Review Questions

Friday, February 09, 2024 6:33 PM

Question 1
Requirements Analysis:
Conceptual Database Design: The information gathered in the require⍰ments analysis step is used to develop a high-level description of the data to be stored in the database, along with the constraints known to hold over this data. This step is often carried out using the ER model and is dis⍰cussed in the rest of this chapter. The ER model is one of several high-level, or semantic, data models used in database design.
Logical Database Design: We must choose a DBMS to implement our database design, and convert the conceptual database design into a database schema in the data model of the chosen DBMS. We will consider only relational DBMSs, and therefore, the task in the logical design step is to convert an ER schema into a relational database schema.
Schema Refinement: The fourth step in database design is to analyze the collection of relations in our relational database schema to identify po⍰tential problems, and to refine it. In contrast to the requirements analysis and conceptual design steps, which are essentially subjective, schema re⍰finement can be guided by some elegant and powerful theory.
Physical Database Design: In this step, we consider typical expected workloads that our database must support and further refine the database design to ensure that it meets desired performance criteria. This step may simply involve building indexes on some tables and clustering some tables, or it may involve a substantial redesign of parts of the database schema obtained from the earlier design steps. We discuss physical design and database tuning in Chapter 20. 6. Application and Security Design: Any software project that involves a DBMS must consider aspects of the application that go beyond the database itself. Design methodologies like UML (Section 2.7) try to ad⍰dress the complete software design and development cycle. Briefly, we must identify the entities (e.g., users, user groups, departments) and processes involved in the application. We must describe the role of each entity in ev⍰ery process that is reflected in some application task, as part of a complete workflow for that task. For each role, we must identify the parts of the database that must be accessible and the parts of the database that must not be accessible, and we must take steps to ensure that these access rules are enforced.

Q1
Requirement analysis
Conceptual DB design → high level description of data
to be stored in the DB, along with constraints
Step carried out by ER Model

Logical DB design: logical schema
schema refinement
physical DB design
Application and security design

QUESTION 2
An entity is an object in the real world that is distinguishable from other objects. Examples include the following: the Green Dragonzord toy,
It is often useful to identify a collection of similar entities. Such a collection is called an entity set. Note that entity sets need not be disjoint; the collection of toy department employees and the collection of appliance department employees may both contain employee John Doe (who happens to work in both departments). We could also define an entity set called Employees that contains both the toy and appliance department employee sets. An entity is described using a set of attributes. All entities in a given entity set have the same attributes; this is what we mean by similar. (This statement is an oversimplification, as we will see when we discuss inheritance hierarchies in Section 2.4.4, but it suffices for now and highlights the main idea.) Our choice of attributes reflects the level of detail at which we wish to represent information about entities. For example, the Employees entity set could use name, social security number (ssn), and parking lot (lot) as attributes. In this case we will store the name, social security number, and lot number for each employee. However, we will not store, say, an employee's address (or gender or age)
A key is a minimal set of attributes whose values uniquely identify an entity in the set. There could be more than one candidate key; if so, we designate one of them as the primary key.

Q2
entity
entity set
attribute
key → minimal set of attributes needed whose values
uniquely identify an entity in the set

QUESTION 3
A relationship is an association among two or more entities.
As with entities, we may wish to collect a set of similar relationships into a relationship set. A relationship set can be thought of as a set of n-tuples: {(e1, . . . , en) | e1 ∈ E1, . . . , en ∈ En} Each n-tuple

denotes a relationship involving n entities e1 through en, where entity ei is in entity set Ei.
A relationship can also have descriptive attributes. Descriptive attributes are used to record information about the relationship, rather than about any one of the participating entities;

Q3 relationship is an association among 2 or more entities
relationship set consists of similar relationships
each relationship can have descriptive attributes
they record info about the relationship, rather than about any of the participating entities
[illegible] on some entity set

QUESTION 4
The restriction that each department has at most one manager is an example of a key constraint, and it implies that each Departments entity appears in at most one Manages relationship in any allowable instance of Man⍰ages.The key constraint on Manages tells us that a department has at most one manager. A natural question to ask is whether every department has a man⍰ager. Let us say that every department is required to have a manager. This requirement is an example of a participation constraint; the participation of the entity set Departments in the relationship set Manages is said to be total. A participation that is not total is said to be partial.
Dependents is an example of a weak entity set. A weak entity can be iden⍰tified uniquely only by considering some of its attributes in conjunction with the primary key of another entity, which is called the identifying owner.
The following restrictions must hold: The owner entity set and the weak entity set must participate in a one⍰to-many relationship set (one owner entity is associated with one or more weak entities, but each weak entity has a single owner). This relationship set is called the identifying relationship set of the weak entity set. The weak entity set must have total participation in the identifying rela⍰tionship set. For example, a Dependents entity can be identified uniquely only if we take the key of the owning Employees entity and the pname of the Dependents entity. The set of attributes of a weak entity set that uniquely identify a weak entity for a given owner entity is called a partial key of the weak entity set.

