By Marina Barsky

Structured Query Language SQL

Lecture 8

SET OPERATIONS

Union, Intersection, and Difference in SQL

- If two SQL queries produce relations with compatible set of attributes then we can combine the queries using the set operations:

(*«subquery»*) UNION (*«subquery»*)

(*«subquery»*) INTERSECT (*«subquery»*)

(*«subquery»*) EXCEPT (*«subquery»*)

- The brackets are mandatory.
- The operands must be queries; you can’t simply use a relation name.

Example

```
(SELECT name
 FROM Took
 WHERE grade > 95)
        UNION
(SELECT name
 FROM Took
 WHERE grade < 50);
```

Bags vs Sets in
Union, Intersection and Difference

- We saw that a SELECT-FROM-WHERE statement uses bag semantics by default: Duplicates are kept in the result.
- The **set operations use set semantics** by default: Duplicates are eliminated from the result.

Motivation?

- When doing projection in relational algebra, it is harder to eliminate duplicates: one tuple at a time
- When doing **intersection** or **difference**, it is most efficient to **sort** the relations first. At that point you may as well eliminate the duplicates anyway.

Controlling Duplicate Elimination

- We can force the result of a SFW query to be a set:

```
SELECT DISTINCT …
```

- We can force the result of a set operation to be a bag by using ALL:

```
(SELECT sid
 FROM Took
 WHERE grade > 95)
        UNION ALL
(SELECT sid
 FROM Took
 WHERE grade < 50);
```

Bag Union

- **Union**, **intersection**, and **difference** need new definitions for bags.

- An element appears in the **union** of two bags the **sum** of the number of times it appears in each bag.

- Example:

  $\{1,2,1\} \cup \{1,1,2,3,1\}$

  $= \{1,1,1,1,1,2,2,3\}$

Bag Intersection

- An element appears in the **intersection** of two bags the **minimum** of the number of times it appears in either.

- Example:

  $\{1,2,1\} \cap \{1,2,3\}$

  $= \{1,2\}$.

Bag Difference

- An element appears in **difference** $A - B$ of bags as many times as it appears in A, **minus** the number of times it appears in B.
  - But never less than 0 times.

- Example:

$\{1,2,1\} - \{1,2,3\}$

$= \{1\}$.

Beware: Bag Laws != Set Laws

Not all algebraic laws that hold for sets also hold for bags.

Example

- Set union is *idempotent*, meaning that

  $\boldsymbol{S \cup S = S}$.

- However, for bags, if $\boldsymbol{x}$ appears $\boldsymbol{n}$ times in $\boldsymbol{S}$, then it appears 2n times in $S \cup S$.

- Thus $\boldsymbol{S \cup S\ != S}$ in general.

Example

```
create table P (a int, b int);
create table Q (a int, c int);
insert into P values  (1, 151), (2, 123), (3, 432), (1, 333), (1, 345),  (4, 912), (5, 123);
insert into Q values  (1, 44), (3, 88), (3, 12), (9, 12);
```

```
select * from P;       select * from Q;
 a |  b                 a | c
---+-----              ---+----
 1 | 151                1 | 44
 2 | 123                3 | 88
 3 | 432                3 | 12
 1 | 333                9 | 12
 1 | 345               (4 rows)
 4 | 912
 5 | 123
(7 rows)
```

Example: Q - P

- (select a from Q) except (select a from P);

```
9
(1 row)
```

- (select a from Q) except all (select a from P);

```
3
 9
(2 rows)
```

```
select * from Q;
 a | c
---+----
 1 | 44
 3 | 88
 3 | 12
 9 | 12
(4 rows)
```

```
select * from P;
 a |  b
---+-----
 1 | 151
 2 | 123
 3 | 432
 1 | 333
 1 | 345
 4 | 912
 5 | 123
(7 rows)
```

Example: P - Q

- (select a from P) except (select a from Q);
- 2
- 4
- 5
- (3 rows)

- (select a from P) except all (select a from Q);
- 1
- 1
- 2
- 4
- 5
- (5 rows)

```
select * from P;
 a |  b
---+-----
 1 | 151
 2 | 123
 3 | 432
 1 | 333
 1 | 345
 4 | 912
 5 | 123
(7 rows)
```

```
select * from Q;
 a | c
---+----
 1 | 44
 3 | 88
 3 | 12
 9 | 12
(4 rows)
```