**SQL**

**Intro – Rel’n algebra very theoretical, SQL very applicable, practical, but they DO THE SAME THING. I try to alternate between theoretical topics and applications of such topics.**

* Stands for Structured Query Language
* A few different standards for SQL, and different DBMSs implement different parts of it. Some DBMSs introduce their own SQL syntax, though basics are common to all DBs.

How you use SQL to interact with a database.

* Databases often live on large files on “database servers” somewhere.
* Designed to be used by multiple people at once (think Amazon, Facebook, …)
* Databases usually allow multiple connections [draw picture].
* Usually these connections are made by computer programs (Python, C++, web applications in PHP, etc), and these programs send SQL commands to the database and get back results (tabular data).
* We can open a manual SQL connection to a database as well.

Client-server DBMS

Oracle, MS SQL Server, MySQL, PostgreSQL

SQLite

RDBMS embedded in a C library

CREATE TABLE courses (crn integer, year integer, cname text, pLast text, startTime int, room text);

CREATE TABLE grades(sLast text, crn int, grade int);

CREATE TABLE heads (house string, pLast string);

CREATE TABLE profs (pLast string, pFirst string);

CREATE TABLE rooms (room text, maxseats int);

CREATE TABLE students (sLast string, sFirst string, house string, pet string);

SQL is case insensitive.

* Generic form of a query:
  + **SELECT X from Y where Z.**
  + FROM clause Y tells us what tables hold the data we want.
  + WHERE clause (Z) is a condition, analogous to selection in relational algebra.
  + SELECT clause (X) tells us what attributes we want, analogous to projection.
  + More or less equivalent to pi\_X (sigma\_Z (Y))
* Find me the last names of all students in Gryffindor house.
  + Do in relational algebra.
  + Do in SQL [select last from students where house=’Gryffindor’]
  + Technical points
    - Case insensitive
    - Strings in single quotes.
    - Double quotes are only used around names of columns or tables that have special characters (like a space).
* Find me the names of all courses taught in 1997.
  + Bags versus sets.
  + SELECT DISTINCT
  + SELECT \*
* Can do math – select grade\*25 from grades;
  + Select grade\*25 as newgrade…
  + Can select a constant.
* WHERE clause
  + Use =, <>, <, >, <=, >=, AND OR, NOT.
  + For String concat, use ||
  + Concat first name and last name in students:  
     select slast||', '||sfirst from students
* Ask class – find all students grades that are Bs or Cs.
* String pattern matching
  + Uses LIKE, NOT LIKE
  + % is a wildcard (0 or more chars), \_ is any one character.
  + Find all students whose names start with G.
  + Find all students with six letter last names.
* ORDER BY attrib1, attrib2…
  + Each attrib can be followed by ASC or DESC
* Exercises (all these use only one table at a time)
  + Find all Slytherin students with Owls.
  + Find all years where history of magic was taught before noon.
    - 94, 95, 96, 97
  + Find all professors with a hyphen in their last name.
  + Find me all rooms with at least ten seats.
  + Find me last names of all students who have ever failed a class.
  + Solutions:
  + select slast, sfirst from students where pet='Owl'
* 6.2 joins
  + **Cartesian product** – just list each table in the FROM clause separated by commas.
  + Example – partner up Gryffindors with Slytherins.
    - Select \* from students, students
    - select \* from students s1, students s2;
    - select s1.slast, s2.slast from students s1, students s2 where s1.house='Gryffindor' and s2.house='Slytherin'
  + **THETA-JOIN**
    - Use T1 JOIN T2 ON <criteria> to link attributes.
    - Example: We want to see the grades table, put also include student first names.
    - select students.slast, students.sfirst, crn, grade from grades JOIN students ON grades.sLast=students.slast
    - Can leave out periods if attribute names are unambiguous.
  + **ASK** students – query to get a list of courses with prof first and last names (I want CRN, course name, prof first and last)
    - Select crn, cname, profs.pLast, profs.pFirst from courses JOIN profs on courses.pLast=profs.pLast
  + The T1 JOIN T2 ON clause is equivalent to
    - T1, T2 WHERE ...<criteria> but expresses better what you’re doing.
  + Sometimes you will see INNER JOIN. The word INNER is optional.
  + Do this:
    - Get a list of grades with course titles
      * Select \* from grades JOIN courses ON courses.crn=grades.crn
    - ASK get a list of grades with course titles, student first and last names of the students who are enrolled, and their grade in the class. (NEXT step add in) prof first and last names of the prof who is teaching the class.
    - Step 1: Select \* from grades JOIN courses on courses.crn = grades.crn (852 rows)
    - Step 2: Select \* from grades JOIN courses on courses.crn = grades.crn  
      JOIN students on grades.sLast = students.sLast (852 rows)
    - Select cname, students.slast, students.sfirst, grade from grades   
      JOIN courses on courses.crn = grades.crn  
      JOIN students on grades.sLast = students.slast
    - Step 3:   
      Select cname, students.slast, students.sfirst, profs.pLast, profs.pFirst, grade from **grades**

