Intermediate SQL

Joseph Hallett

February 13, 2023

Last time...

We introduced SQL as a language for querying databases

- How to create tables
- How to add and delete data
- How to run basic queries

This time...

More advanced SQL!

- More features, more function
- Other joins
- `NULL`

`NULL` is nothing

There is a special value in SQL to represent missing data: `NULL`.

- But they're *pretty much always* a bad idea
- The logic for comparing them is pretty whacky

NULL = NULL?

Lets say we have a database with the following table:

Person	Fruit
Joseph	Lime
Matt	Apple
Partha	

Lets find everyone who we know what their favourite fruit is!

```
SELECT * FROM fruit WHERE fruit <> NULL;
```

Err..., lets try the opposite?

```
SELECT * FROM fruit WHERE fruit = NULL;
```

Err what?

```
SELECT * FROM fruit WHERE fruit LIKE '%';
```

Person	Fruit
Joseph	Lime
Matt	Apple

So...

```
SELECT * FROM fruit WHERE fruit NOT LIKE '%';
```

NULL is weird...

Because NULL means *attribute missing*...

- The results of comparing with it are ~~just plain stupid~~ *somewhat unexpected*

The simple solution is to declare *everything* as NOT NULL

- And use a higher normal form (5NF) then you'll find they almost entirely disappear

Otherwise you have to memorise a bunch of ~~stupid~~ *special* comparators

```
SELECT * FROM fruit WHERE fruit IS NULL;
```

Person	Fruit
Partha	

```
SELECT * FROM fruit WHERE fruit IS NOT NULL;
```

Person	Fruit
Joseph	Lime
Matt	Apple

Tricky joins

Clearly testing for equality when `NULL` is problematic.

- So what happens when you want to join two tables together with `NULL`'s in them

Person	Fruit
Joseph	Lime
Matt	Apple
Partha	

Fruit	Dish
Apple	Apple crumble
Banana	Banana split
Cherry	
Lime	Daiquiri

What’s my favourite food?

So what might make a nice dish for each of your lecturers?

- (A `NATURAL JOIN` is like a regular `JOIN` but assumes same named columns ought to be equal).

Person	Fruit	Dish
Joseph	Lime	Daiquiri
Matt	Apple	Apple crumble

But what about poor *Partha*? How do we get him to appear in our table?

LEFT and RIGHT JOIN

When doing our previous `JOIN` we wanted only rows that matched…

- Technically called an `INNER JOIN`…

Sometimes we're okay with the database sticking `NULL` in if we want to keep columns where a join *can't* be made…

```
SELECT person, fruit.fruit, dish
FROM fruit
LEFT JOIN recipes
ON fruit.fruit = recipes.fruit;
```

Person	Fruit	Dish
Joseph	Lime	Daiquiri
Matt	Apple	Apple crumble
Partha		

RIGHT JOIN

A `RIGHT JOIN` is like a left join but the other way round…

```
SELECT fruit.fruit, dish, person
FROM fruit
RIGHT JOIN recipes
ON fruit.fruit = recipes.fruit;
```

Fruit	Dish	Person
Lime	Daiquiri	Joseph
Apple	Apple crumble	Matt
	Banana split	

Where has the Banana gone?!

```
SELECT recipes.fruit, dish, person
FROM fruit
RIGHT JOIN recipes
ON fruit.fruit = recipes.fruit;
```

Fruit	Dish	Person
Lime	Daiquiri	Joseph
Apple	Apple crumble	Matt
Banana	Banana split	
Cherry		

(Or just `NATURAL JOIN` and it'll *usually* take care of it…)

```
SELECT fruit, dish, person
FROM fruit
RIGHT NATURAL JOIN recipes;
```

Fruit	Dish	Person
Lime	Daiquiri	Joseph
Apple	Apple crumble	Matt
Banana	Banana split	
Cherry		

One more JOIN!

What if we want to do a LEFT and a RIGHT JOIN at the same time?

```
SELECT *
FROM fruit
FULL OUTER NATURAL JOIN recipes;
```

Person	Fruit	Dish
Joseph	Lime	Daiquiri
Matt	Apple	Apple crumble
Partha		
	Banana	Banana split
	Cherry	

What about statistic functions?

In the last lecture we introduced COUNT as a way of counting how many things exist?

- How may different fruits are in the outer joined table?

```
SELECT *
FROM fruit
FULL OUTER NATURAL JOIN recipes;
```

Person	Fruit	Dish
Joseph	Lime	Daiquiri
Matt	Apple	Apple crumble
Partha		
	Banana	Banana split
	Cherry	

```
SELECT COUNT(fruit)
FROM fruit
FULL OUTER NATURAL JOIN recipes
```

COUNT(fruit)
4

...So it looks like COUNT ignores NULL

Other statistics...

Lets rank fruits!

Fruit	Stars
Apple	0
Banana	4
Cherry	
Lime	5

```
SELECT AVG(stars) AS Average FROM ranking;
```

```
Average
    3.0
```

```
SELECT SUM(stars)/COUNT(fruit) AS Average
FROM ranking;
```

```
Average
      2
```

Remember computers are *awful*

- Multiply count by 1.0 to "*fix*"?
- Also number of stars is *ordinal* data so the *mean* shouldn't be used anyway...

将数据按照一定的顺序排列

What about standard deviation?

The standard deviation is how far something deviates *on average* from the *mean*.

```
SELECT SQRT(AVG(Deviation)) AS STDDEV
FROM (
    SELECT Fruit, Stars, Mean,
        (Stars-Mean)*(Stars-Mean) AS Deviation
    FROM ranking JOIN (
        SELECT AVG(stars) AS Mean
        FROM ranking
    )
    WHERE stars IS NOT NULL
);
```

```
          STDDEV
2.16024689946929
```

You can nest queries inside one another (subqueries!)

- This is a recipe for making your SQL *slow*
- Maybe just use SQL for data retrieval and leave complex stats to statistical programming languges?

So thats SQL!

Tips for using it?

- Don’t overcomplicate things!
- Normal forms make things simpler!
- Avoid `NULL` like the plague