Database Management Systems

Module 01: Course Overview

Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Srijoni Majumdar
Himadri B G S Bhuyan
Gurunath Reddy M

Database System Concepts, 6th Ed.

www.db-book.com

Slides marked with 'PPD' are new or edited

Module Objectives

- To understand the importance of database management systems in modern day applications
- To Know Your Course

Module Outline

- Why Databases?
- KYC: Know Your Course
  - Course Prerequisite
  - Course Outline
  - Course Text Book
  - Course TAs

- **Why Databases?**
- KYC: Know Your Course

WHY DATABASES?

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: 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 can be very **large**
- Databases touch **all aspects** of our lives

University Database Example

- Application program examples
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students, compute grade point averages (GPA) and generate transcripts
- 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

Drawbacks of using file systems to store data (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 access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) 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

- Why Databases?
- **KYC: Know Your Course**

KNOW YOUR COURSE

Course Prerequisites

- **Essential**
  - **Set Theory**
    - Definition of a Set
      - Intentional Definition
      - Extensional Definition
      - Set-builder Notation
    - Membership, Subset, Superset, Power Set, Universal Set
    - Operations on sets:
      - Union, Intersection, Complement, Difference, Cartesian Product
    - De Morgan's Law
    - **MOOCs: Discrete Mathematics**: https://nptel.ac.in/noc/individual_course.php?id=noc16-ma01

Course Prerequisites

- **Essential**
  - **Relations and Functions**
    - Definition of Relations
    - Ordered Pairs and Binary Relations
      - Domain and Range
      - Image, Preimage, Inverse
      - Properties – Reflexive, Symmetric, Antisymmetric, Transitive, Total
    - Definition of Functions
    - Properties of Functions – Injective, Surjective, Bijective
    - Composition of Functions
    - Inverse of a Function
    - **MOOCs: Discrete Mathematics**: https://nptel.ac.in/noc/individual_course.php?id=noc16-ma01

PPD

Course Prerequisites

- **Essential**
  - **Propositional Logic**
    - Truth Values & Truth Tables
    - Operators: conjunction (and), disjunction (or), negation (not), implication, equivalence
    - Closure under Operations
    - **MOOCs: Discrete Mathematics**: https://nptel.ac.in/noc/individual_course.php?id=noc16-ma01
  - **Predicate Logic**
    - Predicates
    - Quantification – Existential, Universal
    - **MOOCs: Discrete Mathematics**: https://nptel.ac.in/noc/individual_course.php?id=noc16-ma01

Course Prerequisites

- **Essential**
  - **Data Structures**
    - Array
    - List
    - Binary Search Tree
      - Balanced Tree
    - B-Tree
    - Hash Table / Map
    - **MOOCs: Design and Analysis of Algorithms**: https://nptel.ac.in/noc/individual_course.php?id=noc17-cs27
    - **MOOCs: Fundamental Algorithms – Design and Analysis**: https://nptel.ac.in/noc/individual_course.php?id=noc16-cs24

Course Prerequisites

- **Essential**
  - **Algorithms and Programming in C**
    - Sorting
      - Merge Sort
      - Quick Sort
    - Search
      - Linear Search
      - Binary Search
      - Interpolation Search
    - **MOOCs: Design and Analysis of Algorithms**: https://nptel.ac.in/noc/individual_course.php?id=noc17-cs27
    - **MOOCs: Introduction to Programming in C**: https://nptel.ac.in/noc/individual_course.php?id=noc17-cs43

Course Prerequisites

- **Desirable**
  - **Object-Oriented Analysis and Design**
    - **MOOCs: Object-Oriented Analysis and Design**: https://nptel.ac.in/noc/individual_course.php?id=noc17-cs25
  - **Programming in C++ / Java**
    - **MOOCs: Programming in C++**: https://nptel.ac.in/noc/individual_course.php?id=noc17-cs24

Course Outline

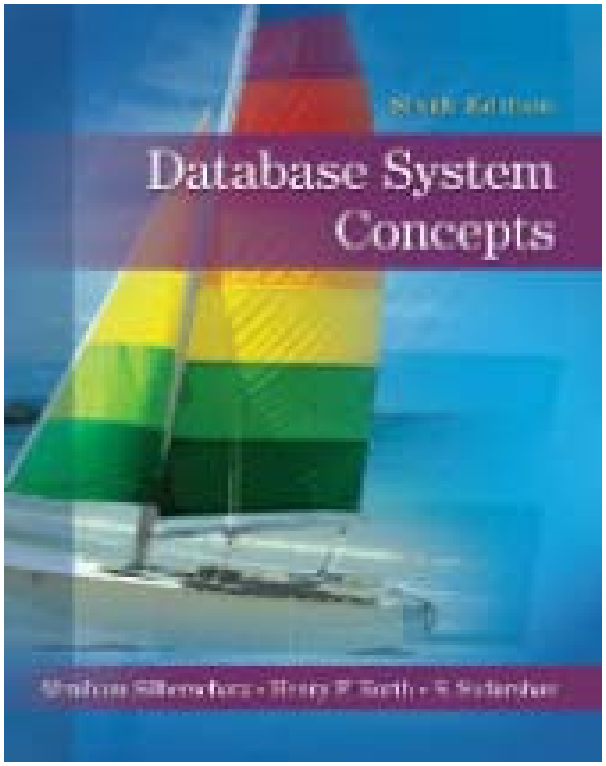
:

Week	Topics
Week 1	Course Overview Introduction to RDBMS
Week 2	Structured Query Language (SQL)
Week 3	Relational Algebra Entity-Relationship Model
Week 4	Relational Database Design
Week 5	Application Development Case Studies Storage and File Structure
Week 6	Indexing and Hashing Query Processing
Week 7	Query Optimization Transactions (Serializability and Recoverability)
Week 8	Concurrency Control Recovery Systems Course Summarization

Application Programmer

DBA / DBMS Developer

Course Textbook



Database System Concepts
Sixth Edition

Avi Silberschatz
Henry F. Korth
S. Sudarshan

McGraw-Hill
ISBN 0-07-352332-1

Website: http://db-book.com/

7th Edition will also do

Course TAs

- Srijoni Majumdar, majumdarsrijoni@gmail.com, 9674474267
- Himadri B G S Bhuyan, himadribhuyan@gmail.com, 9438911655
- Gurunath Reddy M, mgurunathreddy@gmail.com, 9434137638

PPD

Module Summary

- Elucidates the importance of database management systems in modern day applications
- Introduced various aspects of the Course

Instructor and TAs

Name	Mail	Mobile
Partha Pratim Das, Instructor	ppd@cse.iitkgp.ernet.in	9830030880
Srijoni Majumdar, TA	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, TA	himadribhuyan@gmail.com	9438911655
Gurunath Reddy M	mgurunathreddy@gmail.com	9434137638

Slides used in this presentation are borrowed from http://db-book.com/ with kind permission of the authors.

Edited and new slides are marked with "PPD".

Database Management Systems

Module 02: Introduction to DBMS/1

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Srijoni Majumdar
Himadri B G S Bhuyan
Gurunath Reddy M

Database System Concepts, 6th Ed.

www.db-book.com

Slides marked with 'PPD' are new or edited

Module Recap

- Why Databases?
- KYC: Know Your Course
  - Course Prerequisite
  - Course Outline
  - Course Text Book
  - Course TAs

Module Objectives

- To familiarize with the basic notions and terminology of database management systems
- To understand the role of data models and languages
- To understand the approaches to database design

Module Outline

- Levels of Abstraction
- Schema & Instance
- Data Models
  - Relational Databases
- DDL & DML
- SQL
- Database Design

- **Levels of Abstraction**
- Schema & Instance
- Data Models
- DDL & DML
- SQL
- Database Design

LEVELS OF ABSTRACTION

Levels of Abstraction

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

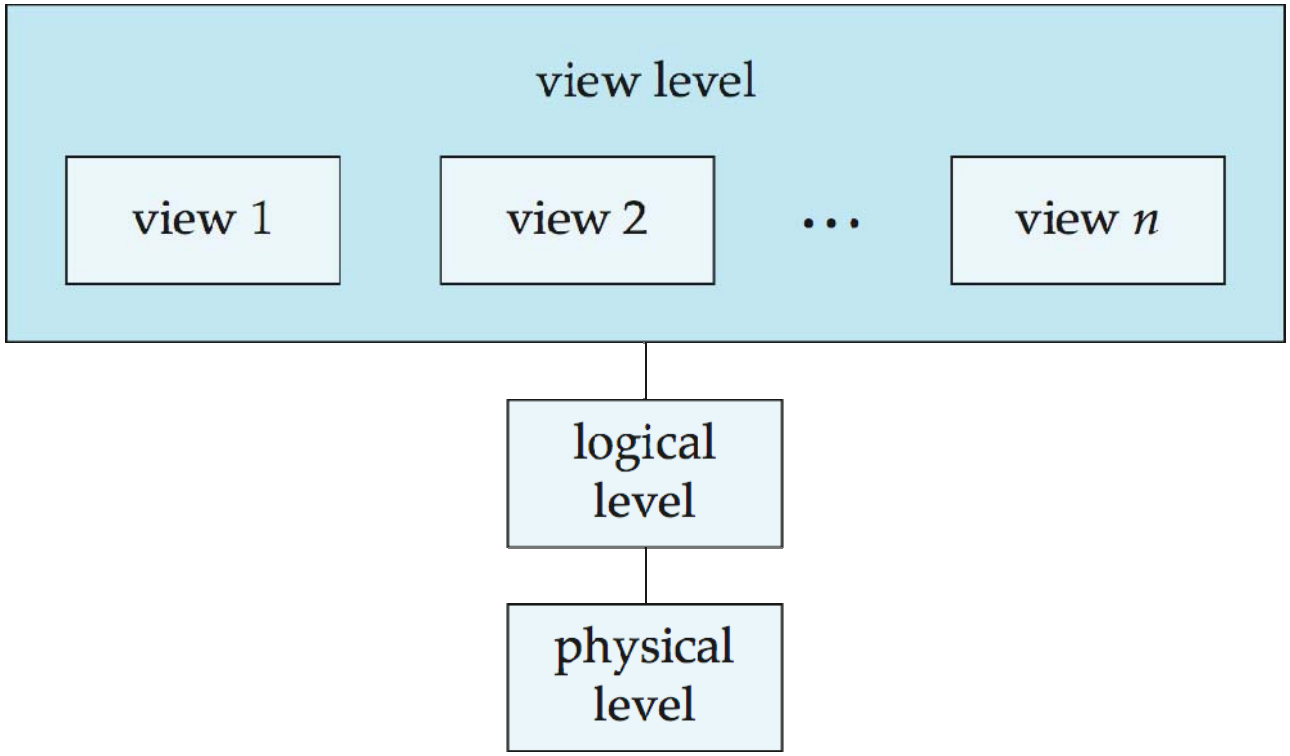
  type *instructor* = **record**

  ID : string;
  name : string;
  dept_name : string;
  salary : 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



