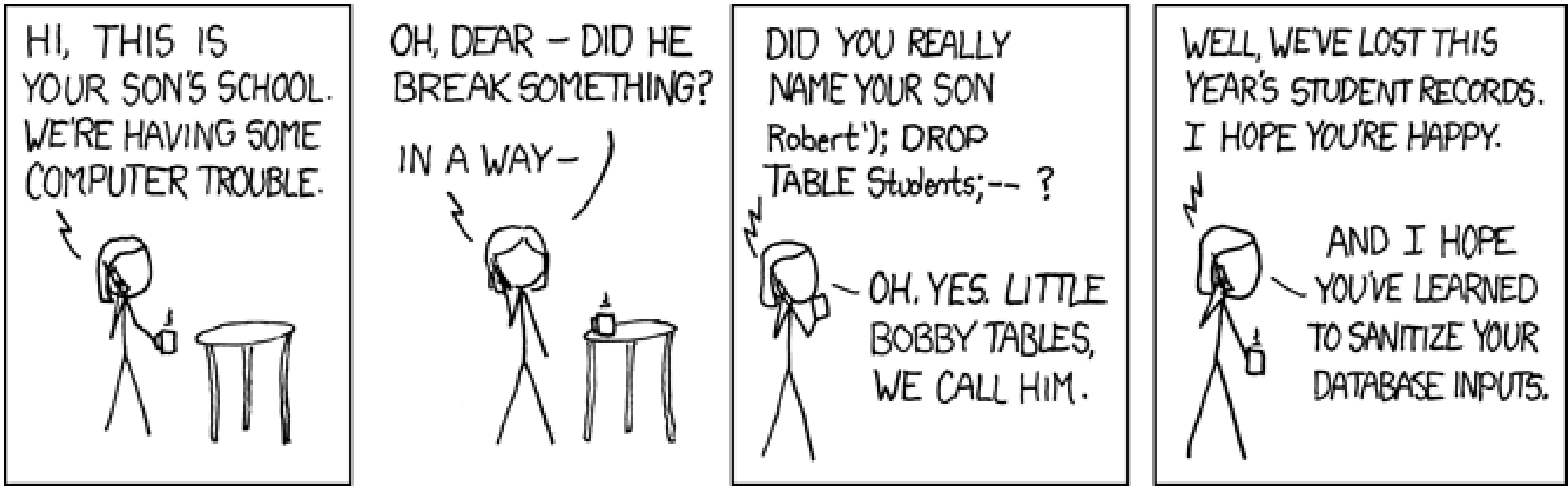
SQL: The Query Language
Part II

R&G - Chapter 5


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY –
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

SQL

- **The form:**

  SELECT $A_1, A_2, \ldots, A_n$
  FROM $r_1, r_2, \ldots, r_m$
  WHERE P

  - A_i represents an attribute
  - r_i represents a relation
  - P is a predicate

- **This query is equivalent to the relational algebra expression:**

$$\Pi_{A_1, A_2, \ldots, A_n}(\sigma_P(r_1 \times r_2 \times \ldots \times r_m))$$

Roadmap of Today’s lecture

- **Set operations**
  - Union
  - Intersect
  - Except

Union

Union: R U S

In SQL:

Subquery 1

UNION

Subquery 2

- The two subqueries must be a valid SQL query (SELECT-FROM block, with WHERE and other clauses optional)
- UNION statement must be union-compatible:
  - The two subqueries must return the same attributes in SELECT clause

UNION with Duplicate Rows

- **UNION excludes duplicate rows**

A
a1
a2
a2
a3

T1

A
a2
a3
a3

T2

```
SELECT A
FROM T1
UNION
SELECT A
FROM T2
```

The query returns

A
a1
a2
a3

UNION of Two SELECT * Clauses

A	B
a1	b1
a2	b1
a3	b2

T1

A	B
a1	b2
a2	b2

T2

```
SELECT *
FROM T1
UNION
SELECT *
FROM T2
```

Both T1 and T2 have A as the key

The query returns

A	B
a1	b1
a1	b2
a2	b1
a2	b2
a3	b2

The records of the same key but different non-key values are considered as different and add into the union result

