CHAPTER

Three

RELATIONAL DATA MODEL AND RELATIONAL DATA BASE CONSTRAINTS

CHAPTER OUTLINE

3.1 RELATIONAL MODEL CONCEPTS

3.2 RELATIONAL MODEL CONSTRAINTS

3.3 RELATIONAL DATABASE SCHEMA

3.4 UPDATE OPERATION AND DEALING WITH CONSTRAINT VIOLATIONS

3.5 DEFINE TRANSACTION

3.1 RELATIONAL MODEL CONCEPTS

Relational data model is the primary data model, which is used widely around the world for data storage and processing. This model is simple and it has all the properties and capabilities required to process data with storage efficiency.

The relational model represents the database as a collection of relations. A relation is nothing but a table of values. Every row in the table represents a collection of related data values. These rows in the table denote a real-world entity or relationship.

The table name and column names are helpful to interpret the meaning of values in each row. The data are represented as a set of relations. In the relational model, data are stored as tables. However, the physical storage of the data is independent of the way the data are logically organized.

Some Popular Relational Database Management Systems are :

- DB2 and Informix Dynamic Server - IBM
- Oracle and RDB - Oracle
- SQL Server and Access - Microsoft

After designing the conceptual model of Database using ER diagram, we need to convert the conceptual model in the relational model which can be implemented using any RDMBS languages like Oracle SQL, MySQL etc.

Relational Model represents how data is stored in Relational Databases. A relational database stores data in the form of relations (tables). Consider a relation Student with attributes Roll_NO, Name, Address, Phone and Age shown in Table 1.

Student

Roll No	Name	Address	Phone	Age
1	Ram	Hyd	9455123451	18
2	Ramesh	Nalgonda	9652431543	18
3	Sujit	Warangal	9156253131	20
4	Suresh	Gadwal		18

Relational Model Concepts

1. **Attribute** : Each column in a Table. Attributes are the properties which define a relation. e.g.; ROLL_NO, NAME

Properties of Relations

- Name of the relation is distinct from all other relations.
- Each relation cell contains exactly one atomic (single) value.
- Each attribute contains a distinct name.
- Attribute domain has no significance.
- tuple has no duplicate value.
- Order of tuple can have a different sequence.

3.2 RELATIONAL MODEL CONSTRAINTS

Every relation has some conditions that must hold for it to be a valid relation. These conditions are called Relational Integrity or Relational Model Constraints.

1. **Entity Integrity Constraint (Integrity Rule 1)**

There must be at least one minimal subset of attributes in the relation, which can identify a tuple uniquely. This minimal subset of attributes is called key for that relation. If there are more than one such minimal subsets, these are called candidate keys.

Employee

EID	Name	Salary	Department	
1	Amit	6,000	Accounts	
2	Sumit	10,000	Computer	
3	Lalit	15,000	Accounts →	This is not allowed because EID is a primary key
4	Deepak	9,000	Electrical	
5	Sandeep	4,000	Civil	

2. **Referential Integrity Rule (Integrity Rule 2)**

A foreign key can be either null or it can have only those values which are present in the primary key with which it is related.

Suppose A be the attribute in relation R1, which is also the primary key in relation R2, then value of A in R1 is either null or same as in relation R2.

Attributes have specific values in real-world scenario. For example, age can only be a positive integer. The same constraints have been tried to employ on the attributes of a relation. Every attribute is bound to have a specific range of values. For example, age cannot be less than zero and telephone numbers cannot contain a digit outside 0-9.

Ex : create table employee

(Eid char (4),

Name char (20),

Age integer (2),

Salary integer,

primary key (Eid),

Check (age>18))

Employee :

EID	Name	Age	Salary
1	Aditya	22	10,000
2	Dinesh	– 18	5,600
3	Sumit	25	8,000
4	Lalit	20	ABC
5	Gaurav	23	11,000

Not allowed because age must be greater than 18

Not allowed because salary has integer data type

4. Key Constraints

In any relation R, if attribute A is primary key then A must have unique value or you can say that primary key attribute must have unique value.

Duplicate values in primary key are invalid.

Primary key or a part of it in any relation cannot be null. Suppose A be the attribute in relation R which is taken as primary key then A must not be null.

Key constraints force that -

- In a relation with a key attribute, no two tuples can have identical values for ke attributes.
- A key attribute cannot have NULL values.