JOIN **courses** on courses.crn = grades.crn

JOIN **students** on grades.sLast = students.slast

JOIN **profs** on courses.pLast = profs.pLast

* NATURAL JOIN
  + Better than theta-join on equality because it’s less typing and SQL will automatically match up the columns with the same names. (but considered **DANGEROUS**)
  + Con – only works when the columns actually do have the same names.
  + **GET** List of students first and last names with their head of house.
  + select sfirst, slast, pLast from students natural join heads;
  + ASK students to compare this query:
    - select \* from students natural join heads;
    - VERSUS this one
    - select \* from students join heads on students.house = heads.house
  + Notice there is only ONE column for the attributes in common between the two tables that are being joined. This is exactly the same differences when using a Natural Join In relational algebra versus a THETA JOIN or CART PROD with some criteria that sets two cols equal to each other.
  + ASK Get a list of courses for 1997 with the number of seats each course can hold.
  + select \*
  + from courses natural join rooms
  + where courses.year=1997

QUICK REVIEW OF JOINS

Cartesian product: SELECT … FROM T1, T2 WHERE ….

Theta join: SELECT … FROM T1 [INNER] JOIN T2 ON [criteria] WHERE …

Natural join: SELECT … FROM T1 NATURAL JOIN T2 WHERE …

* Renaming practice.
  + Find me all courses that are scheduled for the same room and the same time (conflicts within the same year). Return two crns, two titles, year, room, and start time. [hint: 5 pairs of courses]
    - select c1.crn, c1.cname, c2.crn, c2.cname, c1.starttime, c1.room, c1.year from courses c1, courses c2 where c1.room = c2.room and c1.starttime =c2.starttime and c1.year = c2.year and c1.cname < c2.cname;
    - Returns 5 pairs of courses.
  + Find me all grade records of students who are taking a course taught by their head of house.
    - Need info from: courses, grades, heads, students
    - Return student last name, crn, course name, professorlastname.
    - Returns 90 rows.

select \*

from grades natural join courses natural join heads natural join students

* + Find me all grade records of students who are taking a course NOT taught by their head of house. (should return 852-90 = 762 rows)
    - Hint – if you get 3040 rows or 2556 rows, you are partially there.
  + select \*
  + from grades natural join courses natural join students JOIN heads on students.house = heads.house and
  + courses.pLast <> heads.pLast
  + Find me all pairs of pairs of possible tutors – a pair of students who are enrolled in the same CRN where one student has an A and the other student has a D or F. (932 rows)

select \* from students s1 natural join grades g1

join students s2 natural join grades g2

on g1.grade > 2 and g2.grade <= 1

* UNION, INTERSECT, EXCEPT
  + Differs from book – SQLlite doesn’t like parentheses. Pgsql is ok.
  + Get last names of all students and professors.
    - (select slast from students) union (select plast from profs)
  + Get last names of students who have taken courses CRN 1 and CRN2.
    - select studentlast from grades where crn=1 intersect select studentlast from grades where crn=2; [ 7 rows ]
* Ex: get me all profs who teach 9am classes. (prof firstname, last name, start time,
* Ex: get all profs who don’t teach 9am classes.
  + select plast from profs except select plast from courses where starttime = 9
* Ex: all courses taught in rooms with at least ten seats.
* Ex: all profs who have never taught two courses
  + select last from profs except select t.proflast from courses s, courses t where s.year != t.year and s.proflast = t.proflast;
* Ex: all profs who have taught exactly one course.
  + select proflast from courses except select t.proflast from courses s, courses t where s.year != t.year and s.proflast = t.proflast;
* 6.3
* Nested queries or subqueries
* 6.3.1
* If you can write a query and guarantee that the relation that comes back will have exactly one row and exactly one column, you can use this as a subquery to answer a further question.
  + Ex: Find the last name of the head of Padma Patil’s house.
  + Old way: select heads.pLast from students natural join heads where students.slast=’Patil’;
  + New way – How would you answer this query if you had the tables printed out in front of you?
    - What house is Padma Patil in?
    - Who is the head of that house?
    - select house from students where slast=’Patil’;
    - select pLast from heads where house=( select house from students where slast=’Patil’)
  + Ex: Find all classes held in the same room as CRN 42.
    - Old way: Cartesian product with a join of courses with itself.
    - Select \* from courses c1 join courses c2 on c1.room=c2.room where c1.crn=42;
    - New way:
    - What room is CRN 42 held in?
      * Select select room from courses where crn=42
    - What other courses are in that room?
      * select \* from courses where room=(select room from courses where crn=42);
  + Ex: Find the first and last name of the head of Padma Patil’s house.
    - ASK THIS ONE TO CLASS
    - Which tables contain our data?
    - Students (Padma Patil’s house)
    - Heads (who is head of what house?)
    - Profs (first/last name)
    - Do this old way and new way.
    - New way:  
      select \* from profs where last=(select proflast from heads where house=(select house from students where last="Patil"));
    - Old way:
    - Select \* from students natural join heads join profs on profs.last=heads.proflast where students.last=’Patil’;

