



## Lecture 0 of 42

### Database System Concepts: Course Organization and Survey

Monday, 21 August 2006

William H. Hsu

Department of Computing and Information Sciences, KSU

KSOL course page: <http://snipurl.com/va60>

Course web site: <http://www.kddresearch.org/Courses/Fall-2006/CIS560>

Instructor home page: <http://www.cis.ksu.edu/~bhsu>

#### Reading for Next Class:

Chapter 1, Silberschatz *et al.*, 5<sup>th</sup> edition – this week

Syllabus and Introductory Handouts



## Course Administration

- Official Course Page (KSOL): <http://snipurl.com/va60>
- Class Web Page: <http://www.kddresearch.org/Courses/Fall-2006/CIS560>
- Instructional E-Mail Addresses
  - \* [CIS560TA-L@listserv.ksu.edu](mailto:CIS560TA-L@listserv.ksu.edu) (always use this to reach instructor)
  - \* [CIS560-L@listserv.ksu.edu](mailto:CIS560-L@listserv.ksu.edu) (this goes to everyone)
- Instructor: William Hsu, Nichols 213
  - \* Office phone: (785) 532-7905; home phone: (785) 539-7180
  - \* IM: AIM/YIM/MSN [hsuw](#) & [rizanabsith](#), ICQ [28651394](#) & [191317559](#)
  - \* Office hours: after class Mon/Wed/Fri; other times by appointment
- Graduate Teaching Assistant: TBD
  - \* Office location: Nichols 124
  - \* Office hours: to be announced on class web board
- Grading Policy
  - \* Hour exams: 15% each (in-class, closed-book); final (open-book): 30%
  - \* Machine problems, problem sets (8 of 10): 16%; term project: 17%
  - \* Class participation: 7% (3% attendance, 2% questions, 2% answers)





## How To Get an A in This Course

- **A Story from Dr. Gerard G. L. Meyer, Johns Hopkins University**
- **Ask Questions**
  - \* Ask for (more) examples, another explanation, etc. if needed (“don’t be shy”)
  - \* All students (especially remote students): post in class web board
    - ⇒ Unclear points – bring to class as well
    - ⇒ “When will X happen”?
  - \* Fastest way to reach instructor: instant messaging (ICQ, MSN Messenger)
  - \* Notify TA, KDD system administrators of any computer problems
- **Be Aware of Resources**
  - \* Check with instructor or GTA about
    - ⇒ Handouts, lectures, grade postings
    - ⇒ Resources online
  - \* Check with classmates about material from missed lecture
- **Start Machine Problems (and Problem Sets) Early**
  - \* How to start virtuous (as opposed to vicious) cycle
  - \* Don’t cheat



## Homework Assignments: Problem Sets and Machine Problems

- **PS1 assigned Wed 23 Aug 2006, due Wed 06 Sep 2006**
- **MP2 assigned Wed 06 Sep 2006, due Fri 22 Sep 2006**
  - \* Submit using K-State Online
  - \* HW page: <http://www.kddresearch.org/Courses/Fall-2006/CIS560/Homework>
- **Model solutions: 2 class days after due date**
- **Graded assignments: ≤ 7 days after due date**
- **Machine Problem: Relational Joins**
  - \* Problem specifications to be posted on homework page before 06 Sep 2006
  - \* Languages: C/C++ & Java
  - \* MP guidelines
    - ⇒ Work individually
    - ⇒ Generate standard output files and test against partial standard solution
    - ⇒ No late submissions except with documented excusal (medical, etc.)





## Chapter 1: Introduction

- **Purpose of Database Systems**
- View of Data
- Database Languages
- **Relational Databases**
- **Database Design**
- **Object-based and semistructured databases**
- Data Storage and Querying
- **Transaction Management**
- **Database Architecture**
- Database Users and Administrators
- Overall Structure
- History of Database Systems



## Database Management System (DBMS)

- DBMS contains information about a particular enterprise
  - \* Collection of interrelated data
  - \* Set of programs to access the data
  - \* An environment that is both *convenient* and *efficient* to use
- Database Applications:
  - \* Banking: all transactions
  - \* Airlines: reservations, schedules
  - \* Universities: registration, grades
  - \* Sales: customers, products, purchases
  - \* Online retailers: order tracking, customized recommendations
  - \* Manufacturing: production, inventory, orders, supply chain
  - \* Human resources: employee records, salaries, tax deductions
- Databases touch all aspects of our lives





