Relational SQL select

cs380

G Towell,
Sep 17

Last class

- Select within a single table
  - from — which table(s)
  - where — what part of the table
  - order by — how to sort
    - ORDER BY colName [asc,desc]
  - group by — put together like items
    - GROUP BY colName
  - limit — show only a certain number of rows
    - LIMIT 10

Exercises

Instructions: If you cannot think of a query for the whole answer, break down the question and write a query for some portion of the answer. Typically this will have more data than you need.

- List all departure delays
- List all unique departure delays.
  - use select distinct
  - use group by
- What was the largest departure delay?
  - and what flight number(s) had that delay?
- What is the smallest arrival delay?
  - actual delay, not early or on-time
  - How main times did it happen?
  - On which flight numbers did this arrival delay happen?
- Which is greater, the number of destinations or the number of origins?
  - Use two queries
- How many carriers are in this database. How many actually fly into PHL? (Out of PHL?)
- How many carriers fly from each location from which you can fly directly to PHL?

```
select count(distinct carrier),origin from flights group by origin;
```

- How many flights go past midnight?

```
select  count(*) from flights where hour(arrivaltime)<hour(departuretime);
```

- How many flights from each origin landed in PHL on the day after they took off?

```
select count(origin), origin from flights where origin!='phl' and hour(departuretime)>hour(arrivaltime) group by origin;
```

```
MariaDB [flight]> describe airports;
+---------+-------------+------+-----+---------+-------+
| Field   | Type        | Null | Key | Default | Extra |
+---------+-------------+------+-----+---------+-------+
| Name    | varchar(60) | YES  |     |         |       |
| Country | varchar(50) | YES  |     | NULL    |       |
| TLA     | char(3)     | NO   | PRI |         |       |
+---------+-------------+------+-----+---------+-------+
3 rows in set (0.001 sec)
```

```
MariaDB [flight]> describe carriers;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| Acronym  | varchar(3)  | NO   | PRI |         |       |
| Name     | varchar(50) | YES  |     | NULL    |       |
| CallSign | varchar(50) | YES  |     | NULL    |       |
| Country  | varchar(50) | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
4 rows in set (0.001 sec)
```

```
MariaDB [flight]> describe flights;
+----------------+------------+------+-----+-----------+-------+
| Field          | Type       | Null | Key | Default   | Extra |
+----------------+------------+------+-----+-----------+-------+
| Date           | date       | YES  |     | curdate() |       |
| DepartureTime  | time       | YES  |     | curtime() |       |
| ArrivalTime    | time       | YES  |     | curtime() |       |
| Carrier        | varchar(3) | YES  | MUL | NULL      |       |
| FlightNum      | int(11)    | YES  |     | -1        |       |
| ArrivalDelay   | int(11)    | YES  |     | 0         |       |
| DepartureDelay | int(11)    | YES  |     | 0         |       |
| Origin         | char(3)    | YES  | MUL | NULL      |       |
| Destination    | char(3)    | YES  | MUL | NULL      |       |
| Distance       | int(11)    | YES  |     | 0         |       |
| Cancelled      | tinyint(1) | YES  |     | 0         |       |
+----------------+------------+------+-----+-----------+-------+
11 rows in set (0.001 sec)
```

Beyond SELECT

- UNIX information security
- MySQL (and more generally RDBMS) permissions
- INSERT
- DELETE
- UPDATE
- CREATE / DROP

Grab a subset and select from that

- Anywhere you have a table name you can have a select statement, this allows working with subsets
  - SELECT eee.ccc, eee.ooo from (select count(origin) as ccc, origin ooo from flights group by origin) as eee;
  - *This example is pretty silly*
- Sometimes it is convenient to do the subsetting up front
  - WITH
    - I like using WITH, but hardcore SQL people do not seem to use it a lot
  - with eee(cnt, org) as (select count(origin), origin from flights where Origin!='phl' group by origin), fff(mxx) as (select max(eee.cnt) from eee) select eee.cnt,eee.org from eee,fff where eee.cnt=fff.mxx;
  - with eee as (select count(origin) as cnt, origin as org from flights where Origin!='phl' group by origin), fff as (select max(eee.cnt) as mxx from eee) select eee.cnt,eee.org from eee, fff where eee.cnt=fff.mxx;
    - Note use of two tables in the “from” (perfectly legal and natural)
      - using two tables like this can be inefficient
    - Note: WITH puts the alias before the thing aliased. Almost everything else puts the alias after.
      - WITH alias as (…)
      - Select name1 as alias1, …

