Quiz 1:

1. Which of the following is not a DBMS
    a. SQL
2. Which of the following is correct
    a. Database System = DBMS + Database
3. Which of the following does not come under Actors on the Scene
    a. Tool Developers
4. Which of the following is not correct
    a. Database is lists of rows and columns
5. Which data model do you use to explain it to a user or a customer
    a. Conceptual Data Model
6. Is metadata the same as database schema and data model?
    a. False
7. E-R model belongs to which type of data model
    a. Conceptual Data Model
8. Commonly, web applications use which type architecture
    a. Three Tier Client-Server Architecture
9. In E-R model which of the following is not correct
    a. All of the above are correct
10. Which of the following is correct
    a. Note: More than one answer can be correct
    b. DDL declarations are irreversible and difficult to undo
    c. DML declarations are reversible

Quiz 2:

1. Which of the following is not a type of attribute
    a. Simple attribute
2. The arrow direction in EER model represents direction of
    a. Specialization
3. What is the circle called in EER model
    a. Specialization circle
4. ____________ is the process of defining a set of subclasses of a superclass
    a. Specialization
5. Which of the following is a type of Specialization
    a. Predicate-defined
    b. Attribute-defined
    c. User-defined
6. Which of the following can you have in Specialization circle
    a. Union
    b. Overlapping
    c. Disjoint
7. It is necessary that every entity in a superclass be a member of some subclass
    a. False
8. Sub-class entity “inherits” all attributes of super-class
    a. True
9. Each attribute is connected to its entity type
    a. True

10. ______________ has a constraint that every subclass has only one superclass

    a. Hierarchy

Lecture 1:

- DBMS & DBS
    - Database Management System (DBMS): **A software package/ system to facilitate the creation and maintenance of a computerized database**. Example: The university uses a DBMS to manage its database. This software helps in creating, updating, and retrieving data efficiently. Examples of DBMS include MySQL, Oracle, or Microsoft SQL Server.
    - Database: **A collection of related data.** Example: The university's database is a structured collection of information about students, courses, faculty, grades, and other relevant details.
    - Data: **Known facts that can be recorded and have an implicit meaning**. Example: In the university database, data includes information like student names, course codes, grades, faculty details, and enrollment dates
- Mini World
    - Mini-world: **Some part of the real world about which data is stored in a database.** Example: The mini-world in this case is the university itself, representing a portion of the real world where data is being recorded, such as student grades and transcripts.
- Data Abstraction
    - A data model is used to hide storage details and present the users with a conceptual view of the database.

        - Programs refer to the data model constructs rather than data storage details

- Actors on Scene Vs Workers Behind Scene
    - Actors on the scene
        - Database administrators:
            - Responsible for authorizing access to the database, for coordinating and monitoring its use, acquiring software and hardware
            - resources, controlling its use and monitoring efficiency of operations.
        - Database Designers:
            - Responsible for defining the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs
        - End-users: They use the data for queries, reports and some of them update the database content. End-users can be categorized into:
            - Casual: access database occasionally when needed
            - Naïve or Parametric: they make up a large section of the end-user population.
                - They use previously well-defined functions in the form of “canned transactions” against the database.
                - Users of Mobile Apps mostly fall in this category

        - Bank-tellers or reservation clerks are parametric users who do this activity for an entire shift of operations.
        - Social Media Users post and read information from websites

- Sophisticated:
    - These include business analysts, scientists, engineers, others thoroughly familiar with the system capabilities.
    - Many use tools in the form of software packages that work closely with the stored database.
- Stand-alone
    - Mostly maintain personal databases using ready-to-use packaged applications.
    - An example is the user of a tax program that creates its own internal database.
    - Another example is a user that maintains a database of personal photos and videos.
- System Analysts and Application Developers
    - This category currently accounts for a very large proportion of the IT workforce.
        - System Analysts: They understand the user requirements of naïve and sophisticated users and design applications including canned transactions to meet those requirements.
        - Application Programmers: Implement the specifications developed by analysts and test and debug them before deployment.