Example of Union

- **Schema**
  - Boats (<u>bid</u>, bname, color)
  - Sailors(<u>sid</u>, sname, rating, age)
  - Reserves(<u>sid</u>, <u>bid</u>, day)

Query 1: Find ID of sailors who've reserved a red <u>or</u> a green boat

Solution 1 (without set operations)

$$\pi_{sid}(\sigma_{color='red' \vee color='green'} Boats \bowtie \text{Re}\,serves)$$

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

Note: AND is always processed before OR. Without () around `B.color='red' OR B.color='green'`, the WHERE clause will be evaluated as: `(R.bid=B.bid AND B.color='red') OR B.color='green';`

Example of Union

- **Schema**
  - Boats (bid, bname, color)
  - Sailors(sid, sname, rating, age)
  - Reserves(sid, bid, day)

Query 1: Find ID of sailors who've reserved a red or a green boat

Solution 2 (with set union operation)

```
SELECT  R.sid
FROM  Boats B NATURAL JOIN Reserves R
WHERE B.color='red'
UNION
SELECT R.sid
FROM  Boats B NATURAL JOIN Reserves R
WHERE B.color='green';
```

INTERSECT

Intersection: R ∩ S

In SQL:

Subquery 1

INTERSECT

Subquery 2

- The two subqueries must be a valid SQL query (SELECT-FROM block, with WHERE and other clauses optional)
- INTERSECT must be union-compatible:
  - The two subqueries must have the same attributes in SELECT clause

INTERSECT with Duplicate Rows

- **INTERSECT excludes duplicate rows**

A
a1
a1
a2
a3

T1

A
a2
a3
a3

T2

```
SELECT A
FROM T1
INTERSECT
SELECT A
FROM T2
```

The query returns

A
a2
a3

Example of Intersection

- **Schema**
    - Boats (<u>bid</u>, bname, color)
    - Sailors(<u>sid</u>, sname, rating, age)
    - Reserves(<u>sid</u>, <u>bid</u>, day)

Query 2: Find ID of sailors who've reserved a red <u>and</u> a green boat

Solution 1 (with set operation)

```
SELECT sid
FROM Boats B NATURAL JOIN Reserves R
WHERE B.color='red'
INTERSECT
SELECT sid
FROM Boats B NATURAL JOIN Reserves R
WHERE B.color='green';
```

Query 2: Find ID of sailors who’ve reserved a red **and** a green boat (continue)

- Can we write equivalent SQL statement WITHOUT using set operations?
- Is the following solution correct?

```
SELECT R.sid
FROM Boats B NATURAL JOIN Reserves R
WHERE B.color=‘red’ AND B.color=‘green’
```

Query 2: Find ID of sailors who’ve reserved a red **and** a green boat

Is the following solution correct?

```
SELECT R.sid
FROM Boats B NATURAL JOIN Reserves R
WHERE B.color=‘red’ AND B.color=‘green’
```

NO single boat of both red and green colors. The query will return empty answer.

Query 2: Find ID of sailors who’ve reserved a red **and** a green boat

- **Schema**
    - Boats (bid, bname, color)
    - Sailors(sid, sname, rating, age)
    - Reserves(sid, bid, day)

- Can we write equivalent SQL statement without using set operations?
    - Hint: join with Boats table TWICE (one for red boat, one for green boat)

Query 2: Find ID of sailors who've reserved a red **and** a green boat

- **Schema**
    - Boats (bid, bname, color)
    - Sailors(sid, sname, rating, age)
    - Reserves(sid, bid, day)
- Can we write equivalent SQL statement without using set operations?
    - Hint: join with Boats table TWICE (one for red boat, one for green boat)

```
SELECT R1.sid
FROM Boats B1, Boats B2, Reserves R1, Reserves R2
WHERE   B1.color='red'  //B1 only has red boats
      AND B2.color='green'; //B2 only has green boats
      AND R1.bid=B1.bid   // Natural join R1&B1 (resv. of red boats)
      AND R2.bid=B2.bid   // Natural join R2&B2 (resv. of green boats)
      AND R1.sid=R2.sid   // Reservation of same sailor
```