- Levels of Abstraction
- **Schema & Instance**
- Data Models
- DDL & DML
- SQL
- Database Design

SCHEMA AND INSTANCE

Schemas and Instances

- Similar to types and variables in programming languages
- **Schema**
  - **Logical Schema** – the overall logical structure of the database
    - Analogous to type information of a variable in a program
    - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
    - Customer Schema

Name	Customer ID	Account #	Aadhaar ID	Mobile #

    - Account Schema

Account #	Account Type	Interest Rate	Min. Bal.	Balance

  - **Physical Schema**– the overall physical structure of the database

Schemas and Instances

- **Instance**
  - The actual content of the database at a particular point in time
  - Analogous to the value of a variable
  - Customer Instance

Name	Customer ID	Account #	Aadhaar ID	Mobile #
Pavan Laha	6728	917322	182719289372	9830100291
Lata Kala	8912	827183	918291204829	7189203928
Nand Prabhu	6617	372912	127837291021	8892021892

  - Account Instance

Account #	Account Type	Interest Rate	Min. Bal.	Balance
917322	Savings	4.0%	5000	7812
372912	Current	0.0%	0	291820
827183	Term Deposit	6.75%	10000	100000

Schemas and Instances

- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Analogous to independence of ‘Interface’ and ‘Implementation’ in Object-Oriented Systems
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

- Levels of Abstraction
- Schema & Instance
- **Data Models**
- DDL & DML
- SQL
- Database Design

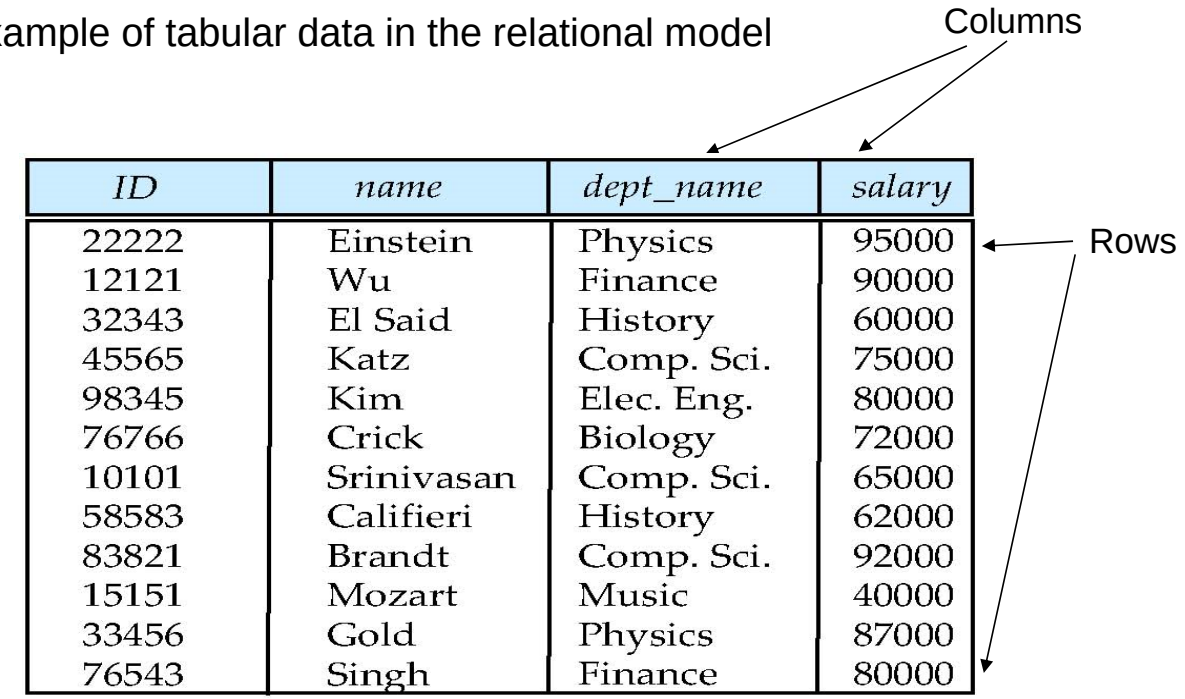
DATA MODELS

Data Models

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- **Relational model** (we focus in this course)
- Entity-Relationship data model (mainly for database design)
- Object-based data data models (Object-oriented and Object-relational)
- Semi-structured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model

Relational Model

- All the data is stored in various tables
- Example of tabular data in the relational model



ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

A Sample Relational Database

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

- Levels of Abstraction
- Schema & Instance
- Data Models
- **DDL & DML**
- SQL
- Database Design

DDL AND DML

Data Definition Language (DDL)

- Specification notation for defining the database schema
  - Example: **create table** *instructor* (
    ID **char**(5),
    name **varchar**(20)**,**
    dept_name **varchar**(20),
    salary **numeric**(8,2))
- DDL compiler generates a set of table templates stored in a ***data dictionary***
- Data dictionary contains metadata (i.e., data about data)
  - Database schema
  - Integrity constraints
    - Primary key (ID uniquely identifies instructors)
  - Authorization
    - Who can access what

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
  - **Pure** – used for proving properties about computational power and for optimization
    - **Relational Algebra** (we focus in this course)
    - Tuple relational calculus
    - Domain relational calculus
  - **Commercial** – used in commercial systems
    - SQL is the most widely used commercial language

- Levels of Abstraction
- Schema & Instance
- Data Models
- DDL & DML
- **SQL**
- Database Design

SQL

SQL

- The most widely used commercial language
- SQL is NOT a Turing machine equivalent language
  - Cannot be used to solve all problems that a C program, for example, can solve
- To be able to compute complex functions SQL is usually embedded in some higher-level language
- 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

- Levels of Abstraction
- Schema & Instance
- Data Models
- DDL & DML
- SQL
- **Database Design**

DATABASE DESIGN

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

Database Design (Cont.)