            - Business Analysts: There is an increasing need for such people who can analyze vast amounts of business data and real-time data (“Big Data”) for better decision making related to planning, advertising, marketing etc.

        - Database Users – Actors behind the Scene
            - System Designers and Implementors: Design and implement DBMS packages in the form of modules and interfaces and test and debug them. The DBMS must interface with applications, language compilers, operating system components, etc.
            - Tool Developers: Design and implement software systems called tools for modeling and designing databases, performance monitoring, prototyping, test data generation, user interface creation, simulation etc. that facilitate building of applications and allow using database effectively.
            - Operators and Maintenance Personnel: They manage the actual running and maintenance of the database system hardware and software environment

- Data Models
    - It defines the structure of the data along with how the DBMS will store, access and update the database
    - Data Model store the underlying structure of the database
    - Data Model is Data + Relationship + Semantics + Constraints
    - By explicitly determining the structure of your data, these models support a variety of use cases, including database modeling, information system design, and process development in support of a consistent, clean exchange of data.

    - Data Models are only defined for structured data
        - Gather the required attributes from the user
        - Database Designers will design the database
        - Database Administrators will implement the design

Lecture 2:

- Data Model Definition
    - Data Model: A set of concepts to describe the structure of a database, the operations for manipulating these structures, and certain constraints that the database should obey
- Constraints
    - Constraints are used to define the database structure
    - Constraints typically include elements (and their data types) as well as groups of elements (e.g. entity, record, table), and relationships among such groups
    - Constraints specify some restrictions on valid data; these constraints must be enforced at all times
- Conceptual ER Model
    - https://www.geeksforgeeks.org/data-models-in-dbms/
    - Conceptual (high-level, semantic) data models:
        - Provide concepts that are close to the way many users perceive data.
        - (Also called entity-based or object-based data models.)
- Database Schema

    - Database Schema:
        - The description of a database.
        - Includes descriptions of the database structure, data types, and the constraints on the database.
    - Schema Diagram:
        - An illustrative display of (most aspects of) a database schema.

    - Schema Construct:
        - A component of the schema or an object within the schema, e.g., STUDENT, COURSE.
- Database Instance
    - The actual data stored in a database at a particular moment in time. This includes the collection of all the data in the database.
    - Also called database state (or occurrence or snapshot).
    - The term instance is also applied to individual database components, e.g. record instance, table instance, entity instance
- DML & DDL
    - https://www.geeksforgeeks.org/difference-between-ddl-and-dml-in-dbms/
    - Data Manipulation Language (DML)
        - High-Level or Non-procedural Languages: These include the relational language SQL
        - May be used in a standalone way or may be embedded in a programming language
        - Low Level or Procedural Languages:
        - These must be embedded in a programming language

- Data Definition Language (DDL):
  - Used by the DBA and database designers to specify the conceptual schema of a database.
  - In many DBMSs, the DDL is also used to define internal and external schemas (views).
  - In some DBMSs, separate storage definition language (SDL) and view definition language (VDL) are used to define internal and external schemas.
    - SDL is typically realized via DBMS commands provided to the DBA and database designers
- Data Manipulation Language (DML):
  - Used to specify database retrievals and updates
  - DML commands (data sublanguage) can be embedded in a general-purpose programming language (host language), such as COBOL, C, C++, or Java.
  - A library of functions can also be provided to access the DBMS from a programming language

        - Alternatively, stand-alone DML commands can be applied directly (called a query language)

- SQL is Declarative
    - SQL is a high level or non-procedural language
        - High-level: closer to human language, designed to be easy for humans to read and write.
        - Declarative (Non-procedural): Allows programmers to retrieve data without explicitly hardcoding the steps
    - This means it is “set”-oriented and specify what data to retrieve rather than how to retrieve it
        - Also called: Declarative Languages
- Centralized and Client Server DBMS
    - Centralized DBMS
        - Combines everything into single system including- DBMS software, hardware, application programs, and user interface processing software.
        - User can still connect through a remote terminal - however, all processing is done at centralized site
    - Client Server DBMS
        - Clients
            - Provide appropriate interfaces through a client software module to access and utilize the various server resources.