Query 2: Find ID of sailors who've reserved a red **and** a green boat

- **Schema**
    - Boats (bid, bname, color)
    - Sailors(sid, sname, rating, age)
    - Reserves(sid, bid, day)

- Can we write equivalent SQL statement without using set operations?
    – Question: can we use one single Reserves table in the two joins (as shown below)?

```
SELECT R.sid
FROM Boats B1, Boats B2, Reserves R
WHERE     R.bid=B1.bid
      AND R.bid=B2.bid
      AND B1.color='red'
      AND B2.color='green';
```

Query 3: Find name of sailors who've reserved at least 2 different boats

- **Schema**
  - Boats (<u>bid</u>, bname, color)
  - Sailors(<u>sid</u>, sname, rating, age)
  - Reserves(<u>sid</u>, <u>bid</u>, day)
- **Write the query without using aggregate function**
- Hint: join with Boats table TWICE (for two different reservations)

```
SELECT S1.sname
FROM Reserves R1, Reserves R2, Sailors S1, Sailors S2
WHERE R1.sid=R2.sid  // same sailor
  AND R1.bid<>R2.bid  //Two different boats
  AND R1.sid=S1.sid   //natural join R1 & S1
  AND R2.sid=S2.sid;  //natural join R2 & S2
```

EXCEPT

Set difference: R - S

In SQL:

Subquery 1
EXCEPT
Subquery 2

- The two subqueries must be a valid SQL query (SELECT-FROM block, with WHERE and other clauses optional)
- EXCEPT must be union-compatible:
  - The two subqueries must have the same attributes in SELECT clause

EXCEPT

- EXCEPT (sometimes also use MINUS)
- EXCEPT only takes the distinct rows of queries

A
a1
a1
a2
a2
a3

T1

A
a2
a3
a3

T2

```
SELECT A
FROM T1
EXCEPT
SELECT A
FROM T2
```

The query returns

A
a1

Example of EXCEPT

Query 4: find ID of sailors who've reserved a red boat **but never reserved** a green boat

```
SELECT sid
FROM Boats B NATURAL JOIN Reserves R
WHERE B.color='red'
EXCEPT
SELECT sid
FROM Boats B NATURAL JOIN Reserves R
WHERE B.color='green'
```

Query 5: Find name of sailors who never reserved a boat

- **Schema**
    - Boats (bid, bname, color)
    - Sailors(sid, sname, rating, age)
    - Reserves(sid, bid, day)

Is this solution correct?

```
SELECT sname
FROM Sailors
EXCEPT
SELECT sname
FROM Sailors NATURAL JOIN Reserves;
```

Query 5: Find name of sailors who never reserved a boat

- **Schema**
  - Boats (<u>bid</u>, bname, color)
  - Sailors(<u>sid</u>, sname, rating, age)
  - Reserves(<u>sid</u>, <u>bid</u>, day)

Is this solution correct?

```
SELECT sname
FROM Sailors
EXCEPT
SELECT sname
FROM Sailors NATURAL JOIN Reserves
```

- The solution is incorrect, as the sailors may have duplicated names, while some reserved boats, some don’t.
- How to fix?

Query 5: find name of sailors who never reserved a boat

```
CREATE TABLE Temp_Sid AS
SELECT sid
FROM Sailors
EXCEPT
SELECT sid
FROM Sailors NATURAL JOIN Reserves;

SELECT sname
FROM Temp_Sid NATURAL JOIN Sailors;
```

Learned lesson:

- Be careful when applying projection on non-key attributes. Duplicates may be trouble makers!