- Is there any problem with this relation?

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Module Summary

- Familiarized with the basic notions and terminology of database management systems
- Introduced the role of data models and languages
- Introduced the approaches to database design

Instructor and TAs

Name	Mail	Mobile
Partha Pratim Das, Instructor	ppd@cse.iitkgp.ernet.in	9830030880
Srijoni Majumdar, TA	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, TA	himadribhuyan@gmail.com	9438911655
Gurunath Reddy M	mgurunathreddy@gmail.com	9434137638

Slides used in this presentation are borrowed from http://db-book.com/ with kind permission of the authors.

Edited and new slides are marked with "PPD".

Database Management Systems

Module 03: Introduction to DBMS/2

Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Srijoni Majumdar
Himadri B G S Bhuyan
Gurunath Reddy M

Database System Concepts, 6th Ed.

Slides marked with 'PPD' are new or edited

Module Recap

- Levels of Abstraction
- Schema & Instance
- Data Models
  - Relational Databases
- DDL & DML
- SQL
- Database Design

Module Objectives

- To understand models of database management systems
- To familiarize with major components of a database engine
- To familiarize with database internals and architecture
- To understand the historical perspective

Module Outline

- Database Design
- OO Relational Model
- XML
- Database Engine
  - Storage Management
  - Query Processing
  - Transaction Management
- Database Users and Administrators
- Database Internals & Architecture
- History of DBMS

- **Database Design**
- OO Relational Model
- XML
- Database Engine
- Database Users and Administrators
- Database Internals & Architecture
- History of DBMS

DATABASE DESIGN

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

Database Design (Cont.)

- Is there any problem with this relation?

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Design Approaches

- Need to come up with a methodology to ensure that each of the relations in the database is “good”
- Two ways of doing so:
  - Entity Relationship Model (Chapter 7)
    - Models an enterprise as a collection of *entities* and *relationships*
    - Represented diagrammatically by an *entity-relationship diagram:*
  - Normalization Theory (Chapter 8)
    - Formalize what designs are bad, and test for them

- Database Design
- **OO Relational Model**
- XML
- Database Engine
- Database Users and Administrators
- Database Internals & Architecture
- History of DBMS

OBJECT-RELATIONAL DATA MODELS

Object-Relational Data Models

- Relational model: flat, “atomic” values
- 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

- Database Design
- OO Relational Model
- **XML**
- Database Engine
- Database Users and Administrators
- Database Internals & Architecture
- History of DBMS

XML: EXTENSIBLE MARKUP LANGUAGE

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

- Database Design
- OO Relational Model
- XML
- **Database Engine**
- Database Users and Administrators
- Database Internals & Architecture
- History of DBMS

DATABASE ENGINE

Database Engine

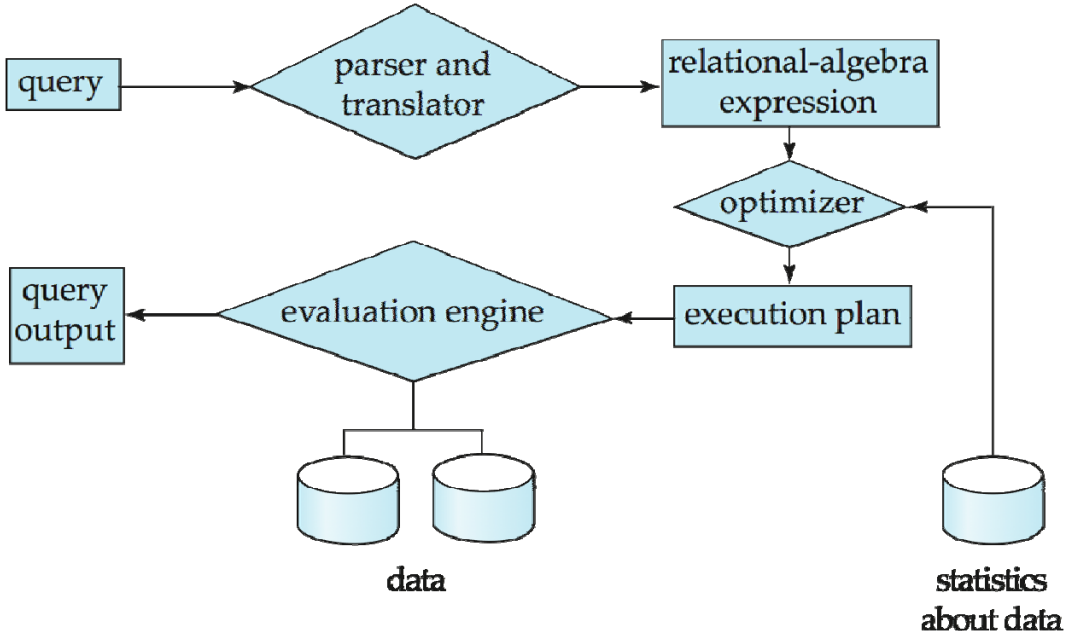
- Storage manager
- Query processing
- Transaction manager