        - Clients may be diskless machines or PCs or Workstations with disks with only the client software installed.
        - Connected to the servers via some form of a network.
        - (LAN: local area network, wireless network, etc.)
    - DBMS Server
        - Provides database query and transaction services to the clients
        - Relational DBMS servers are often called SQL servers, query servers, or transaction servers
        - Applications running on clients utilize an Application Program Interface (API) to access server databases via standard interface such as:
            - ODBC: Open Database Connectivity standard
            - JDBC: for Java programming access

- 2 tier vs 3 tier client server architecture
    - Two Tier Client-Server Architecture
        - Client and server must install appropriate client module and server module software for ODBC or JDBC
        - A client program may connect to several DBMSs, sometimes called the data sources.
        - In general, data sources can be files or other non-DBMS software that manages data.
        - See Chapter 10 for details on Database Programming
    - Three Tier Client-Server Architecture
        - Common for Web applications
        - Intermediate Layer called Application Server or Web Server:

        - Stores the web connectivity software and the business logic part of the application used to access the corresponding data from the database server
        - Acts like a conduit for sending partially processed data between the database server and the client.
    - Three-tier Architecture Can Enhance Security:
        - Database server only accessible via middle tier
        - Clients cannot directly access database server
        - Clients contain user interfaces and Web browsers
        - The client is typically a PC or a mobile device connected to the Web

Lecture 3:

- Overview Database Design Process
    - Two main activities:
        - Database design
        - Applications design
    - Focus in this chapter on conceptual database design
        - To design the conceptual schema for a database application
    - Applications design focuses on the programs and interfaces that access the database
        - Generally considered part of software engineering

