Introduction to SQL and the Relational Model

Data Boot Camp!

May 22, 2013

Michael Cafarella

Relational Databases

- The most common kind is a *relational database*
- The software is called a Relational Database Management System (RDBMS)
  - Oracle, IBM's DB2, Microsoft's SQLServer, MySQL, SQLite, etc
- Your dataset is "a database", managed by an RDBMS

AID	Name	Country	Sport
1	Mary Lou Retton	USA	Gymnastics
2	Jackie Joyner-Kersee	USA	Track
3	Michael Phelps	USA	Swimming

Relational Databases

- A relational database is a set of "relations" (aka tables)
- Each relation has two parts:
  - Instance (a table, with rows (aka tuples, records), and columns (aka fields, attributes))
    - # Rows = cardinality
    - # Columns = degree / arity
  - Schema
    - Relation name
    - Name and type for each column
    - E.g., Student (sid int, name varchar(128), gpa real)

Instance of Athlete Relation

AID	Name	Country	Sport
1	Mary Lou Retton	USA	Gymnastics
2	Jackie Joyner-Kersee	USA	Track
3	Michael Phelps	USA	Swimming

What is the schema? **(aid: integer, name: string, country: string, sport:string)**

Cardinality & Degree? **Cardinality = 3, Degree = 4**

Relational Query Languages

- RDBMS do lots of things, but mainly:
  - Keeps data safe
  - Gives you a powerful query language
- Queries written *declaratively*
  - In contrast to *procedural* methods
- RDBMS is responsible for efficient evaluation
  - System can optimize for efficient query execution, and still ensure that the answer does not change
- Most popular query language is SQL

Creating Relations in SQL

- Create the Athlete relation
  - Type constraint enforced when tuples added or modified

```
CREATE TABLE Athlete
(aid INTEGER,
 name CHAR(30),
 country CHAR(20),
 sport CHAR(20))
```

- Create the Olympics relation

```
CREATE TABLE Olympics
(oid INTEGER,
 year INTEGER,
 city CHAR(20))
```

- Create the Compete relation

```
CREATE TABLE Compete
(aid INTEGER,
 oid INTEGER)
```

The SQL Query Language

- Find all athletes from USA:

```
SELECT *
FROM Athlete A
WHERE A.country = 'USA'
```

AID	Name	Country	Sport
1	Mary Lou Retton	USA	Gymnastics
2	Jackie Joyner-Kersee	USA	Track
3	Michael Phelps	USA	Swimming

- Print only the names and sports:

```
SELECT A.name, A.sport
FROM Athlete A
WHERE A.country = 'USA'
```

Name	Sport
Mary Lou Retton	Gymnastics
Jackie Joyner-Kersee	Track
Michael Phelps	Swimming

Querying Multiple Relations

- What does the following query compute?

```
SELECT O.year
FROM Athletes A, Olympics O, Compete C
WHERE A.aid = C.aid AND O.oid = C.oid
        AND A.name = 'Michael Phelps'
```

Find the years when Michael Phelps competed in the Olympics

Adding & Deleting Tuples

- Can insert a single tuple using:

```
INSERT INTO  Athlete (aid, name, country, sport)
VALUES  (4, 'Johann Koss', 'Norway', 'Speedskating')
```

- Can delete all tuples satisfying some condition (e.g., name = Smith):

```
DELETE
FROM Athlete A
WHERE A.name = 'Smith'
```

Destroying & Altering Relations

DROP TABLE Olympics

Destroys the relation Olympics.

(Schema information and tuples are deleted)

Basic SQL Query

Attributes from input relations

Optional

List of relations

SELECT [DISTINCT] attr-list

FROM relation-list

WHERE qualification

Attr1 **op** Attr2

OPS: <, >, =, <=, >=, <>

Combine using AND, OR, NOT

(Conceptual) Evaluation:

1. Take cross-product of relation-list
2. Select rows satisfying qualification
3. Project columns in attr-list
   (eliminate duplicates only if DISTINCT)

Example of Basic Query

- Schema:
  - Sailors (<u>sid</u>, sname, rating, age)
  - Boats (<u>bid</u>, bname, color)
  - Reserves (<u>sid, bid, day</u>)
- Find the names of sailors who have reserved boat #103

```
SELECT S.sname
FROM Sailors S, Reserves R
WHERE S.sid = R.sid AND R.bid = 103
```

Example of Basic Query

Reserves

sid	bid	day
22	101	10/10
58	103	11/12

Sailors

sid	sname	rating	age
22	dustin	7	45
58	rusty	10	35
31	lubber	8	55

Reserves x Sailors

sid	bid	day	sid	sname	rating	age
22	101	10/10	22	dustin	7	45
22	101	10/10	58	rusty	10	35
22	101	10/10	31	lubber	8	55
58	103	11/12	22	dustin	7	45
58	103	11/12	58	rusty	10	35
58	103	11/12	31	lubber	8	55

Example of Basic Query

```
SELECT DISTINCT sname
FROM Sailors S, Reserves R
WHERE S.sid = R.sid AND R.bid = 103
```

What's the effect of adding DISTINCT?

Another Example

- Schema:
  - Sailors (<u>sid</u>, sname, rating, age)
  - Boats (<u>bid</u>, bname, color)
  - Reserves (<u>sid, bid, day</u>)
- Find the colors of boats reserved by a sailor named rusty

```
SELECT B.color
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND
  S.sname = 'rusty'
```

Note on Range Variables

- Needed when same relation appears twice in FROM clause

```
SELECT S1.sname, S2.sname
FROM Sailors S1, Sailors S2
WHERE S1.age > S2.age
```

What does this Query compute?