Key constraints are also referred to as Entity Integrity Const

Employee :

EID	Name	Age
1	Aditya	22
2	Sumit	19
3	Deepak	25
2	Manoj	24
4	Dheeraj	28

Invalid ← (EID = 2)

5. **Tuple Uniqueness Constraints :** In any relation R, all tuples in relation R must have distinct values. In other words Duplicate tuples within a single relation are not allowed.

Employee :

EID	Name	Age
1	Aditya	22
2	Sumit	19
3	Deepak	25
2	Sumit	19

Not allowed (Duplicate tuple) ← (EID = 2, Sumit, 19)

3.3 RELATIONAL DATABASE SCHEMA

A set S of relation schemas that belong to the same database is called relational database schema. S is the name of the whole database schema – S = {R1, R2... Rn} – R1, R2... Rn are the names of the individual relation schemas within the database S.

Example : Company database with 6 relation database schemas.

Employee :

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno

Department :

Dname	Dnumber	Mgr_ssn	Mgr_str_date

Dept_Locations

Dnumber	Dlocation

A single transaction may involve any number of retrieval operations and any number of update operations. These retrievals and updates will together form an atomic unit of work against the database. For example, a transaction to apply a bank withdrawal will typically read the user account record, check if there is a sufficient balance, and then update the record by the withdrawal amount.

A large number of commercial applications running against relational databases in **OnLine Transaction Processing (OLTP)** systems are executing transactions at rates that reach several hundred per second.

REVIEW QUESTIONS

One Mark Questions :

1. Define attribute.
2. Define relation schema.
3. Define tuple.
4. Define degree.
5. Define cardinality.
6. What is NULL value ?
7. Define relation instance.
8. Define update operation.
9. Define transaction.

Three Mark Questions :

1. Write the properties of relations.
2. List the relational model constraints.
3. Write about Referential Integrity rule.
4. Write about Domain constraints in detail.
5. Write an example to convert ER-diagram to relational schema.

Five Mark Questions :

1. Explain relational model constraints.
2. Explain relational database schema.
3. Give the steps to convert ER diagram to relational database schema/table.
4. Explain different update operation constraint violations.

Customer ID	Customer	Status Name	DeptID
101	Amazon	Active	1
102	Google	Active	2
103	Flipkart	Active	3
104	Apple	Active	3

Customer ID	Customer	Status Name	Dept ID
101	Amazon	Active	1
102	Google	Active	2
103	Flipkart	Active	3

In the above-given example, CustomerName= "Apple" is deleted from the table.

The Delete operation could violate referential integrity if the tuple which is deleted is referenced by foreign keys from other tuples in the same database.

The Delete operation can violate only referential integrity. This occurs if the tuple being deleted is referenced by foreign keys from other tuples in the database. To specify deletion, a condition on the attributes of the relation selects the tuple (or tuples) to be deleted. Here are some examples.

Operation :

Delete the Customer tuple with CustomerName='Apple'

Result : Acceptable.

Operation :

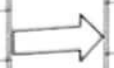
Delete the Department tuple with DeptID = 1.

Result : This deletion is not acceptable, because there are tuples in Customer relation that refer to this tuple. Hence, if the tuple in Departmet is deleted, referential integrity violations will result.

3.5 DEFINE TRANSACTION

A database application program running against a relational database typically executes one or more transactions. A **transaction** is an executing program that includes some database operations, such as reading from the database, or applying insertions, deletions, or updates to the database. At the end of the transaction, it must leave the database in a valid or consistent state that satisfies all the constraints specified on the database schema.

Customer ID	Customer	Status Name	DeptID
101	Amazon	Active	1
102	Google	Active	2
103	Flipkart	Active	3
104	Apple	Active	3



Customer ID	Customer	Status Name	Dept ID
101	Amazon	Active	1
102	Google	Inactive	2
103	Flipkart	Active	3
104	Apple	Active	3

The Update (or Modify) operation is used to change the values of one or more attributes in a tuple (or tuples) of one relation R. It is necessary to specify a condition on the attributes of the relation to select the tuple (or tuples) to be modified. Here are some examples.

Operation :

Update the status of Customer tuple with CustomerName = 'Google' from 'Active' to 'Inactive'

Result : Aceptable.

Operation :

Update the DeptID of the Customer tuple with CustomerID=101 to 2.

Result : Acceptable.

