

SQL: Part I

CPS 116
Introduction to Database Systems

Announcements (September 13)

- ❖ Homework #1 due next Tuesday
 - Do we need a help session on Monday?
- ❖ Course project assigned today
 - Choice of "standard" or "open"
 - One- to three-person teams
 - Two milestones + demo/report
 - Milestone #1 due in 4 weeks, right after fall break

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 (+OLAP, XML, etc.)

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: SPJ statement

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

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 2007 - 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:
 $\pi_{1.SID, 2.SID}$
 $(\rho_{s1} Enroll \bowtie_{1.CID = 2.CID \wedge 1.SID > 2.SID} \rho_{s2} Enroll)$
 - 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: $\pi_L(\sigma_P(R_1 \times \dots \times R_m))$
 - 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

13

Enroll		$\pi_{SID} Enroll$	
SID	CID	SID	
102	CPS116	102	
102	CPS114	102	
102	CPS116	102	
207	CPS116	207	
207	CPS116	207	
207	CPS114	207	
-	-	-	

SELECT SID
FROM Enroll;

SID
102
102
102
207
207
207
-

A case for bag semantics

14

- ❖ Efficiency
 - Saves time of eliminating duplicates
- ❖ Which one is more useful?
 - $\pi_{GPA} Student$
 - 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

15

- ❖ 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

16

- ❖ SELECT {DISTINCT} E_1, E_2, \dots, E_n
FROM R_1, R_2, \dots, 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, \dots, t_m :
 Compute and output E_1, E_2, \dots, E_n as a row
IF DISTINCT is present
 Eliminate duplicate rows in output
- ❖ t_1, t_2, \dots, t_m are often called tuple variables

SQL set and bag operations

17

- ❖ UNION, EXCEPT, INTERSECT
 - Set semantics
 - Duplicates in input tables, if any, are first eliminated
 - Exactly like set \cup , $-$, and \cap 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

18

Bag1	Bag2														
<table border="1"><tr><th>fruit</th></tr><tr><td>apple</td></tr><tr><td>apple</td></tr><tr><td>orange</td></tr></table>	fruit	apple	apple	orange	<table border="1"><tr><th>fruit</th></tr><tr><td>apple</td></tr><tr><td>orange</td></tr><tr><td>orange</td></tr></table>	fruit	apple	orange	orange						
fruit															
apple															
apple															
orange															
fruit															
apple															
orange															
orange															
Bag1 UNION ALL Bag2		Bag1 INTERSECT ALL Bag2													
<table border="1"><tr><th>fruit</th></tr><tr><td>apple</td></tr><tr><td>apple</td></tr><tr><td>orange</td></tr><tr><td>apple</td></tr><tr><td>orange</td></tr><tr><td>orange</td></tr></table>	fruit	apple	apple	orange	apple	orange	orange	<table border="1"><tr><th>fruit</th></tr><tr><td>apple</td></tr></table>	fruit	apple	<table border="1"><tr><th>fruit</th></tr><tr><td>apple</td></tr><tr><td>orange</td></tr></table>	fruit	apple	orange	
fruit															
apple															
apple															
orange															
apple															
orange															
orange															
fruit															
apple															
fruit															
apple															
orange															
Bag1 EXCEPT ALL Bag2															
<table border="1"><tr><th>fruit</th></tr><tr><td>apple</td></tr><tr><td>orange</td></tr><tr><td>apple</td></tr><tr><td>orange</td></tr><tr><td>orange</td></tr></table>	fruit	apple	orange	apple	orange	orange	<table border="1"><tr><th>fruit</th></tr><tr><td>apple</td></tr></table>	fruit	apple						
fruit															
apple															
orange															
apple															
orange															
orange															
fruit															
apple															