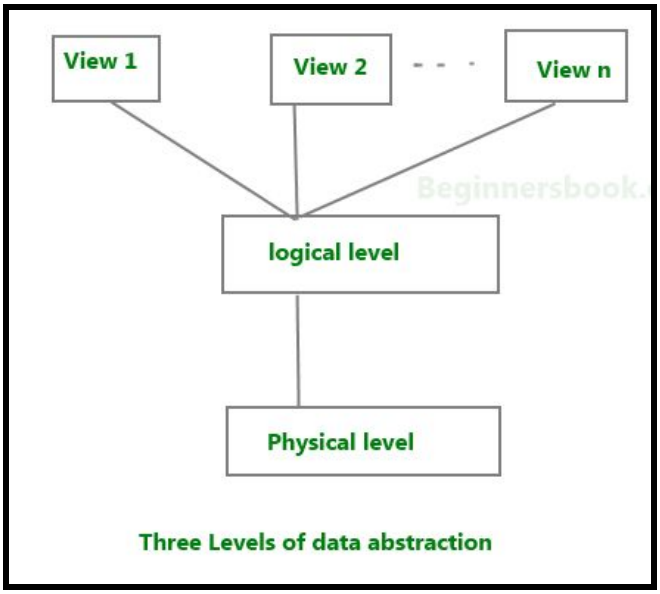
Lecture Notes:

- **Databases are used in multiple places, such as:**
  - Enterprise Information:
    - Sales
    - Accounting
    - Human Resources
    - Manufacturing
    - Online Retailers
  - Banking and Finance:
    - Banking
    - Credit Card Transactions
    - Finance
  - Other Applications:
    - Universities
    - Airlines
    - Hospitals
- **History of Databases:**
  - In the 1960s data storage changed from tape to **direct access**. This allowed shared interactive data use.
  - A tape drive provides **sequential access** storage, unlike a hard disk drive, which provides direct access storage. A disk drive can move to any position on the disk in a few milliseconds, but a tape drive must physically wind tape between reels to read any one particular piece of data. As a result, tape drives have very large average access times.
  - Early databases were **navigational** which was very inefficient for searching. Edgar Codd created a new system in the 1970s based on the **relational model**.
  - In a relational database, data is stored in tables and users access data by doing queries. In a navigational database, data is accessed by defining the path to find the desired data.
  - In the late 1970s and early 1980s, SQL was developed based on the relational model and is the foundation of current databases.
  - In the 2000s, with increasingly large datasets, new XML databases and NoSQL databases are becoming more prevalent.
- **Why use databases:**
  - Commercialized management of large amounts of data
  - Ability to update and maintain data
  - Keep track of relationships between subsets of the data
  - Efficient access and searching capabilities
  - Multiple users can access and share data
  - Ability to limit access to a portion of the data according to user type and enables security of data
  - Minimizes redundancy of multiple data sets
  - Enables consistency constraints
  - Allows users an abstract view of the data which hides the details of how the data are stored and maintained.

- **Data Abstraction:**
  - **Physical Level:**
    - The lowest level.
    - Describes how the data are actually stored.
    - You can get the complex data structure details at this level.
    - These details are often hidden from the programmers.
  - **Logical Level:**
    - The middle level.
    - Describes what data are stored in the database and what relationships exist between the data.
    - Implementing the structure of the logical level may require complex physical low level structures. However, users of the logical level don't need to know about this. We refer to this as the **physical data independence**.
  - **View Level:**
    - The highest level of abstraction.
    - Describes only a small portion of the database.
    - Allows user to simplify their interaction with the database system.
    - The user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how the data is stored and what data is stored.



- **Relational Model:**
  - A **relational database** is a collection of tables each having a unique name.
  - Each **table** also known as a **relation**.
  - **Rows** are referred to as **tuples**.
  - **Columns** are referred to as **attributes**.
  - **Database Schema:** The logical design of the database. It will give you all the tables in that database. It is the consolidation of all the relational schemas. It is the overall design of the database.