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 OS file manager
  - Efficient storing, retrieving and updating of data
- Issues:
  - Storage access
  - File organization
  - Indexing and hashing

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

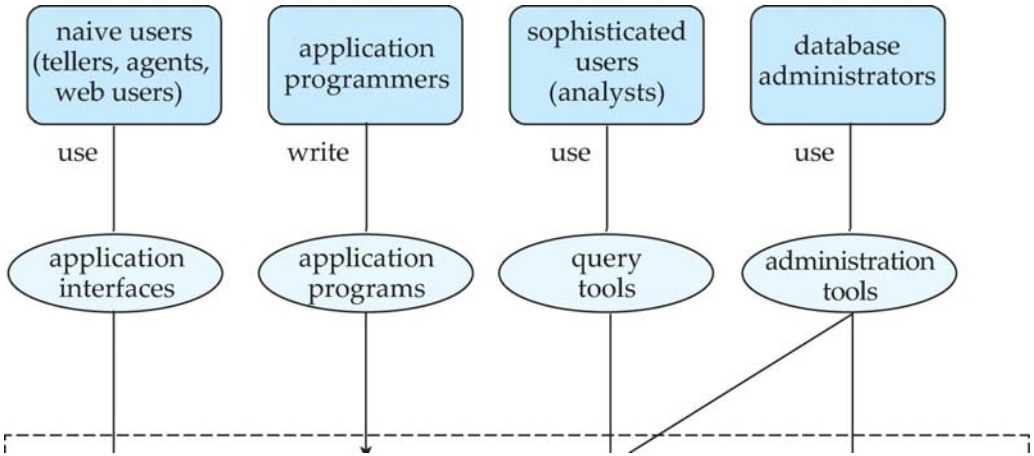
Transaction Management

- What if the system fails?
- What if more than one user is concurrently updating the same data?
- 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 Design
- OO Relational Model
- XML
- Database Engine
- **Database Users and Administrators**
- Database Internals & Architecture
- History of DBMS

DATABASE USERS AND ADMINISTRATOR

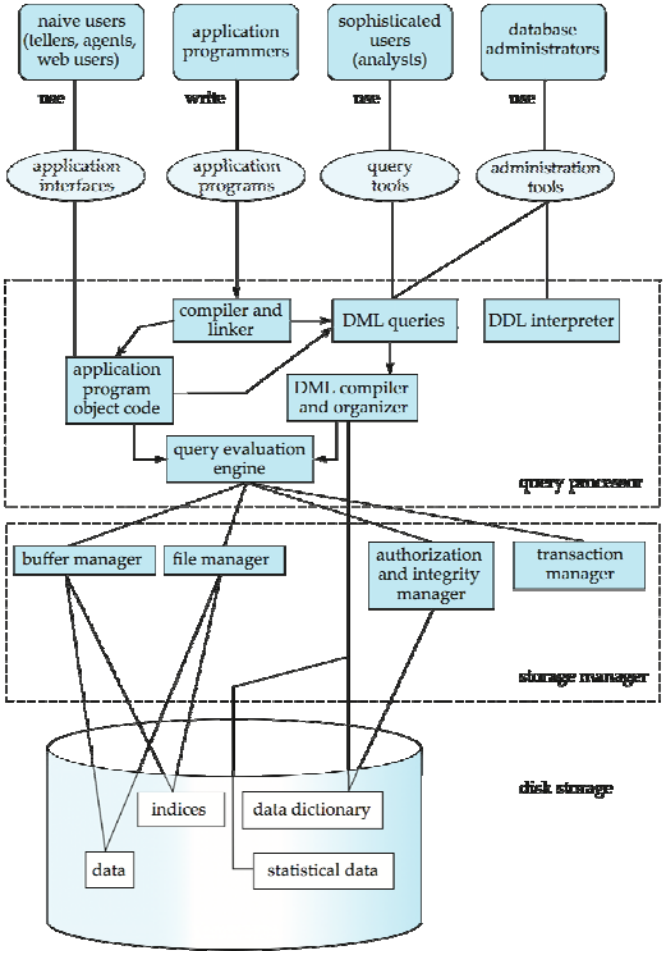
Database Users and Administrators



- Database Design
- OO Relational Model
- XML
- Database Engine
- Database Users and Administrators
- **Database Internals & Architecture**
- History of DBMS

DATABASE INTERNALS AND ARCHITECTURE

Database System Internals



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

- Database Design
- OO Relational Model
- XML
- Database Engine
- Database Users and Administrators
- Database Internals & Architecture
- **History of DBMS**

HISTORY OF DBMS

History of Database Systems

- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage
    - Tapes provided only sequential access
  - Punched cards for input
- Late 1960s and 1970s:
  - Hard disks allowed 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
- Early 2000s:
  - XML and XQuery standards
  - Automated database administration
- Later 2000s:
  - Giant data storage systems
    - Google BigTable, Yahoo PNuts, Amazon, ..

Module Summary

- Introduced models of database management systems
- Familiarized with major components of a database engine
- Familiarized with database internals and architecture

Instructor and TAs

Name	Mail	Mobile
Partha Pratim Das, Instructor	ppd@cse.iitkgp.ernet.in	9830030880
Srijoni Majumdar, TA	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, TA	himadribhuyan@gmail.com	9438911655
Gurunath Reddy M	mgurunathreddy@gmail.com	9434137638

Slides used in this presentation are borrowed from http://db-book.com/ with kind permission of the authors.

