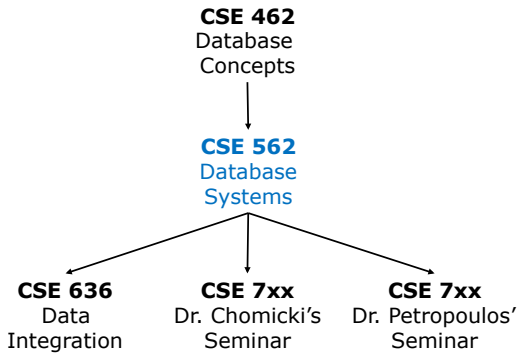
CSE 562
Database Systems

Introduction

Some slides are based or modified from originals by
Database Systems: The Complete Book,
Pearson Prentice Hall 2nd Edition
©2008 Garcia-Molina, Ullman, and Widom

cse@buffalo

UB CSE Database Courses



Resources

- **Instructor:** Dr. Michalis Petropoulos
  Office Hours: Tue & Thu @ 2-3pm
  Location: 210 Bell Hall
- **TA:** Xinglei Zhu
  Office Hours: Wednesday @ 12-2pm
  Location: 329 Bell Hall
- **Web Page**
  http://www.cse.buffalo.edu/~mpetropo/CSE562-SP10/
- **Newsgroup**
  sunyab.cse.562

Goals of the Course

- **Study key DBMS design issues:**
  - Query Languages, Indexing, Query Compilation, Query Execution, Query Optimization
  - Recovery, Concurrency and other transaction management issues
  - Data warehousing, Column-Oriented DBMS
  - Database as a Service (DaaS), Cloud DBMS
- **Ensure that:**
  - You are comfortable using a DBMS
  - You acquire hands-on experience by implementing the internal components of a relational database engine

Course Evaluation

- **Homework Assignments:** 15%
- **Midterm Exam:** 20% (in class)
- **Final Exam:** 30% (in class)
- **Project:** 35%
  - Teams of 2
- **Late Submission Policy:**
  - Submissions less than 24 hours late will be penalized 10%
  - Submissions more than 24 hours but less than 48 hours late will be penalized 25%

Reading Material

Textbook

- Database Systems: The Complete Book (2nd Edition)
  - by *Garcia-Molina, Ullman and Widom*

Also Recommended

- Database System Concepts (5th Edition)
  - by Avi Silberschatz, Henry F. Korth and S. Sudarshan
- Database Management Systems (3rd Edition)
  - by *Ramakrishnan and Gehrke*
- Foundations of Databases
  - by *Abiteboul, Hull and Vianu*

Prerequisites

- Chapters 2 through 8 of the textbook:
  - Relational Model, Entity/Relationship (E/R) Model
  - Constraints, Functional Dependencies
  - Views, Triggers
  - Database query languages (Relational Algebra, SQL)
- Solid background in Data Structures and Algorithms
- Significant programming experience in Java
- Some knowledge of Compilers
- Curiosity! You should ask a lot of questions!

Let's Get Started!

- Isn't Implementing a Database System Simple?

Relations ⇨ Statements ⇨ Results

Let's Get Started!

Introducing the

MEGATRON 3000

Database Management System

- The latest from Megatron Labs
- Incorporates latest relational technology
- UNIX compatible
- Lightweight & cheap!

Megatron 3000 Implementation Details

- Relations stored in files (ASCII)
  - e.g., relation Movie

```
Wild # Lynch # Winger
Sky # Berto # Winger
Reds # Beatty # Beatty
…
```

- Directory file (ASCII)

```
Movie # Title # STR # Director # STR # Actor # STR …
Schedule # Theater # STR # Title # STR …
…
```


Megatron 3000 Sample Sessions

```
% MEGATRON3000
Welcome to MEGATRON 3000!
&

…

& quit
%
```


Megatron 3000 Sample Sessions

```
& select *
  from Movie #

Title   Director  Actor
Wild    Lynch     Winger
Sky     Berto     Winger
Reds    Beatty    Beatty
Tango   Berto     Brando
Tango   Berto     Winger
Tango   Berto     Snyder

&
```

Megatron 3000 Sample Sessions

```
& select Theater, Movie.Title
  from Movie, Schedule
  where Movie.Title = Schedule.Title
        AND Actor = "Winger" #

  Theater Title
  Odeon   Wild
  Forum   Sky

&
```


Megatron 3000

- To execute `select * from Movie where condition`
  - (1) Read dictionary to get Movie attributes
  - (2) Read Movie file, for each line:
    - (a) Check condition
    - (b) If OK, display

Megatron 3000

- To execute

```
select Theater, Movie.Title
from Movie, Schedule
where Movie.Title = Schedule.Title
            AND Actor = "Winger"
```

  - (1) Read dictionary to get Movie, Schedule attributes
  - (2) Read Movie file, for each line:
    - (a) Read Schedule file, for each line:
      - (i) Create join tuple
      - (ii) Check condition
      - (iii) Display if OK

What's wrong with the Megatron 3000 DBMS?

- Tuple layout on disk
  - Change string from 'Cat' to 'Cats' and we have to rewrite file
  - Deletions are expensive

What's wrong with the Megatron 3000 DBMS?