## Purpose of Database Systems

- In the early days, database applications were built directly on top of file systems
- Drawbacks of using file systems to store data:
  - \* **Data redundancy and inconsistency**
    - ⇒ Multiple file formats, duplication of information in different files
  - \* **Difficulty in accessing data**
    - ⇒ Need to write a new program to carry out each new task
  - \* **Data isolation — multiple files and formats**
  - \* **Integrity problems**
    - ⇒ Integrity constraints (e.g. account balance > 0) become “buried” in program code rather than being stated explicitly
    - ⇒ Hard to add new constraints or change existing ones



## Purpose of Database Systems (Cont.)

- Drawbacks of using file systems (cont.)
  - \* **Atomicity of updates**
    - ⇒ Failures may leave database in an inconsistent state with partial updates carried out
    - ⇒ Example: Transfer of funds from one account to another should either complete or not happen at all
  - \* **Concurrent access by multiple users**
    - ⇒ Concurrent accessed needed for performance
    - ⇒ Uncontrolled concurrent accesses can lead to inconsistencies
      - ◆ Example: Two people reading a balance and updating it at the same time
  - \* **Security problems**
    - ⇒ Hard to provide user access to some, but not all, data
- Database systems offer solutions to all the above problems





## Levels of Abstraction

- **Physical level:** describes how a record (e.g., customer) is stored.
- **Logical level:** describes data stored in database, and the relationships among the data.  

```
type customer = record
  customer_id : string;
  customer_name : string;
  customer_street : string;
  customer_city : integer;
end;
```
- **View level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.



## View of Data

An architecture for a database system

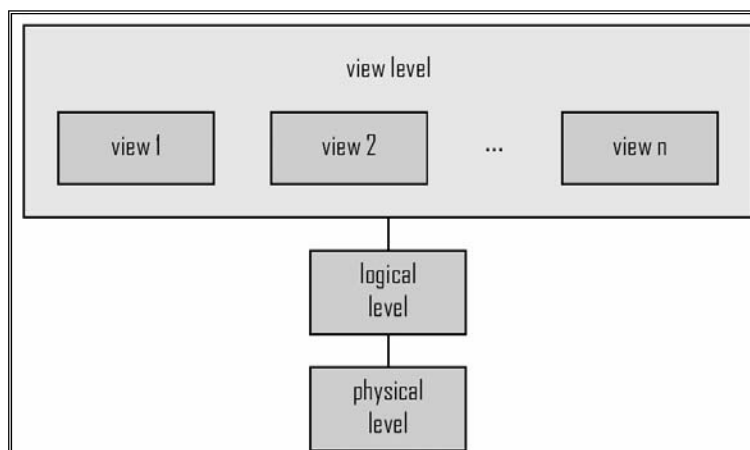
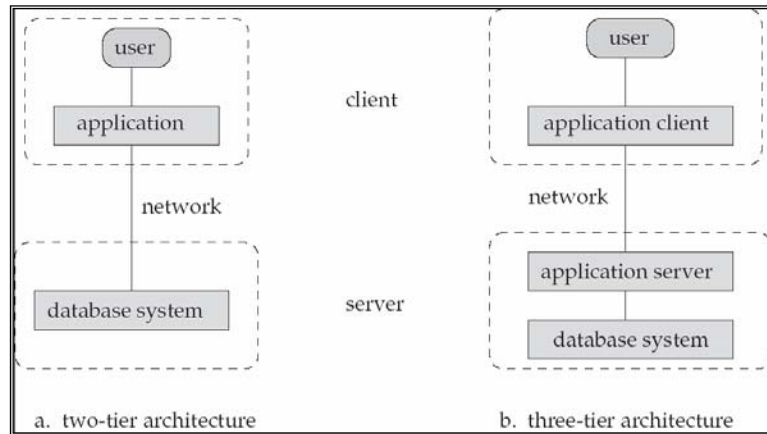




Figure 1.7



## Instances and Schemas

- Similar to types and variables in programming languages
- **Schema** – logical structure of the database
  - \* Example: The database consists of information about a set of customers and accounts and the relationship between them)
  - \* Analogous to type information of a variable in a program
  - \* Physical schema: database design at the physical level
  - \* Logical schema: database design at the logical level
- **Instance** – actual content of the database at a particular point in time
  - \* Analogous to the value of a variable
- **Physical Data Independence** – ability to modify the physical schema without changing the logical schema
  - \* Applications depend on the logical schema
  - \* In general, interfaces between various levels and components should be well defined so that changes in some parts do not seriously influence others



