CompSci 516
Data Intensive Computing Systems

Lecture 24
Review and Wrap-up

Instructor: Sudeepa Roy
Disclaimer

• Only a subset of the topics
• Not an indicator of what can appear in the final exam
• Topics covered may not appear
• Topics not covered may appear
• If we have time at the end, we can revisit any topic in the course
SQL, RA, RC, Datalog
RC

- Student(sid, name)
- Enrollment(sid, cid, points)

- Find names of all students who never scored less than 90 points in any course
- i.e. any course s/he took, s/he scored $\geq 90$
TRC (in class)

• Student(sid, name)
• Enrollment(sid, cid, points)

• Find names of all students who never scored less than 90 points in any course
• i.e. any course s/he took, s/he scored >= 90

\[ \exists n \mid \exists s \in \text{Student} \{ (n.\text{name} = s.\text{name}) \}
\land \neg \exists e \in \text{Enrollment} \exists c \{ (e.\text{sid} = s.\text{sid}) \land (e.\text{points} < 90) \} \]
TRC

- Student(sid, name)
- Enrollment(sid, cid, points)

- Find names of all students who never scored less than 90 points in any course
- i.e. any course s/he took, s/he scored $\geq 90$

$$\exists N \mid \exists S \in \text{Student} \quad N.name = S.name$$
• Student(sid, name)
• Enrollment(sid, cid, points)

• Find names of all students who never scored less than 90 points in any course
• i.e. any course s/he took, s/he scored >= 90

\[
\exists N \mid \exists S \in \text{Student} \quad n, name = S, name \\
\wedge \left[ \forall E \in \text{Enrollment} \left( \begin{array}{c}
S, sid = E, sid \\
\wedge
(E, points < 90)
\end{array} \right) \right]
\]
TRC (in class)

- Student(sid, name)
- Enrollment(sid, cid, points)

- Find names of all students who never scored less than 90 points in any course
  - i.e. any course s/he took, s/he scored >= 90

\[
\{ N \mid \exists S \in \text{Student} \quad \text{n.name} = S.\text{name} \\
\quad \land \quad \land \exists E \in \text{Enrollment} \quad (S.\text{sid} = E.\text{sid}) \\
\quad \land \quad \land \exists E \in \text{Enrollment} \quad (E.\text{points} \geq 90) \}
\]

why does not "\&" = AND work here?
TRC

- Student(sid, name)
- Enrollment(sid, cid, points)

Find names of all students who never scored less than 90 points in any course

i.e. any course s/he took, s/he scored >= 90

\[ \{ N \mid \exists S \in \text{Student} \quad N.name = S.name \wedge \left( \forall E \in \text{Enrollment} \quad (S.sid = E.sid) \implies (\exists points \geq 90) \right) \} \]
Two are equivalent

∀ x ∈ (x ≠ y) = ¬ (∃ x ∈ (x = y))
RA (in class)

- Student(sid, name)
- Enrollment(sid, cid, points)
- Find names of all students who never scored less than 90 points in any course

\[
\begin{align*}
\text{\{ } & \text{\forall s \in Stu \ s.t. \ N.name = s.name} \nonumber \\
& \land \left( \forall e \in \text{Enroll} \left( (s, s.id = e, s.id) \right) \right) \nonumber \\
& \land \left( \forall e \in \text{Enroll} \left( (e, \text{points} < 90) \right) \right) \nonumber \\
\text{\}} \nonumber
\end{align*}
\]
Remember

- You can write RA expressions or logical query plan
- For “-” or “union” or “intersection”, both sides have to have the same schema
Datalog (in class)

- Student(sid, name)
- Enrollment(sid, cid, points)
- Find names of all students who never scored less than 90 points in any course

\[
\{ \text{N} \mid \exists \text{S} \in \text{Stu} \quad \text{N} \cdot \text{name} = \text{S} \cdot \text{name} \wedge \forall \text{E} \in \text{Enroll} \quad (\text{S} \cdot \text{sid} = \text{E} \cdot \text{sid}) \wedge (\text{E} \cdot \text{points} < 90) \} \]

\[\text{NotSoGood(name)} : - \quad \text{Enrollment}(\text{sid}, \text{cid} \quad \text{points} < 90),\]

\[\text{Good(name)} : - \quad \neg \text{NotSoGood(name)}, \text{Student}(\text{sid}, \text{name})\]
Datalog (in class)

Better soln.

\[
\{ n \mid \forall s \in \text{Stu} \ n.\text{name} = s.\text{name} \\
\land \forall e \in \text{Enroll} \ (s, s.\text{id} = e, \text{sid}) \\
\land (e.\text{points} < 90) \} 
\]

