SQL

CPS 216
Advanced Database Systems

SQL

- SQL: Structured Query Language
  - Pronounced “S-Q-L” or “sequel”
  - The query language of every commercial DBMS
- A brief history
  - System R
  - SQL89
  - SQL92 (SQL2)
  - SQL3 (still under construction)

Table creation

- CREATE TABLE *table_name* (…, $column\_name_i$ $column\_type_i$, …);
- Example
  - 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));

SQL is case insensitive

SFW queries

- 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 (more or less) to relational algebra query

4

Example: reading a table

- SELECT * FROM Student;
  - Single-table query; no cross product
  - WHERE clause is optional
  - "*" is a shorthand for "all columns"

5

Example: selection and projection

- Names of students under 18
- When was Lisa born?
  -
    - SELECT list can contain calculations
    - String literals are enclosed in single quotes (case sensitive)

6

Example: join

- SIDs and names of students taking courses with the word “Database” in their titles
  –

➢Many, many more built-in predicates such as LIKE

➢Okay to omit the *table_name* in *table_name.column_name* if *column_name* is unique

7

Example: rename

- SIDs of all pairs of classmates
  –

– “AS” is optional; in fact Oracle doesn’t like it in the FROM clause

8

Set versus bag semantics

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

9

Set versus bag example

SID	CID
142	CPS 216
142	CPS 214
123	CPS 216
857	CPS 216
857	CPS 130
456	CPS 214
...	...

π_{SID} (Enroll) SELECT SID FROM Enroll;

10

A case for bag semantics

- Efficiency
- Which one is more useful?

  π_{GPA} (Student)

  SELECT GPA FROM Student;
- Besides, SQL provides the option of set semantics with DISTINCT

11

Example: forcing set semantics

- SIDs 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;
  - SELECT DISTINCT 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;

12

Operational semantics of SFW

- SELECT [DISTINCT] E_1, E_2, …, E_n
  FROM R_1, R_2, …, 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, …, t_m:
  Compute and output E_1, E_2, …, E_n
  If DISTINCT is present
  Eliminate duplicates in output

13

Set and bag operations

- UNION, EXCEPT, INTERSECT
  - Set semantics
  - Exactly like set ∪, −, ∩ in relational algebra
- UNION ALL, EXCEPT ALL, INTERSECT ALL
  - Bag semantics
  - Bag union: sum the two counts (the times an element appears in the two bags)
  - Bag difference: proper-subtract the two counts
  - Bag intersection: take the minimum of the two counts

14

Examples of bag operations

R

A
apple
apple
orange

S

A
apple
orange
orange

R UNION ALL *S*

R EXCEPT ALL *S*

R INTERSECT ALL *S*

15

Example of set versus bag operations

Enroll(SID, CID), ClubMember(club, SID)

– (SELECT SID FROM ClubMember)
EXCEPT
(SELECT SID FROM Enroll)

– (SELECT SID FROM ClubMember)
EXCEPT ALL
(SELECT SID FROM Enroll)

16

Table expressions

- Use query result as a table
  - In set and bag operations, FROM clauses, etc.
  - A way to "nest" queries
- Example: names of students who are in more clubs than class

17

Scalar subqueries

- A query that returns a single row can be used as a value in WHERE, SELECT, etc.
- Example: students at the same age as Bart

- Runtime error if subquery returns more than one row

18

IN subqueries

- "IN" checks if something is in the result of the subquery
- Example: students at the same age as (any) Bart

19

EXISTS subqueries

- "EXISTS" checks if the result of a subquery is empty
- Example: students at the same age as (any) Bart
  –
  – It's a correlated subquery — a subquery that refers to values in a surrounding query

20

Operational semantics of subqueries

```
SELECT * FROM Student AS S
  WHERE EXISTS
  (SELECT * FROM Student
    WHERE name = 'Bart' AND age = S.age);
```

- For each row S in Student
  – Evaluate the subquery with the appropriate value of S.age
  – If the result of the subquery is not empty, output S.*
- The query optimizer reserves the right to process the query in any other equivalent way

21

Scoping rule of subqueries

```
SELECT * FROM Student AS S
  WHERE EXISTS
  (SELECT * FROM Student
    WHERE name = 'Bart' AND age = S.age);
```

- To find out which table a column belongs to
  - Start with the immediately surrounding query
  - If not found, look in the one surrounding that, and repeat if necessary
- Use renaming to avoid confusion

22

Quantified subqueries

- A quantified subquery can be used as a value in a comparison predicate

  … WHERE something > ANY | ALL (*subquery*)…
- ANY: existential quantifier (exists)
- ALL: universal quantifier (for all)
- Beware
  - In common parlance, "any" and "all" seem to be synonyms
  - In SQL, ANY really means SOME

23

Examples of quantified subqueries

- Which students have the highest GPA?

24

Summary

- Bag semantics
  - Richer semantics, greater efficiency, but just not “relational”
- SELECT-FROM-WHERE
  - A canonical form for queries with any nesting of selection, projection, and join
  - Most queries are in this form
- Subqueries
  - More declarative (recall the highest GPA query)
  - But no more expressive
    - Try translating other forms of subqueries into (NOT) EXISTS, which in turn can be translated into join (and difference)

25

Aggregates

- COUNT, SUM, AVG, MIN, MAX
- Example: number of students under 18, and their average GPA
  –
  - COUNT(*) counts the number of rows

26

Aggregates with DISTINCT

- Example: How many students are taking classes?

27

GROUP BY

- SELECT … FROM … WHERE … GROUP BY *list_of_columns*;
- Operational semantics
  - Compute FROM ($\times$)
  - Compute WHERE (σ)
  - Compute GROUP BY: group results according to the values of GROUP BY columns
  - Compute SELECT for each group (π)

28

GROUP BY example

- Find the average GPA for each age group

29

GROUP BY example with data

SELECT age, AVG(GPA) FROM Student GROUP BY age;

SID	*name*	*age*	*GPA*
142	Bart	10	2.3
857	Lisa	8	4.3
123	Milhouse	10	3.1
456	Ralph	8	2.3
…	…	…	…

Compute GROUP BY: group results according to the values of GROUP BY columns

Compute SELECT for each group

30

Restriction on SELECT

- If any aggregate is used, then every column referenced in SELECT must be either
  - Aggregated, or
  - A GROUP BY column
- Example: Which students have the highest GPA?

SID	*name*	*age*	*GPA*
142	Bart	10	2.3
857	Lisa	8	4.3
123	Milhouse	10	3.1
456	Ralph	8	2.3
...	...	...	...

31

HAVING

- SELECT… FROM… WHERE… GROUP BY… HAVING *condition*;
- Operational semantics
  - Compute FROM ($\times$)
  - Compute WHERE (σ)
  - Compute GROUP BY: group results according to the values of GROUP BY columns
  - Compute HAVING (another σ over the groups)
  - Compute SELECT for each group (π)

32

HAVING examples

- Find the average GPA for each age group over 10
  –
- List the average GPA for each age group with more than a hundred students
  –

33

Next time

- NULLs
- Outerjoins
- Updates
- Constraints
- Triggers

34