	B
54321	CS-101	1	Fall	2017	A-
54321	CS-190	2	Spring	2017	B+
55739	MU-199	1	Spring	2018	A-
76543	CS-101	1	Fall	2017	A
76543	CS-319	2	Spring	2018	A
76653	EE-181	1	Spring	2017	C
98765	CS-101	1	Fall	2017	C-
98765	CS-315	1	Spring	2018	B
98988	BIO-101	1	Summer	2017	A
98988	BIO-301	1	Summer	2018	*null*

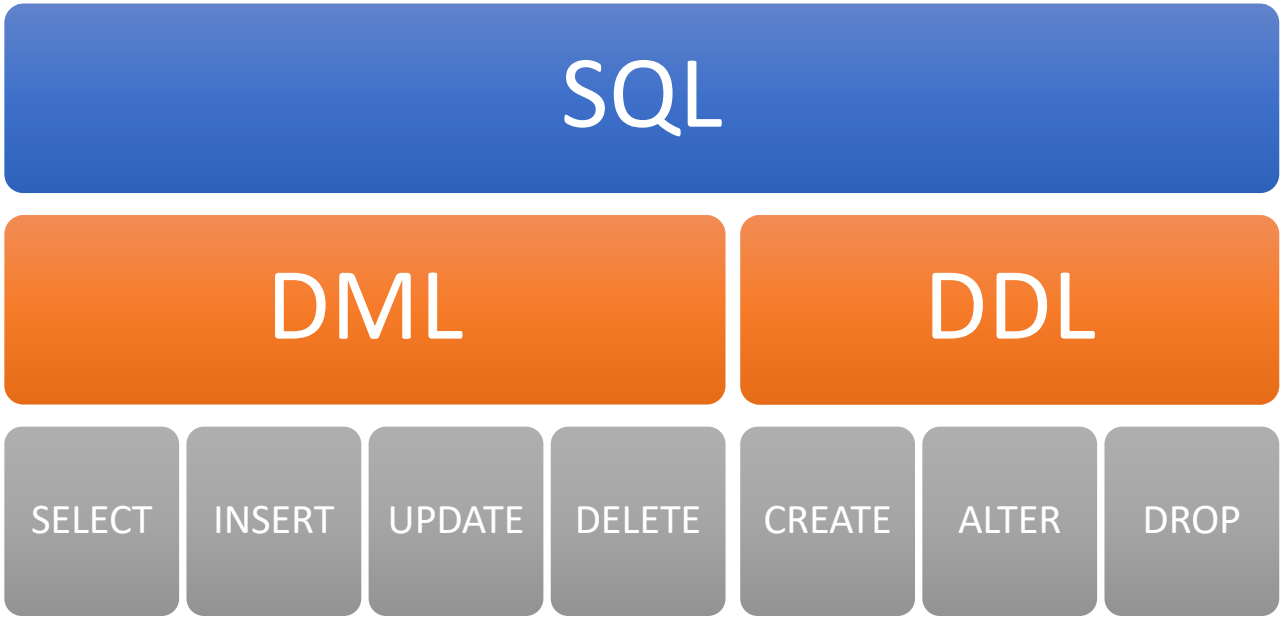
Running Examples

- Relations (tables): *student*

student relation

ID	name	dept_name	tot_cred
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
19991	Brandt	History	80
23121	Chavez	Finance	110
44553	Peltier	Physics	56
45678	Levy	Physics	46
54321	Williams	Comp. Sci.	54
55739	Sanchez	Music	38
70557	Snow	Physics	0
76543	Brown	Comp. Sci.	58
76653	Aoi	Elec. Eng.	60
98765	Bourikas	Elec. Eng.	98
98988	Tanaka	Biology	120

SQL Commands



INSERT

- Basic syntax
  - Insert data into every column:
    - **INSERT INTO** *tablename*
      VALUES (*col1_value, col2_value, …*)
    - Must list values in the same order as in the table schema
    - If some data values are unknown, must type NULL
    - For character sequences, use quotation marks
      - Single quotation marks are preferred (but double quotation marks are allowed)
      - Value in quotations is case-sensitive
  - Insert data into selected columns
    - **INSERT INTO** *tablename* (*col1_name, col3_name, col4_name, …*)
      VALUES (*col1_value, col3_value, col4_value, …*)

same order, same number, same datatype.