Edited and new slides are marked with "PPD".

Database Management Systems

Module 04: Introduction to Relational Model/1

Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Srijoni Majumdar
Himadri B G S Bhuyan
Gurunath Reddy M

Database System Concepts, 6th Ed.

Slides marked with 'PPD' are new or edited

Module Recap

- Database Design
- OO Relational Model
- XML
- Database Engine
  - Storage Management
  - Query Processing
  - Transaction Management
- Database Users and Administrators
- Database Internals & Architecture
- History of DBMS

Module Objectives

- To understand attributes and their types
- To understand the mathematical structure of relational model – schema, instance and keys
- To familiarize with different types of relational query languages

Module Outline

- Attribute Types
- Relation Schema and Instance
- Keys
- Relational Query Languages

Example of a Relation

attributes
(or columns)

ID	*name*	*dept_name*	*salary*
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

tuples
(or rows)

- **Attribute Types**
- Relation Schema and Instance
- Keys
- Relational Query Languages

ATTRIBUTES

Attribute Types

- The set of allowed values for each attribute is called the **domain** of the attribute
  - **Roll #**: Alphanumeric string
  - **First Name, Last Name**: Alpha String
  - **DoB**: Date
  - **Passport #**: String (Letter followed by 7 digits) – nullable
  - **Aadhaar #**: 12-digit number
  - **Department**: Alpha String
- Attribute values are (normally) required to be **atomic**; that is, indivisible
- The special value ***null*** is a member of every domain. Indicated that the value is "unknown"
- The null value causes complications in the definition of many operations

Roll #	First Name	Last Name	DoB	Passport #	Aadhaar #	Department
15CS10026	Lalit	Dubey	27-Mar-1997	L4032464	1728-6174-9239	Computer
16EE30029	Jatin	Chopra	17-Nov-1996	null	3917-1836-3816	Electrical

- Attribute Types
- **Relation Schema and Instance**
- Keys
- Relational Query Languages

SCHEMA AND INSTANCE

Relation Schema and Instance

- $A_1, A_2, \ldots, A_n$ are *attributes*
- $R = (A_1, A_2, \ldots, A_n)$ is a *relation schema*

  Example:

  instructor = (*ID, name, dept_name, salary*)

- Formally, given sets $D_1, D_2, \ldots. D_n$ a **relation *r*** is a subset of

  $D_1 \times D_2 \times \ldots \times D_n$

  Thus, a relation is a set of *n*-tuples $(a_1, a_2, \ldots, a_n)$ where each $a_i \in D_i$
- 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

Relations are Unordered

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

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

- Attribute Types
- Relation Schema and Instance
- **Keys**
- Relational Query Languages

KEYS

Keys

- Let K $\subseteq$ R
- *K* is a **superkey** of *R* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)*
  - Example: {*ID*} and {ID,name} are both superkeys of *instructor*
- Superkey *K* is a **candidate key** if *K* is minimal
  - Example: {*ID*} is a candidate key for *Instructor*
- One of the candidate keys is selected to be the **primary key**
  - Which one?
- A **surrogate key** (or synthetic key) in a database is a unique identifier for either an *entity* in the modeled world or an *object* in the database
  - The surrogate key is *not* derived from application data, unlike a *natural* (or *business*) key which is derived from application data

Keys

- **Super Key**: Roll #, {Roll #, DoB}
- **Candidate Keys**: Roll #, {First Name, Last Name}, Passport #, Aadhaar #
  - Passport # cannot be a key. Why?
  - Null values are allowed for Passport # (a student may not have a passport)
- **Primary Key**: Roll #
- **Secondary / Alternate Key**: {First Name, Last Name}, Aadhaar #
- **Simple key**: Consists of a *single attribute*
- **Composite Key**: {First Name, Last Name}
  - Consists of more than one attribute to uniquely identify an entity occurrence
  - One or more of the attributes, which make up the key, are not simple keys in their own right

Roll #	First Name	Last Name	DoB	Passport #	Aadhaar #	Department
15CS10026	Lalit	Dubey	27-Mar-1997	L4032464	1728-6174-9239	Computer
16EE30029	Jatin	Chopra	17-Nov-1996	null	3917-1836-3816	Electrical
15EC10016	Smriti	Mongra	23-Dec-1996	G5432849	2045-9271-0914	Electronics
16CE10038	Dipti	Dutta	02-Feb-1997	null	5719-1948-2918	Civil
15CS30021	Ramdin	Minz	10-Jan-1997	X8811623	4928-4927-5924	Computer

Keys

- **Foreign key** constraint: Value in one relation must appear in another
  - **Referencing** relation
    - Enrolment: Foreign Keys – Roll #, Course #
  - **Referenced** relation
    - Students, Courses
- A **compound key** consists of *more than one attribute* to uniquely identify an entity occurrence
  - Each attribute, which makes up the key, is a simple key in its own right
  - {Roll #, Course #}