Operation :

Update the DeptID of the Customer tuple with CustomerID = 102 to 5.

Result : Unacceptable, because it violates referential integrity.

Operation :

Update the CustomerID of the Customer tuple with CustomerID = 102 to 103.

Result : Unacceptable, because it violates primary key constraint by repeating a value that already exists as a primary key in another tuple.

Delete Operation :

To specify deletion, a condition on the attributes of the relation selects the tuple to be delete.

Department :

DeptID	Location
1	Delhi
2	Hyderabad
3	Mumbai

The Insert operation provides a list of attribute values for a new tuple t that is to be inserted into a relation R. Insert can violate any of the four types of constraints. Domain constraints can be violated if an attribute value is given that does not appear in the corresponding domain or is not of the appropriate data type. Key constraints can be violated if a key value in the new tuple t already exists in another tuple in the relation r(R). Entity integrity can be violated if any part of the primary key of the new tuple t is Null. Referential integrity can be violated if the value of any foreign key in t refers to a tuple that does not exist in the referenced relation. Here are some examples to illustrate this discussion.

Operation :

Insert <NULL, 'TCS', 'Active',2> into Customer.

Result : This insertion violates the entity integrity constraint (NULL for the primary key CustomerID), so it is rejected.

Operation :

Insert <104, 'CTS', 'InActive',1> into Customer.

Result : This insertion violates the key constraint because another tuple with the same CustomerID value already exists in the Customer relation, and so it is rejected.

Operation :

Insert <106, 'Ebay','Active',4> into Customer.

Result : This insertion violates the referential integrity constraint specified on DeptID in Customer because no corresponding referenced tuple exists in Department with DeptID=4.

Operation :

Insert <106,'Ebay', 'Active',3> into Customer.

Result : This insertion satisfies all constraints, so it is acceptable.

Update Operation :

You can see that in the below-given relation table CustomerName= 'Google' is updated from Active to Inactive.

1. **SELECT** : We use this clause to specify the data or information that we require to answer the query.
2. **FROM** : We use this clause to specify the source of data for answering the query which can be a data table, an existing query or a combination of both.
3. **WHERE** : We use this clause to specify the conditions that we apply to narrow down the choice of data in order to extract the data information that is desirable in the SELECT clause.

3.4 UPDATE OPERATION AND DEALING WITH CONSTRAINT VIOLATIONS

There are three basic update operations that can change the states of relations in the data-base : Insert, Delete, and Update (or Modify). They insert new data, delete old data, or modify existing data records. Insert is used to insert one or more new tuples in a relation, Delete is used to delete tuples, and Update (or Modify) is used to change the values of some attributes in existing tuples. Whenever these operations are applied, the integrity constraints specified on the relational database schema should not be violated.

- Insert is used to insert data into the relation
- Delete is used to delete tuples from the table.
- Update or Modify allows you to change the values of some attributes in existing tuples.

Whenever one of these operations are applied, integrity constraints specified on the relational database schema must never be violated.

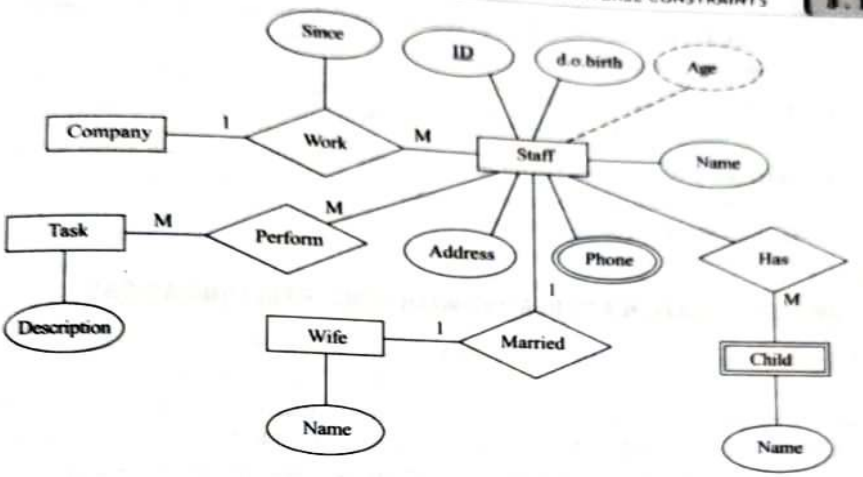
Insert Operation :

The insert operation gives values of the attribute for a new tuple which should be inserted into a relation.

