Introduction to Data Management

Review

Alyssa Pittman
Based on slides by Jonathan Leang, Dan Suciu, et al

Paul G. Allen School of Computer Science and Engineering
University of Washington, Seattle

October 28, 2019
Midterm review

Logistics:

- midterm in class on Wednesday
  - same room, same time, same length
- closed book and device
- allowed 1 page front-and-back of HANDWRITTEN notes
- material through last Wednesday’s lecture is fair game
Recap – Joins

- Join to combine data from different tables
  - Nested-loop semantics
  - Inner join (the most common)
  - Outer joins can preserve information
  - Self join pattern

https://www.dofactory.com/sql/join
Recap - 3-valued logic

Comparisons with null result in *unknown*

false = 0
ture = 1
unknown = .5

Formal definitions:

\[ C_1 \text{ AND } C_2 = \min(C_1, C_2) \]
\[ C_1 \text{ OR } C_2 = \max(C_1, C_2) \]
\[ \text{NOT } C = 1 - C \]
Recap – Aggregate Functions

- We need summaries of data because we are often trying to **make decisions** and **succinctly convey information**
  - SELECT `COUNT(*)` FROM AnimeVideoViews …
  - SELECT `SUM(cost)` FROM CoffeeReceipts …
  - SELECT `AVG(price)` FROM CarDealers …
  - SELECT `MAX(score)` FROM StudentGrades …
  - SELECT `MIN(price)` FROM AveLunchPrices …

\[ \text{AGG(attr)} \] computes \text{AGG} over non-NULL values
\[ \text{AGG(DISTINCT attr)} \] is also possible
Recap: Grouping

**SQL Query:**

\[
\text{SELECT Product, SUM(quantity) FROM Purchases GROUP BY Product HAVING SUM(quantity) > 20}
\]

<table>
<thead>
<tr>
<th>Product</th>
<th>Price</th>
<th>Quantity</th>
<th>Month</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>3</td>
<td>20</td>
<td>Jan</td>
</tr>
<tr>
<td>Bagel</td>
<td>1.50</td>
<td>20</td>
<td>Feb</td>
</tr>
<tr>
<td>Banana</td>
<td>0.5</td>
<td>50</td>
<td>Feb</td>
</tr>
<tr>
<td>Banana</td>
<td>5</td>
<td>10</td>
<td>March</td>
</tr>
<tr>
<td>Apple</td>
<td>4</td>
<td>10</td>
<td>March</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product</th>
<th>SUM(quantity)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bagel</td>
<td>40</td>
</tr>
<tr>
<td>Banana</td>
<td>60</td>
</tr>
</tbody>
</table>
Recap: Semantics

First evaluate the FROM clause
Next evaluate the WHERE clause
Group the attributes in the GROUPBY
Eliminate groups based on HAVING
Sort the results based on ORDER BY
Last evaluate the SELECT clause

FWGHOS™
Recap - General form

```
SELECT  S
FROM  R_1, ..., R_n
WHERE  C_1
GROUP BY  a_1, ..., a_k
HAVING  C_2
ORDER BY  O
```

S, O = any attributes $a_1, ..., a_k$ and/or any aggregates, but no other attributes

C1 = any condition on the attributes in $R_1, ..., R_n$

C2 = any condition on the aggregate expressions and attributes $a_1, ..., a_k$
Recap: The Witnessing Problem

```sql
SELECT P1.Name, MAX(P2.Salary)
FROM Payroll AS P1, Payroll AS P2
WHERE P1.Job = P2.Job
GROUP BY P2.Job, P1.Salary, P1.Name
HAVING P1.Salary = MAX(P2.Salary)
```