Maxima and minima

- Just getting the correct answer for max can be tricky
  - select max(flightnum), date, departuretime from flights;
    - Not correct — only one row, and that row is wrong anyway!
- Need to use a subquery: These two are equivalent
  - select * from flights as fl where fl.flightnum = (select max(flightnum) from flights);
  - with eee as (select max(flightnum) as mx from flights) select fl.flightnum, fl.date from flights as fl, eee where fl.flightnum=eee.mx;
  - Note, use two tables in outer select!
- Max in group:
  - What airport has the most flights into PHL?
    - would like: select max(count(origin)) cc, origin oo from flights where origin!='phl' group by origin;
      - this does not work!
  - select count(origin) as cc, origin from flights where origin!='phl' group by origin order by cc desc limit 1;
    - Works but really ugly and inefficient — using a sort to select 1 item.
    - This is OK on small tables (and can be very efficient), but not OK on big tables

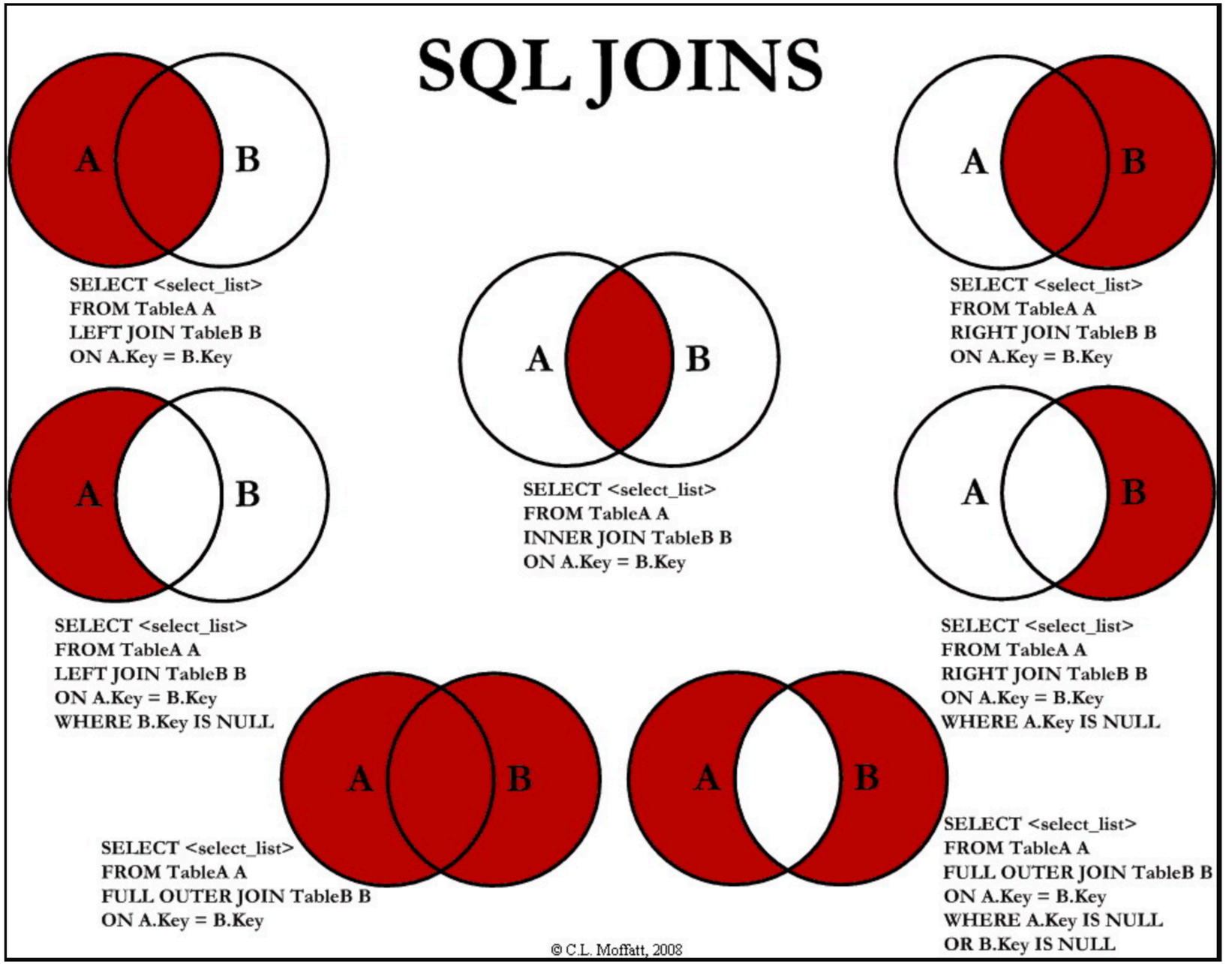
Join, putting the relationship into RDBMS