Consider the following relations Customer and Department

Customer :

CustomerID	Customer Name	Status	DeptID
101	Amazon	Active	1
102	Google	Active	2
103	Flipkart	Active	3
104	Apple	Active	3



The relational schema for the ER Diagram is given below as:

Company(CompanyID, name, address)

Staff(StaffID, dob, address, ***WifeID***)

Child(ChildID, name, ***StaffID***)

Wife (WifeID, name)

Phone(PhoneID, phoneNumber, ***StaffID***)

Task (TaskID, description)

Work(WorkID, ***CompanyID, StaffID***, since)

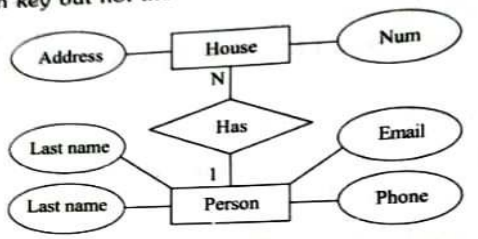
Perform(PerformID, ***StaffID, TaskID***)

Interaction with Databases

The Structured Query Language or SQL makes it easy for the user to interact with the database. It is a comprehensive database language and contains statements for data definition, query, and update.

SQL also facilitates the migration of the users from one database application to another database application. Data Query Language (DQL) is a subset of SQL and is mostly in use to get the answer of most of the basic queries. The basic set of queries consists of :

(iv) 1 : N Relationships : For simplicity, use attributes in the same way as 1:1 relationship but we have only one choice as opposed to two choices. For instance, the Person can have a House from zero to many, but a House can have only one Person. To represent such relationship the personid as the Parent node must be placed within the Child table as a foreign key but not the other way around as shown next :

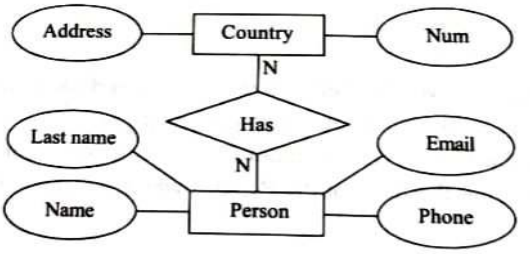


It should convert to:

Persons(personid, name, lastname, email)

House (houseid, num, address, ***personid***)

(v) N : N Relationships : We normally use tables to express such type of relationship. This is the same for N – ary relationship of ER diagrams. For instance, The Person can live or work in many countries. Also, a country can have many people. To express this relationship within a relational schema we use a separate table as shown below :



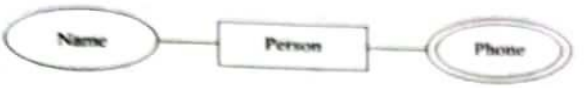
Person(personid, name, lastname, email)

Countrie(countryid, name, code)

HasRelation (hasrelatid, ***personid, countryid***)

Example to convert ER-diagram to relational schema

(ii) Multi-Valued Attributes : A multi-valued attribute is usually represented with a double line oval.



If you have a multi-valued attribute, take the attribute and turn it into a new entity or table of its own. Then make a 1:N relationship between the new entity and the existing one. In simple words.

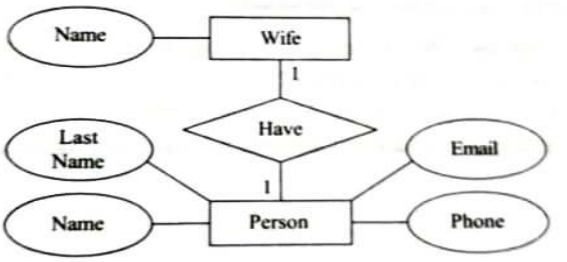
(a) Create a table for the attribute.

(b) Add the primary (id) column of the parent entity as a foreign key within the new table as shown below :

Person(personid , name, lastname, email)

Phone(phoneid , ***personid***, phone)

(iii) 1 : 1 Relationships :



To keep it simple and even for better performances at data retrieval, I would personally recommend using attributes to represent such relationship. For instance, let us consider the case where the Person has or optionally has one wife. You can place the primary key of the wife within the table of the Persons which we call in this case Foreign key as shown below.

