Database Management System (DBMS)
- DBMS contains information about a particular domain
  - 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!
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.
- **View level**: application programs hide details of data types. Views can also hide information (such as an employee’s salary) for security purposes.

```plaintext
type instructor = record
    id : string;
    name : string;
    dept_name : string;
    salary : integer;
end;
```

### View of Data

An architecture for a database system

```
view 1    view 2    ...    view n
```

### Instances and Schemas

- Similar to types and variables in programming languages
- **Schema** – the 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** – 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
  - 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.
- **Logical Data Independence** – the ability to modify the logical schema without changing the applications
  - For example, add new information to each employee

### 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
- **Other newer (or revived) models**:
  - **Key-value**

### Relational Model

- **Relational model** (Chapter 2)
- **Example of tabular data in the relational model**

```
<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>dept_name</th>
<th>salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>Physics</td>
<td>95000</td>
</tr>
<tr>
<td>12345</td>
<td>Wu</td>
<td>Finance</td>
<td>60000</td>
</tr>
<tr>
<td>32343</td>
<td>El Said</td>
<td>History</td>
<td>76000</td>
</tr>
<tr>
<td>45678</td>
<td>Katz</td>
<td>Comp. Sci.</td>
<td>69000</td>
</tr>
<tr>
<td>98765</td>
<td>Kim</td>
<td>ELEC</td>
<td>80000</td>
</tr>
<tr>
<td>78901</td>
<td>Crick</td>
<td>Biology</td>
<td>72000</td>
</tr>
<tr>
<td>10101</td>
<td>Sevinivasan</td>
<td>Comp. Sci.</td>
<td>60000</td>
</tr>
<tr>
<td>98765</td>
<td>Coulam</td>
<td>History</td>
<td>69000</td>
</tr>
<tr>
<td>80912</td>
<td>Brandt</td>
<td>Comp. Sci.</td>
<td>92000</td>
</tr>
<tr>
<td>15151</td>
<td>Mozart</td>
<td>Music</td>
<td>60000</td>
</tr>
<tr>
<td>33333</td>
<td>Gold</td>
<td>Physics</td>
<td>80000</td>
</tr>
<tr>
<td>76543</td>
<td>Sough</td>
<td>Finance</td>
<td>90000</td>
</tr>
</tbody>
</table>
```

(a) The instructor table

(b) The department table

---

©Silberschatz, Korth and Sudarshan
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 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)
    - Referential integrity (references constraint in SQL)
      - e.g., dept_name value in any instructor tuple must appear in department relation
  - Authorization

SQL

- SQL: widely used declarative (non-procedural) language
  - Example: Find the name of the instructor with ID 22222
    
    ```sql
    select name
    from instructor
    where instructor.ID = '22222'
    ```
  - Example: Find the ID and building of instructors in the Physics dept.
    
    ```sql
    select instructor.ID, department.building
    from instructor, department
    where instructor.dept_name = department.dept_name
      and department.dept_name = 'Physics'
    ```
- 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
- Chapters 3, 4 and 5

Database Design

- The process of designing the general structure of a database:
  - Logical Design – Deciding on the database schema. Database design requires that we find a "good" representation of the information from an application domain (e.g., banking) as a collection of relation schemas.
  - Business decision – What information 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?

- Is there any problem with this design?

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>salary</th>
<th>dept_name</th>
<th>building</th>
<th>budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>9500</td>
<td>Physics</td>
<td>Watson</td>
<td>70000</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>9000</td>
<td>Finance</td>
<td>Painter</td>
<td>120000</td>
</tr>
<tr>
<td>32343</td>
<td>El Said</td>
<td>6000</td>
<td>History</td>
<td>Painter</td>
<td>50000</td>
</tr>
<tr>
<td>45965</td>
<td>Katz</td>
<td>7500</td>
<td>Comp. Sci.</td>
<td>Taylor</td>
<td>100000</td>
</tr>
<tr>
<td>96345</td>
<td>Kim</td>
<td>8000</td>
<td>Elec. Eng.</td>
<td>Taylor</td>
<td>85000</td>
</tr>
<tr>
<td>76766</td>
<td>Crick</td>
<td>7200</td>
<td>Biology</td>
<td>Watson</td>
<td>90000</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>6500</td>
<td>Comp. Sci.</td>
<td>Taylor</td>
<td>100000</td>
</tr>
<tr>
<td>59583</td>
<td>Caliieri</td>
<td>6200</td>
<td>History</td>
<td>Painter</td>
<td>50000</td>
</tr>
<tr>
<td>83821</td>
<td>Brandt</td>
<td>9200</td>
<td>Comp. Sci.</td>
<td>Taylor</td>
<td>100000</td>
</tr>
<tr>
<td>15151</td>
<td>Montart</td>
<td>6000</td>
<td>Music</td>
<td>Packard</td>
<td>80000</td>
</tr>
<tr>
<td>33456</td>
<td>Gold</td>
<td>8700</td>
<td>Physics</td>
<td>Watson</td>
<td>70000</td>
</tr>
<tr>
<td>76553</td>
<td>Sogh</td>
<td>8000</td>
<td>Finance</td>
<td>Painter</td>
<td>120000</td>
</tr>
</tbody>
</table>

