Carnegie Mellon Univ.
Dept. of Computer Science
15-415/615 - DB Applications

C. Faloutsos – A. Pavlo

Lecture#7: Fun with SQL (Part 2)

Administrivia

- HW2 is due next Monday.

```
mysql> SELECT due_date FROM homeworks WHERE assignment = 'HW2';
+---------------------+
| due_date            |
+---------------------+
| 2015-09-28 15:00:00 |
+---------------------+
1 row in set (0.00 sec)

mysql>
```


Last Class

- SELECT/INSERT/UPDATE/DELETE
- Table Definition (DDL)
- NULLs
- String/Date/Time/Set/Bag Operations
- Output Redirection/Control
- Aggregates/Group By

Today's Jam

- Complex Joins
- Views
- Nested Queries
- Common Table Expressions
- Triggers
- Database Application Example

Example Database

STUDENT

sid	name	login	age	gpa
53666	Trump	trump@cs	45	4.0
53688	Bieber	jbieber@cs	21	3.9
53655	Tupac	shakur@cs	26	3.5

ENROLLED

sid	cid	grade
53666	Pilates101	C
53688	Reggae203	D
53688	Topology112	A
53666	Massage105	D

COURSE

cid	name
Pilates101	Pilates
Reggae203	20th Century Reggae
Topology112	Topology + Squirrels
Massage105	Massage & Holistic Therapy

Join Query Grammar

```
SELECT ...
  FROM table-name1 join-type table-name2
    ON qualification
[WHERE ...]
```

- **Join-Type**: The type of join to compute.
- **Qualification**: Expression that determines whether a tuple from table1 can be joined with table2. Comparison of attributes or constants using operators =, ≠, <, >, ≤, and ≥.

INNER JOIN

sid	name	login	age	gpa
53666	Trump	trump@cs	45	4.0
53688	Bieber	jbieber@cs	21	3.9
53655	Tupac	shakur@cs	26	3.5

sid	cid	grade
53666	Pilates101	C
53688	Reggae203	D
53688	Topology112	A
53666	Massage105	D

```
SELECT name, cid, grade
  FROM student, enrolled
 WHERE student.sid = enrolled.sid
```

name	cid	grade
Bieber	Reggae203	D
Bieber	Topology112	A
Trump	Massage105	D
Trump	Pilates101	C

OUTER JOIN

sid	name	login	age	gpa
53666	Trump	trump@cs	45	4.0
53688	Bieber	jbieber@cs	21	3.9
53677	Tupac	shakur@cs	26	3.5

sid	cid	grade
53666	Pilates101	C
53688	Reggae203	D
53688	Topology112	A
53666	Massage105	D

```
SELECT name, cid, grade
  FROM student LEFT OUTER JOIN enrolled
    ON student.sid = enrolled.sid
```

name	cid	grade
Bieber	Reggae203	D
Bieber	Topology112	A
Trump	Massage105	D
Trump	Pilates101	C
Tupac	NULL	NULL

OUTER JOIN

sid	name	login	age	gpa
53666	Trump	trump@cs	45	4.0
53688	Bieber	jbieber@cs	21	3.9
53677	Tupac	shakur@cs	26	3.5

sid	cid	grade
53666	Pilates101	C
53688	Reggae203	D
53688	Topology112	A
53666	Massage105	D

```
SELECT name, cid, grade
  FROM enrolled RIGHT OUTER JOIN student
    ON student.sid = enrolled.sid
```