<table>
<thead>
<tr>
<th>UserID</th>
<th>Name</th>
<th>Job</th>
<th>Salary</th>
<th>UserID</th>
<th>Name</th>
<th>Job</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>Jack</td>
<td>TA</td>
<td>50000</td>
<td>123</td>
<td>Jack</td>
<td>TA</td>
<td>50000</td>
</tr>
<tr>
<td>123</td>
<td>Jack</td>
<td>TA</td>
<td>50000</td>
<td>345</td>
<td>Allison</td>
<td>TA</td>
<td>60000</td>
</tr>
<tr>
<td>345</td>
<td>Allison</td>
<td>TA</td>
<td>60000</td>
<td>345</td>
<td>Allison</td>
<td>TA</td>
<td>60000</td>
</tr>
<tr>
<td>345</td>
<td>Allison</td>
<td>TA</td>
<td>60000</td>
<td>123</td>
<td>Jack</td>
<td>TA</td>
<td>50000</td>
</tr>
<tr>
<td>567</td>
<td>Magda</td>
<td>Prof</td>
<td>90000</td>
<td>567</td>
<td>Magda</td>
<td>Prof</td>
<td>90000</td>
</tr>
<tr>
<td>567</td>
<td>Magda</td>
<td>Prof</td>
<td>90000</td>
<td>789</td>
<td>Dan</td>
<td>Prof</td>
<td>100000</td>
</tr>
<tr>
<td>789</td>
<td>Dan</td>
<td>Prof</td>
<td>100000</td>
<td>789</td>
<td>Dan</td>
<td>Prof</td>
<td>100000</td>
</tr>
<tr>
<td>789</td>
<td>Dan</td>
<td>Prof</td>
<td>100000</td>
<td>567</td>
<td>Magda</td>
<td>Prof</td>
<td>90000</td>
</tr>
</tbody>
</table>
Recap – RA Operators

- These are all the operators you will see in this class
  - We’ll profile these one at a time

Join \( \bigotimes \)
Cartesian Product \( \times \)
Selection \( \sigma \)
Projection \( \pi \)
Union \( \bigcup \)
Intersection \( \cap \)
Difference \( - \)
Grouping & Aggregation \( \gamma \)
Sort \( \tau \)
Duplicate Elimination \( \delta \)

RA
Extended RA
Recap - RA Equivalencies

```
SELECT P.Name, R.Car
FROM Payroll AS P, Regist AS R
WHERE P.UserID = R.UserID;
```
Recap – Basic SQL to RA Conversion

- The general plan structure for a “flat” SQL query

\[ \pi \tau \sigma \gamma \]

- FROM & WHERE
- GROUP BY & aggregates
- HAVING
- ORDER BY
- SELECT

\[ \sigma \Join \times \cdots \]

Tables
Decorrelation and Unnesting

Correlated

```
SELECT S.sid
FROM Supplier S
WHERE S.state = 'WA' AND
    NOT EXISTS (SELECT *
                 FROM Inventory I
                 WHERE I.sid = S.sid AND
                       I.price > 100);
```

Decorrelated

```
SELECT S.sid
FROM Supplier S
WHERE S.state = 'WA' AND
    S.sid NOT IN (SELECT I.sid
                   FROM Inventory I
                   WHERE I.price > 100);
```
Decorrelation and Unnesting

Nested

```
SELECT S.sid
FROM Supplier S
WHERE S.state = 'WA' AND
  S.sid NOT IN (SELECT I.sid
                FROM Inventory I
                WHERE I.price > 100);
```

Unnesteed

```
(SELECT S.sid
 FROM Supplier S
 WHERE S.state = 'WA')
 EXCEPT
 (SELECT I.sid
  FROM Inventory I
  WHERE I.price > 100)
```
Decorrelation and Unnesting

\[
\left( \text{SELECT} \ S.\text{sid} \\
\text{FROM} \ Supplier \ S \\
\text{WHERE} \ S.\text{state} = 'WA' \right) \\
\text{EXCEPT} \\
\left( \text{SELECT} \ I.\text{sid} \\
\text{FROM} \ Inventory \ I \\
\text{WHERE} \ I.\text{price} > 100 \right)
\]
Some terminology:
- **Facts**: tuples in the database
- **Rules**: queries

```
.decl Actor(...) 
.decl Movie(...) 
.decl Casts(...) 

Actor(123, "Robert", "Downey"). 
Actor(345, "Gal", "Gadot"). 
Movie(000, "Iron Man", 2008). 
Casts(123, 000). 
Casts(123, 999). 
Casts(345, 888).
```