Example: Changing the budget of the ‘Physics’ department

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>salary</th>
<th>dept_name</th>
<th>building</th>
<th>budget</th>
</tr>
</thead>
<tbody>
<tr>
<td>22222</td>
<td>Einstein</td>
<td>9500</td>
<td>Physics</td>
<td>Watson</td>
<td>70000</td>
</tr>
<tr>
<td>12121</td>
<td>Wu</td>
<td>9000</td>
<td>Finance</td>
<td>Painter</td>
<td>120000</td>
</tr>
<tr>
<td>32343</td>
<td>El Said</td>
<td>6000</td>
<td>History</td>
<td>Painter</td>
<td>50000</td>
</tr>
<tr>
<td>45965</td>
<td>Katz</td>
<td>7500</td>
<td>Comp. Sci.</td>
<td>Taylor</td>
<td>100000</td>
</tr>
<tr>
<td>96345</td>
<td>Kim</td>
<td>8000</td>
<td>Elec. Eng.</td>
<td>Taylor</td>
<td>85000</td>
</tr>
<tr>
<td>76766</td>
<td>Crick</td>
<td>7200</td>
<td>Biology</td>
<td>Watson</td>
<td>90000</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
<td>6500</td>
<td>Comp. Sci.</td>
<td>Taylor</td>
<td>100000</td>
</tr>
<tr>
<td>59583</td>
<td>Caliieri</td>
<td>6200</td>
<td>History</td>
<td>Painter</td>
<td>50000</td>
</tr>
<tr>
<td>83821</td>
<td>Brandt</td>
<td>9200</td>
<td>Comp. Sci.</td>
<td>Taylor</td>
<td>100000</td>
</tr>
<tr>
<td>15151</td>
<td>Montart</td>
<td>6000</td>
<td>Music</td>
<td>Packard</td>
<td>80000</td>
</tr>
<tr>
<td>33456</td>
<td>Gold</td>
<td>8700</td>
<td>Physics</td>
<td>Watson</td>
<td>70000</td>
</tr>
<tr>
<td>76553</td>
<td>Sogh</td>
<td>8000</td>
<td>Finance</td>
<td>Painter</td>
<td>120000</td>
</tr>
</tbody>
</table>
Design Approaches

- Normalization Theory (Chapter 8)
  - Formalize what designs are "good", and test for them
  - Translate a "bad" into a "good" design
- Entity Relationship Model (Chapter 7)
  - Models an domain as a collection of entities and relationships
    - Entity: a "thing" or "object" in the domain 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

The Entity-Relationship Model

- Models a domain as a collection of entities and relationships
  - Entity: a "thing" or "object" in the domain 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

- Relational model: flat, "atomic" values
  - E.g., integer
- 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.

Semistructured Data: XML and JSON

- XML: 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 used to be the basis for many data interchange formats
  - A wide variety of tools is available for parsing, browsing and querying XML documents/data
- JSON: Javascript Object Notation
  - Semistructured data format similar to XML, but simpler
  - Well integrated with web technologies
  - Is widely used today

Storage Management

- Storage manager is a program module that provides the interface between the low-level data stored in the database (on disk) 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

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
- Need to search for a good plan (low costs)
  - Traversing the search space of alternative ways (plans) to compute the query result
  - This is called query optimization

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.
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

History (cont.)

- Later 2000s:
  - Scalable data storage systems
    - Google BigTable, Yahoo PHNuts, Amazon, ...
  - Scalable distributed query processing
    - Hive, Spark SQL, Impala, Apache Flink, ...
  - Scalable transaction processing
    - H-store, Spanner, F1, ...
  - Scalable machine learning
    - Tensorflow
  - Software-Hardware co-design (e.g., Oracle Sparc M7)

Recap

- Why databases?
- What do databases do?
- Data independence
  - Physical and Logical
- Database design
- Data models
  - Relational, object, XML, network, hierarchical
- Query languages
  - DML
  - DDL
- Architecture and systems aspects of database systems
  - Recovery
  - Concurrency control
  - Query processing (optimization)
  - File organization and indexing
- History of databases

Outline

- Introduction
- Relational Data Model
- Formal Relational Languages (relational algebra)
- SQL
- Database Design
- Transaction Processing, Recovery, and Concurrency Control
- Storage and File Structures
- Indexing and Hashing
- Query Processing and Optimization

End of Chapter 1
(a) Two-tier architecture

(b) Three-tier architecture