

EECS 584: Advanced Database Systems

Prof. Kristen LeFevre
klefevre [at] umich.edu
4705 CSE

9/8/09

EECS 584, Fall 2009

1

Course Overview

- Regular class meetings:
 - MW 1:30-3 PM 3427 EECS
 - *Note: Class starts at 1:40 on Michigan time*
- Reserve discussion period (F 2:30-3:30)
- Office Hours:
 - W 3:30-5, and by appointment

9/8/09

EECS 584, Fall 2009

2

Course Overview

- Advanced topics in data management
 - Advanced concurrency control
 - Query processing and optimization
 - Advanced indexing methods
 - Parallel & distributed databases
 - Next-generation data models
 - Data mining & analysis
 - Data on the web
 - Topics in data privacy & security

9/8/09

EECS 584, Fall 2009

3

Course Overview

- Also, practice important research skills:
 - Reading & critically evaluating original research papers
 - Communicating technical material, orally and in written form
 - Small-scale original research project
- Course Website:
www.eecs.umich.edu/~klefevre/eecs584
- Prerequisites: EECS 484 (Introductory Databases), equivalent coursework, or instructor's permission

9/8/09

EECS 584, Fall 2009

4

Papers & Reviews

- This is a paper-reading course
 - No official textbook; papers available on course website
 - Background / reference: Database Management Systems (3rd Edition) by Ramakrishnan & Gehrke
- Will cover 1-2 papers per class
- Read paper and post a short *reaction* by 8 PM the night before class
 - Reactions should be posted to Ctools Discussion
 - See course website for instructions

9/8/09

EECS 584, Fall 2009

5

Seminar Format

- Typical class meeting:
 - Student paper presentation (40 minutes)
 - Short break
 - Student-led discussion (30 minutes)
- Paper presentation goals:
 - Motivate the paper, provide background
 - Highlight key contributions
 - Explain important technical points
 - examples are great!
 - *Beware of presenting too much technical detail!*

9/8/09

EECS 584, Fall 2009

6

Seminar Format

- Student presenters must read designated paper and prepare slides by *Friday* the week before presentation
 - Use template on course website
- **Mandatory** meeting with instructor to go over slides and get feedback
 - Friday (2:30-3): student(s) presenting the next Monday
 - Friday (3-3:30): student(s) presenting the next Wednesday
- Will post your slides on the web after class

9/8/09

EECS 584, Fall 2009

7

Seminar Format

- Discussion Format
 - Another student in charge of leading discussion
 - Provide structure
 - Be prepared with key questions from the paper
- Examples of good discussion questions:
 - Locking and optimistic concurrency control can both be used to provide transactional semantics. When would it be better to use locking? What would it take for a DBMS to implement both kinds of concurrency control?
 - I thought the experimental evaluation missed some key points. In particular, it did not address system throughput, which is important in the real world. If we were to redo this experimental study, what would we do differently?
- **All students are expected to participate in discussion!**

9/8/09

EECS 584, Fall 2009

8

Course Project

- Major component of the course
- Small-scale original research project!
- Opportunity to study a database-related problem in depth
- More specific details and suggested topics coming next week

9/8/09

EECS 584, Fall 2009

9

Grading

Midterm Exam 30%	Open-book exam <i>Tentatively</i> November 23 1:30-3:30
Project 40%	Based on final paper and presentation / demo
Paper Reactions 10%	Spot-check (S/U)
Class Participation 20%	Paper presentations Leading and participating in discussion

9/8/09

EECS 584, Fall 2009

10

Logistics

- By *Friday*, view course schedule, and send instructor an ordered list of topics for which you want to present or lead discussion
 - Can't make any guarantees, but will try to honor requests
- Student presentations begin September 23
 - Will notify students at least 1 week in advance
 - Schedule will be finalized after add/drop deadline
 - Max = 2 presentations per student
- First paper: Codd "A relational data model..."
 - Review due Sunday, 8PM

9/8/09

EECS 584, Fall 2009

11

Introduce Another Student

- For the next 5 minutes, with the person sitting next to you...
 - What is your name?
 - Where are you from?
 - What is your affiliation / program at UM?
 - What is your background in data management?
 - What do you hope to get out of this course?