Find all names of movies made in 2012
```
Q1(n) :- Movie(_, n, 2012).
```

Find all actor names cast in movies made in 2012
```
Q2(f, l) :- Movie(mid, _, 2012), 
           Casts(aid, mid), 
           Actor(aid, f, l).
```

Find all actor names cast in movies made in 2008 and 2012
```
Q3(f, l) :- Actor(aid, f, l), Casts(aid, mid1), Movie(mid1, _, 2012), 
           Casts(aid, mid2), Movie(mid2, _, 2008).
```
In general, we return **all values of head variables** such that there **exists existential variable** values described by the query.

\[ Q_2(f, l) : \text{Movie}(\text{mid}, _, 2012), \text{Casts}(\text{aid}, \text{mid}), \text{Actor}(\text{aid}, f, l). \]

\[ Q_2(f, l) = \exists \text{mid}. \exists \text{aid}. \exists x. (\text{Movie}(\text{mid}, x, 2012) \land \text{Casts}(\text{aid}, \text{mid}) \land \text{Actor}(\text{aid}, f, l)) \]
**Fixed-Point Semantics**

- **Main Idea:** Keep executing until no new results are added (until a fixed point is reached).

\[
Q(a, b) \Leftarrow \text{Edge}(a, b).
\]

\[
Q(a, b) \Leftarrow Q(a, x), \text{Edge}(x, b).
\]
Datalog grouping/aggregation: not on midterm!

Khang has volunteered to give preliminary HW4 Datalog feedback (yes/no whether questions were correct)
Find all descendents of Bob who are not descendents of Alice

// compute descendents of x
D(x,y) :- ParentChild(x,y).
D(x,y) :- D(x,z), ParentChild(z,y).

// desc of bob but not alice
Ans(x) :- D('Bob',x), !D('Alice',x).

All possible answers

<table>
<thead>
<tr>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carol</td>
</tr>
<tr>
<td>Dave</td>
</tr>
<tr>
<td>Eve</td>
</tr>
<tr>
<td>Fred</td>
</tr>
<tr>
<td>Gil</td>
</tr>
</tbody>
</table>

All non-answers

<table>
<thead>
<tr>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carol</td>
</tr>
<tr>
<td>Eve</td>
</tr>
<tr>
<td>Fred</td>
</tr>
<tr>
<td>Gil</td>
</tr>
</tbody>
</table>
Unsafe Rules

- We want our query results bound to existing values only
- **Unsafe queries are those queries that do not always have the same finite results.**
  - Domain-dependent answers or
  - Infinite solutions

\[
U_1(x, y) \leftarrow \text{ParentChild("Alice",} x), \ y \neq \text{"Bob"}
\]

Holds for every \( y \) other than “Bob” – infinite!

\[
U_2(x) \leftarrow \text{ParentChild("Alice",} x), \ !\text{ParentChild}(x, y)
\]

We get all \( x \) still – \( y \) is unbound, so there always exists some \( y \) that \( x \) is not parent of

\[
U_3(\text{minId, } y) \leftarrow \text{minId} = \min x : \{ \text{Actor}(x, y, _) \}
\]

Unclear what \( y \) is – like our SQL aggregate
Safe Rules

- Datalog rules are safe if **every variable appears in some positive relational atom**
  - Positive atom implicitly restricts/defines domain of its variables

\[ \text{U1}(x,y) \ :- \ \text{ParentChild}(“Alice”,x), \ \text{ParentChild}(y,\_), \ y \neq “Bob” \]

Now the domain of Bob is restricted
A Datalog program is stratified if it can be partitioned in strata.

Rules in strata 1\ldots n can be used in aggregates or negation in strata n+1.

