



Lecture 01 of 42

Database Architecture: Client-Server Models, Relational DB Definitions

Wednesday, 23 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.*, 5th edition – this week

Syllabus and Introductory Handouts



Lecture Outline

- Reading for Next Class: Chapter 2, Silberschatz *et al.* 5e
- Today and Friday: Basic Relational DB Principles
 - * Relations
 - * Database definitions
 - ⇒ Records
 - ⇒ Fields
 - ⇒ Tables
 - ⇒ Relations
- Next Week: Relational Algebra and SQL Intro
 - * Relational operators: PROJECT, SELECT, JOIN
 - * Variations on relational joins
 - * Implementation
- Examples and Exercises
 - * Look at Chapter 2 examples
 - * Exercises: relational algebra formulas



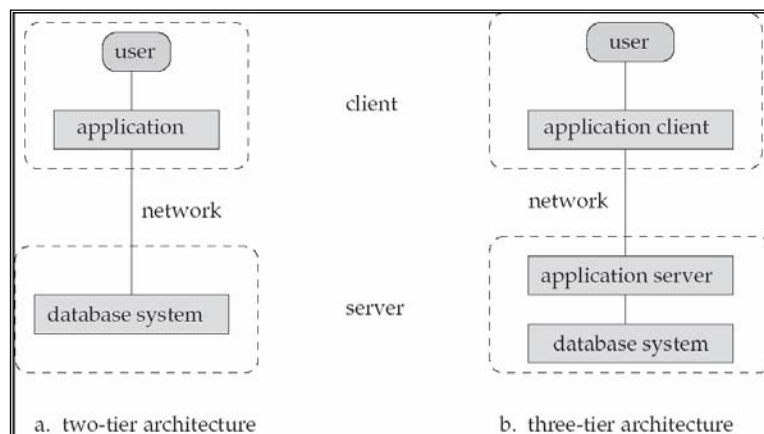


Course Administration

- 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 (always use this to reach instructor)
 - * 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)



Client-Server Architecture: Two-Tier and Three-Tier



Excerpts from Database System Concepts, 5^e

© 2005 Silberschatz, Korth and Sudarshan
See www.db-book.com for conditions on re-use



Review: 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



Review: 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

Database System Concepts, 5th Ed.

©Silberschatz, Korth and Sudarshan
See www.db-book.com for conditions on re-use





Review: A Sample Relational Database

customer_id	customer_name	customer_street	customer_city
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

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

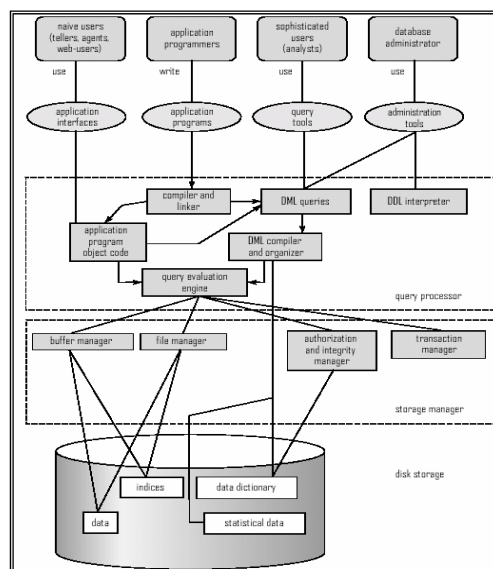
(b) The account table

customer_id	account_number
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



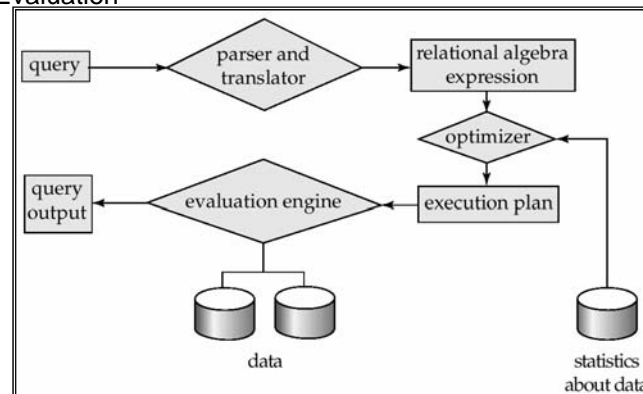
Review: Overall System Structure





Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



Query Processing (Cont.)

- Alternative ways of evaluating a given query
 - * Equivalent expressions
 - * Different algorithms for each operation
- Cost difference between a good and a bad way of evaluating a query can be enormous
- Need to estimate the cost of operations
 - * Depends critically on statistical information about relations which the database must maintain
 - * Need to estimate statistics for intermediate results to compute cost of complex expressions



Transaction Management

- A **transaction** is a collection of operations that performs a single logical function in a database application
- **Transaction-management component** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
- **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.