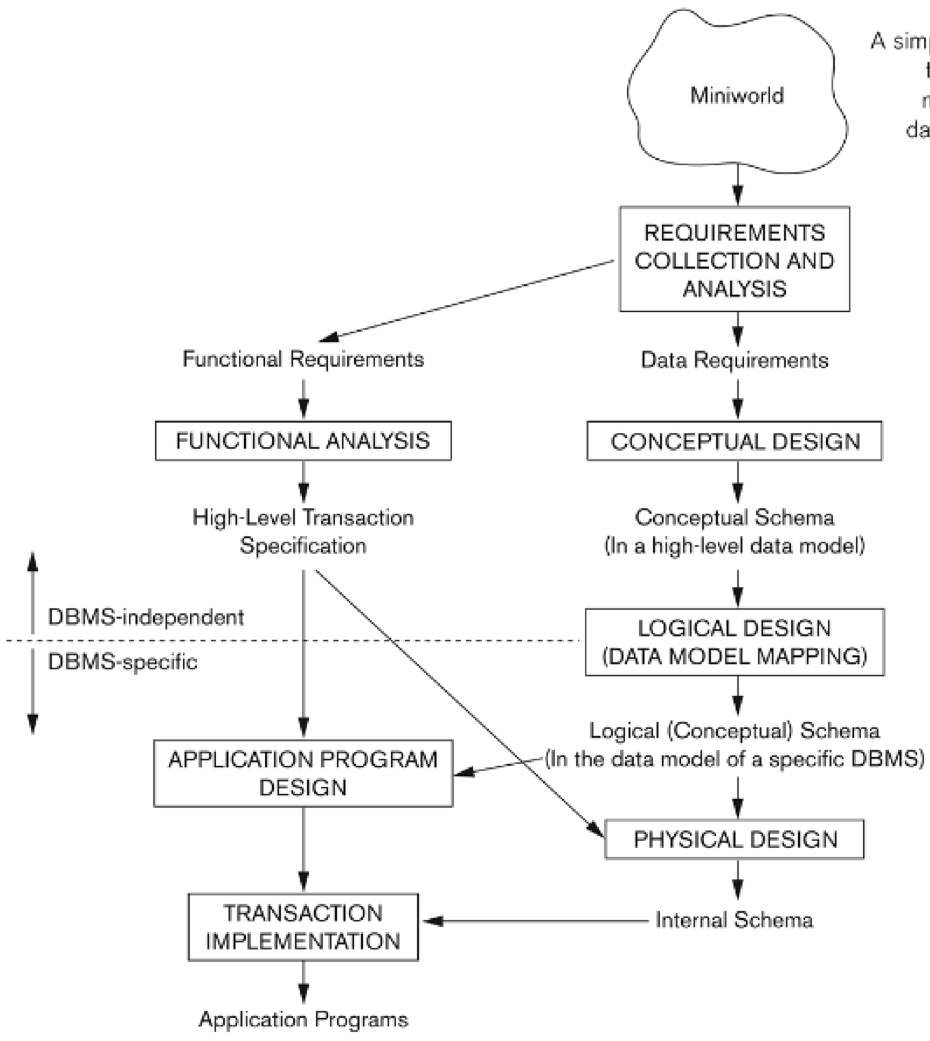

Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-Level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

Figure 3.1
A simplified diagram to illustrate the main phases of database design.

- Simple attribute
  - Each entity has a single atomic value for the attribute. For example, SSN or Sex
- Multi Valued attribute
  - An entity may have multiple values for that attribute. For example, Color of a CAR or PreviousDegrees of a STUDENT.
  - Denoted as {Color} or {PreviousDegrees}.
  - In general, composite and multi-valued attributes may be nested arbitrarily to any number of levels, although this is rare.

        - For example, PreviousDegrees of a STUDENT is a composite multi-valued attribute denoted by {PreviousDegrees (College, Year, Degree, Field)}
    - Multiple PreviousDegrees values can exist
    - Each has four subcomponent attributes:
        - College, Year, Degree, Field
- Composite Attribute
    - The attribute may be composed of several components. For example:
        - Address(Apt#, House#, Street, City, State, ZipCode, Country), or
        - Name(FirstName, MiddleName, LastName).
        - Composite may form a hierarchy where some components are themselves composite.
- Weak Entity
    - An entity that does not have a key attribute and that is identification-dependent on another entity type.
    - A weak entity must participate in an identifying relationship type with an owner or identifying entity type
    - Entities are identified by the combination of:
        - A partial key of the weak entity type
        - The particular entity they are related to in the identifying relationship type
    - Example:
        - A DEPENDENT entity is identified by the dependent’s first name, and the specific EMPLOYEE with whom the dependent is related
        - Name of DEPENDENT is the partial key
        - DEPENDENT is a weak entity type

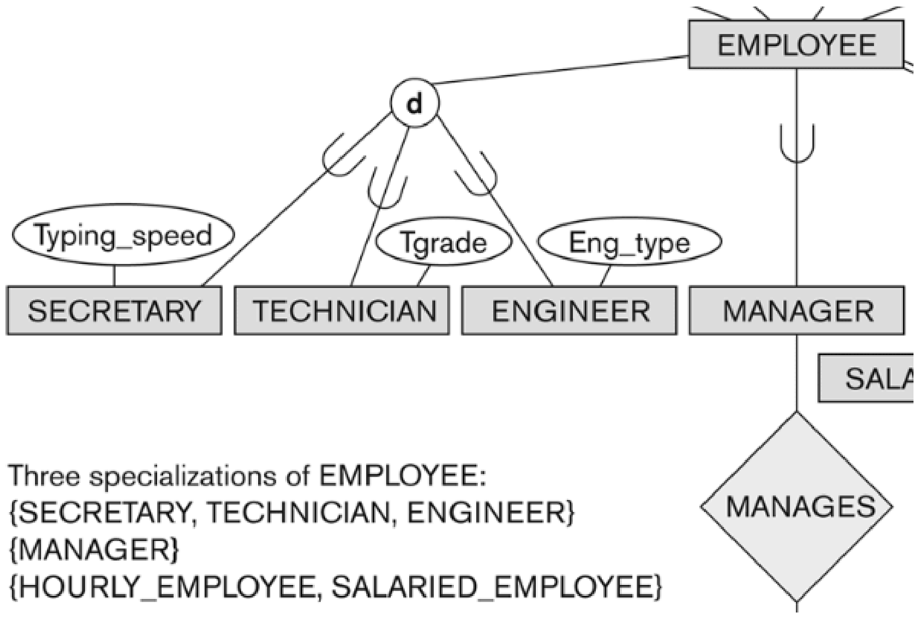
        - EMPLOYEE is its identifying entity type via the identifying relationship type DEPENDENT_O
- Strong Entity
    - A strong entity in the context of Entity-Relationship (ER) modeling in database design refers to an entity that does not depend on any other entity for its identification. It has a primary key, which uniquely identifies each entity instance within its entity set. This primary key is inherent to the strong entity itself and does not rely on any external reference or foreign key to establish its uniqueness.
- Notation for Constraints on Relationships (1:1, 1:N, N:1, M:N)
    - **1:1 (One-to-One):**
        - 1:1 means that a single entity in one table is related to a single entity in another table
    - **1:N (One-to-Many):**
        - 1:1 means that a single entity in one table is related to multiple entities in another table
    - **N:1 (Many-to-One):**
        - N:1 means that there are many entities related to a single entity in another table
    - **M:N (Many-to-Many):**
        - M:N means there are multiple entities related to multiple entities in another table
- Alternative (min,max) notation
    - Specified on each participation of an entity type E in a relationship type R
    - Specifies that each entity e in E participates in at least min and at most max relationship instances in R
    - Default(no constraint): min=0, max=n (signifying no limit)

