CS 184 Course Web Server

- All course information will be posted on line

- URL:

- Class Notes available before class on the Web.
Course Info

- **TAs**
  - Koushik Niyogi
    - Office and office Hours:
      - Monday and Wednesday: 2:00-3:00 PM
  - Rajat Mathur
    - Office and office Hours: TBA

- **Instructor**
  - Office Hours:
  - Tue 11-12 pm
  - (send email)
  - Email: sharad@ics.uci.edu
  - to contact me urgently, send email and mark subject line as CS 184 URGENT
Desiderata

- **Course Text:** (either of following two books will suffice)
  - A First Course in Database Systems, Ullman and Widom
    - we will cover the entire book
  - Database Systems Concepts, Silberschatz, Korth, and Sudarshan
    - we will cover chapters 1-9

- **Software:**
  - Course will involve significant programming.
  - You will get exposure to database programming in DB2
Desiderata (cont.)

Course Requirements:
- Problem sets  ~ approx. every week to 10 days.
  - Total not to exceed 8
- Midterm
- Final (comprehensive)

Grades:
- Problem sets 15%
- Personal Database Assignment (project)        15%
- Midterm            30%
- Final                40%
Policies

- **Late Assignments**
  - No grace period after due date.... except under exceptional circumstances
  - job interviews, out of town trip, breaks etc do not qualify as exceptional circumstances!

- **Working in Groups**
  - do your homework problem sets in group size not to exceed 3
  - learn more
  - get better grades
  - get used to working in groups (important to employers)

- **Do exams individually!!**
Material Covered in CS 184

- Four aspects of studying DBMSs
  - Modeling and design of databases
    - allows exploration of issues before committing to an implementation
  - Programming: queries and DB operations like update.
    - SQL == “intergalactic dataspeak”
  - DBMS implementation
  - Effect of technology and application advances to database technology.

- CS 184 == (1) + (2)
- CS 214 == (3)
- CS215 == (3) + (4)
Database Management Environment

**Database**: collection of interrelated information about world being modeled

**DBMS**: general purpose software to define, create, modify, retrieve, delete and manipulate a database

**Vendors**: Informix, Oracle, O2, Sybase, IBM, DEC
Traditional DBMS Goals

- Efficient management of (faster than files) large amounts of (gigabytes) of persistent (outlasts creator), reliable (outlasts crashes) shared information (multiple users).

- DBMS Users:
  - small and large corporations
    - E-commerce companies, banks, airlines, transportation companies, corporate databases, government agencies, defense.
  - Anyone you can think of!
Databases and File Systems

- DBMSs evolved from file systems.
- DBMSs provide many features that traditional file systems do not.
  - Support for concurrent access and data sharing. Data consistency in presence of concurrency
  - Reliability in presence of failures and system crashes.
  - Efficient associative access to very large amounts of data
  - A high level Query language (SQL) to define, create, access, and manipulate data. Support for unanticipated queries
  - Support for multiple data views
  - Security and authorization
  - Data abstraction
  - Prevention of data redundancy and inconsistencies
Data Abstraction

- **program data independence:**
  - ability to hide details of how data is stored and maintained from application programs

- **program-operation independence:**
  - ability to hide details of operation implementation from application programs (Object-Orientation)
Data Abstraction

- Hiding system complexity, physical storage details from users and application programs

Physical level

Logical Level

View 1

View 2

View n

Physical description of data, storage organization

Conceptual representation

Customized views
Schemas and Instances

- **Instance:**
  - set of data currently instantiated in database
  - changes frequently

- **Schema:**
  - overall design, structure, and constraints over the database
  - referred to as metadata
  - changes infrequently

- **Example:**

  **Schema**
  
  **Tables**
  - Emp (ename, dep#)
  - Dept (dep#, dname, mgr)

  **Constraints**
  - each department has a single manager

  **Instance**
  
<table>
<thead>
<tr>
<th>Emp</th>
</tr>
</thead>
<tbody>
<tr>
<td>(John, 10), (Cindy, 15), (Martha, 10)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dept</th>
</tr>
</thead>
<tbody>
<tr>
<td>(10, Toy, John), (15, Sales, Cindy)</td>
</tr>
</tbody>
</table>
Data Model

- Set of concepts and tools used to describe the database schema
- Different schemas at different abstraction levels:
  - physical schema: describes physical organization of data
  - logical schema: describes data at conceptual level
  - sub schema: defines data at view level
- Different models used describe schemas at different abstraction levels
Types of Data Models

- **Object based Logical Models**
  - Used to describe schema at view and logical levels.
  - Support abstract view of data as objects, relationships, constraints
  - Example: Entity Relationship Model, Functional data Model, Semantic Model, Object Oriented Model like ODL
Types of Data Model (cont.)

- **Record-Based Logical Models**
  - Used to define data at view and logical levels.
  - Provide a high level description of implementation.
  - Examples: Relational Model, Hierarchical Model, Network Model.

- **Physical Models**
  - Used to describe data at implementation level.
  - Examples: Frame Memory Model, Unifying Model.
Entity-Relationship Model

Example of schema in the entity-relationship model
Entity Relationship Model (Cont.)

