Database Management Systems (CPTR 312)

Preliminaries

- Me: Raheel Ahmad
  - Ph.D., Southern Illinois University
  - M.S., University of Southern Mississippi
  - B.S., Zakir Hussain College, India
- Contact: Science 116, rahmad@manchester.edu, 982-5314
  - Tues: 9:00 am - 12:00 am, Thurs: 10:00 am - 12:00 am
- Email me with subject starting with **CPTR312**
- http://users.manchester.edu/Facstaff/RAhmad/classes/312/index.htm
  - Also, Angel's course webpage has a link to above

Preliminaries

- Course
    - Science 142, MWThF: 9 – 9:50 am
- Databases:
    - Crucial
    - Insightful
    - Challenging
- Discuss problems early, often
- Assignments, quizes, tests
- Slides will be available online
- Keep up to date with the deadlines and due dates

Introduction to Databases

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 for a community of users
  - 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
  - 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; *most pervasive software*

History

- 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

History

- 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 access 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 is stored.
    - data structures used; byte level strorage
- **Logical level**: describes the data stored in database, and the relationships among the data.
- lowest level at which programmers and admin interact with database

type *customer* = **record**

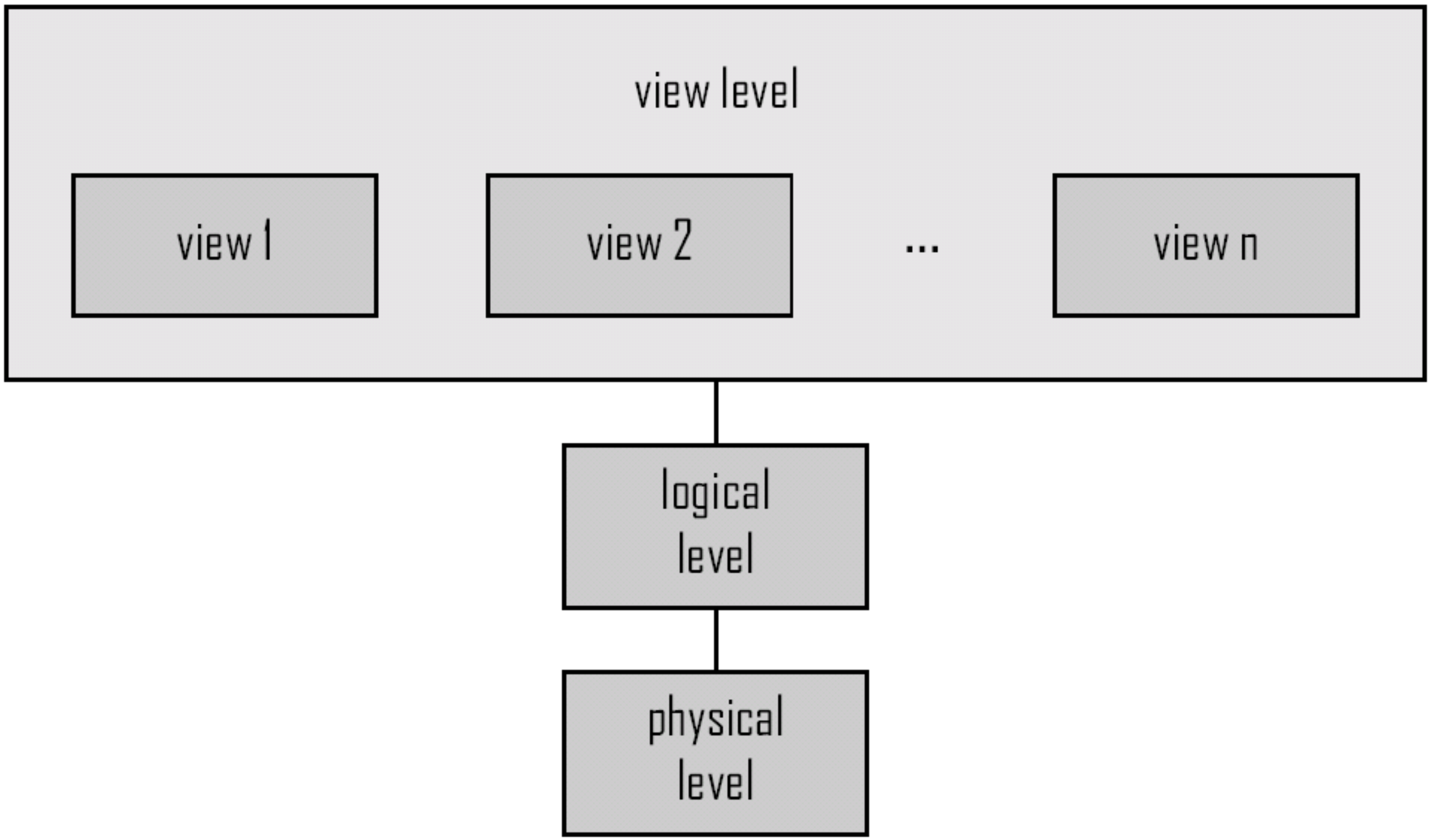
customer_id : string;
customer_name : string;
customer_street : string;
customer_city : integer;

