CPSC 304 – March 1, 2018 Administrative notes

- Reminder: tutorial this week, but none is due
- Reminder: Midterm #2: March 8@7pm in CIRS 1250 (same room as last time)
  - Closed notes, closed book, closed neighbour
  - Bring ID
  - Will cover through the end of Datalog (I'll put in a stopping point slide)

Let's talk about the project for a minute

- As posted on Piazza, next check point moved to Monday, March 5
- Reminder: these checkpoints are ungraded
- PHP vs. JDBC
- Group work

Taking it to the next level

Say you're planning a beach vacation

And you wanted to find if it's possible to get from YVR to OGG (that's on Maui)

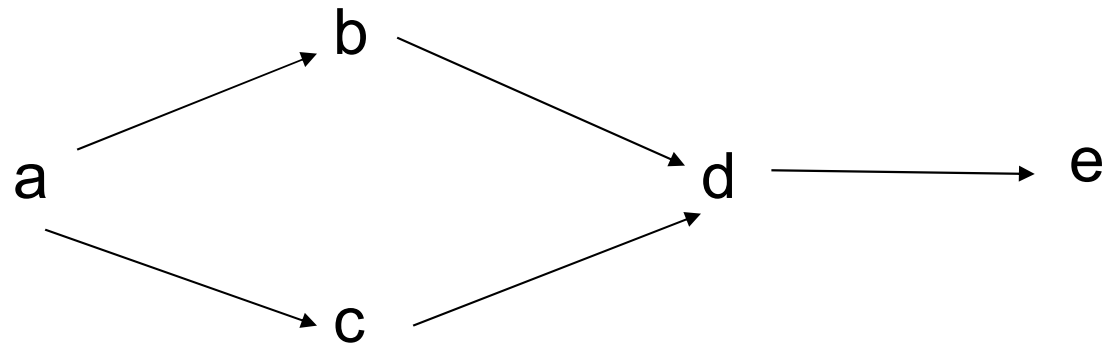
Your available information:

Flight(airline,num,origin,destination)

Now what?

A more general Example: Transitive Closure

Suppose we represent a graph w/ relation *Edge(X,Y):*
Edge(a,b), Edge (a,c), Edge(b,d), Edge(c,d), Edge(d,e)



How can I express the query: *Find all paths*

```
Path(X, Y)   :-  Edge(X, Y).
Path(X, Y)   :-  Path(X, Z), Path(Z, Y).
```

Evaluating Recursive Queries

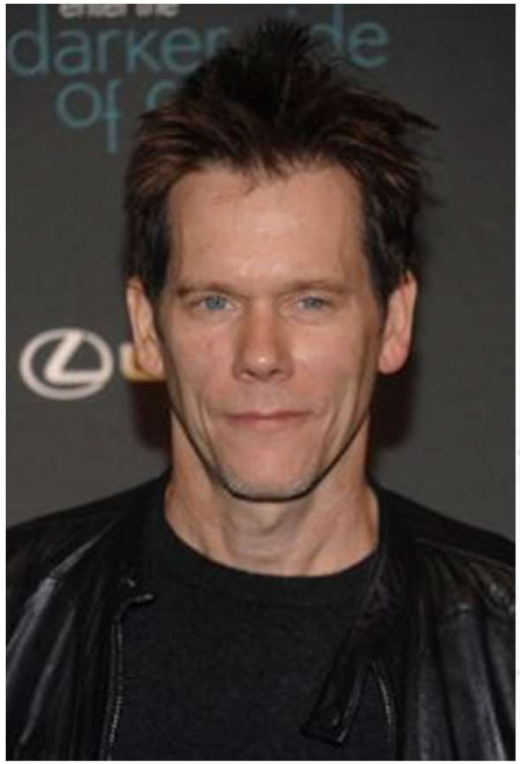
Path(X, Y) :- Edge(X, Y).
Path(X, Y) :- Path(X, Z), Path(Z, Y).