- Search expensive; no indexes
  - Cannot find tuple with given key quickly
  - Always have to read full relation

What's wrong with the Megatron 3000 DBMS?

- Brute force query processing
  For example:

```
select Theater, Movie.Title
from Movie, Schedule
where Movie.Title=Schedule.Title
and Actor = "Winger"
```

- Much better if
  - Use index to select tuples with "Winger" first
  - Use index to find theaters where qualified titles play
- Or
  - Sort both relations on Title and merge-join
- Exploit cache and buffers

What's wrong with the Megatron 3000 DBMS?

- Concurrency control & recovery
- No reliability
  - Can lose data
  - Can leave operations half done

What's wrong with the Megatron 3000 DBMS?

- Security
- Interoperation with other systems
- Consistency enforcement

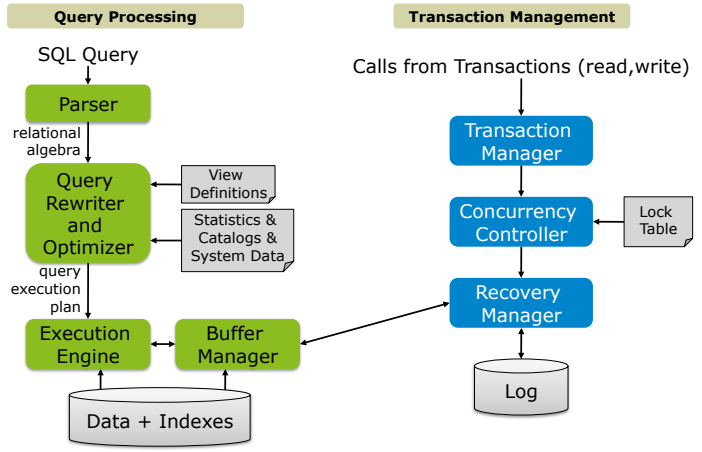
Course Topics

- Hardware Aspects
- Physical Organization Structure
  - Records in blocks, dictionary, buffer management,…
- Indexing
  - B-Trees, hashing,…
- **Query Processing**
  - parsing, rewriting, logical/physical operators, algebraic and cost-based optimization, semantic optimization…
- Crash Recovery
  - Failures, stable storage,…

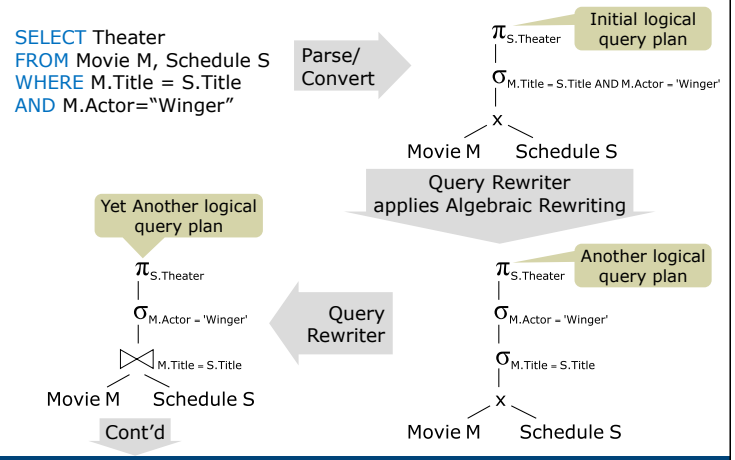
Course Topics

- Concurrency Control
  - Correctness, locks, deadlocks…
- On-Line Analytical Processing
  - Map-Reduce,…
- Database as a Service (DaaS)
- Cloud DBMS

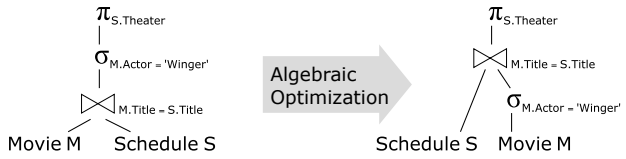
Database System Architecture



Example Journey of a Query



Example Journey of a Query (cont'd)



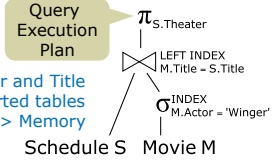
Algebraic Optimization

What are the rules used for the transformation of the query?

How do we evaluate the cost of a possible execution plan?

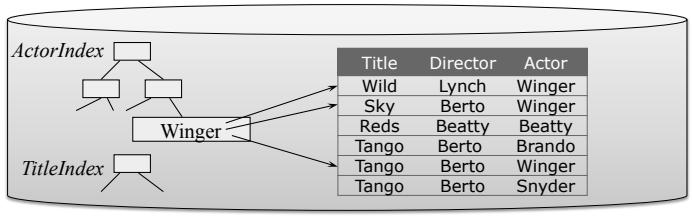
Cost-based Optimization

Query Execution Plan



Index on Actor and Title
Unsorted tables
Tables >> Memory

The Journey of a Query (cont'd)



EXECUTION ENGINE

find "Winger" tuples using ActorIndex
  for each "Winger" tuple
    find tuples t with the same title using TitleIndex
    project the attribute Actor of t

How is the table arranged on the disk?

Are tuples with the same Actor value clustered?

What is the exact structure of the index (tree, hash,…)?