Vidyavardhini's College of Engineering and Technology

Department of Artificial Intelligence & Data Science

Experiment No.2
Mapping ER/EER to Relational schema model.
Date of Performance:
Date of Submission:

Aim :- Prepare the schema for Relational Model with the ER/ERR diagram, drawn for the identified case study in experiment no.1.

Objective :- To map the Entity Relationship (ER) / Extended Entity-Relationship (EER) Diagram to Relational Model schema and learn to incorporate various schema-based constraints.

Theory:

Mapping an Entity-Relationship (ER) model to a relational database schema involves translating the conceptual model represented in the ER diagram into tables and relationships in a relational database management system (DBMS). Here are the general rules for mapping ER to a schema in a DBMS:

1. Entities to Tables:

a. Each entity in the ER diagram corresponds to a table in the relational schema. b.
The attributes of the entity become the columns of the table.
c. The primary key of the entity becomes the primary key of the table.

2. Relationships to Tables:
a. Many-to-Many Relationships:
i. Convert each many-to-many relationship into a new table.
ii. Include foreign key columns in this table to reference the participating entities.
iii. The primary key of this table may consist of a combination of the foreign keys from the
participating entities
b. One-to-Many and One-to-One Relationships:
i. Represented by foreign key columns in one of the participating tables.
ii. The table on the "many" side of the relationship includes the foreign key column referencing the table on the "one" side.
iii. The foreign key column typically references the primary key of the related table.

3. Attributes to Columns:
a. Each attribute of an entity becomes a column in the corresponding table.
b. Choose appropriate data types for each attribute based on its domain and constraints.
c. Ensure that attributes participating in relationships are represented as foreign keys when
needed

4. Primary and Foreign Keys:

    a. Identify the primary key(s) of each table based on the primary key(s) of the corresponding entity

    b. Ensure referential integrity by defining foreign keys in tables to establish relationships between them.

    c. Foreign keys should reference the primary key(s) of related tables.

d. Ensure that foreign keys have appropriate constraints, such as ON DELETE CASCADE or ON UPDATE CASCADE, to maintain data integrity.

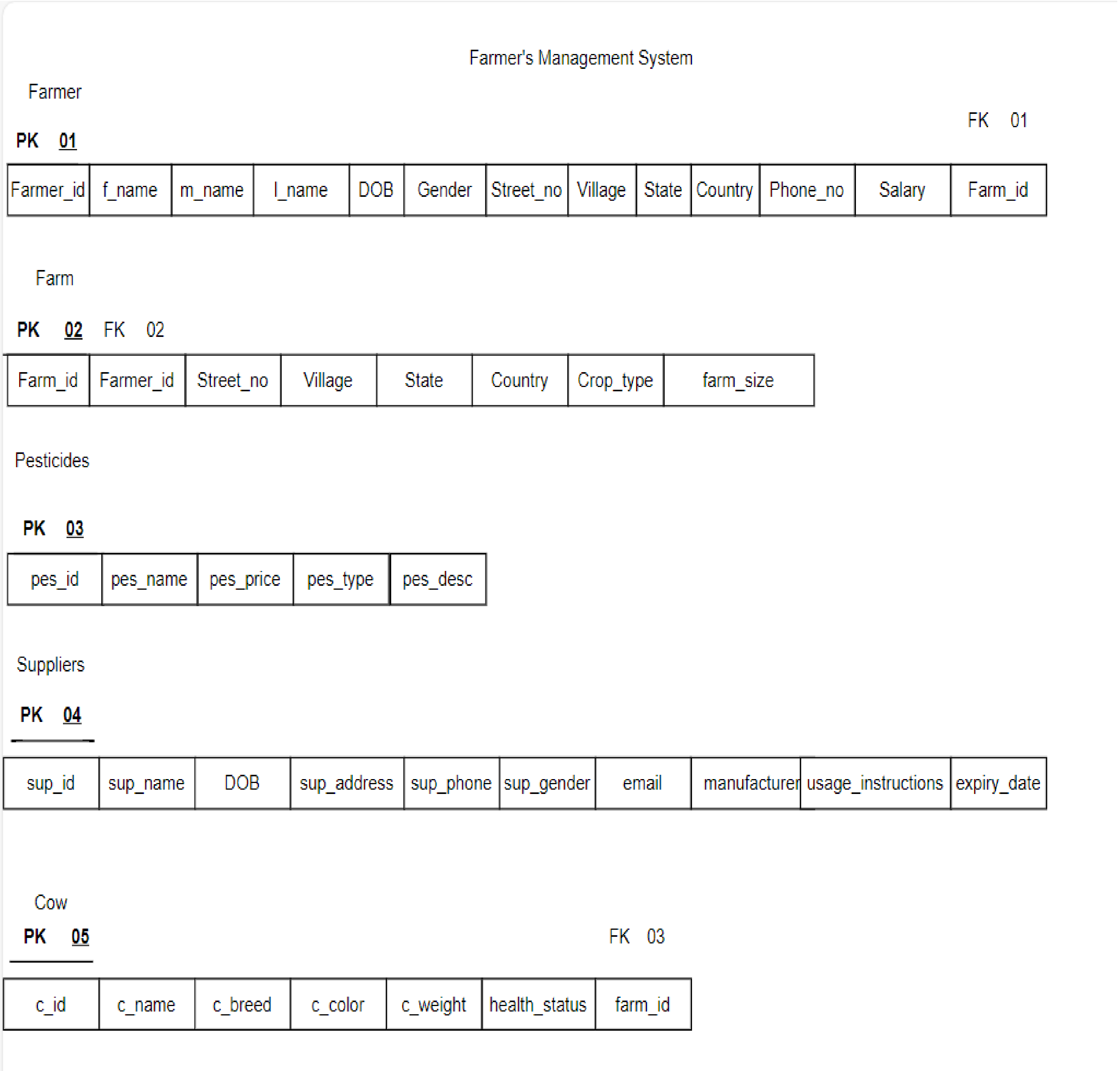
5. Cardinality Constraints:
   a. Use the cardinality constraints from the ER diagram to determine the multiplicity of relationships in the relational schema.