    - Must have min<max, min>=0, max>=1
    - Derived from the knowledge of mini-world constraints
    - Examples
        - A department has exactly one manager and an employee can manage at most one department.
            - Specify (0,1) for participation of EMPLOYEE in MANAGES
            - Specify (1,1) for participation of DEPARTMENT in MANAGES
        - An employee can work for exactly one department but a department can have any number of employees
            - Specify (1,1) for participation of EMPLOYEE in WORKS_FOR
            - Specify (0,n) for participation of DEPARTMENT in WORKS_FOR
- Ternary Relationship
    - Relationships of Higher Degree
        - Relationship types of degree 2 are called binary
        - **Relationship types of degree 3 are called ternary** and of degree n are called n-ary
        - In general, an n-ary relationship is not equivalent to n binary relationships
        - Constraints are harder to specify for higher-degree relationships (n > 2) than for binary relationships
    - Discussion of n-ary relationships (n > 2)
        - In general, 3 binary relationships can represent different information than a single ternary relationship (see Figure 3.17a and b on next slide)
        - If needed, the binary and n-ary relationships can all be included in the schema design (see Figure 3.17a and b, where all relationships convey different meanings)

        - In some cases, a ternary relationship can be represented as a weak entity if the data model allows a weak entity type to have multiple identifying relationships (and hence multiple owner entity types) (see Figure 3.17c)

<u>Lecture 4:</u>

- Superclass & Subclass (Specialization, Generalization)
    - https://www.geeksforgeeks.org/enhanced-er-mode
    - An entity type may have additional meaningful subgroupings of its entities
    - Each of these subgroupings is a subset of a superclass entity, and the corresponding entities are subclasses
    - Note: An entity that is member of a subclass represents the same real-world entity as some member of the superclass:
    - The subclass member is the same entity in a distinct specific role
    - An entity cannot exist in the database merely by being a member of a subclass; it must also be a member of the superclass
    - A member of the superclass can be optionally included as a member of any number of its subclasses
- Representing Specialization in EER
    - Specialization is the process of defining a set of subclasses of a superclass
    - Arrow direction represents specialization

- 
EMPLOYEE
d
Typing_speed
Tgrade
Eng_type
SECRETARY
TECHNICIAN
ENGINEER
MANAGER
SALA
MANAGES
Three specializations of EMPLOYEE:
{SECRETARY, TECHNICIAN, ENGINEER}
{MANAGER}
{HOURLY_EMPLOYEE, SALARIED_EMPLOYEE}