9/8/09

EECS 584, Fall 2009

12

Warp-Speed Review

Suggested Background Reading: R&G Chapters 1,3,4,5

- What is a database?
 - A very large integrated collection of data
- A Database Management System (DBMS) is a software package designed to store and manage databases
- Why use a DBMS?
 - Data independence and efficient access
 - Reduced application development time
 - Data integrity and security
 - Uniform data administration
 - Concurrent access, crash recovery

9/8/09

EECS 584, Fall 2009

13

Data Independence

- Applications insulated from how data is structured and stored.
- One of the important motivations for relational model (see Codd's paper)

9/8/09

EECS 584, Fall 2009

14

Relational Data Model

- A *data model* is a collection of concepts for describing data
- A *schema* is a description of a particular collection of data, using the given data model
- The *relational model* is the most widely used today
 - Main concept: *relation*, basically a table with rows and columns.
 - Every relation has a *schema*, which specifies the name of the relation, name and type (domain) of each column.
 - Can think of a relation as a set of rows or *tuples*. (i.e., all rows are distinct)

9/8/09

EECS 584, Fall 2009

15

Example Instance of Students Relation

Specifies schema

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Order of rows is not important

All rows are distinct

Sets (classic relational) vs. Multisets (SQL)

9/8/09

EECS 584, Fall 2009

16

Integrity Constraints

- Key constraints
 - *Minimal subset of attributes* is a unique identifier for a tuple
 - One key identified as *primary key*
- Foreign key constraints
 - Information stored in one relation linked to another relation
 - Primary key attributes reference foreign key attributes
- General constraints

9/8/09

EECS 584, Fall 2009

17

Relational Query Languages

- A major strength of the relational model: supports simple, powerful *querying* of data.
- Queries can be written intuitively, and the DBMS is responsible for efficient evaluation.
 - *Query optimizer* extensively re-orders operations, but ensures that the answer does not change.
- Relational model supports powerful QLs:
 - Formal foundation based on logic.
 - Allows for much optimization.
- Query Languages **!=** programming languages!
 - QLs not expected to be "Turing complete"

9/8/09

EECS 584, Fall 2009

18

Formal Relational Query Languages

- Two mathematical Query Languages form the basis for “real” languages (e.g. SQL), and for implementation:
 - Relational Calculus:** Lets users describe what they want, rather than how to compute it. (Non-operational, *declarative*.)
 - Relational Algebra:** More *operational*, very useful for representing execution plans.

9/8/09

EECS 584, Fall 2009

19

Preliminaries

- A query is applied to *relation instances*, and result also a relation instance.
 - Schemas of input* relations are fixed
 - The *schema for the result* of a given query is also *fixed!* Determined by definition of query language constructs.

9/8/09

EECS 584, Fall 2009

20

Relational Calculus

- Declarative** - Specify what the query should return, not how to compute it
- Two flavors: Tuple relational calculus, Domain relational calculus

9/8/09

EECS 584, Fall 2009

21

Tuple Relational Calculus

- Variables take tuples as values
- TRC queries of the form: $\{T \mid p(T)\}$, where T is a tuple variable, and p(T) is a boolean formula describing T
- What does this query return?
 - The set of all tuples *t* for which *p(t)* evaluates to True
 - The language for writing p() is a subset of first-order logic

9/8/09

EECS 584, Fall 2009

22

Tuple Relational Calculus

- Let *Rel* be a relation, *R* and *S* be tuple variables, *a* and *b* be attributes
- Let *op* be in the set $\{<, >, =, \geq, \leq, \neq\}$
- An *atomic formula* is one of:
 - $R \in \text{Rel}$
 - $R.a \text{ op } S.b$
 - $R.a \text{ op constant}$, or $\text{constant op } R.a$
- A *formula* is recursively defined using logical operators $\wedge, \vee, \neg, \Rightarrow$ and quantifiers \exists, \forall

