**The Relational Database Schema**

**Learning Objectives**

*After completing this topic, you should be able to*

* *describe how to outline a relationship database schema*
* *describe the concept of referential integrity*

**1. Schemas and referential integrity**

The usability of a relational database depends on its architecture and on the extent to which the database can maintain consistency and minimize redundancy.

Various types of schemas constitute the architecture of a relational database, while referential integrity ensures the consistency and redundancy criteria. You can build an efficient relational database and manage it effectively if you correctly understand how the schemas and their components interact.

Each database architecture is made up of three types of schemas to identify the structure and concepts for the database's system. This is known as the three-schema architecture.

**Internal schema**

The internal schema provides details of how and where data is physically stored. In other words, this schema represents the physical structure of a database. The internal schema is not relational.

**Conceptual schema**

The logical structure of a database is represented by the conceptual schema. This schema includes the abstraction of all the database entities, the relationships among them, their attributes, and constraints. This schema hides the complexities of the physical schema and presents data in a simplified format. This format is relational, and database administrators and developers can easily customize and use it.

**External schema**

The external schema represents the data that is displayed to users. This schema is based on the conceptual schema. The remaining part of the database is hidden from users.  
  
A database administrator can customize the external schema for each user according to the user's access rights and permissions. This ensures that users can only view and modify the data they are authorized to access. The external schema is relational because visible data is in the form of relational tables.

When managing databases, you mainly have to work with the conceptual and external schemas.  
  
In the conceptual schema, data is organized into a group of tables. Each table is formally known as a relation. This is because each row of data in a table relates to a particular entity, such as a specific employee.  
  
Each relation, or table, has various components associated with it. These include

**Graphic**

*The example depicts a table named Employees. This table has one header row and three data rows. Some of the column headings in the table are Employee ID, First Name, Last Name, E-mail ID, Phone Number, and Salary. The three data rows display data that relates to employees with the IDs A00001, A00002, and A00003, respectively.*

* attributes
* tuples
* domains
* data types
* constraints, and
* keys

Each attribute represents a type of value that you want to associate with every entity in a table. For example, in the Employees table, the Employee ID and Salary attributes represent the ID and salary of every employee. An attribute is informally known as a column, and the name of an attribute serves the same purpose as a column name.

Unlike an attribute or column, a tuple contains all the values associated with a specific entity – for example, the ID, name, salary, and address of a specific employee. Each value maps to a specific column. A tuple is informally known as a row.

The contents of rows can be stored in any order in the internal schema, and you don't need to be aware of their physical location. To access or modify the contents, you only have to use a logical database query. For example, you can use a SELECT query to retrieve the salary of an employee from the Employees table and an INSERT query to add details about a new employee to the table.

Each value you add to a row must conform to a certain specification. This is known as the domain of the corresponding column and specifies what values are allowed to be entered for that specific column. For example, the domain of the Salary column could be numerical values greater than zero, while the domain of the E-mail Address column includes valid e-mail addresses. You should define the domain of each column.

You need to define a data type for each column. A data type signifies the type of data a column can contain. For example, the data type for the Salary column would be a money data type, and the data type for both the First Name and E-mail Address columns would be an alphanumeric data type.

**Note**

*There are multiple types of data types, and each database system can use a different set of data types.*

A column can contain any value that maps to its associated data type and domain. If you want to limit these values, you need to define constraints for the column's domain. For example, you can define a constraint that specifies the lower and upper bounds for the Salary column. This type of constraint is known as a domain constraint.

Apart from using domain constraints to specify lower or upper bounds, you can also use them to

* prevent any column value – such as the first name of an employee – from being left blank
* specify the format and length of data types – such as the format of dates and the lengths of integers and strings
* prevent users from entering column values that don't conform to the specified data type, format, or length – for example, you can prevent users from entering a string under a column whose data type is integer – and
* prevent or minimize data duplication and enhance data integrity

Domain constraints are associated with columns. However, you can also define other types of constraints for columns and even for a row or an entire table.  
  
Two specific constraint types are

**check and**

A check constraint allows data to be inserted only if it satisfies a Boolean expression. You can define check constraints at column or table level. An example of the former is a constraint that allows an age value to be inserted only if it is greater than 18. An example of the latter is a constraint that ensures that when you update an employee's salary to a value greater than $30,000, that their compensation level is above 2%.

**unique**

To ensure that each value under a column is unique, you should apply a unique constraint to the column. For example, you should apply a unique constraint to an Employee ID column so that each employee has a different number. You should use discretion and foresight when applying unique constraints. For example, you shouldn't apply a unique constraint to a column such as First Name or Salary. That's because multiple employees can have the same first name or salary.