Good style to always use range variables anyway…

ORDER BY clause

- Most of the time, results are unordered
- You can change this with the ORDER BY clause

Attribute(s) in ORDER BY clause must be in SELECT list.

Find the names and ages of all sailors, in increasing order of age

```
SELECT S.sname, S.age
FROM Sailors S
ORDER BY S.age [ASC]
```

Find the names and ages of all sailors, in decreasing order of age

```
SELECT S.sname, S.age
FROM Sailors S
ORDER BY S.age DESC
```

ORDER BY clause

```
SELECT S.sname, S.age, S.rating
FROM Sailors S
ORDER BY S.age ASC, S.rating DESC
```

What does this query compute?

Find the names, ages, and rankings of all sailors.

Sort the result in <u>increasing</u> order of age.

If there is a tie, sort those tuples in decreasing order of rating.

Union

- Find names of sailors who have reserved a red or a green boat

```
SELECT  S.sname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid
  AND (B.color = 'red' OR B.color = 'green'))
```

```
SELECT  S.sname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
UNION
SELECT S.sname
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid and R.bid = B.bid AND B.color = 'green'
```

Intersect

- Find sids of sailors who have reserved a red and a green boat

```
SELECT  S.sid
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid AND R.bid = B.bid AND B.color = 'red'
INTERSECT
SELECT S.sid
FROM Sailors S, Reserves R, Boats B
WHERE S.sid = R.sid and R.bid = B.bid AND B.color = 'green'
```

What if we replace sid with sname (non-key)?

More Set Operators

- Also EXCEPT (set difference)
- (Painful) SQL oddities
  - For UNION, default is to eliminate duplicates!
  - To keep duplicates, must specify UNION ALL

Aggregate Operators

COUNT (*)
COUNT ([DISTINCT] A)
SUM ([DISTINCT] A)
AVG ([DISTINCT] A)
MAX (A) *Can use Distinct*
MIN (A) *Can use Distinct*

single column

```
SELECT  COUNT (*) FROM  Sailors S
```

```
SELECT COUNT (DISTINCT S.name)
FROM Sailors S
```

```
SELECT  AVG (S.age)
FROM  Sailors S
WHERE  S.rating=10
```

```
SELECT  AVG ( DISTINCT S.age)
FROM  Sailors S
WHERE  S.rating=10
```

```
SELECT  S.sname FROM  Sailors S
WHERE  S.rating= (SELECT  MAX(S2.rating) FROM  Sailors S2)
```

GROUP BY

- Conceptual evaluation
  - Partition data into groups according to some criterion
  - Evaluate the aggregate for each group

Example: *For each rating level, find the age of the youngest sailor*

```
SELECT  MIN (S.age), S.rating
FROM  Sailors S
GROUP BY  S.rating
```

How many tuples in the result?

GROUP BY and HAVING

SELECT	[DISTINCT] *target-list*
FROM	*relation-list*
WHERE	*qualification*
GROUP BY	*grouping-list*
HAVING	*group-qualification*

Target-list contains:

- Attribute names (subset of grouping-list)
- Aggregate operations (e.g., min(age))

Conceptual Evaluation:

1. Eliminate tuples that don't satisfy qualification
2. Partition remaining data into groups
3. Eliminate groups according to group-qualification
4. Evaluate aggregate operation(s) for each group

Find the age of the youngest sailor with age >= 18, for each rating with at least 2 <u>such</u> sailors

```
SELECT  S.rating,  MIN (S.age)
FROM  Sailors S
WHERE  S.age >= 18
GROUP BY  S.rating
HAVING  COUNT (*) >= 2
```

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
71	zorba	10	16.0
64	horatio	7	35.0
29	brutus	1	33.0
58	rusty	10	35.0

rating	age
1	33.0
7	45.0
7	35.0
8	55.5
10	35.0

rating	
7	35.0

Answer relation

NULL Values in SQL

- NULL represents 'unknown' or 'inapplicable'
- Query evaluation complications
  - Q: Is (rating > 10) true when rating is NULL?
  - A: Condition evaluates to 'unknown' (not T or F)
- What about AND, OR connectives?
  - Need 3-valued logic
- WHERE clause eliminates rows that <u>don't evaluate to true</u>

p	q	p AND q	p OR q
T	**T**	T	T
T	**F**	F	T
T	**U**	U	T
F	**T**	F	T
F	**F**	F	F
F	**U**	F	U
U	**T**	U	T
U	**F**	F	U
U	**U**	U	U

NULL Values Example

What does this query return?

```
SELECT sname
FROM sailors
WHERE age > 45
        OR age <= 45
```

sailors

sid	sname	rating	age
22	dustin	7	45
58	rusty	10	NULL
31	lubber	8	55

NULL Values in Aggregates

- NULL values generally ignored when computing aggregates
  - Modulo some special cases (see textbook)

SELECT AVG(age)
FROM sailors

Returns 50!

sailors

sid	sname	rating	age
22	dustin	7	45
58	rusty	10	NULL
31	lubber	8	55

For each red boat, find the number of reservations for this boat*

```
SELECT B.bid,  COUNT (*) AS scount
FROM   Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid AND B.color= 'red'
GROUP BY  B.bid
```

```
SELECT B.bid,  COUNT (*) AS scount
FROM   Sailors S, Boats B, Reserves R
WHERE S.sid=R.sid AND R.bid=B.bid
GROUP BY  B.bid
HAVING    B.color = 'red'
```

Would this work?

note: one color per <u>bid</u>