no col2.

no value for col2. ⇒ col2 will have null value

INSERT

- Add a new tuple to *course*
  - **INSERT INTO** *course*
    VALUES ('CS-437', 'Database Systems', 'Comp. Sci.', 4);

- or equivalently
  - **INSERT INTO** *course* (*course_id*, *title*, *dept_name*, *credits*)
    VALUES ('CS-437', 'Database Systems', 'Comp. Sci.', 4);

- Add a new tuple to *student* with *tot_creds* set to null
  - **INSERT INTO** *student*
    VALUES ('3003', 'Green', 'Finance', *null*);

INSERT

- A foreign key specifies that an attribute from one relation has to map to a tuple in another relation
  - Value in one relation must appear in another relation

Relation: instructor

ID	name	dept_name	salary
76766	Crick	Biology	72000.00
83821	Brandt	Comp. Sci.	92000.00
45565	Katz	Comp. Sci.	75000.00

If — — History — then get rejected since there is no History in department table.

Relation: department

dept_name	building	budget
Biology	Watson	90000.00
Comp. Sci.	Taylor	100000.00
Elec. Eng.	Taylor	85000.00

INSERT

- A foreign key specifies that an attribute from one relation has to map to a tuple in another relation
  - Value in one relation must appear in another relation

- Make sure all foreign keys that new row references have already been added to database
  - One cannot insert a foreign key value unless the corresponding value exists in the referenced relation

INSERT

- Inserting results of other SELECT query
  - Make each student in the Music department who has earned more than 144 credit hours an instructor in the Music department with a salary of $18,000
    - **INSERT INTO** *instructor*
      SELECT *ID, name, dept_name, 18000*
      FROM *student*
      WHERE *dept_name* = 'Music' **AND** *total_cred* > 144;
  - The **SELECT FROM WHERE** statement is evaluated fully before any of its results are inserted into the relation
    - Otherwise queries like
      INSERT INTO *table1* **SELECT** * **FROM** *table1*
      would cause problem

UPDATE

- Basic syntax
  - Updating a table
    - **UPDATE** *tablename*
      SET *col1_name = new_col1_value, col2_name = new_col2_value, …;*
  - Updating a table with conditions
    - **UPDATE** *tablename*
      SET *col1_name = new_col1_value, col2_name = new_col2_value, …*
      WHERE *predicate;*

UPDATE

- Give a 5% salary raise to all instructors
  - **UPDATE** *instructor*
    SET *salary* = *salary* * 1.05

- Give a 5% salary raise to those instructors who earn less than 70000
  - **UPDATE** *instructor*
    SET *salary* = *salary* * 1.05
    WHERE *salary* < 70000;

- Give a 5% salary raise to instructors whose salary is less than average
  - **UPDATE** *instructor*
    SET *salary* = *salary* * 1.05
    WHERE *salary* < (**SELECT AVG**(salary) **FROM** *instructor*);

evaluated first ;;

UPDATE

- Increase salaries of instructors whose salary is over $100,000 by 3%, and all others by a 5%
  - Write two UPDATE statements:

① **UPDATE** *instructor*
SET *salary* = *salary* * 1.03
WHERE *salary* > 100000;

② **UPDATE** *instructor*
SET *salary* = *salary* * 1.05
WHERE *salary* <= 100000;

순서 중요함 이런상황에서.

②→① 일 경우 두번다 적용될수 있음.

