Conceptual Data Models for Database Design for Competitive Exams

Database Design Process

- The database design process consists of a number of steps listed below. We will focus mainly on step 2, the conceptual database design, and the models used during this step.
  - **Step 1: Requirements Collection and Analysis**
    - Prospective users are interviewed to understand and document data requirements
    - This step results in a concise set of user requirements, which should be detailed and complete.
    - The functional requirements should be specified, as well as the data requirements. Functional requirements consist of user operations that will be applied to the database, including retrievals and updates.
    - Functional requirements can be documented using diagrams such as sequence diagrams, data flow diagrams, scenarios, etc.
  - **Step 2: Conceptual Design**
    - Once the requirements are collected and analyzed, the designers go about creating the conceptual schema.
    - Conceptual schema: concise description of data requirements of the users, and includes a detailed description of the entity types, relationships and constraints.
    - The concepts do not include implementation details; therefore the end users easily understand them, and they can be used as a communication tool.
    - The conceptual schema is used to ensure all user requirements are met, and they do not conflict.
  - **Step 3: Database Implementation**
    - Many DBMS systems use an implementation data model, so the conceptual schema is transformed from the high-level data model into the implementation data model.
    - This step is called logical design or data model mapping, which results in the implementation data model of the DBMS.
  - **Step 4: Physical Design**
    - Internal storage structures, indexes, access paths and file organizations are specified.

- Application programs are designed and implemented.

Entity Relationship (ER) Model

- The most popular high-level conceptual data model is the ER model. It is frequently used for the conceptual design of database applications.
  - The diagrammatic notation associated with the ER model, is referred to as the ER diagram. ER diagrams show the basic data structures and constraints.

Entity Types, Entity Sets, Attributes and Keys

- The basic object of an ER diagram is the entity. An entity represents a 'thing' in the real world.
- Examples of entities might be a physical entity, such as a student, a house, a product etc, or conceptual entities such as a company, a job position, a course, etc.
- Entities have attributes, which basically are the properties/characteristics of a particular entity.
- **Examples of entities and attributes**:
  - There are several types of entities. Including:
    - Simple vs. Composite
    - Single-valued vs. Multi-valued
    - Stored vs. Derived

Entity	Attributes	Values
Car	Color	Red
	Make	Volkswagen
	Model	Bora
	Year	2000

Example of Design

Simple vs. Composite Attributes

- Composite attributes can be divided into smaller subparts, which represent more basic attributes, which have their own meanings.
- A common example of a composite attribute is Address. Address can be broken down into a number of subparts, such as Street Address, City, Postal Code. Street Address may be further broken down by Number, Street Name and Apartment/Unit number.
- Attributes that are not divisible into subparts are called simple or atomic attributes.

- Composite attributes can be used if the attribute is referred to as the whole, and the atomic attributes are not referred to. For example, if you wish to store the Company Location, unless you will use the atomic information such as Postal Code, or City separately from the other Location information (Street Address etc) then there is no need to subdivide it into its component attributes, and the whole Location can be designated as a simple attribute.
- What are examples of other composite attributes?

Single-Valued vs. Multi-Valued Attributes

- Most attributes have a single value for each entity, such as a car only has one model, a student has only one ID number, an employee has only one data of birth. These attributes are called single-valued attributes.
- Sometimes an attribute can have multiple values for a single entity, for example, a doctor may have more than one specialty (or may have only one specialty) , a customer may have more than one mobile phone number, or they may not have one at all. These attributes are called multi-valued attributes.
- Multi-valued attributes may have a lower and upper bounds to constrain the number of values allowed. For example, a doctor must have at least one specialty, but no more than 3 specialties.

Stored vs. Derived Attributes

- If an attribute can be calculated using the value of another attribute, they are called derived attributes.
- The attribute that is used to derive the attribute is called a stored attribute.
- Derived attributes are not stored in the file, but can be derived when needed from the stored attributes.

Null Valued Attributes

- There are cases where an attribute does not have an applicable value for an attribute. For these situations, the value null is created.
- A person who does not have a mobile phone would have null stored at the value for the Mobile Phone Number attribute.
- Null can also be used in situations where the attribute value is unknown. There are two cases where this can occur, one where it is known that the attribute is valued, but the value is missing, for example hair color. Every person has a hair color, but the information may be missing. Another situation is if mobile phone number is null, it is not known if the person does not have a mobile phone or if that information is just missing.

Complex Attributes

- Complex attributes are attributes that are nested in an arbitrary way.
- For example a person can have more than one residence, and each residence can have more than one phone, therefore it is a complex attribute that can be represented as:
- {Multi-valued attributes are displayed between braces}
- (Complex Attributes are represented using parentheses)
- E. g.
- {AddressPhone ({Phone (AreaCode, Phone Number) } , Address (StreetAddress (Number, Street, ApartmentNumber) , City, State, Zip)) }

Entity Types, Entity Sets, Keys and Value Sets

Entity Types and Entity Sets

- An **entity type** defines a collection of entities that have the same attributes. Each entity type in the database is described by its name and attributes. The entity share the same attributes, but each entity has its own value for each attribute.