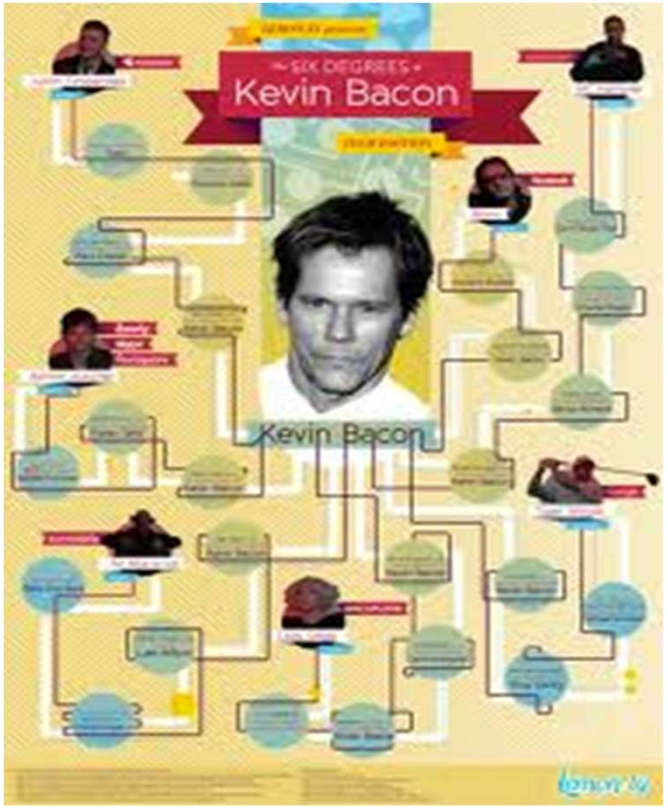
Semantics: evaluate the rules until a *fixed point:*

Iteration #0: Edge: {(a,b), (a,c), (b,d), (c,d), (d,e)}
Path: {}
Iteration #1: Path: {(a,b), (a,c), (b,d), (c,d), (d,e)}
Iteration #2: Path gets the new tuples: (a,d), (b,e), (c,e)
Path: {(a,b), (a,c), (b,d), (c,d), (d,e), (a, d), (b,e), (c, e)}
Iteration #3: Path gets the new tuple: (a,e)
Path: {(a,b), (a,c), (b,d), (c,d), (d,e), (a, d), (b,e), (c, e), (a,e)}
Iteration #4: Nothing changes → Stop.

Note: # of iterations depends on the data. Cannot be anticipated by only looking at the query!

A fun Example





More examples

- Given:
  Movie(id, title)
  Actor(id, name)
  Role(movie-id, actor-id, character)
- Find names of actors who have “Bacon numbers” (assume there’s only one “Kevin Bacon”)

CoStars(Aid,Bid):-Role(Mid,Aid,_), Role(Mid,Bid,_)
CoStars(Aid,Bid):- CoStars(Aid,Cid), CoStars(Cid,Bid)
Bacon_N(B):-Actor(Aid, “Kevin Bacon”), CoStars(Aid,Bid), Actor(Bid,B)

Skip the stuff on Magic Sets

- That's Datalog
- It's simple
- It's based on logic
- It's easy to see the join patterns (especially with anonymous variables)

Learning Goals Revisited

- Given a set of tuples (an input relation) and rules, compute the output relation for a Datalog program.
- Write Datalog programs to query an input relation.
- Explain why we want to extend query languages with recursive queries. Provide good examples of such queries.
- Explain the importance of safe queries, and what makes a Datalog query safe.

Midterm #2 covers through here.

CPSC 304
Introduction to Database Systems

Structured Query Language (SQL)

Textbook Reference
Database Management Systems: Chapter 5

Databases: the continuing saga

When last we left databases…

- We had decided they were great things
- We knew how to conceptually model them in ER diagrams
- We knew how to logically model them in the relational model
- We knew how to normalize our database relations
- We could formally specify queries in Relational Algebra and Datalog

Now: how do most people write queries? SQL!

Learning Goals

- Given the schemas of a relation, create SQL queries using: SELECT, FROM, WHERE, EXISTS, NOT EXISTS, UNIQUE, NOT UNIQUE, ANY, ALL, DISTINCT, GROUP BY and HAVING.
- Show that there are alternative ways of coding SQL queries to yield the same result. Determine whether or not two SQL queries are equivalent.
- Given a SQL query and table schemas and instances, compute the query result.
- Translate a query between SQL and RA.
- Comment on the relative expressive power of SQL and RA.
- Explain the purpose of NULL values and justify their use. Also describe the difficulties added by having nulls.
- Create and modify table schemas and views in SQL.
- Explain the role and advantages of embedding SQL in application programs.
- Write SQL for a small-to-medium sized programming application that requires database access.
- Identify the pros and cons of using general table constraints (e.g., CONSTRAINT, CHECK) and triggers in databases.