## Data Models

- A collection of tools for describing
  - \* Data
  - \* Data relationships
  - \* Data semantics
  - \* Data constraints
- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semistructured data model (XML)
- Other older models:
  - \* Network model
  - \* Hierarchical model



## 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
  - \* **Declarative (nonprocedural)** – user specifies what data is required without specifying how to get those data
- SQL is the most widely used query language





## Data Definition Language (DDL)

- Specification notation for defining the database schema  
Example: `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
    - ⇒ Specifies the storage structure and access methods used
  - \* Integrity constraints
    - ⇒ Domain constraints
    - ⇒ Referential integrity (**references** constraint in SQL)
    - ⇒ Assertions
  - \* Authorization



## Relational Model

- Example of tabular data in the relational model

<i>customer_id</i>	<i>customer_name</i>	<i>customer_street</i>	<i>customer_city</i>	<i>account_number</i>
192-83-7465	Johnson	12 Alma St.	Palo Alto	A-101
192-83-7465	Johnson	12 Alma St.	Palo Alto	A-201
677-89-9011	Hayes	3 Main St.	Harrison	A-102
182-73-6091	Turner	123 Putnam St.	Stamford	A-305
321-12-3123	Jones	100 Main St.	Harrison	A-217
336-66-9999	Lindsay	175 Park Ave.	Pittsfield	A-222
019-28-3746	Smith	72 North St.	Rye	A-201

Attributes





## A Sample Relational Database

<i>customer_id</i>	<i>customer_name</i>	<i>customer_street</i>	<i>customer_city</i>
192-83-7465	Johnson	12 Alma St.	Palo Alto
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

<i>account_number</i>	<i>balance</i>
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

<i>customer_id</i>	<i>account_number</i>
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table



## SQL

- **SQL:** widely used non-procedural language
  - \* Example: Find the name of the customer with customer-id 192-83-7465
 

```
select customer.customer_name
from customer
where customer.customer_id = '192-83-7465'
```
  - \* Example: Find the balances of all accounts held by the customer with customer-id 192-83-7465
 

```
select account.balance
from depositor, account
where depositor.customer_id = '192-83-7465' and
depositor.account_number = account.account_number
```
- 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



## Database Design

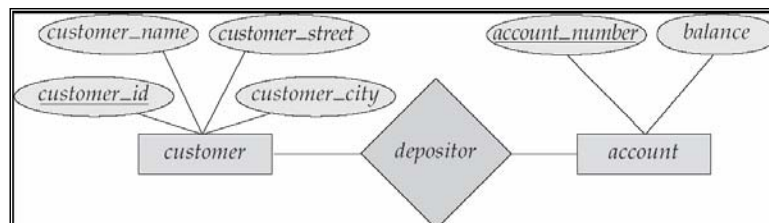
The process of designing the general structure of the database:

- Logical Design – Deciding on the database schema. Database design requires that we find a “good” collection of relation schemas.
  - \* Business decision – What attributes should we record in the database?
  - \* Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- Physical Design – Deciding on the physical layout of the database



## The Entity-Relationship Model

- Models an enterprise as a collection of *entities* and *relationships*
  - \* Entity: a “thing” or “object” in the enterprise that is distinguishable from other objects
    - ⇒ Described by a set of *attributes*
  - \* Relationship: an association among several entities
- Represented diagrammatically by an *entity-relationship diagram*:





## Object-Relational Data Models

- Extend the relational data model by including object orientation and constructs to deal with added data types.
- Allow attributes of tuples to have complex types, including non-atomic values such as nested relations.
- Preserve relational foundations, in particular the declarative access to data, while extending modeling power.
- Provide upward compatibility with existing relational languages.



## XML: Extensible Markup Language

- Defined by the WWW Consortium (W3C)
- Originally intended as a document markup language not a database language
- The ability to specify new tags, and to create nested tag structures made XML a great way to exchange **data**, not just documents
- XML has become the basis for all new generation data interchange formats.
- A wide variety of tools is available for parsing, browsing and querying XML documents/data





## Storage Management

- **Storage manager** is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible to the following tasks:
  - \* Interaction with the file manager
  - \* Efficient storing, retrieving and updating of data
- Issues:
  - \* Storage access
  - \* File organization
  - \* Indexing and hashing



## Overall System Structure