6.3.2

* Subqueries that return one column, but multiple rows
* A subquery can return a relation as well, and there are interesting things you can do with that. There are some SQL operators that you can use with subqueries that return relations (rather than scalars):
  + EXISTS R [true if R is not empty]
  + s IN R is true if s is somewhere in R. [ok for sqlite]
  + s NOT IN R [also ok for sqlite]
  + s > ALL R [ pgsql ok ]
  + s > ANY R [ pg sql ok ]
* (IN) Get me all the classes taught by a head of a house.
  + select \* from courses where pLast in (select pLast from heads);
* (NOT IN)
  + select \* from courses where pLast not in (select pLast from heads);
* Practice: Find me all courses in 1996 that Harry Potter could take that don’t conflict (start time) with a class he’s already in. [3 rows]
  + First question – what are the start times of courses HP is already in?
  + select starttime from courses natural join grades where sLast='Potter' and year=1996
  + second question – find me all courses in 1996 that have different start times.
  + Select \* from courses where year=1996 and starttime NOT IN (select starttime from courses natural join grades where sLast='Potter' and year=1996)
  + …..AND name NOT IN (select name from courses natural join grades where sLast='Potter' and year=1996)
* Get me a listing of all care of magical creatures classes that where taught in years before hagrid starting teaching the course.
  + When did Hagrid teach the course?
  + select year from courses where cname like 'Care%' and plast='Hagrid'
  + get all courses before that year
  + select \* from courses where name like ‘Care%’ and year < ALL ( subquery )
    - SQLite doesn’t support this, other DBMSs do, but there’s a different way to do it once we learn aggregation (MIN)
  + If you want to do “<> ALL”, use NOT IN //// “= ANY” use IN
* [6.3.3] subqueries involving more than one col and possibly more than one row **[ok pgsql]**  
    
  Get me all courses taught immediately after another course in the same room.
  + select \* from courses where (year,room,starttime) in (select year,room,starttime+1 from courses);  
      
    doesn’t work in SQLite.
  + Can do with theta-join.
  + select \* from courses c1 join courses c2 on c1.year=c2.year and c1.room=c2.room and c1.starttime=c2.starttime+1
* Correlated subqueries 6.3.4
  + So far all of our subqueries can stand alone, in that you could copy and paste the subquery and run it separately.
  + However, there are some situations where you want the subquery to be able to access a piece of information from the outer query.
  + Analogous to nested for loops where inner loop uses/doesn’t use outer loop variable.
  + Students who have a unique pet within their house.
  + Consider nested for loops over students (write code on board):
    - Loop over every student
      * Inner loop: get all OTHER students in the house and look at their pets.
      * Is my pet not one of those types? Then it's unique.
  + select \* from students s1 where pet not in (select pet from students s2 where s2.house=s1.house

and s2.last != s1.last)

* + - "Weasley" "Ron" "Gryffindor" "Rat"
    - "Edgecombe" "Marietta" "Ravenclaw" "Toad"
    - "Patil" "Padma" "Ravenclaw" "Cat"
* 6.3.5
  + Queries in FROM clause
    - Get me all Gryffindor failing grades. (last name, house, crn, grade)
    - select slast,house,crn,grade from students natural join grades where house='Gryffindor' and grade = 0;
    - select slast,house,crn,grade from (select slast,house from students where house='Gryffindor') as S natural join grades where grade=0;

Return to “pizza places frequented by students from only one college”

Database schema:

Person(name, age, school) name is a key

Frequents(name, pizzeria) (name, pizzeria) is a key

Eats(name, pizza) (name, pizza) is a key

Serves(pizzeria, pizza, price) (pizzeria, price) is a key

school is either “Rhodes” or “U of M”

pizzeria is the name of a pizza restaurant (e.g., “Memphis Pizza Café,” “Broadway Pizza,” etc)

pizza is a type of pizza (e.g., “pepperoni,” “cheese,” “pineapple,”…)

R = person natural join frequents [so now we have name, age, school, pizzeria]

Select \* from person natural join frequents

* Grouping/aggregation
* Count(\*) or COUNT(attrib), count (distinct attrib), sum, max, min, avg
  + Find me how many total classes harry Potter has taken.
    - select count(\*) from grades where sLast='Potter'
  + Find me his GPA, highest grade, lowest grade.
    - GPA: select avg(grade) from grades where sLast='Potter'
    - Highest grade: select max(grade) from grades where sLast='Potter'
  + Turns out harry Potter has repeated some classes (has enrolled in duplicate classes with the same course name). Find total distinct classes.
    - select count(distinct cname) from grades natural join courses where sLast='Potter'
    - Results = 8
* Group by, having
  + Find out how many courses were offered every year.
    - select year, count(\*) from grades natural join courses group by year
  + Find me how many courses harry potter took each year.
    - select year, count(\*) from grades natural join courses group by year where sLast='Potter'
  + Find me his GPA by year.
    - select year, avg(grade) from grades natural join courses

where sLast ='Potter'

group by year

* + Find me his GPA by year, discounting courses where he failed.
    - select year, avg(grade) from grades natural join courses

where sLast ='Potter' and grade>0

group by year

* + In the select clause, when doing aggregation, only two kinds of things may appear:
    - Aggregations on an attrib
    - Indiv attribs that are grouped by
  + **CLASSPRAC**
  + Find me a list of all profs and the number of courses they have ever taught.
    - select pLast, count(\*) from courses group by proflast
  + List of every CRN and the number of people enrolled in that CRN
    - select crn, count(\*) from grades group by crn
  + Find me a list of all courses taught with the number of times they’ve been taught.
    - Select name, count(\*) from courses group by name
  + List of every CRN and the number of people enrolled in that CRN, plus course name
    - select crn, enrolled, name from

(select crn, count(\*) as enrolled from grades group by crn) AS S

natural join courses

* PAUSE: This is a useful query. Let's expand it to get all information from the grades table.

select \* from

(select crn, count(\*) as enrolled from grades group by crn) AS S

natural join courses

CREATE VIEW enrollment AS  
[above query]

**HAVING**

* + Find me courses taught twice in the same year.
    - Old way = Cartesian product
    - select name, year, count(\*) from courses group by name, year
    - select name, year, count(\*) from courses group by name, year having count(\*)=2
  + Find the earliest class start time.
    - select min(starttime) from courses
  + Find the earliest class start time each year.
    - select year, min(starttime) from courses group by year
  + Find me a list of course names that are always taught at 10am or later.
    - How to think about this:
    - Get list of classes grouped by course name,
    - Look at start times within each group
    - Take minimum
    - select min(starttime),name from courses group by name having min(starttime)>9;

WRAPUP

* + How would we get the most popular course?
  + select max(enrolled) from enrollment
  + But how do we get all the auxiliary information here? Like the name of the course and who's teaching it?
  + Can't just add those to the above query. Try it; doesn't work. using where doesn't work. using having doesn't work.
  + Use subquery:
  + select \* from enrollment where enrolled=(select max(enrolled) from enrollment)
  + What about most popular course per year?
  + select max(enrolled), year from enrollment group by year
  + select \* from
  + (select max(enrolled) as maxenrolled), year from enrollment group by year) as
  + join
  + enrollment
  + on maxenrolled

old stuff

List of every course name along with largest enrollment in that room (cname, enrollment)  
  
select max(enrolled), cname from

(select crn, count(\*) as enrolled from grades group by crn) as S

natural join courses group by cname

But now what if we want the year each of those was taught?

Can't just add YEAR to the above query.

* Find highest enrollment course [19]
  + select max(enroll) from

(select crn, count(\*) as enroll from grades group by crn) natural join courses

* Find the name(s) of this highest enrollment course
  + select \* from

((select crn, count(\*) as enroll from grades group by crn) natural join courses)

where enroll=

(select max(enroll) from

(select crn, count(\*) as enroll from grades group by crn) natural join courses

)

* Find most popular CRN every year

select max(enroll), year from

(

select crn, count(\*) as enroll from grades group by crn

) natural join courses

group by year

* Most popular CRN [highest enrollment] (per year), (per house)
* Most popular teacher
* Select from where group by having order by