Coming up in SQL...

- Data Definition Language (reminder)
- Basic Structure
- Set Operations
- Aggregate Functions
- Null Values
- Nested Subqueries
- Modification of the Database
- Views
- Integrity Constraints
- Putting SQL to work in an application

The SQL Query Language

- Need for a standard since relational queries are used by many vendors
- Consists of several parts:
  - Data Definition Language (DDL)
    (a blast from the past (Chapter 3))
  - Data Manipulation Language (DML)
    - Data Query
    - Data Modification

Creating Tables in SQL(DDL) Revisited

- A SQL relation is defined using the **create table** command:

  create table *r* (A_1 D_1, A_2 D_2, ..., A_n D_n,
  (integrity-constraint$_1$),
  ...,
  (integrity-constraint$_k$))

- *Integrity constraints can be:*
  - *primary and candidate keys*
  - *foreign keys*
- Example:

```
CREATE TABLE Student
       (sid     CHAR(20),
        name  CHAR(20),
        address  CHAR(20),
        phone  CHAR(8),
        major    CHAR(4),
        PRIMARY KEY (sid))
```

Domain Types in SQL Reference Sheet

- **char(n).** Fixed length character string with length *n.*
- **varchar(n).** Variable length character strings, with maximum length *n.*
- **int.** Integer (machine-dependent).
- **smallint.** Small integer (machine-dependent).
- **numeric(p,d).** Fixed point number, with user-specified precision of *p* digits, with *d* digits to the right of decimal point.
- **real, double precision.** Floating point and double-precision floating point numbers, with machine-dependent precision.
- **float(n).** Floating point number, with user-specified precision of at least *n* digits.

- Null values are allowed in all the domain types.
  To prohibit null values declare attribute to be **not null**
- **create domain** in SQL-92 and 99 creates user-defined domain types

  create domain *person-name* **char**(20) **not null**

Date/Time Types in SQL Reference Sheet

- **date.** Dates, containing a (4 digit) year, month and date
  - E.g. **date** '2001-7-27'
- **time.** Time of day, in hours, minutes and seconds.
  - E.g. **time** '09:00:30' **time** '09:00:30.75'
- **timestamp**: date plus time of day
  - E.g. **timestamp** '2001-7-27 09:00:30.75'
- **Interval**: period of time
  - E.g. Interval '1' day
  - Subtracting a date/time/timestamp value from another gives an interval value
  - Interval values can be added to date/time/timestamp values
- Relational DBMS offer a variety of functions to
  - extract values of individual fields from date/time/timestamp
  - convert strings to dates and vice versa
  - For instance in Oracle (date is a timestamp):
    - TO_CHAR(date, format)
    - TO_DATE(string, format)
    - format looks like: 'DD-Mon-YY HH:MI.SS'

Running Example (should look familiar)

Movie(MovieID, Title, Year)

StarsIn(MovieID, StarID, Character)

MovieStar(StarID, Name, Gender)

Basic SQL Query

- SQL is based on set and relational operations
- A typical SQL query has the form:

  select $A_1, A_2, ..., A_n$
  from $r_1, r_2, ..., r_m$
  where P

  - A_is represent attributes
  - r_is represent relations
  - P is a predicate.

SELECT	*target-list*
FROM	*relation-list*
WHERE	*qualification*

π → SELECT clause
σ → WHERE clause
$\bowtie$ →FROM and WHERE clause

- The result of a SQL query is a table (relation)
- By default, duplicates are not eliminated in SQL relations, which are bags or multisets and not sets
- Let's compare to relational algebra…

Basic SQL/RA Comparison example 1

- Find the titles of movies

Basic SQL/RA Comparison example 1

- Find the titles of movies

$$\pi_{\text{Title}}(\text{Movie})$$

- In SQL, π is in the SELECT clause
- Select only a subset of the attributes

```
SELECT   Title
FROM     Movie
```

- Note duplication can happen!