- Basic idea of join — get two sets / subsets and put then together
  - preferably using primary / foreign keys
    - Not required but can be 1000x (or more) faster
- So, suppose you want a table with all flights departing PHL before noon on Jan 1
  - select * from flights where month(date)=1 and day(date)=1 and hour(departuretime)<12 and origin='phl';
  - Problem — Three letter acronyms
  - Solution — JOIN
  - select f.carrier, f.date, f.departuretime, a.name, a.country from flights as f join airports as a on f.destination=a.tla where month(date)=1 and day(date)=1 and hour(departuretime)<12 and origin='phl';
- Now and do more — specify that destination should be outside USA!
  - select f.carrier, f.date, f.departuretime, a.name, a.country from flights as f join airports as a on f.destination=a.tla where month(date)=1 and day(date)=1 and hour(departuretime)<12 and origin='phl' and a.country!='usa';

Keeping the join going

- Still have ugly acronyms for the airline
  - Use another Join!!!
  - select c.name, f.date, f.departuretime, a.name, a.country from flights as f join airports as a on f.destination=a.tla join carriers as c on c.acronym=f.carrier where month(date)=1 and day(date)=1 and hour(departuretime)<12 and origin='phl' and a.country!='usa';
- Finally, find all departures from PHL in January to a US destination on a non-US based carrier
  - select c.name, c.country, f.date, f.departuretime, a.name, a.country from flights as f join airports as a on f.destination=a.tla join carriers as c on c.acronym=f.carrier where month(date)=1 and day(date)=1 and origin='phl' and a.country='usa' and c.country not like '% States';

- Inner
- Left
- Right
- Full Outer
- “”
  - same as inner



https://www.codeproject.com/Articles/33052/Visual-Representation-of-SQL-Joins

More Selecting

Hurricanes

```
show tables;
+--------------------+
| Tables_in_hurricane |
+--------------------+
| hurricane          |
| observation        |
| typename           |
+--------------------+
```

```
describe typename;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| type  | varchar(2)  | NO   | PRI | NULL    |       |
| name  | varchar(30) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

```
describe hurricane;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| HID   | char(8)     | NO   | PRI | NULL    |       |
| Name  | varchar(25) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

```
describe observation;
+-------------+-----------------------------------------------------------------------------+------+-----+---------+-------+
| Field       | Type                                                                        | Null | Key | Default | Extra |
+-------------+-----------------------------------------------------------------------------+------+-----+---------+-------+
| HID         | char(8)                                                                     | YES  | MUL | NULL    |       |
| date        | date                                                                        | YES  |     | NULL    |       |
| time        | time                                                                        | YES  |     | NULL    |       |
| type        | enum('','TY','PT','ET','ST','TD','TS','HU','EX','SD','SS','LO','WV','DB') | YES  |     | NULL    |       |
| latitude    | float                                                                       | YES  |     | NULL    |       |
| latitudehemi | char(1)                                                                    | YES  |     | NULL    |       |
| longitude   | float                                                                       | YES  |     | NULL    |       |
| longitudehemi | char(1)                                                                   | YES  |     | NULL    |       |
| maxSustained | int(11)                                                                    | YES  |     | NULL    |       |
+-------------+-----------------------------------------------------------------------------+------+-----+---------+-------+
```

Searching for Hurricanes

- Name of Easternmost point (in the western hemisphere) hurricanes whose names start with A
- Name of Easternmost hurricane
  - Note this uses a join on a non-key
    - This can be very slow.
    - BUT there is only 1 observation on one side of join
- Name of first observed storm in database (with the full name of the type) and an actual name!
- First observation for each named storm in database (with the full name of the type)
- Last observation of each named hurricane when the storm was of type ‘db’.
  - Be very careful when interpreting answers when you have grouped and maxed. The data in the row will not necessarily be forrm the max row

Thoughts about efficiency

- Avoid queries with multiple tables in “from”
  - This will generate a new temporary table with |M|*|N| rows
    - e.g. select count(*)from hurricane as hu, observation as ob;
      - 51840*1893 = 98133120 (and the query took 3 seconds to run!
    - If you throw a where onto this, it will still generate the huge table before cutting it down
      - select hu.hid, hu.name, ob.type from hurricane as hu, observation as ob where hu.hid=ob.hid
  - better to use join
    - select hu.hid, hu.name, ob.type from hurricane as hu inner join observation as ob on hu.hid=ob.hid;

Joining efficiency

inner join only

- **join order matters!**
  - because if we can join two tables that will reduce the number of rows needed to be processed by subsequent steps, then our performance will improve.
- SO as a general rule:
  - Specify the largest table first.
  - Next, specify the smallest table. The contents of the second, third, and so on tables are all transmitted across the network. You want to minimize the size of the result set from each subsequent stage of the join query. The most likely approach involves joining a small table first, so that the result set remains small even as subsequent larger tables are processed.
  - Join the next smallest table, then the next smallest, and so on.
  - For example, if you had tables BIG, MEDIUM, SMALL, and TINY, the logical join order to try would be BIG, TINY, SMALL, MEDIUM.