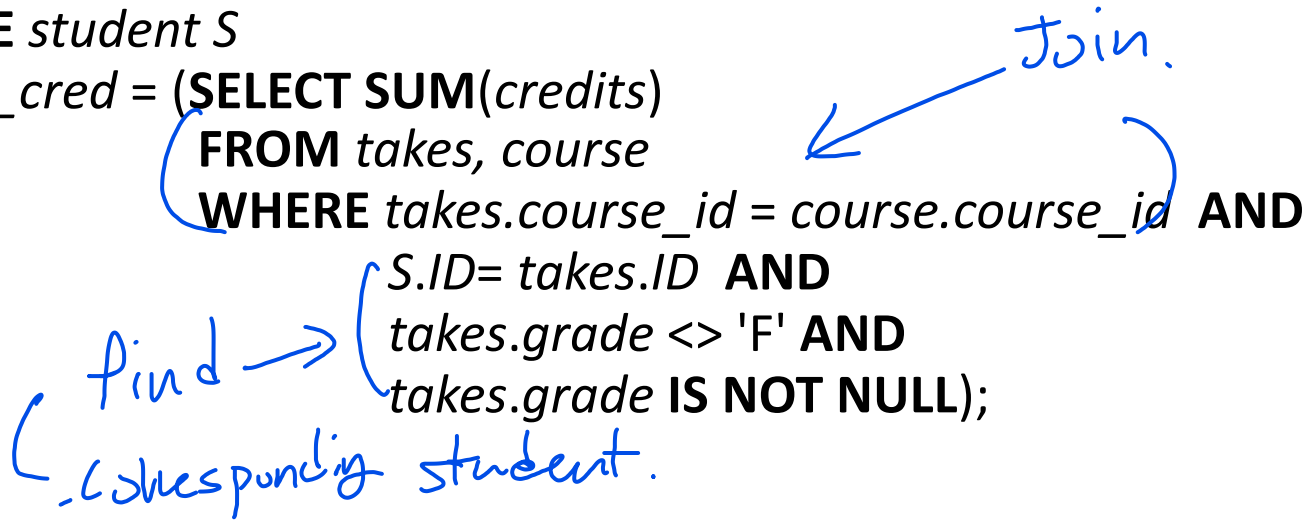
  - The order is important
  - Can be done better using the **case** statement (next slide)

CASE Statement for Conditional Update

- The following query is equivalent to the previous UPDATE queries
  - **UPDATE** *instructor*
    SET *salary* = **CASE**
      WHEN *salary* <= 100000 **THEN** *salary* * 1.05
      ELSE *salary* * 1.03
    END

UPDATE with Scalar Subqueries

- Recompute and update *tot_creds* value for all students
  - **UPDATE** *student S*
    SET *tot_cred* = (**SELECT SUM**(*credits*)
    FROM *takes, course*
    WHERE *takes.course_id = course.course_id* **AND**
    S.ID= takes.ID **AND**
    takes.grade <> 'F' **AND**
    takes.grade **IS NOT NULL**);



DELETE

- Basic syntax
  - To remove specific rows
    - **DELETE FROM** *tablename*
      WHERE *predicate*;
  - To remove all rows
    - **DELETE FROM** *tablename*;
    - This is equivalent to **TRUNCATE**:
      TRUNCATE (**TABLE**) *tablename*;
    - One cannot truncate a table with foreign key constraints
      - Must disable the constraints first (we will cover **ALTER** when we study SQL DDL):
        ALTER TABLE *tablename*
        DISABLE CONSTRAINT *constraint_name*;

DELETE

- Delete all instructors
  - **DELETE FROM** *instructor*;

- Delete all instructors from the Finance department
  - **DELETE FROM** *instructor*
    WHERE *dept_name*= 'Finance';

- Delete all tuples in the instructor relation for those instructors associated with a department located in the Watson building
  - **DELETE FROM** *instructor*
    WHERE *dept name* **IN** (**SELECT** *dept name*
    FROM *department*
    WHERE *building* = 'Watson');

DELETE

- Delete all instructors whose salary is less than the average salary of instructors
  - Example: **DELETE FROM** *instructor*
    WHERE *salary* < (**SELECT AVG** (*salary*)
    FROM *instructor*);

- Issue: as we delete tuples from *instructor*, the average salary changes
  - Solution used in SQL:
    1. First, compute **AVG**(*salary*) and find all tuples to delete
    2. Next, delete all tuples found above (without recomputing **AVG** or retesting the tuples)

EOF

- Coming next:
  - Nested subqueries
  - Set membership (SOME, ALL, EXISTS)