**CHAPTER FOUR**

**CONCEPTUAL DATA MODEL**

**Introduction**

Conceptual modelling is an important phase in designing a successful database application. This chapter follows the traditional approach of concentrating on database structures and constrains during database design.

**Terminology**

**Model:** A representation of a real world thing.

**Types of Data Models**

A database model is a collection of logical constructs used to represent the data structure and relationships found within the database.

**1. Object Based Logical Models**

They are used in describing data at the conceptual and view levels. They provide fairly flexible structuring capabilities and allow data constraints to be specified explicitly. They include:

• E - R Model

• Object Oriented Model

• Binary Model

• Semantic Data Models

• Info-logical Data Model

• Function

**2. Record Based Logical Models**

These are models used in describing data at the conceptual and view levels. They are used to specify the overall logical structure of the database and to provide a higher-level description implementation. Record based models are so named because the database is structured in fixed format records of several types. This is in contrast to many of the object-based models whose richer structure often leads to variable length records at physical level. The three most widely accepted record based data models are the Relational, network and hierarchical models.

 **Relational model**

- This model uses a collection of tables to represent both data and the relationship among those data.



- Data is organized into independent 2-dimensional arrays called tables or relations.

- Each relation has multiple columns (fields or attributes) each with a unique name.

- The rows (records or tuple) represent one record.

- The size of table refers to the number of rows while the degree of table refers to the number of columns.

- This model differs from network and hierarchical in that it does not use pointers or links but relates records by values that they contain a concept that allows a mathematical foundations to be defined.

Table

|  |  |  |
| --- | --- | --- |
| Borrower name | ID | Address |
| Odhiambo | A-1345 | 45 kisumu |
| Legei | A-567 | 67 kabarak |

*Table 3.1: An example of relational database table*

Database is a collection of tables

 **Network model**

- Data in this model are represented by collection of records (as c or c++ type), and relationships among data are represented by links, which can be viewed as pointers.

- The records in the database are organized as a collection of arbitrary graphs.

 **Hierarchical model**

- Is similar to the network model in the sense that data and relationships among data are represented by records and links respectively but it differs from network model in that records are organized as collections of trees rather than arbitrary graphs.

**3. Physical Data Models**

These are models that are used to describe data at the lowest level. They are very few in number and the two widely known ones are:

i. Unifying model

ii. Frame memory model

NB: Like the E-R model, the object-oriented model is based on a collection of object where an object contains values stored in instance variables with the object.

**The E- R (Entity Relationship) Model**

It is based on a perception over a real world, which consists of a collection of basic objects called entities and relationships among this objects. An entity is an object that is distinguished from other objects via a specific set of attributes.

Components of E-R diagram

(i) Rectangles: - They represent entity sets. (ii) Ellipses: - represent attributes

(iii) Diamond: - represents relationship sets

(iv) Lines - Link attributes to entities and entity sets to relationship sets

(v) Double ellipses: - represent multi-value attributes

(vi) Dashed ellipses: - denote derived attributes

(vii) Double lines: - indicate total participation of an entity in relationship sets.

**E-R Model Basic Concepts**

The model employs the following components:

• Entity sets

• Relationship sets

• Attributes

**1. Entity sets**

An entity is a thing or object in the real world that is distinguishable from all other objects *(objects about which database is designed to store information)*. It may be concrete e.g. a person or a book or it may be abstract e.g. a loan, holiday a concept etc. An entity set is a set of entities of the same type that share the same properties or attitudes e.g. a set of all persons who are customers of a bank*.(a collection of specific entities that the database currently contains*)

**2. Entity class**

Entire collection of all possible entities of a specific type that database might contain i.e. its an abstract description of an object.

(These are classes of distinct things about which data is recorded in the system)

An entity is usually represented as a rectangle containing its name in singular noun e.g. considers a library system (database) with entities book and borrower.

*BOOK* BORROWER

*Figure 3.1: Sample entities*

**3. Entity attributes**

They are descriptive properties or characteristics possessed by each member of an entity set.

**(i.e.** a descriptive value associated with an entity) the designation of an attribute for an entity set expresses that the database stores similar information concerning each entity in the entity set.

Attributes are used to uniquely identify individual entities within an entity class.

Attributes are represented as ellipses containing the name/property of the attribute.

Example attributes for a library database

Title Price

ISBN

Book

*: Some attributes of the Book entity*

**Characteristics of Attributes**

An attribute as used in E—R model can be characterized by the following attribute types: -

**i. Simple and Composite attributes –**

A simple attribute cannot be subdivided into subparts e.g. title in entity book where as composite attributes can be subdivided in subparts (other attributes) e.g. author\_name (first name, middle name, last name). Composite attributes are necessary if a user wishes to refer to entire attribute on some occasions and to only a component of the attributes on other occasions.