This is going to be more interesting if you can write your own queries

- Your Oracle (DBMS) accounts on the department servers are ready
- Your login is "ora_[cs ugrad userid]" Your password is "a[your student id]“. To login:
  - Log onto remote.ugrad.cs.ubc.ca.
  - Start sqlplus (the interface to Oracle by typing "sqlplus [Oracle login]@ug" ("ug" is the specific database to access)
  - Additional information can be found on Piazza. Note: if you run into problems getting things to work, make sure that:
    - You're logging onto remote.ugrad.cs.ubc.ca
    - Login to the undergraduate machine using your regular user name and password
    - You have the "@ug" at the end of logging in (that says what database to access)
    - you don't forget to prepend your student ID with an "a" for your sql login
- Scripts for most data is on the exercises page

Clicker Question: SQL projection

- Given the table scores: what is result of SELECT Score1, Score2 FROM Scores
- Which of the following rows is in the answer?

Team1	Team2	Score1	Score2
Dragons	Tigers	5	3
Carp	Swallows	4	6
Bay Stars	Giants	2	1
Marines	Hawks	5	3
Ham Fighters	Buffaloes	1	6
Lions	Golden Eagles	8	12

A. (1,2)
B. (5,3)
C. (8,6)
D. All are in the answer
E. None are in the answer

clickerprojection.sql

Clicker Question: SQL projection

- Given the table scores: what is result of SELECT Score1, Score2 FROM Scores
- Which of the following rows is in the answer?

Team1	Team2	Score1	Score2
Dragons	Tigers	5	3
Carp	Swallows	4	6
Bay Stars	Giants	2	1
Marines	Hawks	5	3
Ham Fighters	Buffaloes	1	6
Lions	Golden Eagles	8	12

A. (1,2)
B. (5,3) Correct
C. (8,6)
D. All are in the answer
E. None are in the answer

In SQL, σ is in *Where* clause

```
SELECT   *
FROM     Movie
WHERE    Year > 1939
```

You can use:

attribute names of the relation(s) used in the FROM.
comparison operators: =, <>, <, >, <=, >=
apply arithmetic operations: rating*2
operations on strings (e.g., "||" for concatenation).
Lexicographic order on strings.
Pattern matching: s LIKE p
Special stuff for comparing dates and times.

Basic SQL/RA Comparison example 2

Find female movie stars

Basic SQL/RA Comparison example 2

Find female movie stars

$$\sigma_{\text{Gender} = \text{‘female’}}\text{MovieStar}$$

```
SELECT   *
FROM     MovieStar
WHERE    Gender = 'female'
```

Clicker Question: Selection

- Consider Scores(Team, Opponent, RunsFor, RunsAgainst) and query

```
SELECT *
FROM Scores
WHERE
  RunsFor > 5
```

- Which tuple is in the result?

A. (Swallows, Carp, 6, 4)
B. (Swallows, Carp, 4)
C. (12)
D. (*)

Team	Opponent	RunsFor	RunsAgainst
Dragons	Tigers	5	3
Carp	Swallows	4	6
Bay Stars	Giants	2	1
Marines	Hawks	5	3
Ham Fighters	Buffaloes	1	6
Lions	Golden Eagles	8	12
Tigers	Dragons	3	5
Swallows	Carp	6	4
Giants	Bay Stars	1	2
Hawks	Marines	3	5
Buffaloes	Ham Fighters	6	1
Golden Eagles	Lions	12	8

clickerselection.sql

Clicker Question: Selection

- Consider Scores(Team, Opponent, RunsFor, RunsAgainst) and query

```
SELECT *
FROM Scores
WHERE
  RunsFor > 5
```

- Which tuple is in the result?

A. (Swallows, Carp, 6, 4)
B. (Swallows, Carp, 4)
C. (12)
D. (*)

answer A

Team	Opponent	RunsFor	RunsAgainst
Dragons	Tigers	5	3
Carp	Swallows	4	6
Bay Stars	Giants	2	1
Marines	Hawks	5	3
Ham Fighters	Buffaloes	1	6
Lions	Golden Eagles	8	12
Tigers	Dragons	3	5
Swallows	Carp	6	4
Giants	Bay Stars	1	2
Hawks	Marines	3	5
Buffaloes	Ham Fighters	6	1
Golden Eagles	Lions	12	8

Selection & Projection – together forever in SQL

We can put these together:

- What are the names of female movie stars?

- What are the titles of movies from prior to 1939?