9/8/09

EECS 584, Fall 2009

23

TRC Examples

- Suppose we have three relations
 - Sailors(sid, sname, rating, age)
 - Boats(bid, bname, color)
 - Reserves(sid, bid, day)
- Find names and ages of sailors with rating above 7

$$\{P \mid \exists S \in \text{Sailors}(S.\text{rating} > 7 \wedge P.\text{sname} = S.\text{sname} \wedge P.\text{age} = S.\text{age})\}$$
- Find names of sailors who have reserved a red boat

$$\{P \mid \exists S \in \text{Sailors} \exists R \in \text{Reserves} (R.\text{sid} = S.\text{sid} \wedge P.\text{sname} = S.\text{sname} \wedge \exists B \in \text{Boats} (B.\text{bid} = R.\text{bid} \wedge B.\text{color} = \text{'red'}))\}$$

9/8/09

EECS 584, Fall 2009

24

Relational Algebra

- **Operational** - Specify how to compute the query result using a well-defined set of operators
- Codd's Theorem - Any query expressed in relational calculus can be expressed in relational algebra (and visa versa)

9/8/09

EECS 584, Fall 2009

25

Relational Algebra

Basic operations:

- **Selection** (σ) Selects a subset of rows from relation.
- **Projection** (π) Deletes unwanted columns from relation.
- **Cross-product** (\times) Allows us to combine two relations.
- **Set-difference** ($-$) Tuples in reln. 1, but not in reln. 2.
- **Union** (\cup) Tuples in reln. 1 and tuples in reln. 2.

Additional operations:

- Intersection, *join*, division, renaming: Not essential, but (very!) useful.

Since each operation returns a relation, **operations can be composed** (Algebra is "closed").

9/8/09

EECS 584, Fall 2009

26

Projection

- Deletes attributes that are not in *projection list*.
- **Schema** of result contains exactly the fields in the projection list
- Projection operator has to eliminate *duplicates!*
 - *Note: real systems typically don't do duplicate elimination unless the user explicitly asks for it.*

sname	rating
yuppy	9
lubber	8
guppy	5
rusty	10

$\pi_{sname, rating}(S)$

age
35.0
55.5

$\pi_{age}(S)$

9/8/09

EECS 584, Fall 2009

27

Selection

- Selects rows that satisfy *selection condition*.
- No duplicates in result!
- **Schema** of result identical to schema of input relation.
- **Result** relation can be the *input* for another relational algebra operation!

sid	sname	rating	age
28	yuppy	9	35.0
58	rusty	10	35.0

$\sigma_{rating > 8}(S)$

sname	rating
yuppy	9
rusty	10

$\pi_{sname, rating}(\sigma_{rating > 8}(S))$

9/8/09

EECS 584, Fall 2009

28

Union, Intersection, Set-Difference

sid	sname	rating	age
22	dustin	7	45.0
44	guppy	5	35.0
28	yuppy	9	35.0

S1

sid	sname	rating	age
31	lubber	8	55.5
58	rusty	10	35.0

S2

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0
44	guppy	5	35.0
28	yuppy	9	35.0

S1 \cup S2

- All of these operations take two input relations, which must be *union-compatible*

What about duplicates?

9/8/09

EECS 584, Fall 2009

29

Cross-Product

- S1 x R1: Each row of S1 is paired with each row of R1.

sid	sname	rating	age
22	dustin	7	45.0
31	lubber	8	55.5
58	rusty	10	35.0

x

sid	bid	day
22	101	10/10/96
58	103	11/12/96

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96
22	dustin	7	45.0	58	103	11/12/96
31	lubber	8	55.5	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
58	rusty	10	35.0	22	101	10/10/96
58	rusty	10	35.0	58	103	11/12/96

9/8/09

EECS 584, Fall 2009

30

Joins

- **Condition Join:** $R \bowtie_c S = \sigma_c(R \times S)$
- **Equi-Join:** A special case of condition join where the condition c contains only **equalities**.

sid	sname	rating	age	sid	bid	day
22	dustin	7	45.0	22	101	10/10/96
31	lubber	8	55.5	58	103	11/12/96
50	rusty	10	35.0			

$\bowtie_{sid=sid}$

(sid)	sname	rating	age	(sid)	bid	day
22	dustin	7	45.0	22	101	10/10/96

9/8/09

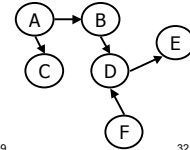
EECS 584, Fall 2009

31

Limitations of Relational Algebra

- **Transitive Closure**
 - e.g., If relation represents direct flights from airport x to airport y , transitive closure answers question "Is it possible to get from x to y (in any number of flights)?"
- For any particular instance of Edges, there is a query to compute transitive closure
 - What is it?
- There's no RA expression for transitive closure in arbitrary instance of Edges
 - Why not?