**ii. Single valued and Multi valued Attribute: -**

The social security number or ID number can only have a single value at any instance and therefore its said to be single valued. An attribute like dependant name can take several values ranging from o-n thus it is said to be multi valued.

iii. **Null Attributes:** -

A null value is used when an entity does not have a value for an attribute (not applicable or not required or value missing or value does not exist) e.g. dependent name.

iv. **Calculated attribute** - The value for this type of attribute can be derived from the values of other related attributes or entities e.g.

i. Employment length value can be derived from the value for the start date and the current date.

ii. Loans held can be a count of the number of loans a customer has.

**Relationship Sets**

A relationship is an association amongst several entities while a relationship set is a set of relationships of the same tuple. It is a mathematical relation on n>2 possible non-distinct entity sets e.g. consider 2 entity sets, loan and branch. A relationship set loan,

branch can be defined to denote association between a bank loan and the branch in which that loan is obtained.

Example

Consider 2 entity sets Customer and loan.

A relationship set - A borrower can be defined to denote the association between customers and the bank loans that the customers have.

Customer

Borrower

Loan

*Figure 3.3: A relationship between two entities*

**Weak Entity Set**

This is an entity set that does not have sufficient attributes to form a primary e.g. an entity set payments comprising of the attributes payment number, payment date and payment amount. Although each payment entity is distinct, payment for different loan e.g. may share the same payment number thus this entity set does not have a primary key.

**Strong Entity Set**

This is an entity set that has a primary key. For weak entity set to be meaningful it must be part of a one to many relationships.

**Types of Relationships (cardinality mapping)**

(Meaningful interactions or associations between entities)

Mapping cardinalities or cardinality ratios express the number of entities to which another entity can be associated via a relationship set. Mapping cardinalities are most useful in describing binary relationship set although occasionally they contribute to the description of relationship set that involve more than two entity sets.

For binary relationship **R** between entity sets **A** and **B** the mapping cardinality must be one of the following

i. **One to one relationship** (**1:1**) –

An entity in **A** is associated with utmost one entity in **B** and an entity in **B** is associated with at utmost one entity in **A**.

(i.e. for an entity type A there may only be one member of entity type B and for any entity type B there is only one member of entity type A associated with it.

**A B**

a1 b1

a2 b2

a3 b3

*Figure 3.4a: A one-to-one relationship*

Example: -

One managing director works for only one company and one company will employ only one managing director.

Managing director

Works for

Employs

Company

*Figure 3.4b: An example of a one-to-one relationship*

ii. **One to Many relationship *(*1:M*)*** –

An entity in **A** is associated with any number of entities in **B** while an entity in **B** can be associated with at most one entity in **A**.

**A B**

a1 b1 a2 b2 a3 b3

*Figure 3.5a: A one-to-many relationship*

Example: -

One course is done by many students

Course Student

\

*Figure 3.5b: An example of a one-to-many relationship*

(ii) One department may comprise of many employees but one employee will only work in one department.

DIPID Name

ID Name

Addre ss

Department Employee

*Figure 3.5c: Another example of a one-to-many relationship with attributes*

To implement a one to many relationship add a key attribute from related table

E.g. the relationship a above

Employee (ID, name, address) becomes Employee (ID, name, address, departmentID)

The departmentID in employee entity class is referred to as *foreign key* since it’s a primary key for a foreign entity class department.

iii. **Many to many** (**M: N**): -

An entity in **A** is associated with any number entities in **B** and an entity in **B** can be associated with any number of entities in **A**.

Whenever the degree of relationship is many to many we must decompose the relationship to either one to many or many to one.

Example: -

A product can be made of many different types of raw materials while a raw material can be part of (used to make) different products.

Teach

Lecturer Student

Taught by

Made of

Product Raw

Part of

material

*Figure 3.6: Some examples of many-to-many relationships*

Implementing a many to many relationship is more involving than a one to many relationship. We cannot simply treat this as a two one to many relationship since it would result in redundancy



We need to add a new table (relation/entity) then treat a 2- one to many relationships e.g.

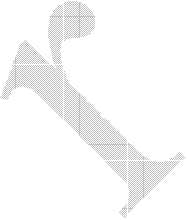
Book Author becomes our entity/new table drawing key attributes from entity Book and Author

