EECS 584:
Advanced Database Systems

Prof. Michael Cafarella
michjc [at] umich.edu
4709 CSE

Slides thanks to K. LeFevre

Course Overview

- Regular class meetings:
  - MW 1:30-3 PM 2166 DOW
  - Note: Class starts at 1:40 on Michigan time
- Reserve discussion period
  (F 2:30-3:30), though we often won’t use it
- Office Hours:
  - W 3:00-4, and by appointment

Course Overview

- Advanced topics in data management
  - Indexing and storage
  - Query processing and optimization
  - Query execution
  - Transactions and concurrency
  - Next-generation data models
  - Data mining & analysis
  - Parallel Databases
  - Web search, data integration, probabilistic databases, privacy, eScience, …

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/~michjc/eecs584
- Prerequisites: EECS 484 (Introductory Databases), equivalent coursework, or instructor’s permission

Papers & Reviews

- You will read 1-2 papers per class
  - No official text; papers available on course website
- Read paper and post a 300-400 word reaction by 8 PM the night before class
  - What problem is addressed? Why important?
  - 1-2 main technical contributions? Describe.
  - 1-2 weaknesses or open questions? Describe and discuss.

Seminar Format

- Typical class meeting:
  - Student-prepared paper presentation (50 minutes)
  - Group discussion (30 minutes)
- Paper presentation goals:
  - Motivate the paper, provide background
  - Highlight key contributions
  - Explain important technical points
    - examples are great!
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 (1:30-2): student(s) presenting the next Monday
  - Friday (2-2:30): student(s) presenting the next Wednesday
- Will post your slides on the web after class

Presentation Advice
- Good presentations come from lots of preparation
  - Give yourself plenty of time to read & understand paper
  - You may have to read cited & related papers to fully understand
  - Revise your slides several times
  - Practice your talk several times
  - Beware of presenting too much technical detail
  - Pick most important 1-2 technical items; summarize the rest
  - Put yourself in authors' shoes
    - Why did they do what they did?
    - Are their decisions still good ones?

Discussion Format
- Instructor will lead discussion
- Be prepared with key questions from the paper
- I'll ask specific students questions if needed

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!

Course Project
- Major component of the course
- Small-scale original research project
- Opportunity to study a database-related problem in depth
- Work in groups of 2
- Not in DB? Choose a project that fits your other interests
- Choose your project by early October

Midterm Exam
- 30%
  - Open-book exam
  - Tentatively November 22 1:30-3:30
Project
- 40%
  - Based on final paper and presentation / demo
Paper Reactions
- 10%
  - Spot-check (S/U)
Class Participation
- 20%
  - 15% Presentations
  - 5% Participating in discussion

Grading

Presentation Logistics
- By Friday Sept. 9, fill out form at http://goo.gl/ZdEul that indicates your preferred dates
  - No guarantees, but will try to honor requests
- Student presentations begin Monday Sept. 19
  - Will notify students at least 1 week in advance
  - Schedule will be finalized after add/drop deadline
- First paper: "What Goes Around Comes Around", Stonebraker & Hellerstein
  - Reaction due Sunday, 8PM
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?

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

Data Independence

- Applications insulated from how data is structured and stored.
- One of the important motivations for relational model (see the papers for next time)

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)

Example Instance of Students Relation

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>5366</td>
<td>Jones</td>
<td>jones@cs</td>
<td>18</td>
<td>3.4</td>
</tr>
<tr>
<td>5368</td>
<td>Smith</td>
<td>smith@eecs</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
</tr>
</tbody>
</table>

Order of rows is not important
All rows are distinct

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
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 $\neq$ programming languages!
  - QLs not expected to be “Turing complete”

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.

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.

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

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

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:
  - $R \in Rel$
  - $R.a \text{ op } S.b$
  - $R.a \text{ op constant } \text{ or constant op } R.a$
- A formula is recursively defined using logical operators $\land, \lor, \neg$ and quantifiers $\exists, \forall$
Since each operation returns a relation, additional operations:

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

Additional operations:
- Intersection (\(\cap\)), division, renaming: Not essential, but (very!) useful.
- Since each operation returns a relation, operations can be composed (Algebra is “closed”).

