IS5102
Database Management Systems
Lecture 10: Intermediate SQL

Alexander Konovalov
alexander.konovalov@st-andrews.ac.uk

(with thanks to Susmit Sarkar)

2021

This week: More SQL

- Modifying Data
- Orderings and Aggregates
- Set Operations
- Integrity Constraints
- Views

Set Operations

- Set operations `UNION`, `INTERSECT`, and `EXCEPT`
  - Each of the above operations automatically eliminates duplicates
- To retain all duplicates use the corresponding multi-set versions

  `UNION ALL`,
  `INTERSECT ALL`
  `EXCEPT ALL`

- Suppose a tuple occurs m times in `r` and n times in `s`, then, it occurs:
  - $m + n$ times in `r UNION ALL s`
  - $\min(m, n)$ times in `r INTERSECT ALL s`
  - $\max(0, m - n)$ times in `r EXCEPT ALL s`

Remark: SQLite only supports `UNION ALL`, but not the other two multi-set versions

- Find courses that ran in Semester 2 of 2019 **or** in Semester 1 of 2020

```
SELECT course_id
  FROM course_runs
 WHERE semester = 2
   AND year = 2019
 UNION
SELECT course_id
  FROM course_runs
 WHERE semester = 1
   AND year = 2020;
```

- Find courses that ran in Semester 2 of 2019 **and** in Semester 2 of 2020

```
SELECT course_id
  FROM course_runs
 WHERE semester = 2
   AND year = 2019
INTERSECT
SELECT course_id
  FROM course_runs
 WHERE semester = 2
   AND year = 2020;
```

- Find courses that ran in Semester 1 of 2019 **but not** in Semester 1 of 2020

```
SELECT course_id
  FROM course_runs
 WHERE semester = 1
   AND year = 2019
EXCEPT
SELECT course_id
  FROM course_runs
 WHERE semester = 1
   AND year = 2020;
```

We can allow a **default value** to be specified

Example:

```
CREATE TABLE student (
   stud_id   CHAR(5),
   name      VARCHAR(20) NOT NULL,
   dept_id   VARCHAR(20),
   tot_cred  NUMERIC(3,0) DEFAULT 0,
   PRIMARY KEY (stud_id),
   FOREIGN KEY (dept_id) REFERENCES department);
```

- The default value of `tot_cred` is set to 0
- When a tuple is inserted if no value is provided its value is set to 0

Ensuring that a value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another relation.

- Example: If `MATH` is a department ID appearing in one of the tuples in the `instructor` relation, then `MATH` also appears in some tuple in the `department` relation.

Formal Definition:
Let `A` be a set of attributes. Let `R` and `S` be two relations that contain attributes `A` and where `A` is the primary key of `S`. `A` is said to be a foreign key of `R` if for any values of `A` appearing in `R` these values also appear in `S`

Cascading Actions in Referential Integrity

```
CREATE TABLE course (
  course_id  VARCHAR(8),
  title      VARCHAR(50),
  dept_id    VARCHAR(20),
  credits    NUMERIC(2,0),
  PRIMARY KEY (course_id),
  FOREIGN KEY (dept_id) references department


 );
```

Cascading Actions in Referential Integrity

```
CREATE TABLE course (
  course_id  VARCHAR(8),
  title      VARCHAR(50),
  dept_id    VARCHAR(20),
  credits    NUMERIC(2,0),
  PRIMARY KEY (course_id),
  FOREIGN KEY (dept_id) references department
  ON DELETE CASCADE
  ON UPDATE CASCADE
 );
```

Alternative actions to cascade: `SET NULL`, `SET DEFAULT`

Demo: see `cascading.sql`

- In some cases, it is not desirable for all users to see the entire logical model (that is, all the actual relations stored in the database.)

- Consider a person who needs to know an instructors name and department, but not the salary. This person should see a relation described, in SQL, by

```
SELECT instr_id, instr_name, dept_id
  FROM instructor;
```

Views

- A view provides a mechanism to hide certain data from the view of certain users.
- Any relation that is not of the conceptual model but is made visible to a user as a "virtual relation" is called a **view**.
- A view is defined using the `CREATE VIEW` statement which has the form

  `CREATE VIEW v AS < query expression >`

  where `<query expression>` is any legal SQL expression. The view name is represented by `v`.

View Definition

- Once a view is defined, the view name can be used to refer to the virtual relation that the view generates.
- View definition is not the same as creating a new relation by evaluating the query expression.
- Rather, a view definition causes the saving of an expression; the expression is substituted into queries using the view.

View Examples

A view of instructors without their salary

```
CREATE VIEW faculty AS
SELECT instr_id, instr_name, dept_id
  FROM instructor;
```

Create a view of department salary totals

```
CREATE VIEW departments_total_salary(dept_code, total_salary) AS
SELECT dept_id, SUM (salary)
  FROM instructor
 GROUP BY dept_id;
```

Views Defined Using Other Views

```
CREATE VIEW acad_year_2020 AS
SELECT semester, course_id, dept_id, title
  FROM course_runs NATURAL JOIN course
 WHERE year  = 2020;

CREATE VIEW cs_acad_year_2020 AS
SELECT semester, course_id, title
  FROM acad_year_2020
 WHERE dept_id= 'CS';
```

Add a new tuple to `faculty` view which we defined earlier

```
INSERT INTO faculty VALUES ('30765', 'James', 'CHEM');
```

This insertion must be represented by the insertion of the tuple

```
('30765', 'James', 'CHEM', NULL)
```

into the `instructor` relation.

WARNING: this feature is not supported in SQLite:

https://www.sqlite.org/omitted.html

Some updates cannot be translated uniquely:

```
CREATE VIEW instructor_info AS
SELECT instr_id, instr_name, building
  FROM instructor, department
 WHERE instructor.dept_id = department.dept_id;

INSERT INTO instructor_info
VALUES ('69987', 'Raul', 'Bute');
```

Which department, if there are multiple departments in Bute?

What if no department is in Bute?

View Update restrictions

Most SQL implementations allow updates only on simple views

- The `FROM` clause has only one database relation.
- The `SELECT` clause contains only attribute names of the relation, and does not have any expressions, aggregates, or distinct specification.
- Any attribute not listed in the `SELECT` clause can be set to `NULL`
- The query does not have a `GROUP BY` or `HAVING` clause.

- When defining a view, simply create a physical table representing the view at the time of creation.
- Can be a cheaper option.
- How are updates handled to the "base" relations on which the view was defined?

Advantages	Disadvantages
Data independence	Update restriction
Improved security	Performance
Reduced complexity	
Convenience	
Customisation	
Data Integrity	

- **[DBSC]** Chapters 4-5, Database System Concepts, Silberschatz, Korth and Sudarshan
- **[DBS]** Chapter 7, Database Systems, Connolly and Begg
- **[DBD]** Chapters 15-16, Database Design, Watt and Eng
- Useful sites
  - http://www.w3schools.com/sql/
  - http://sqlzoo.net/wiki/Main_Page