We want the semantics that every entity in one of these sets is also an Em⍰ployees entity and, as such, must have all the attributes of Employees defined. Therefore, the attributes defined for an Hourly Emps entity are the attributes for Employees plus Hourly Emps. We say that the attributes for the entity set Employees are inherited by the entity set Hourly Emps and that Hourly Emps ISA (read is a) Employees. In addition—and in contrast to class hierarchies in programming languages such as C++—there is a constraint on queries over instances of these entity sets: A query that asks for all Employees entities must consider all Hourly Emps and Contract Emps entities as well.
A class hierarchy can be viewed in one of two ways: Employees is specialized into subclasses. Specialization is the process of identifying subsets of an entity set (the superclass) that share some distinguishing characteristic. Typically, the superclass is defined first, the subclasses are defined next, and subclass-specific attributes and relation⍰ship sets are then added. Hourly Emps and Contract Emps are generalized by Employees. As an⍰other example, two entity sets Motorboats and Cars may be generalized into an entity set Motor Vehicles. Generalization consists of identifying some common characteristics of a collection of entity sets and creating a new entity set that contains entities possessing these common character⍰istics.

Sometimes, we have to model a relationship between a collection of entities and relationships. Suppose that we have an entity set called Projects and that each Projects entity is sponsored by one or more departments. The Spon⍰sors relationship set captures this information. A department that sponsors a project might assign employees to monitor the sponsorship. Intuitively, Moni⍰tors should be a relationship set that associates a Sponsors relationship (rather than a Projects or Departments entity) with an Employees entity. However, we have defined relationships to associate two or more entities.
To define a relationship set such as Monitors, we introduce a new feature of the ER model, called aggregation. Aggregation allows us to indicate that a relationship set (identified through a dashed box) participates in another relationship set

Q4 key constraint : each entity in some [illegible]
appears at most ~~one~~ one relationship in any allowable instance of that relationship
participation Constraint : Total or partial
↓
how much participation is shown by an entity set in a Relationship
weak entity : It can be identified uniquely only by considering some of its attributes in conjunction with the primary key of another entity, which is called identifying owner.
Class hierarchy : inherited attributes
specialization
generalization

QUESTION 5-READ FROM BOOK IN DETAIL(Section 2.5 full)-also check answer for this answer in answer sheet pdf given
QUESTION 6

The usual approach is that the requirements of various user groups are considered, any conflicting requirements are somehow resolved, and a single set of global requirements is generated at the end of the requirements analysis phase. Generating a single set of global requirements is a difficult task, but it allows the conceptual design phase to proceed with the development of a logical schema that spans all the data and applications throughout the enterprise. An alternative approach is to develop separate conceptual schemas for different user groups and then integrate these conceptual schemas.
In some situations, schema integration cannot be avoided; for example, when one organization merges with another, existing databases may have to be integrated. Schema integration is also in�creasing in importance as users demand access to heterogeneous data sources, often maintained by different organizations.

Q6 single set of global requirements after requirement analysis phase
This allows conceptual design phase to proceed with development of a logical schema
<u>Method 2</u>
Separate conceptual schemas for different user groups

QUESTION 7
UML, like the ER model, has the attractive feature that its constructs can be drawn as diagrams. It encompasses a broader spectrum of the software design process than the ER model: Business Modeling: In this phase, the goal is to describe the business processes involved in the software application being developed. System Modeling: The understanding of business processes is used to identify the requirements for the software application. One part of the requirements is the database requirements. Conceptual Database Modeling: This step corresponds to the creation of the ER design for the database. For this purpose, UML provides many constructs that parallel the ER constructs. Physical Database Modeling: UML also provides pictorial represen�tations for physical database design choices, such as the creation of table spaces and indexes. (We discuss physical database design in later chapters, but not the corresponding UML constructs.) Hardware System Modeling: UML diagrams can be used to describe the hardware configuration used for the application.
Activity diagrams show the flow of actions in a business process. Statechart diagrams describe dynamic interactions between system objects. These dia�grams, used in business and system modeling, describe how the external func�tionality is to be implemented, consistent with the business rules and processes of the enterprise. Class diagrams are similar to ER diagrams, although they are more general in that they are intended to model application entities (intuitively, important program components) and their logical relationships in addition to data entities and their relationships. Both entity sets and relationship sets can be represented as classes in UML, together with key constraints, weak entities, and class hierarchies. The term relationship is used slightly differently in UML, and UML's relationships are binary. This sometimes leads to confusion over whether relationship sets in an ER diagram involving three or more entity sets can be directly represented in UML. The confusion disappears once we understand that all relationship sets (in the ER sense) are represented as classes in UML; the binary UML 'relationships' are essentially just the links shown in ER diagrams between entity sets and relationship sets.
ndeed, every class in a UML class diagram is mapped into a table in the cor�responding UML database diagram. UML's database diagrams show how classes are represented in the database and contain additional details about the structure of the database such as integrity constraints and indexes. Links (UML's 'relationships') between UML classes lead to various integrity con�straints between the corresponding tables. Many details specific to the re�lational model (e.g., views, foreign keys, null-allowed fields) and that reflect physical design choices (e.g., indexed fields) can be modeled in UML database diagrams. UML's component diagrams describe storage aspects of the database, such as tablespaces and database partitions), as well as interfaces to applications that access the database. Finally, deployment diagrams show the hardware aspects of the system.

<u>Q7</u> UML $\rightarrow$ diagram structures
Business modelling, system, conceptual DB, physical DB, Hardware System