From	To
A	B
A	C
B	D
D	E
F	D



9/8/09

EECS 584, Fall 2009

32

The SQL Query Language

- Developed by IBM (system R) in the 1970s
- Need for a standard since it is used by many vendors
- Standards:
 - SQL-86
 - SQL-89 (minor revision)
 - SQL-92 (major revision)
 - SQL-99 (major extensions, current standard)

9/8/09

EECS 584, Fall 2009

33

The SQL Query Language

- To find all 18 year old students, we can write:

```
SELECT *
FROM Students S
WHERE S.age=18
```

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2

- To find just names and logins, replace the first line:

```
SELECT S.name, S.login
```

9/8/09

EECS 584, Fall 2009

34

The SQL Query Language

- What does the following query compute?
- ```
SELECT S.name, E.cid
FROM Students S, Enrolled E
WHERE S.sid=E.sid AND E.grade="A"
```

Given the following instances of Enrolled and Students:

| sid   | name  | login      | age | gpa |
|-------|-------|------------|-----|-----|
| 53666 | Jones | jones@cs   | 18  | 3.4 |
| 53688 | Smith | smith@ee   | 18  | 3.2 |
| 53650 | Smith | smith@math | 19  | 3.8 |

| sid   | cid         | grade |
|-------|-------------|-------|
| 53831 | Carnatic101 | C     |
| 53831 | Reggae203   | B     |
| 53650 | Topology112 | A     |
| 53666 | History105  | B     |

we get:

| S.name | E.cid       |
|--------|-------------|
| Smith  | Topology112 |

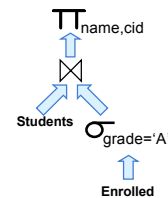
9/8/09

EECS 584, Fall 2009

35

## Relational Algebra "Plan"

```
SELECT S.name, E.cid
FROM Students S, Enrolled E
WHERE S.sid=E.sid AND E.grade="A"
```



9/8/09

EECS 584, Fall 2009

36

## The SQL Query Language

- Many more complex queries
  - See R&G Chapter 5
- Aggregates, GROUP BY, HAVING
  - Important for data cube paper (coming up)
- Nested Queries
- Set Operations

9/8/09

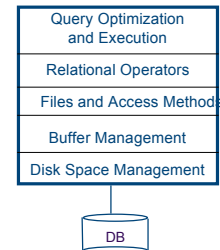
EECS 584, Fall 2009

37

## Database Management Systems (DBMSs)

*Software packages designed to store and manage databases*

- A typical DBMS has a layered architecture.
- The figure does not show the concurrency control and recovery components.



9/8/09

EECS 584, Fall 2009

38

## Data on External Storage

- *Typically, databases are assumed to be larger than main memory.*
- **Disks:** Can retrieve random page at fixed cost
  - But reading several consecutive pages is much cheaper than reading them in random order
- **File organization:** Method of arranging a file of records on external storage.
  - Record id (rid) is sufficient to physically locate record
  - Indexes are data structures that allow us to find the record ids of records with given values in *index search key* fields
- **Architecture:** Buffer manager stages pages from external storage to main memory buffer pool. File and index layers make calls to the buffer manager.

9/8/09

EECS 584, Fall 2009

39

## Buffer Management

- Data read into memory for processing, and written to disk for persistent storage by layer called the buffer manager
- Buffer manager maintains a pool of pages in memory
- Buffer manager implements its own page replacement (separate from OS's virtual memory, file system buffer)
  - Why?
  - We'll return to this later in the semester

9/8/09

EECS 584, Fall 2009

40

## Alternative File Organizations

Many alternatives exist, *each ideal for some situations, and not so good in others:*

- **Heap (random order) files:** Suitable when typical access is a file scan retrieving all records.
- **Sorted Files:** Best if records must be retrieved in some order, or only a 'range' of records is needed.
- **Indexes:** Data structures to organize records via trees or hashing.
  - Like sorted files, they speed up searches for a subset of records, based on values in certain ("search key") fields
  - Updates are much faster than in sorted files.

