Contents

Chapter 1

Introduction

Database is set of stored data that is persistent, integrated, interconnected with as little as possible redundency, based on a data model, under control of a central system, used by multiple user, shared and concurrent.

Indexes are used to improve searching. Explain the following

1. B-tree and B$^+$-tree

2. R-tree (for two dimension trees)

3. ISO and image

4. HDFS

Chapter 2

Entity-Relationship Model

In database design, database designers first need to know the data they are modeling. This includes knowing the *data requirements* and *functional requirements*. Functional requirements are the operations on the data. Once the requirements have been analyzied, the next step is designing a *conceptual schema/model* for the database. Note that this step is not dependent on the DBSM system and is done using high-level conceptual modeling. Then, the database designers must be this conceptual model to a DBSM system, this step is called *logical design* or *data mapping design*. The last step is *physical design* in which the goal is to design the underlying hardware, that is the storage structures, file organization, indicies, access paths, and specifiying physical parameters for the design.

2.1   ER model

Entity-Relationship model is a conceptual model of data.

2.2   Entity

An *entity* is an objecte or a thing that exists independently. Each entity has *attributes* that describe it. For example STUDENT is an entity that is has a name and a student number. Each attribute has some value, base on which we can categorize the attributes into

Definition:

Simple/Atomic Attributes that are not divisible into other attributes. For example "First Name".

Composite Attributes that can be divided into other attributes. For example "Name" can be divided into "First Name", "Middle Name", and "Last Name"

Singlevalued Attributes that can only have one value. For example "Social Security Number".

Multivalued Attributes that can have many values. For example "Phone Number" might have multiple values.

Derived Attributes that can be determined from another attributes which is called the *stored attribute*. For example, "Area" can be determined from "Radius" in case of a CIRCLE entity.

Attributes can have *null* value. We can consider a domain of values _ a set_ for each attribute. To allow nullable and multiplevalued attribute we can consider an attribute to be a subset of this domain.

A set of entities that have similar attribute like STUDENT is called an *entity type*. An important attributes of the entities in an entity type is their *key/identifier attribute* which uniquely determines the specific entity. Some entity types might have more than on key attribute, For example a STUDENT can be determined via its "Student Number" or "Social Security Number".

2.3   Relationship

The interaction between two or more entities is a *relationship*. Mathematically, a relationship is a Cartesian product n entity types. $E_1, \ldots, E_n$ said to *participate* in a relationship type $R = E_1 \times \cdots \times E_n$ and individual entities $e_1, \ldots, e_n$ are said to *paraticipate* in a relationshiple instance $r = (e_1, \ldots, e_n)$. The degree of a relationship is the number of entity types that participate in it. Binary relationship are relationship with two entity types. Binary relationship can have different *cardinality ratios*

Many-to-Many denoted as $M : N$. For example each STUDENT can enroll in multiple COURSEs and each COURSE has many STUDENT enrolled in it.

One-to-Many denoted as $N : 1$. Each DEPARTMENT has many FACULTY but each FACULTY has only one DEPARTMENT.

One-to-One denoted as $1 : 1$. Each DEPARTMENT has one DEAN and each DEAN belongs to one DEPARTMENT.

Total participation or *existence dependecy* is when each entity in the first entity type must be in a relationship with another entity in the second entity type. For example, each FACULTY must have a DEPARTMENT. In *partial participation* some of the entities in the first entity type are in a relationship with the entities in the second entity type. For example, not every FACULTY is a DEAN.

A relationship might have attributes for itself. For instance, a DEAN *MANAGES* a DEPARTMENT from a starting time to some other time.

2.4   Weak entity

An entity type that does not have a key attribute is called a *weak entity*. Those entities that have a key attribute are called *strong entity*. In real world, a weak entity is an entity that is depedent on the existence of another entity, called *identifying/owner entity type*. In ER model we can model this relationship by a total participation relationship which is called *identifying relationship*. A weak entity type normal has one or more *partial key* attributes that distinguish it from the other related weak entities to the same owner entity. For example, a COURSEGROUP entity has the "Semester", "Year", and the "Group Number" attributes of a COURSE entity. Every COURSEGROUP entity must be related to a COURSE entity and here, each of its attributes are partial keys.

By adding weak entities we can reduce the order of a relationship to two and even remove attributes of the relationship and give it to the weak entity.

2.5   ER notation and diagrams

Insert the corresponding diagrams :)

2.6 Enhanced/Extended ER model

2.6.1 Inheritance

For inheritance we need a *supertpye* and *subtype* which inherits attributes of the supertype. We show the concept of inheritance in EER model through *specialization* and *generalization.* Specialization is the process of defining a set of subclasses of an entity type. Generalization is the process of finding common attributes and creating a supertype. A specialization has the following characteristics

Definition:

Total every entity of the supertype must be an instance of at least one one the subtypes.

Partial opposite of total.

Disjoint every entity of the supertype belongs to at most one of the subtypes.

Overlapping opposite of disjoint.

Totalness and disjointness are indepedent of each other and hence we can have four combinations. –insert diagram

Multiple inheritance can occure when a subtype inherites attributes of one or more supertypes. For example, a TEACHING-ASSISTANT is both a STUDENT and an EMPLOYEE.

2.6.2 Union-Type (Category)

When we want to combine different entities, we use Union types. For example, a VEHICLE can be a CAR or a TRUCK.

Definition:

Total every entity of the subtype is one the supertype.

Partial opposite of total.

–insert diagram

If the keys of the supertpyes are from the same domain, then we can have the same key for the subtype. But, it is better to have different key. Also note that, total union type is equivalent to disjoint total inheritance.

We represent both inheritance and union-type with a IS-A relationship.

2.7 Aggregation and association

We need to represent membershipe/being a component/containment relationships as well. IS-A-PART-OF relationships denote the following meanings – insert diagram

- F is a part of E.

- E contains F.

- E has F.

- F is a member of E.

We can also view relationships as an entity (i literally have no fucking clue why they dont make a science out of it and standardize it with better capabilities).

2.8   Traps

There are two traps

Definition:

Loop trap thinking that $(i, s), (s, p), (p, i) \implies (i, s, p)$ is wrong.

Chasm trap failing to model correctly that in a chain of 1:N relationship, the last entity is
"connected" to the first entity, because the participation is not total. (If it is ambiguous, let
me tell that i have seen a million examples and still dont fucking understand, it is so fucking
dumb ‿ or maybe i am :\\).