Database Architecture

The architecture of a database systems is greatly influenced by the underlying computer system on which the database is running:

- Centralized
- Client-server
- Parallel (multi-processor)
- Distributed





History of Database Systems

- 1950s and early 1960s:
 - * Data processing using magnetic tapes for storage
 - ⇒ Tapes provide only sequential access
 - * Punched cards for input
- Late 1960s and 1970s:
 - * Hard disks allow direct access to data
 - * Network and hierarchical data models in widespread use
 - * Ted Codd defines the relational data model
 - ⇒ Would win the ACM Turing Award for this work
 - ⇒ IBM Research begins System R prototype
 - ⇒ UC Berkeley begins Ingres prototype
 - * High-performance (for the era) transaction processing



History (cont.)

- 1980s:
 - * Research relational prototypes evolve into commercial systems
 - ⇒ SQL becomes industrial standard
 - * Parallel and distributed database systems
 - * Object-oriented database systems
- 1990s:
 - * Large decision support and data-mining applications
 - * Large multi-terabyte data warehouses
 - * Emergence of Web commerce
- 2000s:
 - * XML and XQuery standards
 - * Automated database administration





Chapter 2: Relational Model

- Structure of Relational Databases
- Fundamental Relational-Algebra-Operations
- Additional Relational-Algebra-Operations
- Extended Relational-Algebra-Operations
- Null Values
- Modification of the Database



Example of a Relation

<i>account_number</i>	<i>branch_name</i>	<i>balance</i>
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	700
A-217	Brighton	750
A-222	Redwood	700
A-305	Round Hill	350





Basic Structure

- Formally, given sets D_1, D_2, \dots, D_n a **relation** r is a subset of $D_1 \times D_2 \times \dots \times D_n$
Thus, a relation is a set of n -tuples (a_1, a_2, \dots, a_n) where each $a_i \in D_i$

- Example: If

$customer_name = \{\text{Jones, Smith, Curry, Lindsay}\}$

$customer_street = \{\text{Main, North, Park}\}$

$customer_city = \{\text{Harrison, Rye, Pittsfield}\}$

Then $r = \{$
 (Jones, Main, Harrison),
 (Smith, North, Rye),
 (Curry, North, Rye),
 (Lindsay, Park, Pittsfield) $\}$

is a relation over

$customer_name \times customer_street \times customer_city$



Attribute Types

- Each attribute of a relation has a name
- The set of allowed values for each attribute is called the **domain** of the attribute
- Attribute values are (normally) required to be **atomic**; that is, indivisible
 - * Note: multivalued attribute values are not atomic
 - * Note: composite attribute values are not atomic
- The special value *null* is a member of every domain
- The null value causes complications in the definition of many operations
 - * We shall ignore the effect of null values in our main presentation and consider their effect later





Relation Instance

- The current values (*relation instance*) of a relation are specified by a table
- An element t of r is a *tuple*, represented by a *row* in a table

<i>customer_name</i>	<i>customer_street</i>	<i>customer_city</i>
Jones	Main	Harrison
Smith	North	Rye
Curry	North	Rye
Lindsay	Park	Pittsfield

customer

attributes (or columns)

tuples (or rows)



Relations are Unordered

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
- Example: *account* relation with unordered tuples

<i>account_number</i>	<i>branch_name</i>	<i>balance</i>
A-101	Downtown	500
A-215	Mianus	700
A-102	Perryridge	400
A-305	Round Hill	350
A-201	Brighton	900
A-222	Redwood	700
A-217	Brighton	750



Database

- A database consists of multiple relations
- Information about an enterprise is broken up into parts, with each relation storing one part of the information
 - account* : stores information about accounts
 - depositor* : stores information about which customer owns which account
 - customer* : stores information about customers
- Storing all information as a single relation such as *bank(account_number, balance, customer_name, ..)* results in
 - * repetition of information (e.g., two customers own an account)
 - * the need for null values (e.g., represent a customer without an account)
- Normalization theory (Chapter 7) deals with how to design relational schemas



Homework 1: Problem Set

- **Assigned: 23:00 CDT Wed 23 Aug 2006**
- **Due: before midnight CDT Wed 06 Sep 2006**
- **Topics**
 - * Relational database (RDB) basics: client-server, RDB design
 - * Relational algebra warm-up
- **To Be Posted**
 - * [KSOL web site](#)
 - * [KDDresearch.org](#) (URL mailed to class mailing list)
- **Questions and Discussion**
 - * General discussion on class mailing list: CIS560-L@listserv.ksu.edu
 - * Questions for instructor: CIS560TA-L@listserv.ksu.edu
- **Outside References**
 - * Elmasri and Navathe, 4e
 - * Ramakrishnan and Gehrke, 3e

