SQL: Part I

CPS 116

Introduction to Database Systems

Announcements (Thu. Sep. 11)

- Homework #1 due next Tuesday
  - Do we need a help session tomorrow or Monday?
    - Tomorrow (Sep. 12): 3-4pm?
    - Monday (Sep. 15): 4:15-5:15pm?
    - Will email the announcement
- Talk next Monday (Sep. 15), 4-5pm, North 130A
  - http://www.cs.duke.edu/events/?id=00000000938
  - *Flexible Recommendations in CourseRank*
    - Hector Garcia-Molina (Stanford)
      - One of the book authors!
  - Highly recommended!

SQL

- SQL: Structured Query Language
  - Pronounced "S-Q-L" or "sequel"
  - The standard query language supported by most commercial DBMS
- A brief history
  - IBM System R
  - ANSI SQL89
  - ANSI SQL92 (SQL2)
  - ANSI SQL99 (SQL3)
  - ANSI SQL 2003 (added OLAP, XML, etc.)
  - ANSI SQL 2006 (added more XML)

Creating and dropping tables

- CREATE TABLE *table_name*
  (..., *column_name*, *column_type*, ...);
- DROP TABLE *table_name*;
- Examples

```
create table Student (SID integer,
                      name varchar(30), email varchar(30),
                      age integer, GPA float);
create table Course (CID char(10), title varchar(100));
create table Enroll (SID integer, CID char(10));
drop table Student;
drop table Course;
drop table Enroll;
-- everything from -- to the end of the line is ignored.
-- SQL is insensitive to white space.
-- SQL is insensitive to case (e.g., ...Course... is equivalent to
-- ...COURSE...)
```

Basic queries: SFW statement

- SELECT A_1, A_2, ..., A_n
  FROM R_1, R_2, ..., R_m
  WHERE *condition*;
- Also called an SPJ (select-project-join) query
- Equivalent (not really!) to relational algebra query
  $\pi_{A_1, A_2, \ldots, A_n} (\sigma_{condition} (R_1 \times R_2 \times \ldots \times R_m))$

Example: reading a table

- SELECT * FROM Student;
  - Single-table query, so no cross product here
  - WHERE clause is optional
  - * is a short hand for "all columns"

Example: selection and projection

- Name of students under 18
  - `SELECT name FROM Student WHERE age < 18;`
- When was Lisa born?
  - ```
    SELECT 2008 – age
    FROM Student
    WHERE name = 'Lisa';
    ```
  - SELECT list can contain expressions
    - Can also use built-in functions such as SUBSTR, ABS, etc.
  - String literals (case sensitive) are enclosed in single quotes

Example: join

- SID's and names of students taking courses with the word "Database" in their titles
  - ```
    SELECT Student.SID, Student.name
    FROM Student, Enroll, Course
    WHERE Student.SID = Enroll.SID
    AND Enroll.CID = Course.CID
    AND title LIKE '%Database%';
    ```
  - LIKE matches a string against a pattern
    - % matches any sequence of 0 or more characters
  - Okay to omit *table_name* in *table_name*.*column_name* if *column_name* is unique

Example: rename

- SID's of all pairs of classmates
  - Relational algebra query:

    $\pi_{e1.SID, e2.SID}$
    $(\rho_{e1}\ Enroll \bowtie_{e1.CID = e2.CID \wedge e1.SID > e2.SID} \rho_{e2}\ Enroll)$
  - SQL:
    ```
    SELECT e1.SID AS SID1, e2.SID AS SID2
    FROM Enroll AS e1, Enroll AS e2
    WHERE e1.CID = e2.CID
    AND e1.SID > e2.SID;
    ```
  - AS keyword is completely optional

A more complicated example

- Titles of all courses that Bart and Lisa are taking together

```
SELECT c.title
FROM Student sb, Student sl, Enroll eb, Enroll el, Course c
WHERE sb.name = 'Bart' AND sl.name = 'Lisa'
AND eb.SID = sb.SID AND el.SID = sl.SID
AND eb.CID = c.CID AND el.CID = c.CID;
```

Tip: Write the FROM clause first, then WHERE, and then SELECT

Why SFW statements?

- Out of many possible ways of structuring SQL statements, why did the designers choose SELECT-FROM-WHERE?
  - A large number of queries can be written using only selection, projection, and cross product (or join)
  - Any query that uses only these operators can be written in a canonical form: $\pi_L (\sigma_p (R_1 \times \ldots \times R_m))$
    - Example: $\pi_{R.A, S.B} (R \bowtie_{p1} S) \bowtie_{p2} (\pi_{T.C} \sigma_{p3} T) =$
      $\pi_{R.A, S.B, T.C} \sigma_{p1 \wedge p2 \wedge p3} (R \times S \times T)$
  - SELECT-FROM-WHERE captures this canonical form

Set versus bag semantics

- Set
  - No duplicates
  - Relational model and algebra use set semantics
- Bag
  - Duplicates allowed
  - Number of duplicates is significant
  - SQL uses bag semantics by default

Set versus bag example

Enroll

SID	CID
142	CPS116
142	CPS114
123	CPS116
857	CPS116
857	CPS190
456	CPS114
…	…

π_{SID} *Enroll*

SID
142
123
857
456
…

```
SELECT SID
FROM Enroll;
```

SID
142
142
123
857
857
456
…

A case for bag semantics

- Efficiency
  - Saves time of eliminating duplicates
- Which one is more useful?
  - π_{GPA} *Student*
  - `SELECT GPA FROM Student;`
  - The first query just returns all possible GPA's
  - The second query returns the actual GPA distribution
- Besides, SQL provides the option of set semantics with `DISTINCT` keyword

Forcing set semantics

- SID's of all pairs of classmates
  - ```
    SELECT e1.SID AS SID1, e2.SID AS SID2
    FROM Enroll AS e1, Enroll AS e2
    WHERE e1.CID = e2.CID
    AND e1.SID > e2.SID;
    ```
    - Say Bart and Lisa both take CPS116 and CPS114
  - `SELECT DISTINCT e1.SID AS SID1, e2.SID AS SID2`
    `…`
    - With `DISTINCT`, all duplicate (SID1, SID2) pairs are removed from the output

Operational semantics of SFW

- `SELECT [DISTINCT]` $E_1, E_2, \ldots, E_n$
  `FROM` $R_1, R_2, \ldots, R_m$
  `WHERE` *condition*;
- For each t_1 in R_1:
  For each t_2 in R_2: … …
  For each t_m in R_m:
  If *condition* is true over $t_1, t_2, \ldots, t_m$:
  Compute and output $E_1, E_2, \ldots, E_n$ as a row
  If `DISTINCT` is present
  Eliminate duplicate rows in output
- $t_1, t_2, \ldots, t_m$ are often called tuple variables

SQL set and bag operations

- `UNION, EXCEPT, INTERSECT`
  - Set semantics
    - Duplicates in input tables, if any, are first eliminated
  - Exactly like set ∪, −, and ∩ in relational algebra
- `UNION ALL, EXCEPT ALL, INTERSECT ALL`
  - Bag semantics
  - Think of each row as having an implicit count (the number of times it appears in the table)
  - Bag union: sum up the counts from two tables
  - Bag difference: proper-subtract the two counts
  - Bag intersection: take the minimum of the two counts

Examples of bag operations

Bag1

fruit
apple
apple
orange

Bag2

fruit
apple
orange
orange

Bag1 UNION ALL Bag2

fruit
apple
apple
orange
apple
orange
orange

Bag1 EXCEPT ALL Bag2

fruit
apple

Bag1 INTERSECT ALL Bag2

fruit
apple
orange
    ```