- Student(sid, name)
- Enrollment(sid, cid, points)
- Find names of all students who never scored less than 90 points in any course

\begin{align*}
\text{NotSoGood}(\text{sid}) & : - \text{Enrollment}(\text{sid}, \text{cid}, \text{points}), \\
& \quad \text{points} < 90
\end{align*}

\begin{align*}
\text{Good}(\text{name}) & : - \lnot \text{NotSoGood}(\text{sid}) \\
\text{Student}(\text{sid}, \text{name})
\end{align*}
Remember:
Safe Datalog rules for negation

• Any variable that appears under a negated atom (relation name)
• MUST appear under a positive atom without negation as well
• Here “sid” is under NOT, but appears in “Student” too
• So safe and domain-independent
• Also, Datalog did not need same schema for negation as in RA
SQL (in class) – from RA

- Student(sid, name)
- Enrollment(sid, cid, points)
- Find names of all students who never scored less than 90 points in any course

```
(Select name 
 From Student) 
Except 
(Select name 
 From Student S, Enrollment E 
 Where 
 S.sid = E.sid and 
 E.points < 90)
```
• Student(sid, name)
• Enrollment(sid, cid, points)
• Find names of all students who never scored less than 90 points in any course

```sql
select name
from Student S
where not exists (
    select *
    from Enrollment E
    where S.sid = E.sid
    and E.points < 90
)
```
More SQL solutions in Practice Problem 3

- Student(sid, name)
- Enrollment(sid, cid, points)
- Find names of all students who never scored less than 90 points in any course

- NOT EXISTS
  - Checks if the relation is empty
- NOT IN
  - Same schema
- ALL
  - One attribute in select in nested subquery
- HAVING MIN
  - Useful sometimes when you want to avoid nested query
Remember: NULL and 3-valued logic

- $A = \text{NULL}$: wrong
- $A \text{ IS NULL}$: correct

- $\text{NULL} = \frac{1}{2}$, $V = \text{max}$, $\land = \text{min}$, $\text{True} = 1$, $\text{False} = 0$

- $A = \text{UNKNOWN}$, $B = \text{TRUE}$, $C = \text{FALSE}$

\[
(A \land B) \lor C = (\text{TRUE} \lor \text{TRUE}) \lor \text{FALSE} = \text{TRUE} \lor \text{FALSE} = \text{Unknown}
\]
Remember: Order of SQL Evaluation

- Select...
- From...
- Where...
- Group by..
- Having

The condition must hold for the group.
Index
Choices in Index

- k
- search key
- k
- data entry (rid)
- hash
- B⁺ tree

Two levels of page access

Cl → ⌈\frac{N}{M}⌉
Uncl → ⌈\sqrt{N}⌉

N = # satisfying tuple
M = page size
Normal Form
BCNF and 3NF

\[ BC \rightarrow D \]

1. \( A \in X \) e.g. \( AB \rightarrow A \) trivial
2. \( X^+ = \) all attributes \( \equiv X \) is a superkey

3. \( A \) is part of some key (minimal)
Revisit

• Why do we want normalization
  – Or DO NOT WANT it?
• Lossless join
• BCNF decomposition
UNDO log
REDO log
UNDO-REDO log
\[ x = u - \text{old} \]

**UNDO**

\[ (T, x, u) \]

\[ x = v \]

\[ \langle \text{COMMIT} \ T \rangle \]

**REDO**

\[ (T, x, u, v) \]

\[ x = v \]

\[ \langle \text{COMMIT} \ T \rangle \]

\[ \langle \text{COMMIT} \ T \rangle \]

**UNDO/REDO**
UNDO

\[ X = u - \text{old} \]
\[ T : X = u - \text{new} \]

\[ \langle T, X, u \rangle \]

UNDO/REDO

\[ \langle T, X, u, v \rangle \]

UNDO/REDO

\[ \langle T, X, u, v \rangle \]

REDO

\[ \langle T, X, u \rangle \]

\[ \langle \text{COMMIT T} \rangle \]

\[ \langle \text{COMMIT T} \rangle \]

STEAL VS. NO STEAL?
FORCE VS. NO FORCE?

Which combination is good?
\[ x = u - \text{old} \]
\[ T : x = u - \text{new} \]

**UNDO**

\[ <T, x, u> \]
\[ x = u \]
\[ <\text{COMMIT } T> \]

**REDO**

\[ <T, x, u> \]
\[ <\text{COMMIT } T> \]
\[ x = u \]

**UNDO/REDO**

\[ <T, x, u, u> \]
\[ <\text{COMMIT } T> \]
\[ x = u \]

- ✔ STEAL
- ✗ NO STEAL
- ✔ NO FORCE

\[ x = \text{bad for performance} \]
\[ ✔ = \text{good} \]

- ✔ STEAL
- ✔ NO FORCE

\[ \text{why is this combo good?} \]
Wait until T2 and T3 commit or abort.

(If committed, their writes are written to disk by UNDO log policy.)