Students

Roll #	First Name	Last Name	DoB	Passport #	Aadhaar #	Department

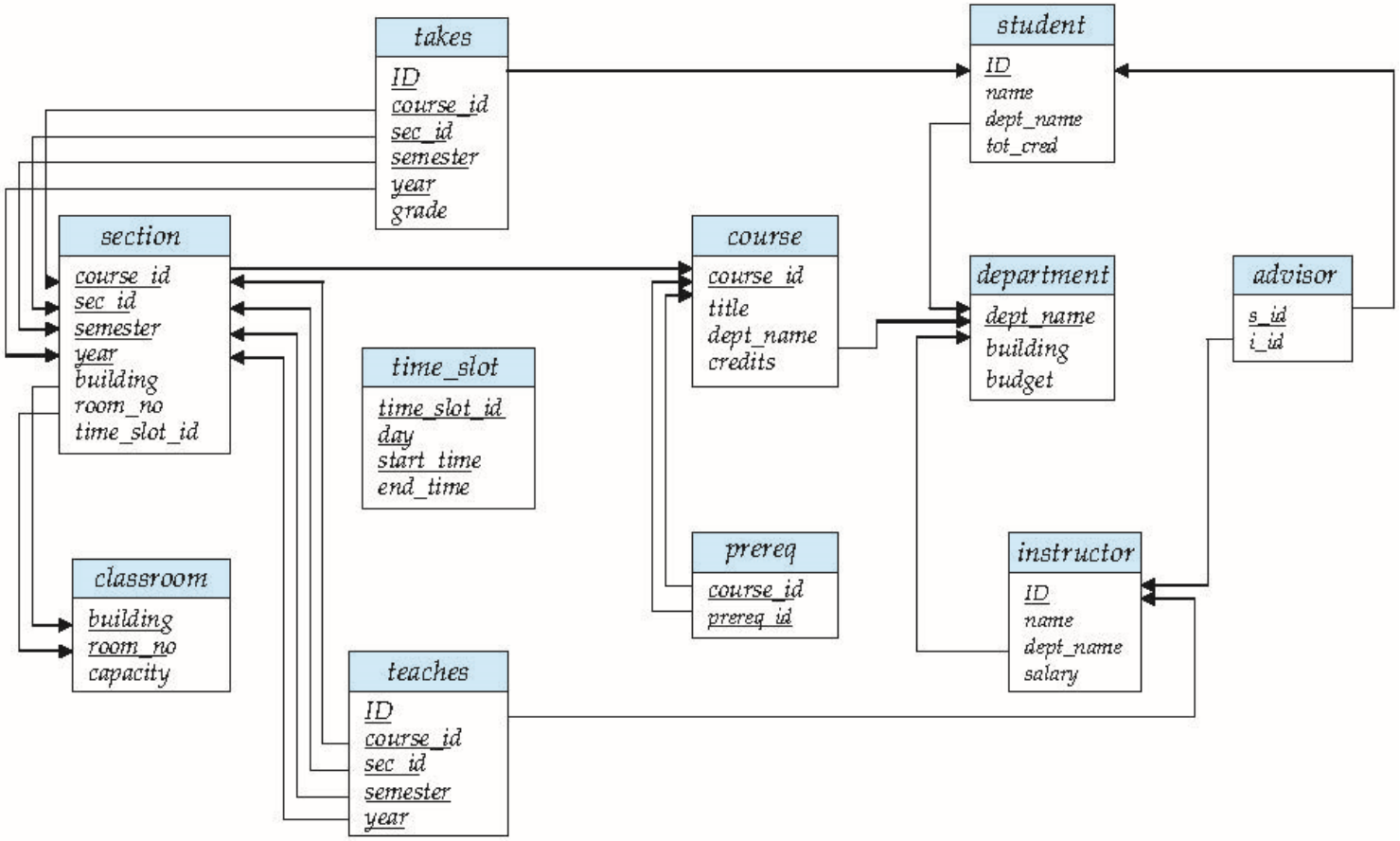
Courses

Course #	Course Name	Credits	L-T-P	Department

Enrolment

Roll #	Course #	Instructor ID

Schema Diagram for University Database



- Attribute Types
- Relation Schema and Instance
- Keys
- **Relational Query Languages**

RELATIONAL QUERY LANGUAGES

Relational Query Languages

Procedural vs. Non-procedural or Declarative Paradigms

- **Procedural programming** requires that the programmer tell the computer what to do
  - That is, ***how*** to get the output for the range of required inputs
  - The programmer must know an appropriate algorithm
- **Declarative programming** requires a more descriptive style
  - The programmer must know ***what*** relationships hold between various entities
- **Example: Square root of n**
  - Procedural
    1. Guess x_0 (close to root of n)
    2. i ← 0
    3. $x_{i+1} \leftarrow (x_i + n / x_i) / 2$
    4. Repeat Step 2 if $| x_{i+1} - x_i | >$ delta
  - Declarative
    - Root of n is m such that $m^2 = n$

Relational Query Languages

- "Pure" languages:
  - Relational algebra
  - Tuple relational calculus
  - Domain relational calculus
- The above 3 pure languages are equivalent in computing power
- We will concentrate on relational algebra
  - Not Turing-machine equivalent
    - Not all algorithms can be expressed in RA
  - Consists of 6 basic operations

Module Summary

- Introduced the notion of attributes and their types
- Taken an overview of the mathematical structure of relational model – schema and instance
- Introduced the notion of keys – primary as well as foreign

Instructor and TAs

Name	Mail	Mobile
Partha Pratim Das, Instructor	ppd@cse.iitkgp.ernet.in	9830030880
Srijoni Majumdar, TA	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, TA	himadribhuyan@gmail.com	9438911655
Gurunath Reddy M	mgurunathreddy@gmail.com	9434137638

Slides used in this presentation are borrowed from http://db-book.com/ with kind permission of the authors.

Edited and new slides are marked with "PPD".