Entity Type Example

- *Entity Type*:
  - Student
- *Entity Attributes*:
  - StudentID,
  - Name,
  - Surname,
  - Date of Birth,
  - Department
  - The collection of all entities of a particular entity type in the database at any point in time is called an entity set. The entity type (Student) and the entity set (Student) can be referred to using the same name.

Entity Set Example

- Entity Type: Student
  - Entity Set:
    - [123, John, Smith, 12/01/1981, Computer Technology]
    - [456, Jane, Doe, 05/02/1979, Mathematics]
    - [789, Semra, Aykan, 02/08/1980, Linguistics]

- The entity type describes the intension, or schema for a set of entities that share the same structure. The collection of entities of a particular entity type is grouped into the entity set, called the extension.

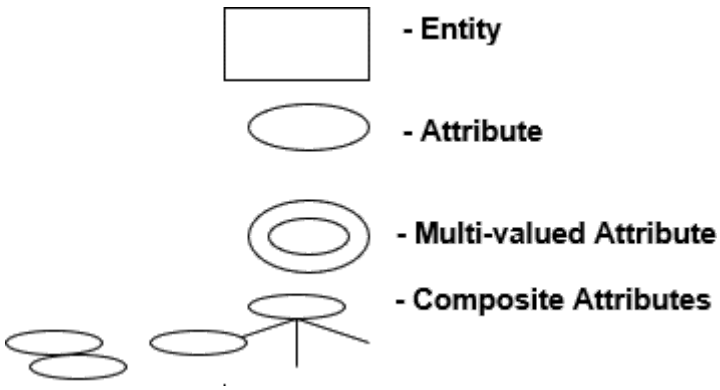
Key Attributes of an Entity Type

- An important constraint on entities of an entity type is the uniqueness constraint.
- A key attribute is an attribute whose values are distinct for each individual entity in the entity set.
- The values of the key attribute can be used to identify each entity uniquely.
- Sometimes a key can consist of several attributes together, where the combination of attributes is unique for a given entity. This is called a composite key.
- Composite keys should be minimal, meaning that all attributes must be included to have the uniqueness property.
- Specifying that an attribute is a key of an entity type means that the uniqueness property must hold true for every entity set of the entity type.
- An entity can have more than one key attribute, and some entities may have no key attribute. Those entities with no key attribute are called weak entity types.

Value Sets (Domains) of Attributes

- Each simple attribute of an entity is associates with a domain of values, or value set, which specifies the set of values that may be assigned to that attribute for each entity. For example, date of birth must be before today's date, and after 01/01/1900, or the Student Name attribute must be a string of alphabetic characters.
- Value sets are not specified in ER diagrams.

ER Diagram Notation



Key Attributes

Company Database Chapter Example

- The company database keeps track of a company's employees, departments and projects. Suppose that after the requirements collection and analysis phase, the database designers provided the following description of the part of the company to be represented by the database.
  - The company is organized into department. Each department has a unique name a unique number and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.
  - A department controls a number of projects, each of which has a unique name, a unique number and a single location.
  - We store each employees name, ID number, address, salary, sex and birth date. An employee is assigned to one department but may work on several projects, which are not necessarily controlled by the same department. We keep track of the number of hours per week that an employee works on each project. We also keep track of the direct supervisor of each employee.
  - We want to keep track of the dependents of each employee for insurance purposes. We keep each dependent's first name, sex, birth date and relationship to the employee.
  - From the information given above, we can identify 4 entities.
    - **Department** – Name, Number, Locations, Manager, and Manager Start Date.

- **Project** – Name, Number, Location and Controlling Department.
- **Employee** – Name, ID Number, Sex, Address, Salary, Birth Date, Department. Both Name and Address can be a composite attribute, however it was not specified in the requirements.
- **Dependent** – Employee, Dependent Name, Sex, Birth Date, Relationship

- The information about the projects an employee works on can be represented in two ways. One, we can include a multi-valued composite, attribute, WorksOn (Project, Hours) in the Employee entity, or we can include a multi-valued composite attribute, Workers (Employee, Hours) .

Relationship Types, Relationship Sets, Roles and Structural Constraints

- Looking at the example above, there are several implicit relationships among the entity types.
  - Whenever an attribute of one entity type refers to another entity type, some relationship exists.
  - For example, Manager of a department refers to an employee who manages the department, Controlling Department of the project, refers to the department that controls the project. Supervisor of an employee refers to the employee who supervises that employee.
  - In an ER diagram, these references are not represented as attributes, but as relationships.

Relationship Types, Sets and Instances

- A relationship type, R, among entities, defines a *relationship set* among entities from the entity types.

Role Names

- Each entity type that participates in a relationship type plays a particular role in the relationship.
- The role name shows the role that an particular entity from the entity type plays in each relationship.
- Example: In the Company diagram, in the Works For relationship type, the employee plays the role of employee or worker, and the department entity plays the role of department or employer.
- In some cases the same entity participates more than once in a relationship type in different roles.
- For example, the Supervision relationship type relates an employee to a supervisor, where both the employee and supervisor are of the same employee entity type, therefore

the employee entity participates twice in the relationship, once in the role of supervisor, and once in the role of supervisee.

Developed by: Mindsprite Solutions