- Constraints on Specialization (Disjoint, Overlapping)
  - Disjointness Constraint:
    - Specifies that the subclasses of the specialization must be disjoint:
    - an entity can be a member of at most one of the subclasses of the specialization
    - Specified by d in EER diagram
  - If not disjoint, specialization is Overlapping:
    - that is the same entity may be a member of more than one subclass of the specialization
    - Specified by o in EER diagram
- Hierarchy & Lattice
  - A subclass may itself have further subclasses specified on it
    - forms a hierarchy or a lattice
  - Hierarchy has a constraint that every subclass has only one superclass (called single inheritance); this is basically a tree structure

  - In a lattice, a subclass can be subclass of more than one superclass (called multiple inheritance)
- Categories - Union Types
  - All of the superclass/subclass relationships we have seen thus far have a single superclass
  - A shared subclass is a subclass in:
    - more than one distinct superclass/subclass relationships
    - each relationships has a single superclass
    - shared subclass leads to multiple inheritance
  - In some cases, we need to model a single superclass/subclass relationship with more than one superclass
  - Superclasses can represent different entity types
  - Such a subclass is called a category or UNION TYPE
  - Example: In a database for vehicle registration, a vehicle owner can be a PERSON, a BANK (holding a lien on a vehicle) or a COMPANY.
    - A category (UNION type) called OWNER is created to represent a subset of the union of the three superclasses COMPANY, BANK, and PERSON
    - A category member must exist in at least one (typically just one) of its superclasses
  - Difference from shared subclass, which is a:
    - subset of the intersection of its superclasses
    - shared subclass member must exist in all of its superclasses

Lecture 5:

- Table = Relation "R"
    - Informally, a relation looks like a table of values.
    - A relation typically contains a set of rows.
    - The data elements in each row represent certain facts that correspond to a real-world entity or relationship
        - In the formal model, **rows are called tuples**
    - Each column has a column header that gives an indication of the meaning of the data items in that column
        - In the formal model, **the column header is called an attribute name** (or just **attribute**)
- Row/Record = Tuple
    - Key of a Relation:
        - Each row has a value of a data item (or set of items) that uniquely identifies that row in the table
            - Called the key
        - In the STUDENT table, SSN is the key
        - Sometimes row-ids or sequential numbers are assigned as keys to identify the rows in a table
            - Called artificial key or surrogate key
- Column = Attribute
- Formal Definitions - Schema
    - The Schema (or description) of a Relation:
        - Denoted by R(A1, A2, ...An)
        - R is the name of the relation

        - The attributes of the relation are A1, A2, ..., An
    - Example:
        - CUSTOMER (Cust-id, Cust-name, Address, Phone#)
        - CUSTOMER is the relation name
        - Defined over the four attributes: Cust-id, Cust-name, Address, Phone#
    - Each attribute has a domain or a set of valid values.
        - For example, the domain of Cust-id is 6 digit numbers
- Table w Attributes: R(A1, A2, ...An)
    - The Schema (or description) of a Relation:
        - Denoted by R(A1, A2, ...An)
        - R is the name of the relation
        - The attributes of the relation are A1, A2, ..., An
    - Example:
        - CUSTOMER (Cust-id, Cust-name, Address, Phone#)
        - CUSTOMER is the relation name
        - Defined over the four attributes: Cust-id, Cust-name, Address, Phone#
- Tuples(Rows): <1234, "John Smith", USA>
    - A tuple is an ordered set of values (enclosed in angled brackets '< … >')
    - Each value is derived from an appropriate domain.
    - A row in the CUSTOMER relation is a 4-tuple and would consist of four values, for example:
        - <632895, "John Smith", "101 Main St. Atlanta, GA 30332", "(404) 894-2000">
        - This is called a 4-tuple as it has 4 values
        - A tuple (row) in the CUSTOMER relation.
    - A relation is a set of such tuples (rows)