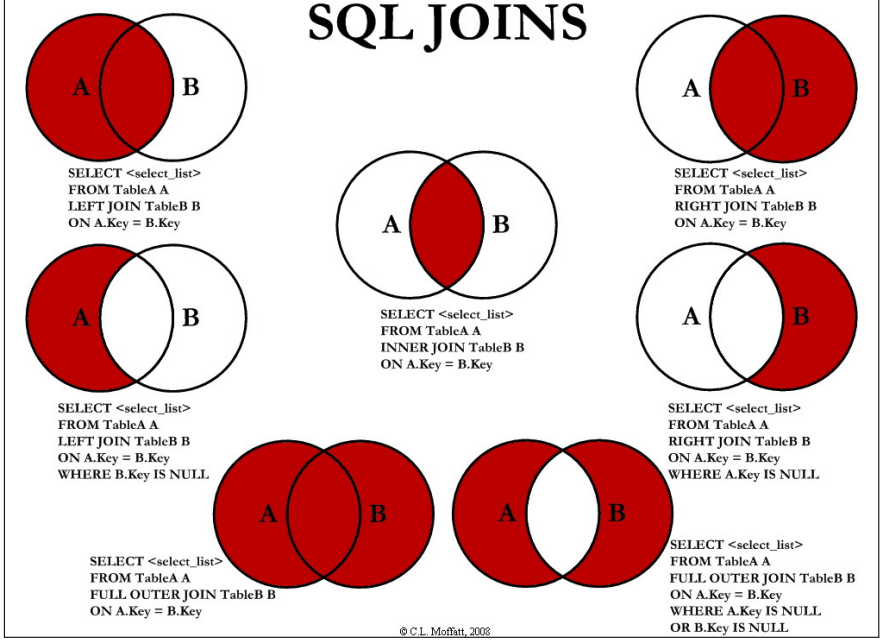
name	cid	grade
Bieber	Reggae203	D
Bieber	Topology112	A
Trump	Massage105	D
Trump	Pilates101	C
Shakur	NULL	NULL

Join Types

```
SELECT * FROM A JOIN B ON A.id = B.id
```

Join Type	Description
INNER JOIN	Join where A and B have same value
LEFT OUTER JOIN	Join where A and B have same value AND where only A has a value
RIGHT OUTER JOIN	Join where A and B have same value AND where only B has a value
FULL OUTER JOIN	Join where A and B have same value AND where A or B have unique values
CROSS JOIN	Cartesian Product

SQL JOINS



Today's Jam

- Complex Joins
- Views
- Nested Queries
- Common Table Expressions
- Triggers
- Database Application Example

Views

- Creates a "virtual" table containing the output from a **SELECT** query.
- Mechanism for hiding data from view of certain users.
- Can be used to simplify a complex query that is executed often.
  - *Won't make it faster though!*

View Example

- Create a view of the CS student records with just their id, name, and login.

```
CREATE VIEW CompSciStudentInfo AS
  SELECT sid, name, login
    FROM student
  WHERE login LIKE '%@cs';
```

Original Table

sid	name	login	age	gpa
53666	Trump	trump@cs	45	4.0
53688	Bieber	jbieber@cs	21	3.9

View

sid	name	login
53666	Trump	trump@cs
53688	Bieber	jbieber@cs

View Example

- Create a view with the average age of the students enrolled in each course.

```
CREATE VIEW CourseAge AS
  SELECT cid, AVG(age) AS avg_age
    FROM student, enrolled
   WHERE student.sid = enrolled.sid
GROUP BY enrolled.cid;
```

cid	avg_age
Massage105	45.0
Pilates101	45.0
Topology112	21.0
Reggae203	21.0

Views vs. SELECT INTO

```
CREATE VIEW AvgGPA AS
  SELECT AVG(gpa) AS avg_gpa FROM student
   WHERE login LIKE '%@cs'
```

```
SELECT AVG(gpa) AS avg_gpa INTO AvgGPA
  FROM student WHERE login LIKE '%@cs'
```

- **INTO**→Creates static table that does not get updated when student gets updated.
- **VIEW**→Dynamic results are only materialized when needed.