- E-R model of real world
  - Entities (objects)
    - E.g. customers, accounts, bank branch
  - Relationships between entities
    - E.g. Account A-101 is held by customer Johnson
    - Relationship set depositor associates customers with accounts

- Widely used for database design
  - Database design in E-R model usually converted to design in the relational model (coming up next) which is used for storage and processing
### Relational Model

- Example of tabular data in the relational model

<table>
<thead>
<tr>
<th>Customer-id</th>
<th>customer-name</th>
<th>customer-street</th>
<th>customer-city</th>
<th>account-number</th>
</tr>
</thead>
<tbody>
<tr>
<td>192-83-7465</td>
<td>Johnson</td>
<td>Alma</td>
<td>Palo Alto</td>
<td>A-101</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>Smith</td>
<td>North</td>
<td>Rye</td>
<td>A-215</td>
</tr>
<tr>
<td>192-83-7465</td>
<td>Johnson</td>
<td>Alma</td>
<td>Palo Alto</td>
<td>A-201</td>
</tr>
<tr>
<td>321-12-3123</td>
<td>Jones</td>
<td>Main</td>
<td>Harrison</td>
<td>A-217</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>Smith</td>
<td>North</td>
<td>Rye</td>
<td>A-201</td>
</tr>
</tbody>
</table>
A Sample Relational Database

(a) The `customer` table

<table>
<thead>
<tr>
<th>customer-id</th>
<th>customer-name</th>
<th>customer-street</th>
<th>customer-city</th>
</tr>
</thead>
<tbody>
<tr>
<td>192-83-7465</td>
<td>Johnson</td>
<td>12 Alma St.</td>
<td>Palo Alto</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>Smith</td>
<td>4 North St.</td>
<td>Rye</td>
</tr>
<tr>
<td>677-89-9011</td>
<td>Hayes</td>
<td>3 Main St.</td>
<td>Harrison</td>
</tr>
<tr>
<td>182-73-6091</td>
<td>Turner</td>
<td>123 Putnam Ave.</td>
<td>Stamford</td>
</tr>
<tr>
<td>321-12-3123</td>
<td>Jones</td>
<td>100 Main St.</td>
<td>Harrison</td>
</tr>
<tr>
<td>336-66-9999</td>
<td>Lindsay</td>
<td>175 Park Ave.</td>
<td>Pittsfield</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>Smith</td>
<td>72 North St.</td>
<td>Rye</td>
</tr>
</tbody>
</table>

(b) The `account` table

<table>
<thead>
<tr>
<th>account-number</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-101</td>
<td>500</td>
</tr>
<tr>
<td>A-215</td>
<td>700</td>
</tr>
<tr>
<td>A-102</td>
<td>400</td>
</tr>
<tr>
<td>A-305</td>
<td>350</td>
</tr>
<tr>
<td>A-201</td>
<td>900</td>
</tr>
<tr>
<td>A-217</td>
<td>750</td>
</tr>
<tr>
<td>A-222</td>
<td>700</td>
</tr>
</tbody>
</table>

(c) The `depositor` table

<table>
<thead>
<tr>
<th>customer-id</th>
<th>account-number</th>
</tr>
</thead>
<tbody>
<tr>
<td>192-83-7465</td>
<td>A-101</td>
</tr>
<tr>
<td>192-83-7465</td>
<td>A-201</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>A-215</td>
</tr>
<tr>
<td>677-89-9011</td>
<td>A-102</td>
</tr>
<tr>
<td>182-73-6091</td>
<td>A-305</td>
</tr>
<tr>
<td>321-12-3123</td>
<td>A-217</td>
</tr>
<tr>
<td>336-66-9999</td>
<td>A-222</td>
</tr>
<tr>
<td>019-28-3746</td>
<td>A-201</td>
</tr>
</tbody>
</table>
Classification of DBMSs based on Data Model

- **Relational DBMSs:**
  - modeling concept: tables and constraints on tables
  - Query Language: SQL
  - Applications: suited for traditional business processing applications

- **Object Oriented DBMSs**
  - modeling concepts: objects, classes, inheritance
  - Query Language: object oriented OQL
  - Applications: suited for CAD databases, CASE databases, office automation

- **Object Relational DBMSs:**
  - incorporate OO concepts into relational model
  - similar functionality as OODBMSs though different in implementations
  - Language extended to process objects.
DBMS Languages

- **Data Definition Language (DDL)**
  - DDL = the language used to describe a schema
  - Data dictionary/directory = a compiled description of a schema

- **Data Manipulation Language (DML)**
  - DML = Language users use to ask questions about (query) the database, and to change the data in the database.

