CPSC 304 – Administrative notes October 16 & October 17, 2024

- Final exam: December 16 @ noon
- Project:
  - Milestone 3: Project check in – due October 25
    - Sign up coming next week
  - Milestone 4: Project implementation – due November 29
  - Milestone 5: Group demo – week of December 2
  - Milestone 6: Individual Assessment – Due November 29
- October 22: Midterm @ 6PM
  - See Piazza for important midterm information, including how to find where you will write the midterm
    - If you can't find your location on PrairieTest, make sure you're on the US site: https://us.prairietest.com/
  - **All blank answers will be marked as incorrect!**
- Lecture before the midterm is open office hours (this Friday/next Tuesday)
- Tutorials: pivoting to project work/open office hours – next week is nominally SQL Plus
- Note that your SQL accounts and repositories go away with term end!

CPSC 304
Introduction to Database Systems

Structured Query Language (SQL)

Textbook Reference
Database Management Systems: Chapter 5

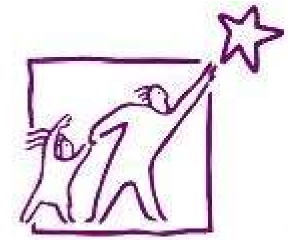
Databases: the continuing saga

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

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

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

- The sqlplus tutorial will walk you through the steps (do it now, or in tutorial next week)
- https://www.students.cs.ubc.ca/~cs-304/resources.html

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(<u>MovieID</u>, Title, Year)

StarsIn(<u>MovieID, StarID</u>, Character)

MovieStar(<u>StarID</u>, Name, Gender)

Basic SQL Query

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

  SELECT $A_1, A_2, \ldots, A_n$
  FROM $r_1, r_2, \ldots, r_m$
  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

Movie(<u>MovieID</u>, Title, Year)
StarsIn(<u>MovieID, StarID</u>, Character)
MovieStar(<u>StarID</u>, Name, Gender)

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!
  - You can get the same value multiple times

Basic SQL/RA Comparison example 1

- You can also refer to an attribute by (relation name).(attribute name)

```
SELECT   Movie.Title
FROM     Movie
```

- Since we are only working with one relation, you don't need to specify where the attribute comes from
  - This is useful when you have multiple relations that share the same attribute name

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

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

- Which of the following rows is in the answer?

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

$$\pi_{score1,score2}(Scores)$$

In SQL, σ is in *WHERE* clause

Movie(MovieID, Title, Year)
StarsIn(MovieID, StarID, Character)
MovieStar(StarID, Name, Gender)

```
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

Movie(<u>MovieID</u>, Title, Year)
StarsIn(<u>MovieID, StarID</u>, Character)
MovieStar(<u>StarID</u>, Name, Gender)

Basic SQL/RA Comparison example 2

Find female movie stars

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

```
SELECT   *
FROM     MovieStar
WHERE    Gender=‘female’
```

Movie(<u>MovieID</u>, Title, Year)
StarsIn(<u>MovieID, StarID</u>, Character)
MovieStar(<u>StarID</u>, Name, Gender)

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. (*)

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

answer A

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?

Movie(<u>MovieID</u>, Title, Year)
StarsIn(<u>MovieID, StarID</u>, Character)
MovieStar(<u>StarID</u>, Name, Gender)

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
```

Movie(MovieID, Title, Year)
StarsIn(MovieID, StarID, Character)
MovieStar(StarID, Name, Gender)

Selection example (dates)

reserves

SID	BID	Day
22	101	2010-10-10
22	102	2010-10-10
22	103	2010-10-08
22	104	2010-07-10
31	102	2010-11-10
31	103	2010-11-06
31	104	2010-11-12
58	102	2010-11-08
58	103	2010-11-12

```
SELECT *
FROM reserves
WHERE day < DATE'2010-11-01'
```

SID	BID	Day
22	101	2010-10-10
22	102	2010-10-10
22	103	2010-10-08
22	104	2010-07-10

Basic SQL/RA comparison example 3

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

  $\pi_{\text{Character, Name}}(\text{StarsIn} \bowtie_{\text{StarsIn.StarID} = \text{MovieStar.StarID}} \text{MovieStar})$

- 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: Simple Joins

- Consider R :

a	b
0	0
0	1
1	0
1	1

S:

a	b
0	0
0	1
1	0
1	1

```
SELECT R.a, R.b, S.a, S.b
    FROM R, S
    WHERE R.b = S.a
```

Compute the results

Which of the following are true:

A. (0,1,1,0) appears twice.

B. (1,1,0,1) appears once.

C. (1,1,1,0) appears once.

D. All are true

E. None are true

Simple Joins Results

R.a	R.b	S.a	S.B
0	0	0	0
1	0	0	0
0	0	0	1
1	0	0	1
0	1	1	0
1	1	1	0
0	1	1	1
1	1	1	1

Clicker Question: Simple Joins

- Consider R :

a	b
0	0
0	1
1	0
1	1

S:

a	b
0	0
0	1
1	0
1	1

```
SELECT R.a, R.b, S.a, S.b
    FROM R, S
    WHERE R.b = S.a
```

Compute the results

Which of the following are true:

A. (0,1,1,0) appears twice. — False. Only R(0,1) and S(1, 0)

B. (1,1,0,1) appears once. — False. R.b <> S.a

C. (1,1,1,0) appears once. — True. R(1,1) and S(1, 0)

D. All are true

E. None are true

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

Example of the cross product obtained from the **first tuple in R** x S x T.

R.a	R.b	S.a	S.b	T.a	T.b
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	1	0
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	0	1
0	0	0	1	1	0
0	0	0	1	1	1
0	0	1	0	0	0
0	0	1	0	0	1
0	0	1	0	1	0
0	0	1	0	1	1
0	0	1	1	0	0
0	0	1	1	0	1
0	0	1	1	1	0
0	0	1	1	1	1

R.a	R.b	S.a	S.b	T.a	T.b
~~0~~	~~0~~	~~0~~	~~0~~	~~0~~	~~0~~
0	0	0	0	0	1
~~0~~	~~0~~	~~0~~	~~0~~	~~1~~	~~0~~
0	0	0	0	1	1
0	0	0	1	0	0
~~0~~	~~0~~	~~0~~	~~1~~	~~0~~	~~1~~
0	0	0	1	1	0
~~0~~	~~0~~	~~0~~	~~1~~	~~1~~	~~1~~
~~0~~	~~0~~	~~1~~	~~0~~	~~0~~	~~0~~
~~0~~	~~0~~	~~1~~	~~0~~	~~0~~	~~1~~
~~0~~	~~0~~	~~1~~	~~0~~	~~1~~	~~0~~
~~0~~	~~0~~	~~1~~	~~0~~	~~1~~	~~1~~
~~0~~	~~0~~	~~1~~	~~1~~	~~0~~	~~0~~
~~0~~	~~0~~	~~1~~	~~1~~	~~0~~	~~1~~
~~0~~	~~0~~	~~1~~	~~1~~	~~1~~	~~0~~
~~0~~	~~0~~	~~1~~	~~1~~	~~1~~	~~1~~

Dropping all the tuples that don’t fulfill the WHERE clause.

Repeat this process for the other tuples in R.

What we get after processing the FROM and WHERE clauses.

R.a	R.b	S.a	S.b	T.a	T.b
0	0	0	0	0	1
0	0	0	0	1	1
0	0	0	1	0	0
0	0	0	1	1	0
0	1	1	0	0	1
0	1	1	0	1	1
0	1	1	1	0	0
0	1	1	1	1	0
1	0	0	0	0	1
1	0	0	0	1	1
1	0	0	1	0	0
1	0	0	1	1	0
1	1	1	0	0	1
1	1	1	0	1	1
1	1	1	1	0	0
1	1	1	1	1	0

R.a	R.b	S.b	T.b
0	0	0	1
0	0	0	1
0	0	1	0
0	0	1	0
0	1	0	1
0	1	0	1
0	1	1	0
0	1	1	0
1	0	0	1
1	0	0	1
1	0	1	0
1	0	1	0
1	1	0	1
1	1	0	1
1	1	1	0
1	1	1	0

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)

So how does a typical SQL query relate to relational algebra then?

SQL:

SELECT $A_1, A_2, \ldots, A_n$
FROM $r_1, r_2, \ldots, r_m$
WHERE P

Is approximately equal to

Relational algebra

$$\pi_{A1,\, A2,\, \ldots,\, An}(\sigma_P\, (r_1 \;\; \mathrm{x}\; r_2 \;\; \mathrm{x} \;\; \ldots \;\; \mathrm{x} \;\; r_m))$$

Difference? Duplicates.