Write all dirty buffers to disk.

Everything updated before START CKPT are written to disk for sure.
UNDO

..........  
<COMMIT T1>
<START CKPT T2, T3>

Wait until T2 and T3 Commit or Abort
(if committed, their Writes are written to Disk by UNDO log policy)

<END CKPT>

CRASH!!!
UNDO UNCOMMITTED ones inside or after START CKPT

REDO

..........  
<COMMIT T1>
<START CKPT T2, T3>

Write changes by Committed trans. e.g. T1 To disk

<END CKPT>

CRASH!!!
REDO COMMITTED ones Inside or after START CKPT

<END CKPT>

UNDO/REDO

..........  
<COMMIT T1>
<START CKPT T2, T3>

Write all dirty buffers to disk
(Everything updated before START CKPT are written to Disk for sure But some extra changes may Be written too)

<END CKPT>

Write all dirty buffers to disk
UNDO UNCOMMITTED ones REDO COMMITTED ones inside or after START CKPT
Remember

- UNDO update: backward
- REDO update: forward
- ARIES is a specific implementation of UNDO/REDO log
Conclusions
(of 516, Fall 2016)
Take-Aways

• DBMS Basics

• DBMS Internals

• Overview of Research Areas

• Hands-on Experience in DB systems

• Some experience in data-related research projects
DB Systems

• Traditional DBMS
  – Postgres, SQL

• Large-scale Data Processing Systems
  – Spark/Scala, AWS

• New DBMS/NOSQL
  – MongoDB

• In addition
  – XML, JSON, JDBC, Python/Java
DB Basics

• SQL
• RA/Logical Plans
• RC
• Datalog
  – Why we needed each of these languages

• Normal Forms
DB Internals and Algorithms

• Storage
• Indexing
• Operator Algorithms
  – External Sort
  – Join Algorithms
• Cost-based Query Optimization
• Transactions
  – Basics
  – Concurrency Control
  – Recovery
Large-scale Processing and New Approaches

• Parallel DBMS
• Distributed DBMS
• Map Reduce
• NOSQL and column-store
Advanced/Research Topics

(in various levels of details)

• Data Warehouse/OLAP/Data Cube
• View Maintenance
• Association Rule Mining
• Acyclic query and join processing
• Worst case optimal join algorithms
• Data integration
Announcements
Syllabus for the final

• **Everything** that has been covered in class is included
  – Unless the slide is marked as "optional"
• **Every topic has the same weight**
  – no lower weights for topics before midterm
• **The problems will be of similar nature as that in the midterm**
  – i.e. you have to know the material well to solve the problems
  – Also there may be some short Q/A
  – keep asking questions on piazza!
• **Lectures 1 to 18 (up to NOSQL) and Lecture 21 (Datalog)**
  you need to know all the details
  – Advanced topics Lecture 19, 20, 22, 23 -- you should have an idea about these areas as covered in the lecture slides
  – again, optional slides are not included
Project Presentation on Friday

• about 75 mins for 10 presentations
• Prepare a talk for 6 mins
  – no group takes more than 8 mins including Q/A
• Have 5-6 slides
  – But depends on the density of the slide
  – The problem, motivation, approach, why your solution is novel or interesting, some interesting results, a demo or a snapshot

• Ask questions to other groups
• We start at 4:40 pm sharp
  – I will send the schedule by tomorrow
• Come early and make sure your laptop connects
HW and Project Report

• All HWs have been graded
  – if you have any question, contact Jung and resolve asap

• Upload your final project report by Friday night
  – the final version will be considered for grading
Final Exam

• Monday, Dec 19
• In class – LSRC A247
• 9:00 am to 12:00 noon

• Office hour schedule may vary in the reading period
• Sudeepa’s office hour
  – Regular one next Monday Dec 5
  – NO office hour Monday Dec 12
  – Office hour Friday Dec 16 afternoon (tentatively from 2:30 to 5:30 pm)
  – Post questions on Piazza during the reading period
  – Answer questions that you know
ACM SIGMOD Student Research Competition Deadline Today 8 pm!

- Just 2 pages – abstract only
- If you want to quickly submit any of your research projects
- Two tracks for undergraduates and Graduates
- Undergraduates can submit as a team, grads cannot
- But if you think your research can be submitted as a conference/workshop paper soon, DO NOT submit it to the competition
  - parallel submissions are not allowed
Course Evaluations are Now Open

• On DukeHub
• Please submit your evaluation soon!
• We need to hear from each of you about your feedback on this course
• It is very important for the course staff to know whether we succeeded in helping you learn the material
  – so that we can improve the course structure in future offerings and do a better job in the future
That’s all!

Hope some of you will further explore Database Systems/Data Management/Data Analysis/Big Data...

In the last class on Friday, we will hear about your work 😊

All the best!