Materialized Views

- Creates a view containing the output from a **SELECT** query that is automatically updated when the underlying tables change.

```
CREATE MATERIALIZED VIEW AvgGPA AS
  SELECT AVG(gpa) AS avg_gpa FROM student
   WHERE login LIKE '%@cs'
```


Today's Jam

- Complex Joins
- Views
- Nested Queries
- Common Table Expressions
- Triggers
- Database Application Example

Nested Queries

- Queries containing other queries

"outer query" *"inner query"*

- Inner query:
  - Can appear in **FROM** or **WHERE** clause

```
SELECT cname FROM customer WHERE
 acctno IN (SELECT acctno FROM account)
```

Think of this as a function that returns the result of the inner query

cname
Johnson
Smith
Jones
Smith

Nested Queries

- Find the names of students in 'Massage105'

```
SELECT name FROM student
 WHERE ...
```

"sid in the set of people that take Massage105"

Nested Queries

- Find the names of students in 'Massage105'

```
SELECT name FROM student
 WHERE ...
   SELECT sid FROM enrolled
    WHERE cid = 'Massage105'
```


Nested Queries

- Find the names of students in 'Massage105'

```
SELECT name FROM student
 WHERE sid IN (
   SELECT sid FROM enrolled
    WHERE cid = 'Massage105'
 )
```

name
Trump

Nested Queries

- **ALL** →Must satisfy expression for all rows in sub-query
- **ANY** →Must satisfy expression for at least one row in sub-query.
- **IN** → Equivalent to '**=ANY()**'.
- **EXISTS** → At least one row is returned.
- *Nested queries are difficult to optimize. Try to avoid them if possible.*

Nested Queries

- Find the names of students in 'Massage105'

```
SELECT name FROM student
 WHERE sid = ANY(
   SELECT sid FROM enrolled
    WHERE cid = 'Massage105'
 )
```

name
Trump

Nested Queries

- Find student record with the highest id.
- This won't work in **SQL-92**:

```
SELECT MAX(sid), name FROM student;
```

- Runs in **MySQL**, but you get wrong answer:

sid	name
53688	Tupac

Nested Queries

- Find student record with the highest id.

```
SELECT sid, name FROM student
 WHERE ...
```

"is greater than every other sid"

Nested Queries

- Find student record with the highest id.

```
SELECT sid, name FROM student
 WHERE sid  is greater than every
   SELECT sid FROM enrolled
```


Nested Queries

- Find student record with the highest id.

```
SELECT sid, name FROM student
 WHERE sid => ALL(
   SELECT sid FROM enrolled
)
```

sid	name
53688	Bieber

Nested Queries

- Find student record with the highest id.

```
SELECT sid, name FROM student
 WHERE sid IN (
   SELECT MAX(sid) FROM enrolled
)
```

```
SELECT sid, name FROM student
 WHERE sid IN (
   SELECT sid FROM enrolled
    ORDER BY sid DESC LIMIT 1
)
```


Nested Queries

- Find all courses that nobody is enrolled in.

```
SELECT * FROM course
 WHERE ...
```

"with no tuples in the 'enrolled' table"

cid	name
Pilates101	Pilates
Reggae203	20th Century Reggae
Karate101	Karate Kid Aerobics
Topology112	Topology + Squirrels
Massage105	Massage & Holistic Therapy

sid	cid	grade
53666	Pilates101	C
53688	Reggae203	D
53688	Topology112	A
53666	Massage105	D

Nested Queries

- Find all courses that nobody is enrolled in.

```
SELECT * FROM course
 WHERE NOT EXISTS(
   tuples in the 'enrolled' table
)
```


Nested Queries

- Find all courses that nobody is enrolled in.

```
SELECT * FROM course
 WHERE NOT EXISTS(
   SELECT * FROM enrolled
    WHERE course.cid = enrolled.cid
)
```

cid	name
Karate101	Karate Kid Aerobics

Today's Jam

- Complex Joins
- Views
- Nested Queries
- Common Table Expressions
- Window Functions
- Triggers
- Database Application Example

Common Table Expressions

- Provides a way to write auxiliary statements for use in a larger query.
- Alternative to nested queries and views.

```
WITH cteName AS (
   SELECT 1
)
SELECT * FROM cteName
```


