Database System Engineering

SWEN304
SQL tutorial

Hui Ma, Yi Mei
hui.ma@ecs.vuw.ac.nz
yi.mei@ecs.vuw.ac.nz

Structured Query Language (SQL)

- We are concerned with three major problems
  - How to define a database schema and its relation schemas?
  - How to store and manipulate data in relations?
  - How to retrieve data from a database?
- SQL is used as
  - a Data Definition Language (DDL)
  - a Data Manipulation Language (DML)
  - a Query Language (QL)

SQL as DDL

- Catalog
- Domains
- Tables (Relation)
- Constraints
- Modify schemas (DROP and ALTER)

Catalog

- The catalog records essential information (meta data) about this database
  - Schema table: information on all relation schemas
  - Attribute table: information on all attributes
  - Data types table: information on all data types (domains)
  - Constraints table: information on all constraints

SQL Definitions

```
CREATE SCHEMA UNIVERSITY AUTHORIZATION huima;


CREATE DOMAIN idno
    AS  INT
    DEFAULT 300001
    NOT NULL
    CONSTRAINT idnoconstr
        CHECK (VALUE > 300000 AND VALUE <= 399999);
```

UNIVERSITY Database

UNIVERSITY ={STUDENT(StudId, Lname, Fname, Major),
COURSE(CourId, Cname, Points, Dept),
GRADES(StudId, CourId, Grade)}

STUDENT			
StudId	**Lname**	**Fname**	**Major**
300111	Smith	Susan	COMP
300121	Bond	James	MATH
300143	Bond	Jenny	MATH
300132	Smith	Susan	COMP

COURSE			
CourId	**Cname**	**Points**	**Dept**
COMP302	DB sys	15	Engineering
COMP301	softEng	20	Engineering
COMP201	Pr & Sys	22	Engineering
MATH214	DisMat	15	Mathematics

GRADES		
StudId	**CourId**	**Grade**
300111	COMP302	A+
300111	COMP301	A
300111	MATH214	A
300121	COMP301	B
300132	COMP301	C
300121	COMP302	B+
300143	COMP201	ω
300132	COMP201	ω
300132	COMP302	C+

Relation Schema: COURSE

```
CREATE TABLE COURSE (
   CourId CHAR(7)  CONSTRAINT cspk  PRIMARY KEY,
   CName CHAR(15) NOT NULL,
   Points INT NOT NULL
      CONSTRAINT pointschk
         CHECK (Points >= 0 AND Points <= 50),
   Dept  CHAR(25)
);
```

With/Without Domain

```
CREATE TABLE STUDENT (
    StudId  INT
       NOT NULL
       DEFAULT 30000
       CONSTRAINT stpk PRIMARY KEY
       CONSTRAINT StIdRange CHECK
          (StudId  BETWEEN 300000 AND 399999),
    LName  CHAR(15) NOT NULL,
    FName  CHAR(15) NOT NULL,
    Major  CHAR(25) DEFAULT 'Comp'
);
```

```
CREATE TABLE STUDENT (
    StudId  idno CONSTRAINT stpk PRIMARY KEY,
    LName  CHAR(15) NOT NULL,
    FName  CHAR(15) NOT NULL,
    Major  CHAR(25) DEFAULT 'Comp'
);
```

Second Key

- Unique and Not Null

```
CREATE TABLE STAFF (
  staff_id  INT PRIMARY KEY,
  ird_number  CHAR(7) NOT NULL UNIQUE,
  address VARCHAR(255)
);
```

DROP

- CASCADE behavior means deleting the construct itself and all the other constructs related to it,
- RESTRICT behavior means that the construct will be deleted only if it is empty (schema), or not referenced by any other construct (like: table, attribute, view)
- e.g. DROP TABLE COURSE RESTRICT

  DROP TABLE STUDENTS CASCADE

ALTER

```
ALTER TABLE STUDENT ADD NoOfPoints INT DEFAULT
   320;

ALTER TABLE GRADES ALTER Grade SET DEFAULT ‘C';

ALTER TABLE GRADES DROP CONSTRAINT gsri ;

ALTER TABLE STUDENT DROP CONSTRAINT stpk
   CASCADE;
```

SQL as DML

UNIVERSITY ={STUDENT(StudId, Lname, Fname, Major),
COURSE(CourId, Cname, Points, Dept),
GRADES(StudId, CourId, Grade)}

IC = {GRADES[Id] $\subseteq$ STUDENT[Id],
GRADES[Course_id] $\subseteq$ COURSE[Course_id]}

STUDENT			
StudId	**Lname**	**Fname**	**Major**
300111	Smith	Susan	COMP
300121	Bond	James	MATH
300143	Bond	Jenny	MATH
300132	Smith	Susan	COMP

COURSE			
CourId	**Cname**	**Points**	**Dept**
COMP302	DB sys	15	Engineering
COMP301	softEng	20	Engineering
COMP201	Pr & Sys	22	Engineering
MATH214	DisMat	15	Mathematics