Selection & Projection – together forever in SQL

We can put these together:

- What are the names of female movie stars?

```
SELECT name
FROM MovieStar
WHERE Gender = 'female'
```

- What are the titles of movies from prior to 1939?

```
SELECT title
FROM Movie
WHERE year < 1939
```

Selection example (dates)

reserves

SID	BID	Day
22	101	1998-10-10
22	102	1998-10-10
22	103	1998-10-08
22	104	1998-07-10
31	102	1998-11-10
31	103	1998-11-06
31	104	1998-11-12
58	102	1998-11-08
58	103	1998-11-12

```
Select *
From reserves
Where day < DATE'1998-11-01'
```

SID	BID	Day
22	101	1998-10-10
B	102	1998-10-10
22	103	1998-10-08
22	104	1998-07-10

Basic SQL/RA comparison example 3

- Find the person names and character names of those who have been in movies

- In order to do this we need to use joins. How can we do joins in SQL?

  π → SELECT clause
  σ → WHERE clause
  $\bowtie$ → FROM and WHERE clause

Joins in SQL

```
SELECT  Character, Name
FROM    StarsIn s, MovieStar m
WHERE   s.StarID = m.StarID
```

- Cross product specified by From clause
- Can alias relations (e.g., “StarsIn s”)
- Conditions specified in where clause

Clicker Question: Joins

- Consider R :

a	b
0	0
0	1
1	0
1	1

S:

a	b
0	0
0	1
1	0
1	1

T:

a	b
0	0
0	1
1	0
1	1

```
SELECT R.a, R.b, S.b, T.b
    FROM R, S, T
    WHERE R.b = S.a AND S.b <> T.b
```

(note: <> == 'not equals')

Compute the results

Which of the following are true:

A. (0,1,1,0) appears twice.
B. (1,1,0,1) does not appear.
C. (1,1,1,0) appears once.
D. All are true
E. None are true

clickerjoin.sql

Clicker Question: Joins

- Consider R :

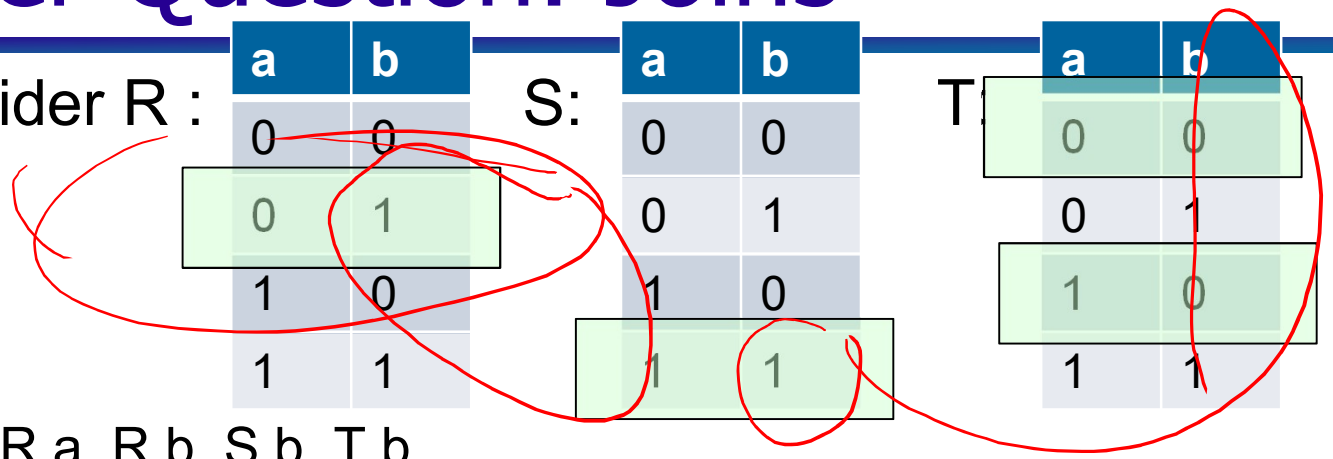
a	b
0	0
0	1
1	0
1	1

S:

a	b
0	0
0	1
1	0
1	1

T:

a	b
0	0
0	1
1	0
1	1



```
SELECT R.a, R.b, S.b, T.b
    FROM R, S, T
    WHERE R.b = S.a AND S.b <> T.b    (note: <> == 'not equals')
```

Compute the results

Which of the following are true:

A. (0,1,1,0) appears twice.
B. (1,1,0,1) does not appear.
C. (1,1,1,0) appears once.
D. All are true
E. None are true

True R(0,1) S(1,1), T(0,0)&
R(0,1), S(1,1), T(1,0),

False: R(1,1), S(1,0), T(0,1)

False: like A but use R(1, 1)