The information about any constraints that you define is stored in the system catalog and database dictionary. The database then accesses the relevant constraints whenever a user tries to add or update a column value and allows the new value to be inserted only if it adheres to the constraints.

Keys enable you to further add constraints to the columns in a table.  
  
Keys can be of various types, such as

* primary key
* surrogate key
* superkey
* candidate key, and
* foreign key

In any table, a primary key comprises single or multiple column values that help to uniquely identify a row. A table can only have one primary key defined on it, and all the columns must have unique values for each row.

**Graphic**

*In the example, the Employee ID column is the primary key.*

Usually a table will have a specific column added to be the primary key. This column is called a surrogate key and has no actual meaning to the row. It is only meant to uniquely identify the rows in the table and is used when no other column can do this. The surrogate key is usually a system generated value that is defined for each row in the table. For example, in the Orders table, you can add a column called Order ID to be the surrogate key. Each time a new order is placed by a customer, the value for this column will be automatically generated to track the number of orders and uniquely identify each one.

One or more columns in a table that are used to uniquely identify a row is known as a superkey.  
  
For example, if the combination of First Name, Last Name, and Phone Number are required to uniquely identify each row in the Employees table, these three columns are the superkey for that table.

**Graphic**

*In the example, First Name, Last Name, and Phone Number form a superkey.*

A candidate key is a column that makes up the superkey. If you remove any column, the key no longer remains a superkey, and so it can't help you to identify a specific row. For example, if you remove the First Name column from the superkey, then Last Name and Phone Number are no longer a superkey because they require the First Name column to uniquely identify a row. Therefore, First Name is a candidate key. This is also the case with the other two columns in the superkey – they need each other, and so each of them is a candidate key.  
  
Tables can have multiple superkeys and candidate keys.

You can create a foreign key in a table to reference the primary key in another table to link the tables together. The foreign key column has the same values as the primary key column. The table with the primary key is called the parent table, and the linked table, or referenced table, is the child table.

**Graphic**

*The example depicts the Employees table along with another table named Projects. The Projects table has one header row and three data rows. Some of the column headings in the table are Employee ID, Project ID, Project Name, and Role. The Employee ID column is a foreign key. This key is linked to the Employee ID column in the Employees table. This Employee ID column is a primary key.*

In the example, the foreign key in the Projects table refers to the primary key in the Employees table. This ensures that any employees working on a project exist in the Employees table before they can work on a project. That is, when a row is added to the Projects table, the value entered as the Employee ID must already be a value in the parent table, Employees.  
  
You can also use this column to perform join operations between the two tables – for example, to find the actual employees' names and the projects they are working on.

By using a foreign key to reference a primary key, the database management system ensures referential integrity. This identifies how the data in the parent and child tables are managed. Referential integrity ensures data consistency and integrity between the two tables.  
  
For example, referential integrity ensures that data is updated or inserted into the parent table before data is updated or inserted into the child table to prevent inconsistent data from being entered.  
  
In addition, when a user tries to delete a row from a child table, the system can either

* prevent the deletion unless the user first deletes rows containing matching column values from the parent table or
* delete rows containing matching column values from the child table automatically

**Note**

*This depends on the RDBMS that is being used for the database.*

Foreign and primary keys also help to establish various types of relationships between tables:

**one-to-one**

In a one-to-one relationship there can only be one occurrence of a value between two tables. For example, each department can only have one manager and each manager can only manage one department.   
*In the example, the Departments and Managers tables have a one-to-one relationship. The Departments table has columns named Department Name and Location and the Managers table has columns such as Employee ID, First Name, Last Name, E-mail ID, Phone Number, and Salary.*

**one-to-many, or**

In a one-to-many relationship, a single row in a table is linked to multiple rows in another table. For example, each department has several employees, but each employee can only be in one department.   
*In the example, the Departments table has a one-to-many relationship with the Employees table. The Employees table has columns such as Employee ID, First Name, Last Name, E-mail ID, Phone Number, and Salary.*

**many-to-many**

In a many-to-many relationship, multiple rows in a table are related to multiple rows in another table. For example, multiple employees might work on a number of different projects, and each project can have more than one employee working on it.   
*In the example, the Employees table has a many-to-many relationship with the Projects table. The Projects table has columns such as Employee ID, Project ID, Project Name, and Role.*

Relationships, keys, and other related components help to organize the conceptual schema.  
  
This schema also contains several other components that enable you to access and manage data.

**Indexes**

An index is a type of database structure that allows data to be quickly retrieve from the table the index is based on. Each index is based on the data that is most frequently queried. Any query based on the columns the index is created on first searches the index, and then uses the link provided in the index to directly access the desired row. It doesn't have to read each row in a table before it can access the desired one. This saves a considerable amount of time.