GRADES		
StudId	**CourId**	**Grade**
300111	COMP302	A+
300111	COMP301	A
300111	MATH214	A
300121	COMP301	B
300132	COMP301	C
300121	COMP302	B+
300143	COMP201	ω
300132	COMP201	ω
300132	COMP302	C+

SQL as DML

- Three commands used to modify the database:
  - `INSERT`, `DELETE`, and `UPDATE`
- Full insert

```
INSERT INTO STUDENT
    VALUES (111111, 'Bole', 'Ann', Math);
```

- Partial insert

```
INSERT INTO STUDENT (FName, LName, StudId )
    VALUES('Ann', 'Bole', 111111);
```

UPDATE and DELETE

```
UPDATE GRADES
 SET Grade = 'A+'
 WHERE  CourId  = 'C302';


DELETE  FROM STUDENT
 WHERE  FName = 'Susan';


DELETE  FROM STUDENT
 WHERE StudId IN
    (SELECT s.StudId
      FROM STUDENT s, GRADES g
      WHERE  s.StudId = g.StudId AND CourId =
                'C302');
```

DROP vs DELETE

- DELETE statement performs conditional based deletion, whereas DROP command deletes entire records in the table
- DELETE statement removes only the rows in the table and it preserves the table structure as same whereas DROP command removes all the data in the table and the table structure
- DELETE operation can be rolled back and it is not auto committed, while DROP operation cannot be rolled back in any way as it is an auto committed statement
- DROP is a DDL statement while DELETE is a DML statement

SQL as QL

- `SELECT` is the basic SQL statement for retrieving data from a database

```
SELECT  [ DISTINCT ]  ⟨attribute_list⟩
FROM  ⟨table_list⟩
[ WHERE  ⟨condition⟩  ]
```

SQL as QL

- SQL does not automatically eliminate duplicate tuples in query results
- Use the keyword `DISTINCT` in the `SELECT` clause
  - Only distinct tuples should remain in the result

The University Database

- Suppose for each student only pass grades are recorded in the database

STUDENT

LName	FName	StudId	Major
Smith	Susan	131313	Comp
Bond	James	007007	Math
Smith	Susan	555555	Comp
Cecil	John	010101	Math

COURSE

PName	CourId	Points	Dept
DB Sys	C302	15	Comp
SofEng	C301	15	Comp
DisMat	M214	22	Math
Pr&Sys	C201	22	Comp

GRADES

StudId	CourId	Grade
007007	C302	A+
555555	C302	D
007007	C301	A
007007	M214	A+
131313	C201	B-
555555	C201	C
131313	C302	D
007007	C201	A
010101	C201	D

Single Table Queries

- Retrieve the first and last names of Comp students

```
SELECT FName, LName
FROM STUDENT
WHERE Major = 'Comp';
```

FName	LNam e
Susan	Smith
Susan	Smith

- Find all different grades

```
SELECT DISTINCT Grade
FROM GRADES ;
```

Grade
A+
A
B-
C

Arithmetic Operations, Sorting

- SQL provides capability to perform four basic arithmetic operations (+, -, *, /) that can be applied to numeric attributes and constants only

```
SELECT 2 + 2;
```

- Sorting of the query result tuples is done using

```
ORDER BY { ⟨attribute_name⟩ [(ASC|DESC)],...}
```

clause after the WHERE clause (ASC is default)

```
SELECT *
FROM GRADES
ORDER BY StudId ASC, CourId DESC;
```

Qualification and Aliasing

- Attributes in different relation schemas can have the same names. How do we prevent ambiguity?
- In the `SELECT` clause, we prefix attributes by table name: `SELECT` STUDENT.StudId …
- In the `FROM` clause, specify a tuple variable from the table: …`FROM` COURSE c, GRADES g, STUDENT s
- In the `WHERE` clause, prefix an attribute by the tuple variable: `WHERE` c.CourId = g.CourId

Join

- To retrieve data from more than one table, we need a new operation: `JOIN`
- There are different joins:
  - `INNER` (theta join, equi-join, natural join)
  - `OUTER` (left, right, full)
  - Most often, we use the equi-join

A JOIN Example

```
SELECT * FROM r1, r2 WHERE r1.B = r2.B ;
```

Join Condition

r1

A	B
1	b1
2	b1
3	b2
4	b3
5	ω

r2

B	C
b1	c1
b2	c1
b3	c1
b4	c2

r1 EQUI-JOIN r2

A	B	B	C
1	b1	b1	c1
2	b1	b1	c1
3	b2	b2	c1
4	b3	b3	c1

- If there is no join condition what is the result?