Remove them? Distinct

Using DISTINCT

- Find the names of actors who have been in at least one movie

```
SELECT   DISTINCT Name
FROM     StarsIn S, MovieStar M
WHERE    S.StarID = M.StarID
```

- Would removing DISTINCT from this query make a difference?
- Note: on the exams, if we ask for a general question like "find all the names", we expect duplicates to be removed. When in doubt, keep the DISTINCT rather than reasoning through if you need it!

Clicker question: distinction

Consider the relation: Scores(Team, Opponent, RunsFor, RunsAgainst) and the query:

```
SELECT DISTINCT Team,
                RunsFor
FROM    Scores
```

Which is true:

A. 1 appears once
B. 5 appears twice
C. 6 appears 4 times
D. All are true
E. None are true

Team	Opponent	Runs For	Runs Against
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

Clicker question: distinction

Consider the relation: Scores(Team, Opponent, RunsFor, RunsAgainst) and the query:

```
SELECT DISTINCT Team,
        RunsFor
FROM    Scores
```

Which is true:

A. 1 appears once
B. 5 appears twice Correct
C. 6 appears four times
D. All are true
E. None are true

Team	Opponent	Runs For	Runs Against
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

clickerdistinction.sql

Join Example

- Find the names of all movie stars who have been in a movie

Join Example

- Find the names of all movie stars who have been in a movie

```
SELECT Name
FROM StarsIn S, MovieStar M
WHERE S.StarID = M.StarID
```

Is this totally correct?

StarID	Name	Gender
1	Harrison Ford	Male
2	Vivian Leigh	Female
3	Judy Garland	Female

MovieID	StarID	Character
1	1	Han Solo
4	1	Indiana Jones
2	2	Scarlett O'Hara
3	3	Dorothy Gale

Harrison Ford will appear twice

Join Example

- Find the names of all movie stars who have been in a movie

```
SELECT Name
FROM StarsIn S, MovieStar M
WHERE S.StarID = M.StarID
```

Is this totally correct?

```
SELECT DISTINCT Name
FROM StarsIn S, MovieStar M
WHERE S.StarID = M.StarID
```

What if two movie stars had the same name?

- What if I run the following query?

```
SELECT DISTINCT StarID, Name
FROM StarsIn S, MovieStar M
WHERE S.StarID = M.StarID
```

Error: Column StarID is ambiguous

Select Project Join example

- What are all the titles of movies with female actors?
- Write in Relational Algebra and SQL

Select Project Join example

- What are all the titles of movies with female actors?
- Write in Relational Algebra and SQL

Relational algebra:

$\pi_{\text{title}}(\sigma_{\text{gender = 'female'}}(\text{Movie} \bowtie \text{StarsIn} \bowtie \text{MovieStar}))$

Select Project Join example

- What are all the titles of movies with female actors?
- Write in Relational Algebra and SQL

Relational algebra:

$\pi_{\text{title}}(\sigma_{\text{gender = 'female'}}(\text{Movie} \bowtie \text{StarsIn} \bowtie \text{MovieStar}))$

SQL

SELECT DISTINCT Title
FROM Movie m, StarsIn s, MovieStar st
WHERE m.MovieID = s.MovieID and s.StarID = st.StarID and gender = ‘female’

Renaming Attributes in Result

- SQL allows renaming relations and attributes using the **as** clause:

  old-name **as** *new-name*

- Example: Find the title of movies and the IDs of all actors in them, and rename "StarID" to "ID"

```
SELECT    Title, StarID AS ID
FROM      StarsIn S, Movie M
WHERE     M.MovieID = S.MovieID
```

Congratulations: You know select-project-join queries

- Very common subset to talk about
  - You saw it in the RA tutorial
- Can do many (but not all) useful things

SQL is *declarative*, not procedural
how do we know? Lets see what
procedural would look like…

Conceptual Procedural Evaluation Strategy

1. Compute the cross-product of *relation-list*.
2. Discard resulting tuples if they fail *qualifications*.
3. Delete attributes that are not in *target-list*.
4. If DISTINCT is specified, eliminate duplicate rows.

Example of Conceptual Procedural Evaluation

SELECT Name

FROM MovieStar M, StarsIn S

WHERE S.StarID = M.StarID AND MovieID = 276

join selection

MovieStar **X** StarsIn

(StarID)	Name	Gender	MovieID	(StarID)	Character
1273	Nathalie Portman	Female	272	1269	Leigh Anne Touhy
1273	Nathalie Portman	Female	273	1270	Mary
1273	Nathalie Portman	Female	274	1271	King George VI
1273	Nathalie Portman	Female	276	1273	Nina Sayers
…	…	…	…	…	…