9/8/09

EECS 584, Fall 2009

41

## Indexes

- An *index* on a file speeds up selections on the *search key fields* for the index.
  - Any subset of the fields of a relation can be the search key for an index on the relation.
- An index contains a collection of *data entries*, and supports efficient retrieval of all data entries  $k^*$  with a given key value  $k$ .
- Index Types from 484
  - B+ Trees
  - Hash-based Indexes
- *This semester -- Advanced indexing techniques*

9/8/09

EECS 584, Fall 2009

42

## Query Optimization

- Given a SQL query, how do we evaluate it efficiently?
- Query Optimizer** – Important component of a DBMS
  - Convert SQL query *blocks* to extended relational algebra expressions
  - Enumerate alternative evaluation plans
  - Choose a plan based on estimated cost

9/8/09

EECS 584, Fall 2009

43

## Query Evaluation Plan

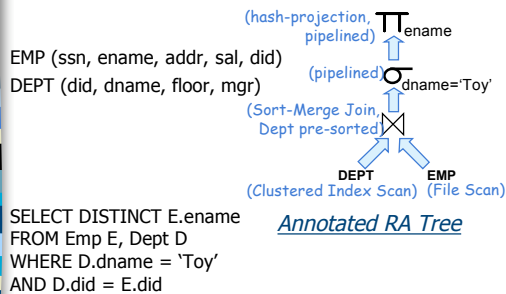
- Annotated, extended RA tree
- Operator Interface: open(), getNext(), close()
- Intermediate Results (multiple ops):
  - Pipelined**: Tuples resulting from one operator fed directly into the next
  - Materialized**: Create a temporary table to store intermediate results

9/8/09

EECS 584, Fall 2009

44

## Query Plan Example



9/8/09

EECS 584, Fall 2009

45

## Transactions

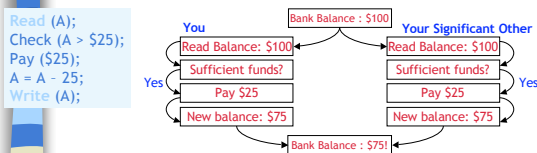
- Foundation for concurrent execution and recovery in DBMS
- Transaction is an **atomic** unit of work
  - E.g., Debit \$500 from my bank account
- Transaction consists of multiple actions
- For performance, DBMS can **interleave** actions from different transactions
- Must guarantee same result as executing transactions **serially**

9/8/09

EECS 584, Fall 2009

46

## Example - Concurrent Execution



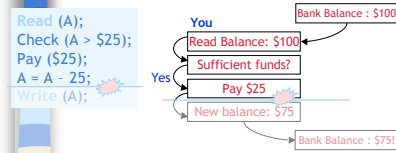
- Interleaving actions of different transactions can cause inconsistency
- DBMS should provide users an illusion of a single-user system
- Could insist on admitting only one transaction at a time
  - Lower utilization: CPU / IO overlap
  - Long running queries starve other queries, reduce overall response time

9/8/09

EECS 584, Fall 2009

47

## Example - Crash Recovery



- DBMS must also guarantee that changes made by partially completed transactions are not seen by other transactions

9/8/09

EECS 584, Fall 2009

48

## The ACID Properties

- **A**tomicity: All actions in the Xact happen, or none happen.
- **C**onsistency: Consistent DB + consistent Xact  $\Rightarrow$  consistent DB
- **I**solation: Execution of one Xact is isolated from that of other Xacts.
- **D**urability: If a Xact commits, its effects persist.

9/8/09

EECS 584, Fall 2009

49

## Summary

- Quick review of the basics
- DBMSs are software packages that manage and provide access to large amounts of data
- Key Ideas
  - Data Models (e.g., relational)
  - Declarative query languages
  - System Architecture (e.g., file organization, buffer management, indexing, query optimization)
  - Transactions, Concurrency Control & Recovery

9/8/09

EECS 584, Fall 2009

50