Explicit Join

- Q1: Retrieve first name, course id and corresponding grades of the student with Student Id = 007007

```
SELECT FName, CourId, Grade
FROM (STUDENT NATURAL JOIN GRADES )
WHERE StudId = 007007 ;
```

- The FROM clause contains a single joined table

Types of Join

- Inner Joins:
  - JOIN, INNER JOIN, EQUI-JOIN, NATURAL JOIN,
- Outer Joins:
  - LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN
  - The keyword OUTER may be omitted
  - CROSS JOIN is used to specify the CARTESIAN PRODUCT operation and should be used only with the utmost care

JOIN

- Using `JOIN` operator

```
SELECT * FROM R1 JOIN R2 ON R1.B = R2.B;
```

R1

A	B
a_1	ω
a_2	b_1

R2

B	C
b_1	c_1
b_2	c_1
b_3	c_1

R1 JOIN R2

A	B	B	C
a_2	b_1	b_1	c_1

- Using `NATURAL JOIN` operator

```
SELECT * FROM R1 NATURAL JOIN R2;
```

A	B	C
a_2	b_1	c_1

LEFT OUTER JOIN

- Every tuple in left table must appear in result
- If no matching tuple
  - Padded with NULL values for attributes of right table

```
SELECT * FROM R1 LEFT JOIN R2 ON R1.B = R2.B;
```

R1

A	B
a_1	ω
a_2	b_1

R2

B	C
b_1	c_1
b_2	c_1
b_3	c_1

R1 LEFT JOIN R2

A	B	B	C
a_1	ω	ω	ω
a_2	b_1	b_1	c_1

RIGHT OUTER JOIN

- Every tuple in right table must appear in result
- If no matching tuple
  - Padded with NULL values for attributes of right table

```
SELECT * FROM R1 RIGHT JOIN R2 ON R1.B = R2.B;
```

R1

A	B
a_1	ω
a_2	b_1

R2

B	C
b_1	c_1
b_2	c_1
b_3	c_1

R1 RIGHT JOIN R2

A	B	B	C
a_2	b_1	b_1	c_1
ω	ω	b_2	c_1
ω	ω	b_3	c_1

FULL OUTER JOIN

- All tuples from both relations must appear in result

```
SELECT * FROM R1 FULL JOIN R2 ON R1.B = R2.B;
```

R1

A	B
a_1	ω
a_2	b_1

R2

B	C
b_1	c_1
b_2	c_1
b_3	c_1

R1 RIGHT JOIN R2

A	B	B	C
a_1	ω	ω	ω
a_2	b_1	b_1	c_1
ω	ω	b_2	c_1
ω	ω	b_3	c_1

Nested Query

- *Retrieve first names of students that passed M214*

```
SELECT FName
FROM STUDENT s
WHERE s.StudId  IN
    (SELECT StudId  FROM GRADES
     WHERE CourId = 'M214' AND Grade IS NOT
     NULL);
```

FName
James

- A nested query defined by using IN (or =ANY) operator can be expressed as a single block query

```
SELECT FName
FROM STUDENT s, GRADES g
WHERE s.StudId = g.StudId  AND g.CourId = 'M214'
AND g.Grade IS NOT NULL;
```

EXISTS and NOT EXISTS

- *Retrieve Id's and surnames of students who passed at least one course:*

```
SELECT s.StudId, s.LName FROM STUDENT s
WHERE EXISTS
     (SELECT *  FROM GRADES
      WHERE s.StudId = StudId AND Grade IS NOT NULL);
```

- *Retrieve Id's and surnames of students who didn't pass any course:*

```
SELECT s.StudId, s.LName FROM STUDENT s
WHERE NOT EXISTS
     (SELECT *  FROM GRADES
      WHERE s.StudId = StudId AND Grade IS NOT NULL );
```

Aggregate Functions

Q2: What is the total point value of the courses in the COMP dept?

```
SELECT SUM(Points)
FROM COURSE
WHERE Dept = 'Comp';
```

sum_Points
52

Q3: How many different first names do COMP majors have?

```
SELECT COUNT(DISTINCT FName) AS NoOfNames
FROM STUDENT
WHERE Major = 'Comp';
```

NoOfNames
1

Grouping

For each student, retrieve the number of courses passed

```
SELECT StudId, COUNT(*)
FROM GRADES
WHERE Grade IS NOT NULL
GROUP BY StudId ;
```

StudId	COUNT(*)
007007	4
131313	1
555555	1

HAVING Clause

Retrieve the number of courses passed for students that passed at least two courses

```
SELECT StudId, COUNT(*)
FROM STUDENT s NATURAL JOIN GRADES g
WHERE Grades IS NOT NULL
GROUP BY s.StudId
HAVING COUNT(*) > 1;
```

StudId	COUNT(*)
007007	4

SQL Views

- Create a view

```
CREATE VIEW StudOccupied AS
    SELECT g.StudId, SUM(Hours ) AS Occupied
    FROM Grades g, Course p
    WHERE g.CourId = p.CourId AND Grade IS NULL
    GROUP BY StudId ;
```

- Deleting a view

```
DROP VIEW StudOccupied;
```