- **Database Instance:** A snapshot of the data in the database. It is the information stored at a particular moment in time. It is the collection of all the tuples of the database at any moment. The instance of a database frequently changes.
- **Relation Schema:** A list of attributes and their corresponding domains. It will never talk about the data, just the design. In a relation schema, you need to relay the name of the relation, the attributes and the primary keys. Typically, we underline the attributes that are the primary keys.
- The difference between database schema and relational schema is given in this example:
  Let's say we have 2 relations, A and B.
  A(a1, a2, a3)
  B(b1, b2, b3)
  Each individual is a relational schema, however, the consolidation of them is a database schema.
- It is useful to have the same attribute in multiple schema so that you can combine them.
- The **domain** is all the valid values which an attribute may contain. It is the data type of the column (E.g. int, str, etc).
- E.g. Suppose we have the following database:
  Instructor Relation

ID	Name	Department
1	Srinivasan	Comp. Sci.
2	Wu	Finance
3	Mozart	Music

  a. A tuple from the above database is:

2	Wu	Finance

  b. An attribute from the above database is:

Department
Comp. Sci.
Finance
Music

  c. The domain of the attribute department is string.
  d. The domain of the attribute ID is integer.
  e. The instructor relation has the schema: **instructor(ID, Name, Department)**.
  The syntax of a relation schema is: **relation_name(attribute1, attribute2, … attributen)**. Furthermore, you need to underline the primary key.

- **Keys:**
  - A **key** is an attribute or set of an attribute which helps you to identify a tuple in a relation. They allow you to find the relationships between two tables.
  - We need keys because:
    1. Keys help you to identify any row of data in a table. In a real-world application, a table could contain thousands of records. Moreover, the records could be duplicated. Keys ensure that you can uniquely identify a table record despite these challenges.
    2. Allows you to establish a relationship between and identify the relation between tables
    3. Help you to enforce identity and integrity in the relationship.
  - We will look at the following types of keys:
    1. **Super Key:** A set of one or more attributes that taken together uniquely identify a tuple in the relation. If an attribute is already a super key and other attributes get added to the set, then the set is still a super key. E.g. Suppose the attribute ID is a super key. If we add other attributes to the set, such as name and phone number, the new set is still a super key.
    2. **Candidate Key:** A super key with no repeated attribute. It is a minimal super key.
       Properties of Candidate key:
       - It must contain unique values
       - Candidate key may have multiple attributes
       - Must not contain null values
       - It should contain minimum fields to ensure uniqueness
       - Uniquely identify each record in a table
    3. **Primary Key:** A candidate key chosen to distinguish between tuples. It is usually denoted in a relation schema by an underline. There is only a single primary key.
    4. **Foreign Key:** A set of attributes in a relation that is a primary key in another relation. The foreign key does not have to be a primary key.

- **E.g.** Suppose we have the relation below:

Instructor Relation

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

  a. Some superkeys for this would be:
    1. {ID}
    2. {name, dept_name}
    3. {ID, dept_name}
  b. The candidate keys for this would be:
    1. {ID}
    2. {name, dept_name}

- **E.g.** Given the below relation schemas, find the foreign key(s). The primary key(s) are all underlined.
  A(a1, a2, a3)
  B(a1, b1, b2)

  Solution:
  The foreign key here is a1 in B. While it is not a primary key, a1 is a primary key in A.

- **E.g.** Given the below relation schemas, find the foreign key(s). The primary key(s) are all underlined.
  A(b1, b2, b3)
  B(b1, b2, b3)

  Solution:
  The b1 in A is not a foreign key because the primary key in B is {b1, b2, b3}. However, the b1 in B is a foreign key as b1 is a primary key in A.

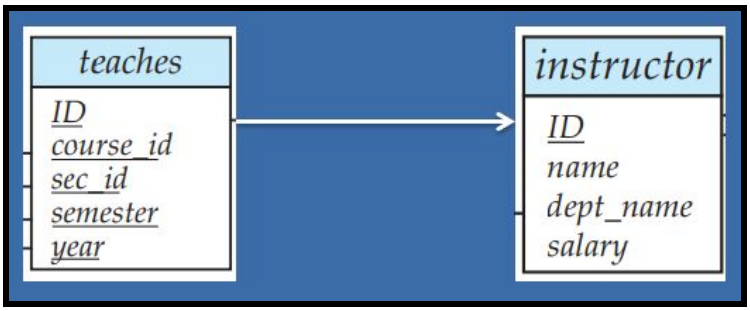
- **E.g.** Given the below relation schemas, find the foreign key(s). The primary key(s) are all underlined.
  instructor(ID,name,dept_name,salary)
  department(dept_name,building,budget)
  teaches(ID,course_id,sec_id,semester,year)

Solution:
The ID in teaches is the foreign key.
We say ID from teaches references instructor.
teaches is the referencing relation.
instructor is the referenced relation.