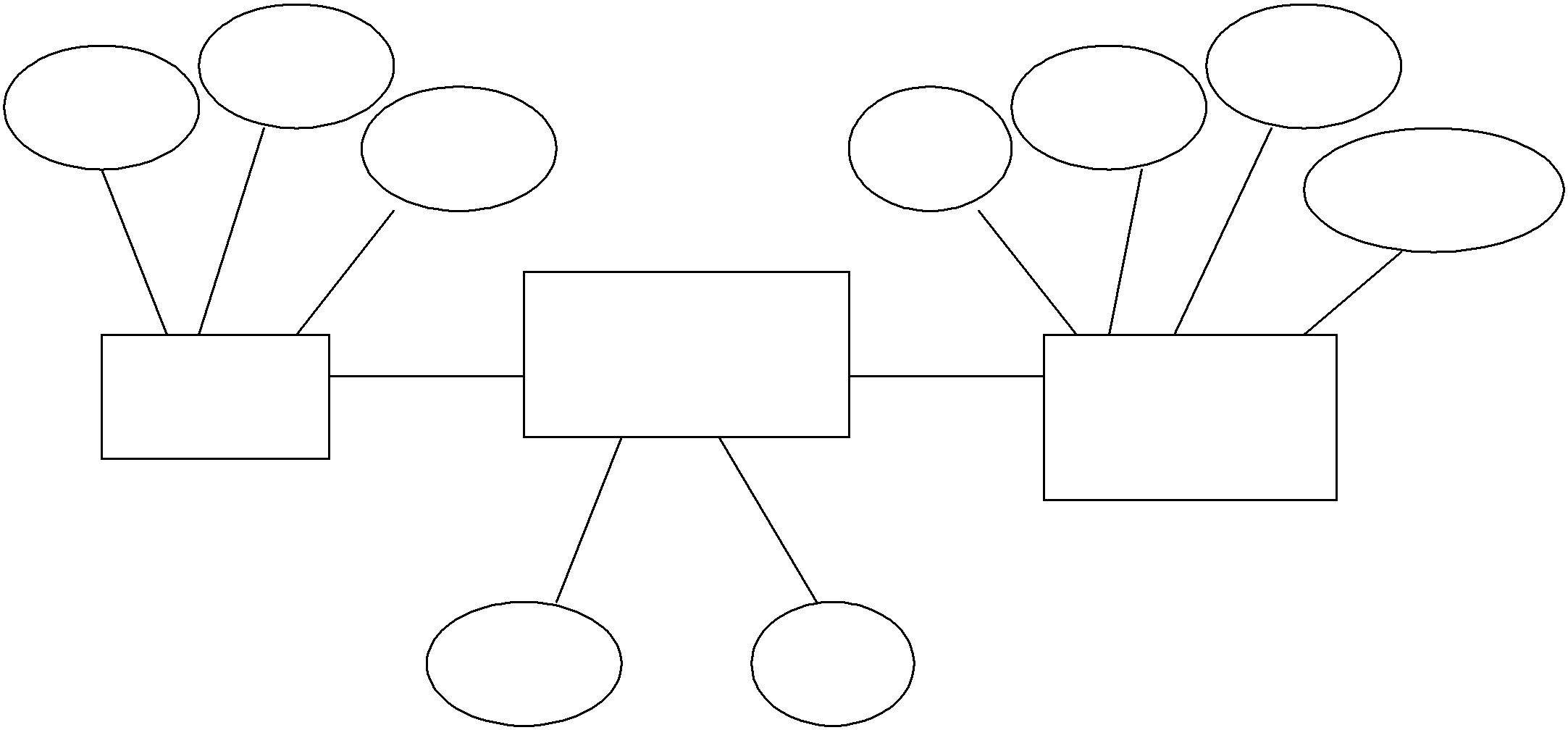
See example below

Example: -

(i) Book to author relationship is a many-to-many. A new entity BookAuthor is added to enable the creation of many

–to-one and one-to-many.





ISBN

Title

Price

SSN

Name

Phone

Address

BookAuthor

Book Author

ISBN

SSN

*Figure 3.7: A many-to-many relationship between Book and Author*

**Referential integrity**

When using foreign keys to implement relationships each value of foreign key must have matching value in related table otherwise *dangling reference*. This restriction is termed as referential constraint and thus ensuring referential restraint is termed ensuring referential integrity.

**Cascading updates and deletions**

This ensures that referential integrity is maintained but should be used with extreme caution.

**Cascading updates**

This means that if a value in a referenced key is changed then all matching entries in the foreign key are automatically changed



**Cascading deletion**

If a value in the referenced key is deleted then all matching records with same value in foreign key are deleted

**Existence Dependencies**

If the existence of an entity X depends on the existence of entity Y, then X is said to be existence dependent on Y. If Y is deleted, so is X. Y is said to be the dominant entity and X is said to be subordinate entity.

**Primary Keys**

These are special type of more general construct candidate keys. A candidate key is a unique identifier and each relation has at least one candidate key. For a given relation, one of the candidate keys is chosen to be the primary key and the rest are called alternate keys.

**Specialization**

An entity set may include sub-groupings of entities that are distinct in some way from other entities in the set. This is called specialization of the entity set e.g. the entity bank account could have different types e.g.

Credit account

Checking account

Savings account - interest rate

Checking account - overdraft amount

Under checking account you could have type:

i. Standard check account ii. Gold checking account

iii. Senior checking account

For the standard if may be divided by number count of checks gold minimum balance and an interest payment.

Senior checking account - age limit

A specialized entity set may be specialized by one or more distinguishing features.

**Aggregation**

This is abstraction through which relationship are heated as higher-level entities e.g. the relationship set borrower and the entity sets customer and loan can be treated as a higher set called borrower as a whole.