   b. Ensure that the constraints are enforced through the appropriate use of primary and foreign keys.

6. Normalization:
   a. Normalize the schema to minimize redundancy and dependency.
   b. Follow normalization rules such as First Normal Form (1NF), Second Normal Form (2NF), Third Normal Form (3NF), etc., to ensure data integrity and minimize anomalies.

7. Indexing and Optimization:
   a. Consider indexing frequently queried columns to improve query performance. b. Evaluate the schema design for optimization opportunities based on query patterns and performance requirements.

Implementation:



Farmer's Management System

Farmer

PK 01 FK 01

Farmer_id	f_name	m_name	l_name	DOB	Gender	Street_no	Village	State	Country	Phone_no	Salary	Farm_id

Farm

PK 02 FK 02

Farm_id	Farmer_id	Street_no	Village	State	Country	Crop_type	farm_size

Pesticides

PK 03

pes_id	pes_name	pes_price	pes_type	pes_desc

Suppliers

PK 04

sup_id	sup_name	DOB	sup_address	sup_phone	sup_gender	email	manufacturer	usage_instructions	expiry_date

Cow

PK 05 FK 03

c_id	c_name	c_breed	c_color	c_weight	health_status	farm_id

Conclusion:

In this practical, we performed the crucial task of translating the conceptual design of the Farmers Management System, represented by an Entity-Relationship (ER) or Enhanced Entity-Relationship (EER) model, into a concrete relational schema. Through this process, we aimed to bridge the gap between the abstract representation of the system and its implementation in a relational database management system (RDBMS).

1. Write definition of relational schema and notations

Ans. A relational schema is a logical representation of the structure of a relational database. It defines the tables, attributes, keys, and relationships that constitute the database. The relational schema provides a blueprint for organizing and storing data in a relational database management system (RDBMS), enabling efficient data storage, retrieval, and manipulation.

Relational schema notations vary, but commonly used symbols and conventions include:

1. **Tables**: Represented as rectangles with the table name at the top. Each attribute is listed below the table name, along with its data type.

2. **Attributes**: Attributes are listed beneath the table name with their respective data types. Primary key attributes are often underlined to denote their uniqueness.

3. **Keys**: Primary keys are typically denoted by an asterisk (*) or the word "PK" next to the attribute name. Foreign keys are indicated similarly, with the word "FK" or by specifying the referenced table and attribute.

4. **Relationships**: Relationships between tables are depicted by lines connecting the related attributes. Cardinality and participation constraints may be indicated using symbols or annotations near the lines.

2. Write various schema-based constraints.

Ans. Primary Key Constraint: This constraint ensures that each row in a table is uniquely identifiable by a primary key attribute or combination of attributes. It prohibits duplicate and null values in the primary key column(s).

Unique Constraint: The unique constraint ensures that the values in one or more columns of a table are unique across all rows. Unlike primary keys, unique constraints allow null values, but if a column is marked as unique, only one row may contain a null value in that column.

Foreign Key Constraint: Foreign key constraints establish relationships between tables by enforcing referential integrity. A foreign key in one table references the primary key in another table, ensuring that every foreign key value must match a primary key value in the referenced table or be null.

Check Constraint: Check constraints define conditions that must be true for every row in a table. They allow you to specify rules that restrict the values allowed in certain columns. For example, a check constraint can ensure that values in a "age" column are greater than zero and less than 120.

Not Null Constraint: The not null constraint ensures that a column cannot contain null values. It requires that every row in the table must have a value for the specified column, preventing the insertion of null values.

Default Constraint: Default constraints specify a default value for a column when no value is explicitly provided during insertion. If a column with a default constraint is not specified in an INSERT statement, the default value will be used.

Domain Constraint: Domain constraints define the allowable range of values for a column based on its data type. For example, a domain constraint might restrict the values in a "gender" column to 'Male' or 'Female'.

Entity Integrity Constraint: Entity integrity constraints ensure that the primary key attribute of a table cannot contain null values, thus guaranteeing the uniqueness of each row in the table.