Person(personid , name, lastname, email , ***wifeid***)

Wife (***wifeid*** , name)

Or vice versa to put the personid as a foreign key within the Wife table as shown below :

Person(personid , name, lastname, email)

Wife (wifeid , name, ***personid***)

Project

Pname	Pnumber	Plocation	Dnum

Works_On

Essn	Pno	Hours

Dependent

Essn	Dependent_name	Sex	Bdate	Relationship

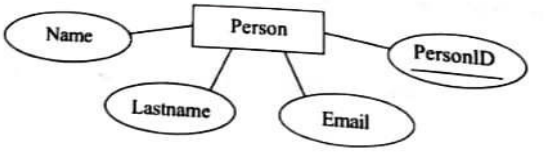
Convert ER Diagram to Relational Database schema/table

The guidelines or rules to design the relational database schema provide a step-by-step procedure. This procedure converts the ER design into a Relational Data model design in order to form the desired database. Following are the steps to convert the ER design into a Relational Data model design :

The ER Model is intended as a description of real-world entities. Although it is constructed in such a way as to allow easy translation to the relational schema model, this is not an entirely trivial process. The ER diagram represents the conceptual level of database design meanwhile the relational schema is the logical level for the database design. We will be following the simple rules :

(i) **Entities and Simple Attributes :** An entity type within ER diagram is turned into a table. You may preferably keep the same name for the entity or give it a sensible name but avoid DBMS reserved words as well as avoid the use of special characters. Each attribute turns into a column (attribute) in the table. The key attribute of the entity is the primary key of the table which is usually underlined. It can be composite if required but can never be null.

Taking the following simple ER diagram :



The initial relational schema is expressed in the following format writing the tab names with the attributes list inside a parentheses as shown below for

Person(personid , name, lastname, email)

Referential integrity constraints work on the concept of Foreign Keys. A foreign key is a key attribute of a relation that can be referred in other relation.

Referential integrity constraint states that if a relation refers to a key attribute of a different or same relation, then that key element must exist.

These are the rules or constraints applied to the database to keep data stable, accurate or consistent. To keep database consistent we have to follow some rules known as integrity rules or integrity constraints.

Employee :

EID	Name	Salary	Dept-ID
1	Amit	6,000	1A
2	Sumit	10,000	2C
3	Lalit	15,000	4F
4	Deepak	9,000	3F
5	Sandeep	4,000	-

Null value is allowed

This is not allowed because Dept-ID is a foreign key and the value 4F is not present in attribute Dept-ID of relation Department.

Department :

Dept-ID	Dept-Name
1A	Accounts
2C	Computer
3E	Electrical
4C	Civil

3. Domain Constraints

The restrictions which we applied on domain are known as domain constraints. These restrictions are applied to every value of attribute. By following these constraints you can keep consistency in database. These restrictions include data types (integer, varchar, char, time format, date format etc.), size of variable, checks (like value not null etc.) etc.

2. **Tables** - In the Relational model the, relations are saved in the table format. It is stored along with its entities. A table has two properties rows and columns. Rows represent records and columns represent attributes.

3. **Tuple** - Each row in the relation is known as tuple. The above relation contains 4 tuples, one of which is shown as :

1	Ram	Delhi	9455123451	18

4. **Relation Schema :** A relation schema represents the name of the relation with its attributes. e.g.; Student (Roll– No, Name, Address, Phone and Age) is relation schema for Student. If a schema has more than 1 relation, it is called Relational Schema.

5. **Degree :** The number of attributes in the relation is known as degree of the relation. The Student relation defined above has degree 5.

6. **Cardinality :** The number of rows present in the table is known as cardinality of the relation. The student relation defined above has cardinality 4.

7. **Column :** Column represents the set of values for a particular attribute. The column Roll_No is extracted from relation Student.

Roll_No
1
2
3
4

8. **Relation Instance** - The set of tuples of a relation at a particular instance of time is called as relation instance. Table 1 shows the relation instance of Student at a particular time. It can change whenever there is insertion, deletion or updation in the database.

9. **Relation Key** - Every row has one, two or multiple attributes, which is called relation key.

10. **Attribute Domain** - Every attribute has some pre-defined value and scope which is known as **attribute domain**.

11. **NULL Values** - The value which is not known or unavailable is called Null value. It is represented by blank space. e.g.; Phone of Student having Roll_No 4 is Null.