// compute descendents of x
D(x,y) :- ParentChild(x,y).
D(x,y) :- D(x,z), ParentChild(z,y).

// name and count
Ans(x) :- D('Bob',x), !D('Alice',x).

---

Stratification

- A Datalog program is stratified if it can be partitioned into strata.
- Rules in strata 1\ldots n can be used in aggregates or negation in strata n+1.

---

Stratum 1: D(x,y) :- ParentChild(x,y).
D(x,y) :- D(x,z), ParentChild(z,y).

Stratum 2: Ans(x) :- D('Bob',x), !D('Alice',x).

Depends on:
- Stratum 1 rules on Stratum 2 rules.
These are all the blocks we will learn about:

- Entity set
- Attribute
- Relationship
- Subclass
- Weak Entity

- is a
**Relation Multiplicity**

- **One-to-one**
- **Many-to-one**
- **Many-to-many**

```sql
CREATE TABLE Company (  
    name VARCHAR(100) PRIMARY KEY,  
    ...);  
CREATE TABLE Product (  
    name VARCHAR(100) PRIMARY KEY,  
    cname VARCHAR(100) REFERENCES Company  
    ...);  
```

- **Product**
  - Beyblade, ...
  - Trolls, ...
- **Company**
  - Hasbro, ...
  - Nyform, ...

Product makes Company
Data Interrelationships

A **Functional Dependency** $A_1, \ldots, A_m \rightarrow B_1, \ldots, B_n$ holds in the relation $R$ if:

$$\forall t, t' \in R, (t.A_1 = t'.A_1 \land \ldots \land t.A_m = t'.A_m \rightarrow t.B_1 = t'.B_1 \land \ldots \land t.B_n = t'.B_n)$$

Informally, some attributes determine other attributes.

Warning! Dependency does not imply causation!
Closure

The **Closure** of the set \( \{A_1, \ldots, A_m\} \), written as \( \{A_1, \ldots, A_m\}^+ \), is the set of attributes \( B \) such that \( A_1, \ldots, A_m \to B \).

A closure finds **everything** a set of attributes determines.

Closure (example)

Given the functional dependencies:
- \( SSN \to Name \)
- \( Name \to Initials \)

We can derive some closures:
- \( Name^+ = \{Name, Initials\} \)
- \( SSN^+ = \{SSN, Name, Initials\} \)
- \( Initials^+ = \{Initials\} \)
- \( \{SSN, Initials\}^+ = \{SSN, Name, Initials\} \)
Finding Keys

**Superkey**

A **Superkey** is a set of attributes \( A_1, ..., A_n \) s.t. for any single attribute \( B \):

\[
A_1, ..., A_n \rightarrow B
\]

In other words, for the set of all attributes \( C \) in the relation \( R \), the set \( \{A_1, ..., A_n\} \) is a superkey iff \( \{A_1, ..., A_n\}^+ = C \)

**Key**

A **Key** is a minimal superkey, i.e. no subset of a key is a superkey.

**Candidate Key**

When a relation has multiple keys, each key is a **Candidate Key**.
A relation \( R \) is in *Boyce-Codd Normal Form (BCNF)* if for every non-trivial dependency, \( X \rightarrow A \), \( X \) is a superkey.

Equivalently, a relation \( R \) is in BCNF if \( \forall X \) either \( X^+ = X \) or \( X^+ = C \) where \( C \) is the set of all attributes in \( R \).
Decomposition

- “Extracting” attributes can be done with **decomposition** (split the schema into smaller parts)
- For this class, decomposition means the following:

\[
R(A_1, \ldots, A_n, B_1, \ldots, B_m, C_1, \ldots, C_k) < R_1(A_1, \ldots, A_n, B_1, \ldots, B_m) < R_2(A_1, \ldots, A_n, C_1, \ldots, C_k)
\]

Some common attributes are present so we can rejoin data.
Last winter’s midterm (solutions), final (solutions)

- note that some topics moved in the quarter, so not all questions are applicable to us