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



Schemas and Instances

- Similar to types and variables in programming languages
- Schema – the logical structure of the database (at every level) ; rarely changes
    - E.g.: DB consists of information for set of customers, accounts, & their relationships
    - 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; most important
    - **View schemas** (*subschemas*)
- Instance – the actual content of the database at a particular point in time
    - Analogous to the value of a variable
- Physical Data Independence – the ability to modify the physical schema without changing the logical schema
    - Applications depend on the logical schema
    - interfaces between various levels should be well defined so that changes in some parts do not seriously influence others.

Data Models

- A collection of conceptual tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- Relational model
  - tables; most widely used
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semi-structured 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
  - Retrieval, insertion, deletion, modification
  - **Query**: statement in DML requesting information
  - DML also known as query language (technically incorrect)
- SQL is the most widely used 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
- Abstraction: DML => physical level algorithms
  - ease of use

Data Definition Language (DDL)

- For defining the database schema
  - Example: **create table** account (
    account-number char(10),
    balance integer)
- Integrity constraints
  - **Domain** constraints (integer, character, date)
  - **Referential** integrity (referenced values across relations)
  - **Assertions** (always valid condition)
    - “every user with loan must have >$1000 balance”
  - **Authorization** (read, insert, modify, delete)
- DDL compiler generates output: a set of tables stored in a **data dictionary**
- Data dictionary (table) contains **metadata** (i.e., data about data)
  - Database schema
  - DD consuted before reading/modifying data

Relational Model

Uses **tables** for data & relations between data
Usually employs **SQL**

A Sample Relational Database

Attributes

Records

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

A table: multiple columns
A column: unique name

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

Relational Model

- Bad schema

Duplication

customer_id	*customer_name*	*customer_street*	*customer_city*	*account_number*
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

- Tables may be stored in files
  - Relational model hides such low-level implementation details

SQL

- SQL: widely used non-procedural language
- **Input**: set of tables + **Constraints** -------> **Output**: 1 table

SQL Query Example I

Find the name of the customer with customer-id 192-83-7465:

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

select *customer.customer_name* — *output*
from *customer* — *input*
where *customer.customer_id* = '192-83-7465' — *constraints*

customer_name
Johnson

SQL Query Example II

Find all customers living in Harrison

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

select *customer.customer_name*
from *customer*
where *customer.customer_city* = 'Harrison'

↓

customer_name
Hayes
Jones

SQL Query Example III

Find the balances of all accounts held by the customer with customer-id 192-83-7465

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

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

↓

account_number	*balance*
A-101	500
A-201	900

SQL DDL

- Provides a rich DDL

create table *account*
(*account_number* **char**(10),
balance **integer**)