- **Storage Definition Language (SDL)**
  - SDL = language to define the internal schema

- **View Definition Language (VDL)**
  - VDL = view definition language
Data Definition Language (DDL)

- Specification notation for defining the database schema
  - E.g.
    ```sql
    create table account (
        account-number char(10),
        balance integer
    )
    ```

- DDL compiler generates a set of tables stored in a data dictionary

- Data dictionary contains metadata (i.e., data about data)
  - database schema
  - Data storage and definition language
    - language in which the storage structure and access methods used by the database system are specified
    - Usually an extension of the data definition language
Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
  - DML also known as query language
- Two classes of languages
  - Procedural – user specifies what data is required and how to get those data
  - Nonprocedural – user specifies what data is required without specifying how to get those data
- SQL is the most widely used query language
SQL

- SQL: widely used non-procedural language
  - E.g. find the name of the customer with customer-id 192-83-7465
    ```sql
    select customer.customer-name
    from customer
    where customer.customer-id = '192-83-7465'
    ```

- Basic SQL has limited expressability
  - cannot implement any arbitrary function in SQL

- Application programs generally access databases through one of
  - Language extensions to allow embedded SQL
  - Application program interface (e.g. ODBC/JDBC) which allow SQL queries to be sent to a database
Application Architectures

- **Two-tier architecture**: E.g. client programs using ODBC/JDBC to communicate with a database
- **Three-tier architecture**: E.g. web-based applications, and applications built using “middleware”
DBMS Interface

- Provides users means to interact with database.
- Where we are today:
  - Menu driven interface, Graphical interface, Forms based interface, Natural Language Interface, WWW connectivity.
- Interface design is a tremendous challenge not only for DBMS researchers but to HCI and human cognition researchers.
- Future interfaces to databases:
  - virtual reality, immersive environments, speech, natural languages, gestures, handwriting, eye tracking brain waves, tactile interfaces, multimodal input and outputs -- combination of more than one modality.
People Involved with DBMSs

- DBMS designers and implementers
- tool designers
- database administrator (DBA)
  - DBA = ‘super-user’ for a database, similar to a system administrator.
  - DBA can define schemas, views, authorization, indexes, tuning parameters, etc.
- application programmers
- database designers
  - interact with users to define database at all levels
- database and system operators.
- end users
- large number of jobs available for each of the above tasks!!
Key Database Technologies

- **Data Models**
  - allow specification of database structure at all the levels of abstraction

- **Design tools**
  - that help in the database design process. These tools automate or facilitate some aspects of the design

- **Access Methods**
  - data structures to support efficient access of data on disk

- **Query Optimization and Processing**
  - efficient query processing techniques for good query performance. These techniques usually minimize the amount of disk I/O

- **Transaction processing techniques**
  - to support concurrent access and reliability in the presence of failures
Need for Query Optimization

- Consider two tables
  - Employee(ename, salary, department)
    - say 1000 entries
  - Manager(mname, department)
    - say 10 entries

- Query:
  - List the names of employees for the department of which Sharad is the manager
Strategies 1

- For each entry M in Manager
  - read record M
  - For each entry E in employees
    - read Entry E
    - If (E.department == M.department) and (M.mname = “sharad”)
      - print E.ename

- Cost Analysis:
  - Outer loop 10 iterations. 1 read operation each time.
  - Inner loop 1000 iterations. 1 entry read each time.
  - total number of reads = 10 + 1000*10 = 10,010.
Strategy 2

- For each entry M in Manager
  - If M.mname = “sharad”
    temp = M.department

- For each entry E in Employees
  - If E.department == temp
    print E.ename

- Cost Analysis:
  - first loop 10 iterations. 1 read operation each time.
  - Second loop 1000 iterations. 1 read operation each times.
  - Total number of reads = 1010.
Transaction Concept

- **Atomicity:**
  - all or nothing execution.

- **Consistency:**
  - execution of a transaction leaves system state as well as the state of the real world consistent.

- **Isolation:**
  - partial effects of a transaction are hidden from each other.

- **Durability:**
  - a successful transactions effects survives future system malfunctions.
Example of Transaction

- **Withdraw $100 checking account using an ATM.**
- **Atomicity:**
  - account debited if and only if user gets money from the ATM.
- **Consistency:**
  - balance of account is always positive.
- **Isolation:**
  - concurrent execution of withdraw, deposit, transfers does not result in an incorrect balance of account.
- **Durability:**
  - After withdraw terminates, and the ATM dispenses money, account reflects that $100 withdrawn despite failures.
Motivation of Isolation

- Consider two transactions--
  - read account A, debit the value by $100 and write the new value to A.
  - read account A, credit the value by $200 and write the new value to A.
- Let initial value of A be $1000.
- Final value should be $1100.
- Consider the following execution if concurrency is permitted:
  - read1(A,1000) read2(A,1000) write2(A,1200) write1(A,900)
  - for the above execution the value of A is 900!
Importance of the Transactions

- **Transaction concept supports:**
  - simple failure semantics -- either all the effects of the transaction appear or none do-- all or nothing
  - an isolated view of the world -- protection from partial effects of concurrently executing transactions

- **Makes application development easy**
  - complex, possibly distributed applications that share data and resources can be developed without explicitly dealing with synchronization and fault-tolerance.
Transactions versus Other Concurrent Programming Environments

- concurrent programs prevalent in a variety of other areas in CS
  - operating systems, parallel programming, distributed systems.
- support Powerful Language Constructs:
  - to specify concurrent behavior of applications (programmers responsibility to deal with failures and concurrency issues).
- In contrast transactions relieve the application programmers of these tasks.