- A **foreign key constraint** is the fact that a foreign key value in one relation must appear in the referenced relation. A foreign key cannot contain data that the primary key in the referenced relation doesn't have. If data in the primary key of the reference relation changes, that change won't be reflected in the foreign key. However, if a user tries to access that information from the foreign key, an error message will appear.

- **E.g.** Given the below relation schemas, find the foreign key constraint. The primary key(s) are all underlined.
teaches(ID,course_id,sec_id,semester,year)
section(course_id,sec_id,semester,year,building,room_number)

  Solution:
  course_id,sec_id,semester,year in teaches has a foreign key constraint on section. teaches is the referencing relation and section is the referenced relation.
- We can depict foreign key constraints and primary keys using a **schema diagram**. In a schema diagram, the relation is in light blue and the primary keys are underlined. Furthermore, there is an arrow from the foreign key attributes in the referencing relation pointing to the primary key of the referenced relation.
Note: Suppose we have 2 relations, A and B, that both have foreign keys that reference each other.
I.e. A(a1, a2, a3, b1) & B(b1, b2, b3, a1) → b1 in A is a foreign key and a1 in B is also a foreign key.
In this case, there would be a double arrowed line connecting A and B.
E.g.



- **Terminology:**
  - **Candidate Key:** A super key with no repeated attribute. It is a minimal super key.
  Properties of Candidate key:
    - It must contain unique values
    - Candidate key may have multiple attributes
    - Must not contain null values
    - It should contain minimum fields to ensure uniqueness
    - Uniquely identify each record in a table

- **Database:** A collection of interrelated data that is relevant to an enterprise.
- **Database Management System (DBMS):** A software package designed to define, manipulate, retrieve and manage data in a database. A DBMS generally manipulates the data itself, the data format, field names, record structure and file structure. It must be convenient and efficient.
- **Database Schema:** The logical design of the database. It will give you all the tables in that database. It is the consolidation of all the relational schemas. It is the overall design of the database.
- **Database Instance:** A snapshot of the data in the database. It is the information stored at a particular moment in time. It is the collection of all the tuples of the database at any moment. The instance of a database frequently changes.
- **Direct Access:** The ability to obtain data from a storage device by going directly to where it is physically located on the device rather than by having to sequentially look for the data at one physical location after another.
- **Domain:** All the valid values which an attribute may contain. It is the data type of the column (E.g. int, str, etc).
- **Foreign Key:** A set of attributes in a relation that is a primary key in another relation. The foreign key does not have to be a primary key.
- **Foreign key constraint:** The fact that a foreign key value in one relation must appear in the referenced relation. A foreign key cannot contain data that the primary key in the referenced relation doesn't have. If data in the primary key of the reference relation changes, that change won't be reflected in the foreign key. However, if a user tries to access that information from the foreign key, an error message will appear.
- **Key:** An attribute or set of an attribute which helps you to identify a tuple in a relation. They allow you to find the relationships between two tables.
- **Navigational Database:** A type of database in which records or objects are found primarily by following references from other objects.
- **Physical data independence:** Allows you to separate conceptual levels from the internal/physical levels. It allows you to provide a logical description of the database without the need to specify physical structures.
- **Primary Key:** A candidate key chosen to distinguish between tuples. It is usually denoted in a relation schema by an underline. There is only a single primary key.
- **Relational database:** A collection of tables each having a unique name.
- **Relational Model:** Represents the database as a collection of table values. A database organized in terms of the relational model is a relational database.
- **Relation Schema:** A list of attributes and their corresponding domains. It will never talk about the data, just the design. In a relation schema, you need to relay the name of the relation, the attributes and the primary keys. Typically, we underline the attributes that are the primary keys.
- **Sequential Access:** A method of retrieving data from a storage device by moving through all the information until the desired data is reached.
- **Super Key:** A set of one or more attributes that taken together uniquely identify a tuple in the relation. If an attribute is already a super key and other attributes get added to the set, then the set is still a super key.