Common Table Expressions

- Find student record with the highest id that is enrolled in at least one course.

```
WITH cteSource (maxId) AS (
   SELECT MAX(sid) FROM enrolled
)
SELECT name FROM student, cteSource
 WHERE student.sid = cteSource.maxId
```


CTEs – Recursion

- Print 1 to 10.

```
WITH RECURSIVE cteSource (counter) AS (
   (SELECT 1)
   UNION ALL
   (SELECT counter + 1 FROM cteSource
     WHERE counter < 10)
)
SELECT * FROM cteSource
```

- ***Postgres CTE Demo!***

Today's Jam

- Complex Joins
- Views
- Nested Queries
- Common Table Expressions
- Triggers
- Database Application Example

Database Triggers

- Procedural code that is automatically executed in response to certain events on a particular table or view in a database.
- **BEFORE**/**AFTER**
  - **INSERT**
  - **UPDATE**
  - **DELETE**

Trigger Example

- Set a timestamp field whenever a row in the enrolled table is updated.
- First we need to add our timestamp field.

```
ALTER TABLE enrolled
  ADD COLUMN updated TIMESTAMP;
```


Trigger Example

- Register a function that sets the 'updated' column with the current timestamp.

```
CREATE OR REPLACE FUNCTION update_col()
  RETURNS TRIGGER AS $$
    BEGIN
       NEW.updated = NOW();
       RETURN NEW;
    END;
$$ language 'plpgsql';
```

Postgres

Trigger Example

- Invoke the *update_col* function when a row in the enrolled table is updated.

```
CREATE TRIGGER update_enrolled_modtime
    AFTER UPDATE ON enrolled
    FOR EACH ROW
      EXECUTE PROCEDURE update_col();
```

Postgres

MySQL Alternative

- Non-standard way to do this just for setting timestamps.

```
CREATE TABLE enrolled (
 :
  updated TIMESTAMP
                DEFAULT CURRENT_TIMESTAMP
              ON UPDATE CURRENT_TIMESTAMP
);
```


Today's Party

- DDLs
- Complex Joins
- Views
- Nested Subqueries
- Triggers
- Database Application Example

Outline of an DB application

- Establish connection with DB server
- Authenticate (user/password)
- Execute SQL statement(s)
- Process results
- Close connection

Database Connection Libraries

- DBMS-specific Libraries
- "Universal" Libraries
  - Open Database Connectivity (**ODBC**)
  - Java Database Connectivity (**JDBC**)
- Application framework libraries

ORM Libraries

- Object-Relational Mapping
- Automatically convert classes into database-backed objects.
- Method calls on objects are automatically converted into SQL queries.
- Removes the tediousness of writing SQL queries directly in application code.

ORM Example

```
class Location(models.Model):
    zipcode = CharField(max_length=5,primary_key=True)
    state = USStateField()
    city = CharField(max_length=64)

class Company(models.Model):
    name = CharField(max_length=64,unique=True)
    address1 = CharField(max_length=128)
    location = ForeignKey(Location)
    website = URLField()
    public = BooleanField(default=True)
```


ORM Example

```
CREATE TABLE location (
  zipcode VARCHAR(5) NOT NULL,
  state CHAR(2) NOT NULL,
  city VARCHAR(64) NOT NULL,
  PRIMARY KEY (zipcode),
);
CREATE TABLE company (
  id INT(11) NOT NULL AUTO_INCREMENT,
  name VARCHAR(64) NOT NULL,
  address1 VARCHAR(128) NOT NULL,
  location_id VARCHAR(5) NOT NULL \
    REFERENCES location (zipcode),
  website VARCHAR(200) NOT NULL,
  public TINYINT(1) NOT NULL,
  PRIMARY KEY (id),
);
```

ORM Libraries

- Standalone:
  - Hibernate (Java)
  - SQLAlchemy (Python)
  - Doctrine (PHP)
- Integrated:
  - Django (Python)
  - ActiveRecord (Ruby on Rails)
  - CakePHP (PHP)

Next Class

- We begin discussing storage internals.
- This material will be important for helping you pick up dates at parties.