Relational Algebra

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

Additional operations:
- Intersection (\(\cap\)), division, renaming: Not essential, but (very!) useful.
- Since each operation returns a relation, operations can be composed (Algebra is “closed”).

**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.

```plaintext
\pi_{\text{sname, rating}}(S)
\pi_{\text{age}}(S)
```

**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!

```plaintext
<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>dusty</td>
<td>9</td>
<td>35.0</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>
```

**Union, Intersection, Set-Difference**

- All of these operations take two input relations, which must be union-compatible.
- What about duplicates?

```plaintext
\sigma_{\text{rating}} > 8(S)
```

```plaintext
S1 \cup S2
```

---

**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:
  \(\{P \mid 3S \subseteq \text{Sailors}(S.\text{rating} > 7 \land P.\text{sname} = S.\text{sname} \land P.\text{age} = S.\text{age})\}\)

- Find names of sailors who have reserved a red boat:
  \(\{P \mid 3S \subseteq \text{Sailors} \exists R \subseteq \text{Reserves} (R.\text{sid} = S.\text{sid} \land P.\text{sname} = S.\text{sname} \land 3B \subseteq \text{Boats} (B.\text{bid} = R.\text{bid} \land B.\text{color} = \text{red}))\}\)

---

**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).
Cross-Product

- $S_1 \times R_1$: Each row of $S_1$ is paired with each row of $R_1$.

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>32</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>58</td>
<td>dusty</td>
<td>10</td>
<td>55.0</td>
</tr>
</tbody>
</table>

$	imes$

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>101</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>103</td>
</tr>
<tr>
<td>58</td>
<td>dusty</td>
<td>10</td>
<td>112</td>
</tr>
</tbody>
</table>

Joins

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

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>101</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>103</td>
</tr>
<tr>
<td>58</td>
<td>dusty</td>
<td>10</td>
<td>112</td>
</tr>
</tbody>
</table>

Limitations of Relational Algebra

- **Transitive Closure**
  - e.g., if a 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?

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)

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

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

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

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

The SQL Query Language

Given the following instances of Enrolled and Students:

- 5366: Jones, Jonesites, 18, 3.4
- 5368: Smith, smithites, 18, 3.2
- 5806: Jones, Jonesites, 18, 3.4
- 5808: Smith, smithites, 18, 3.2
- 5820: Smith, smithites, 18, 3.2

We get:

- 5366: Jones, Jonesites, 18, 3.4
- 5368: Smith, smithites, 18, 3.2
- 5806: Jones, Jonesites, 18, 3.4
- 5808: Smith, smithites, 18, 3.2
- 5820: Smith, smithites, 18, 3.2
Relational Algebra “Plan”

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

Aggregate Operators

- SELECT COUNT (*)
  FROM Students S
- SELECT COUNT (DISTINCT S.name)
  FROM Students S
- SELECT AVG (S.age)
  FROM Students S
  WHERE S.gpa = 4.0

GROUP BY and HAVING

Important for parallel DB and data cube papers (upcoming)

GROUP BY grouping-list
HAVING group-qualification

Conceptual Evaluation:
1. Eliminate tuples that don’t satisfy qualification
2. Partition remaining data into groups
3. Eliminate groups according to group-qualification
4. Evaluate aggregate operation(s) for each group

The SQL Query Language

- Many more complex queries
  - See R&G Chapter 5
- Nested Queries
- Set Operations

Database Management Systems (DBMSs)

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

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.
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

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.

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

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

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

Query Plan Example

```
SELECT DISTINCT E.ename
FROM Emp E, Dept D
WHERE D.dname = 'Toy'
AND D.did = E.did
```

Annotated RA Tree
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

Example - Concurrent Execution

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

Example - Crash Recovery

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

The ACID Properties

- Atomicity: All actions in the Xact happen, or none happen.
- Consistency: Consistent DB + consistent Xact $\Rightarrow$ consistent DB
- Isolation: Execution of one Xact is isolated from that of other Xacts.
- Durability: If a Xact commits, its effects persist.

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