**Triggers**

A trigger is a piece of code that executes when a specific event occurs in a database. For example, a trigger may run when a user tries to access or modify a specific table or row. You use triggers to monitor and thereby eliminate or minimize redundant data.

**Stored procedures**

A stored procedure is a program that enables you to run batch processing tasks and manage data in a database. For example, you can run a single stored procedure to add a row to a table, gather statistical information, and perform calculations. Stored procedures are usually stored in the corresponding databases.

Unlike components such as stored procedures and tables, a view is part of the external schema. Each view is a logical structure that enables you to display tabular data selectively. For example, you can configure a view to display data that's stored in specific columns of one or more tables. You can also create a view that's based on other views or on a collection of tables and views.

Using a view, you can simplify the presentation of data or hide sensitive data. For example, a Customer table can contain sensitive data, such as credit card details. You can create multiple views based on this table, each of which displays data according to the access rights of the intended user type.

**Question**

Identify the correct statements about database schemas.

**Options:**

1. A domain is the range of values a column can have
2. A stored procedure enables you to run batch processing tasks
3. A trigger ensures that a query can quickly retrieve data from a table
4. The external schema enables you to create a different data view for each user type
5. The internal schema organizes database content into tables, columns, and rows

**Answer**

***Option 1:*** *This option is correct. In the conceptual database schema, each column has a domain, which specifies the range of values for the column.*

***Option 2:*** *This option is correct. A stored procedure is a program that is stored in a database. This program enables you to run batch processing tasks and manage data in the database.*

***Option 3:*** *This option is incorrect. An index ensures that a query can quickly retrieve data. In contrast, a trigger is any code that runs only when specific events occur in a database. You can use triggers to monitor redundant data.*

***Option 4:*** *This option is correct. The external schema enables you to selectively create data views. You can customize this schema for each user according to the user's access rights and permissions. The external schema is based on the conceptual schema.*

***Option 5:*** *This option is incorrect. The conceptual schema organizes database content into a logical structure made up of tables, columns, and rows. The internal schema only represents the physical structure of a database.*

**Correct answer(s):**

1. A domain is the range of values a column can have  
2. A stored procedure enables you to run batch processing tasks  
4. The external schema enables you to create a different data view for each user type

**Question**

Which statements about referential integrity are true?

**Options:**

1. It promotes data redundancy
2. It helps to ensure data consistency
3. It ensures that data is inserted in the child table first
4. It requires that each parent row be created before its child row

**Answer**

***Option 1:*** *This option is incorrect. Referential integrity minimizes duplication of data. In other words, it minimizes data redundancy.*

***Option 2:*** *This option is correct. Referential integrity ensures that data is consistent across all the tables in a database. You can use primary and foreign keys, which help to build tabular relationships, as constraints to enforce referential integrity.*

***Option 3:*** *This option is incorrect. Referential integrity ensures that data needs to be inserted into the parent table first before inserting data in the child table.*

***Option 4:*** *This option is correct. A foreign key ensures that each row in a parent table is created or deleted before the corresponding rows in a child table. This promotes referential integrity. You can further strengthen referential integrity by ensuring that rows in a child table are deleted along with related rows in the linked parent table.*

**Correct answer(s):**

2. It helps to ensure data consistency  
4. It requires that each parent row be created before its child row

**Summary**

A database's architecture includes various types of schemas – internal, conceptual, and external. The internal schema represents the physical structure of the database, and the conceptual schema represents its logical structure. The external schema, in contrast, represents the data views available to each user type.  
  
Referential integrity is achieved by using primary and foreign keys. These keys help to maintain data consistency and minimize duplication.  
  
Like keys, indexes, triggers, and stored procedures are also components of the conceptual schema. These components help you to access and manage data.

**Table of Contents**

|  [**Top of page**](http://sumt.wipro.com/ITMS/data/20121227_161050_2388/df_dbfn_a01_it_enus_pif/Content/cca/df_dbfn_a01_it_enus/output/t15/misc/transcript.html#pagetop) |

|  [**Learning Objectives**](http://sumt.wipro.com/ITMS/data/20121227_161050_2388/df_dbfn_a01_it_enus_pif/Content/cca/df_dbfn_a01_it_enus/output/t15/misc/transcript.html#objective) |

| [**1.Schemas and referential integrity**](http://sumt.wipro.com/ITMS/data/20121227_161050_2388/df_dbfn_a01_it_enus_pif/Content/cca/df_dbfn_a01_it_enus/output/t15/misc/transcript.html#section_1) |

| [**Summary**](http://sumt.wipro.com/ITMS/data/20121227_161050_2388/df_dbfn_a01_it_enus_pif/Content/cca/df_dbfn_a01_it_enus/output/t15/misc/transcript.html#section_2) |