Database Management Systems

Module 05: Introduction to Relational Model/2

Partha Pratim Das
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ernet.in

Srijoni Majumdar
Himadri B G S Bhuyan
Gurunath Reddy M

Database System Concepts, 6th Ed.

Slides marked with 'PPD' are new or edited

Module Recap

- Attribute Types
- Relation Schema and Instance
- Keys
- Relational Query Languages

Module Objectives

- To understand relational algebra
- To familiarize with the operators of relational algebra

Module Outline

- Operations
  - Select
  - Project
  - Union
  - Difference
  - Intersection
  - Cartesian Product
  - Natural Join
- Aggregate Operations

- **Operations**
- Aggregate Operations

RELATIONAL OPERATORS

Select Operation – selection of rows (tuples)

- Relation r

A	B	C	D
α	α	1	7
α	β	5	7
β	β	12	3
β	β	23	10

- $\sigma_{A=B \wedge D > 5}(r)$

A	B	C	D
α	α	1	7
β	β	23	10

Project Operation – selection of columns (Attributes)

- Relation *r*:

A	*B*	*C*
α	10	1
α	20	1
β	30	1
β	40	2

- $\Pi_{A,C}(r)$

A	*C*
α	1
α	1
β	1
β	2

=

A	*C*
α	1
β	1
β	2

Union of two relations

- Relations *r, s:*

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- r ∪ s:

A	B
α	1
α	2
β	1
β	3

Set difference of two relations

- Relations *r*, *s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- *r – s:*

A	B
α	1
β	1

Set intersection of two relations

- Relation *r, s*:

A	B
α	1
α	2
β	1

r

A	B
α	2
β	3

s

- $r \cap s$

A	B
α	2

Note: $r \cap s = r - (r - s)$

Joining two relations -- Cartesian-product

- Relations *r, s*:

A	*B*
α	1
β	2

r

C	*D*	*E*
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r* x *s*:

A	*B*	*C*	*D*	*E*
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Cartesian-product – naming issue

- Relations *r, s*:

A	*B*
α	1
β	2

r

B	*D*	*E*
α	10	a
β	10	a
β	20	b
γ	10	b

s

- *r* x *s*:

A	*r.B*	*s.B*	*D*	*E*
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

Renaming a Table

- Allows us to refer to a relation, (say E) by more than one name.

$$\rho_x(E)$$

returns the expression *E* under the name *X*

- Relations *r*

A	*B*
α	1
β	2

r

- *r* x ρ_s (r)

r.A	*r.B*	*s.A*	*s.B*
α	1	α	1
α	1	β	2
β	2	α	1
β	2	β	2

Composition of Operations

- Can build expressions using multiple operations
- Example: $\sigma_{A=C}(r \times s)$

- $r \times s$

A	B	C	D	E
α	1	α	10	a
α	1	β	10	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

- $\sigma_{A=C}(r \times s)$

A	B	C	D	E
α	1	α	10	a
β	2	β	10	a
β	2	β	20	b

Joining two relations – Natural Join

- Let *r* and *s* be relations on schemas *R* and *S* respectively. Then, the "natural join" of relations *R* and *S* is a relation on schema $R \cup S$ obtained as follows:
  - Consider each pair of tuples t_r from *r* and t_s from *s*.
  - If t_r and t_s have the same value on each of the attributes in $R \cap S$, add a tuple *t* to the result, where
    - *t* has the same value as t_r on *r*
    - *t* has the same value as t_s on *s*

Natural Join Example

- Relations r, s:

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

r

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	ε

s

- Natural Join
  - r ⋈ s

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

$$\Pi_{A,\, r.B,\, C,\, r.D,\, E}(\sigma_{r.B = s.B \wedge r.D = s.D}(r \times s))$$

- Operations
- **Aggregate Operations**

AGGREGATION OPERATORS

Aggregate Operators

- Can we compute:
  - SUM
  - AVG
  - MAX
  - MIN

Notes about Relational Languages

- Each Query input is a table (or set of tables)
- Each query output is a table
- All data in the output table appears in one of the input tables
- Relational Algebra is not Turning complete

Summary of Relational Algebra Operators

Symbol (Name)	Example of Use
σ (Selection)	$\sigma_{salary >= 85000}(instructor)$
	Return rows of the input relation that satisfy the predicate.
Π (Projection)	$\Pi_{ID, salary}(instructor)$
	Output specified attributes from all rows of the input relation. Remove duplicate tuples from the output.
x (Cartesian Product)	*instructor* x *department*
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.
∪ (Union)	$\Pi_{name}(instructor) \cup \Pi_{name}(student)$
	Output the union of tuples from the *two* input relations.
– (Set Difference)	$\Pi_{name}(instructor) - \Pi_{name}(student)$
	Output the set difference of tuples from the two input relations.
⋈ (Natural Join)	*instructor* ⋈ *department*
	Output pairs of rows from the two input relations that have the same value on all attributes that have the same name.

Module Summary

- Introduced relational algebra
- Familiarized with the operators of relational algebra

Instructor and TAs

Name	Mail	Mobile
Partha Pratim Das, Instructor	ppd@cse.iitkgp.ernet.in	9830030880
Srijoni Majumdar, TA	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, TA	himadribhuyan@gmail.com	9438911655
Gurunath Reddy M	mgurunathreddy@gmail.com	9434137638

Slides used in this presentation are borrowed from http://db-book.com/ with kind permission of the authors.

Edited and new slides are marked with "PPD".