- Domain
    - A domain has a logical definition:
        - Example: “USA_phone_numbers” are the set of 10 digit phone numbers valid in the U.S.
    - A domain also has a data-type or a format defined for it.
        - The USA_phone_numbers may have a format: (ddd)ddd-dddd where each d is a decimal digit.
        - Dates have various formats such as year, month, date formatted as yyyy-mm-dd, or as dd mm,yyyy etc.
    - The attribute name designates the role played by a domain in a relation:
        - Used to interpret the meaning of the data elements corresponding to that attribute
        - Example: The domain Date may be used to define two attributes named “Invoice-date” and “Payment-date” with different meanings
- ***<u>IMPORTANT</u>*** State & Summary
    - State
        - The relation state is a subset of the Cartesian product of the domains of its attributes
            - each domain contains the set of all possible values the attribute can take.
        - Example: attribute Cust-name is defined over the domain of character strings of maximum length 25
            - dom(Cust-name) is varchar(25)
        - The role these strings play in the CUSTOMER relation is that of the name of a customer.

        - Summary
            - Formally,
                - Given R(A1, A2, ..., An)
                - r(R) ⊂ dom (A1) X dom (A2) X ....X dom(An)
            - R(A1, A2, …, An) is the schema of the relation
            - R is the name of the relation
            - A1, A2, …, An are the attributes of the relation
            - r(R): a specific state (or "value" or “population”) of relation R – this is a set of tuples (rows)
                - r(R) = {t1, t2, …, tn} where each ti is an n-tuple
                - ti = <v1, v2, …, vn> where each vj element-of dom(Aj)
        - Example
            - Let R(A1, A2) be a relation schema:
                - Let dom(A1) = {0,1}
                - Let dom(A2) = {a,b,c}
            - Then: dom(A1) X dom(A2) is all possible combinations:
                - {<0,a> , <0,b> , <0,c>, <1,a>, <1,b>, <1,c> }
            - The relation state r(R) ⊂ dom(A1) X dom(A2)
            - For example: r(R) could be {<0,a> , <0,b> , <1,c> }
                - this is one possible state (or “population” or “extension”) r of the relation R, defined over A1 and A2.
                - It has three 2-tuples: <0,a> , <0,b> , <1,c>
- Characteristics of Relations (Self Describing)
    - Ordering of tuples in a relation r(R):

            - The tuples are not considered to be ordered, even though they appear to be in the tabular form.
    - Ordering of attributes in a relation schema R (and of values within each tuple):
        - We will consider the attributes in R(A1, A2, ..., An) and the values in t=<v1, v2, ..., vn> to be ordered .
            - (However, a more general alternative definition of relation does not require this ordering. It includes both the name and the value for each of the attributes).
            - Example: t= { <name, “John” >, <SSN, 123456789> }
            - This representation may be called as “self-describing”.
- Superkey
    - Superkey of R:
        - Is a set of attributes SK of R with the following condition:
            - No two tuples in any valid relation state r(R) will have the same value for SK
            - That is, for any distinct tuples t1 and t2 in r(R), t1[SK] t2[SK]
            - This condition must hold in any valid state r(R)
- Candidate Key
    - Key of R or Candidate Key:
        - A "minimal" superkey.
        - That is, a key is a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey (does not possess the superkey uniqueness property)
        - A Key is a Superkey but **not** vice versa
- Entity Integrity, Referential Integrity
    - Entity Integrity:

        - The primary key attributes PK of each relation schema R in S cannot have null values in any tuple of r(R).
            - This is because primary key values are used to identify the individual tuples.
            - t[PK] ≠ null for any tuple t in r(R)
            - If PK has several attributes, null is not allowed in any of these attributes
        - Note: Other attributes of R may be constrained to disallow null values, even though
    - Referential Integrity
        - A constraint involving two relations
            - The previous constraints involve a single relation.
        - Used to specify a relationship among tuples in two relations:
            - The referencing relation and the referenced relation.
        - Tuples in the referencing relation R1 have attributes FK (called foreign key attributes) that reference the primary key attributes PK of the referenced relation R2.
            - A tuple t1 in R1 is said to reference a tuple t2 in R2 if t1[FK] = t2[PK].
        - A referential integrity constraint can be displayed in a relational database schema as a directed arc from R1.FK to R2