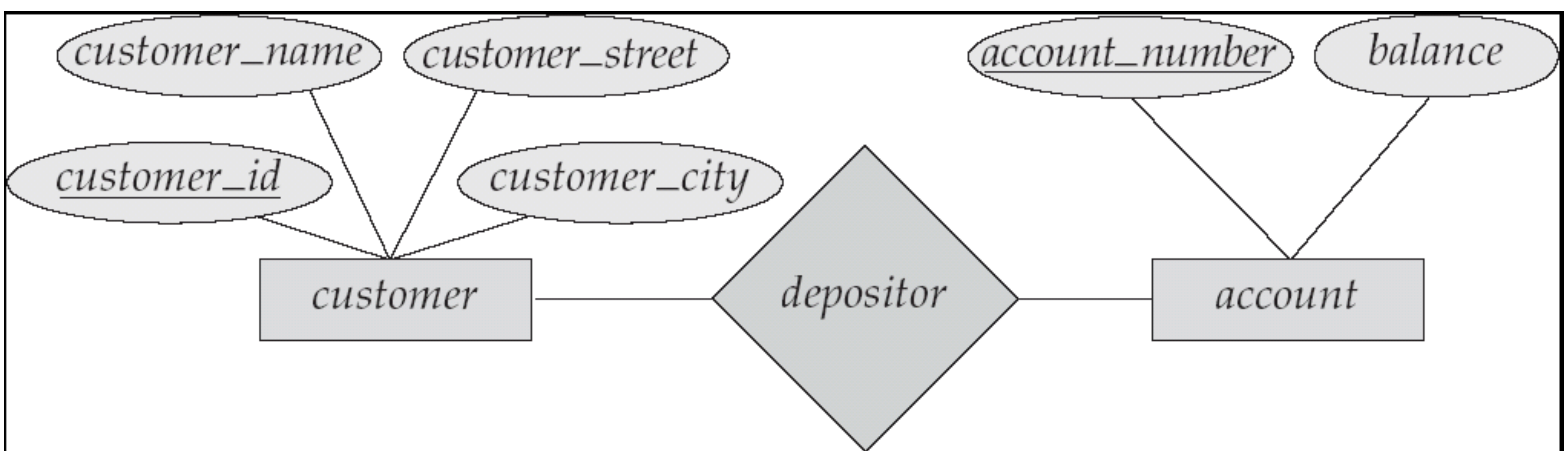
  - creates the *account* table
  - updates data dictionary
- Application programs
  - written in *host* language: C++, Java
  - embeds SQL queries to access data
- Language provides API to send DDL/DML to DB
  - ODBC, JDBC

Database Design

- The process of designing the structure of database (**schema**)
  - everything before data is entered
- User requirements **specification**
- Translate to chosen data model's schema (**conceptual-design**)
  - Relational: which attributes, how to group them into tables
- Check if meets **functional requirements**: e.g. operations to search, update, modify
- Moving to implementation: **logical** & **physical** design
- Logical Design:
  - from conceptual schema to **implementation**: SQL commands
- 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:



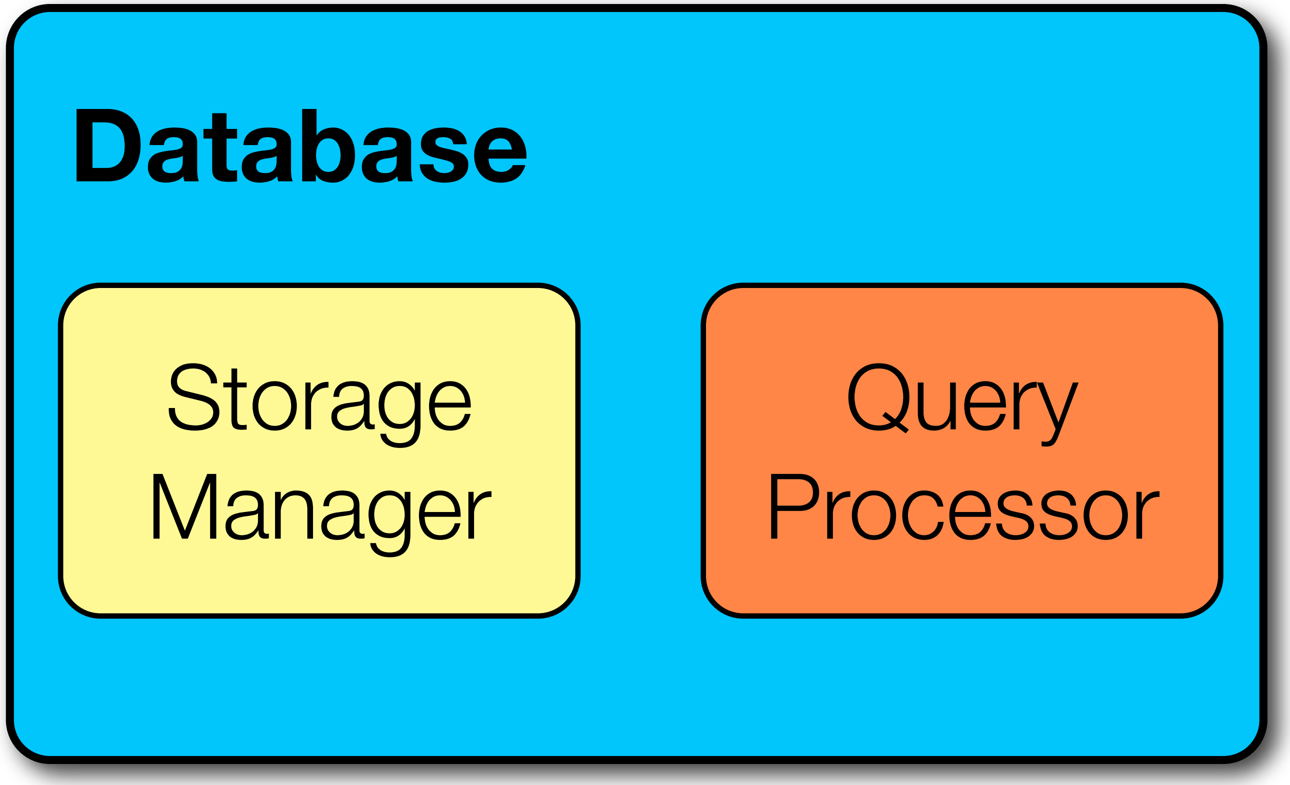
Object-Relational Data Models

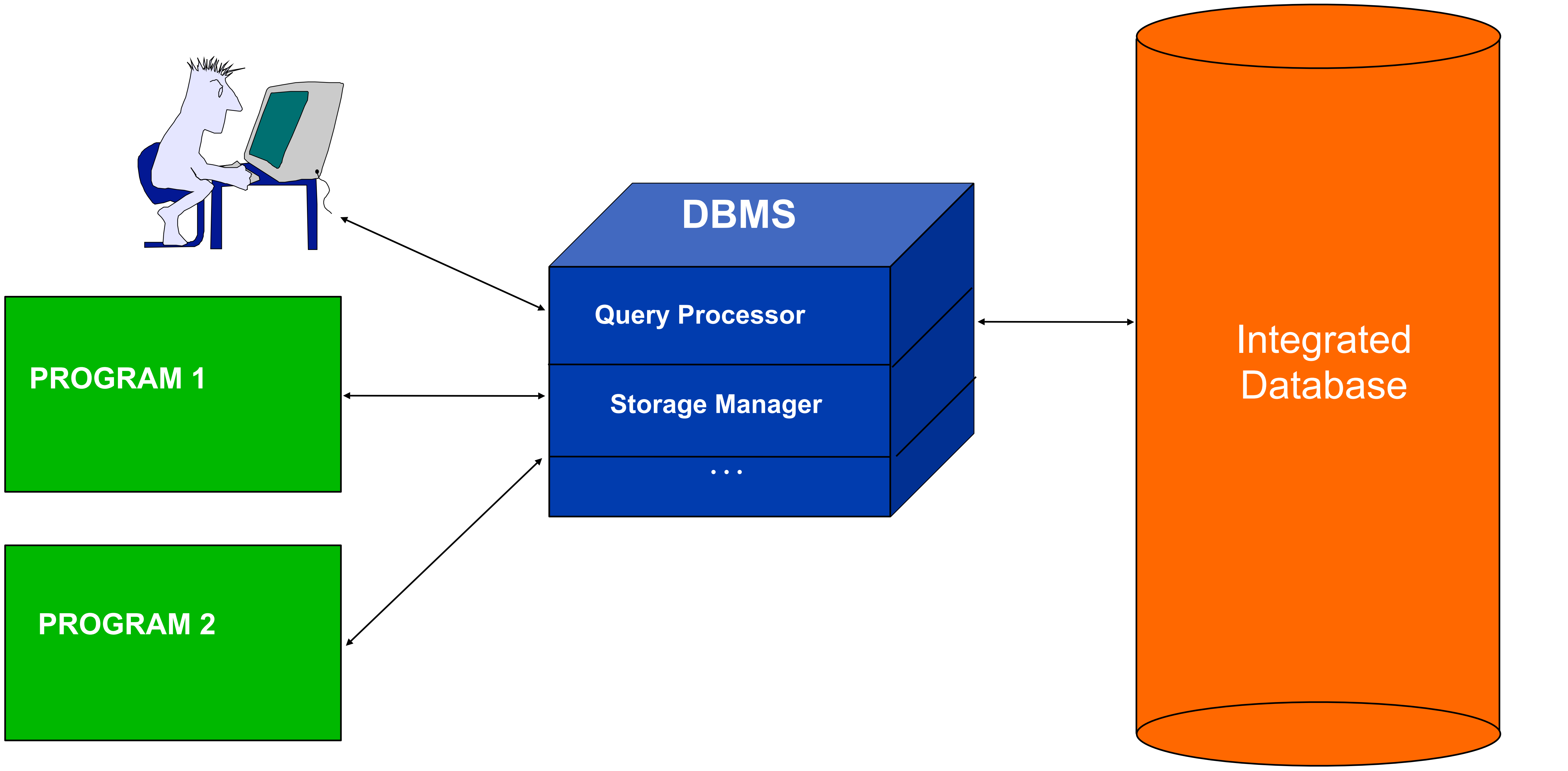
- Extend the relational data model by including **object–orientation**
  - object-identity
  - inheritance
  - encapsulation (information-hiding)
- Allow attributes of a row to have complex types
- **Preserve relational foundations**, in particular the declarative access to data, while extending modeling power

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
- Data of same type, with different attributes
  - flexibility

Data Storage & Querying





DBMS
Query Processor
Storage Manager
. . .
PROGRAM 1
PROGRAM 2
Integrated Database

Storage Management

- Interfaces the low-level data and the application programs and queries
- The storage manager is responsible to the following tasks:
  - Interaction with the file manager (translates DML to filesystem commands)
  - Efficient storing, retrieving and updating of data
- Manages **data files**, **data dictionary**, **indices**
- Issues:
  - File organization
  - Storage access
  - Indexing and hashing

Query Processor

- **DDL Interpreter**
  - interprets DDL statements for the data dictionary
- **DML Compiler**
  - translates DML statements into low-level instructions
- **Query evaluation engine**
  - executes the low-level instructions generated by DML compiler

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

- Architecture influenced by **underlying computer system**
  - Centralized
  - Client-server
  - Parallel (multi-processor)
  - Distributed

Database Users

- **Application programmers** – write applications that interact with the DB
- **Sophisticated users** – Use query language, e.g., SQL
- **Specialized users** – write specialized database applications
- **Naïve users** – use an application written previously
  - Users accessing database over the web, bank tellers, clerical staff
  - Forms & reports

Database Administrator

- Coordinates all activities of the database system
- Database administrator's duties include:
  - Schema definition
  - Storage structure and access method definition
  - Specifying integrity constraints
  - Granting user authority to access the database
  - Monitoring performance and responding to changes in requirements

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

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