**Univent Database Schema – 2NF Justification**

**SQL Code for Database and Tables**

|  |
| --- |
| CREATE DATABASE Univent; |
| USE Univent; |
|  |
| -- USER Table |
| CREATE TABLE User ( |
| user\_id INT PRIMARY KEY, |
| first\_name VARCHAR(50), |
| last\_name VARCHAR(50), |
| age INT, |
| role VARCHAR(50), |
| email VARCHAR(100) UNIQUE |
| ); |
|  |
| -- COLLEGE Table |
| CREATE TABLE College ( |
| college\_id INT PRIMARY KEY, |
| name VARCHAR(100), |
| location VARCHAR(100) |
| ); |
|  |
| -- SUPER\_ADMIN Table |
| CREATE TABLE Super\_Admin ( |
| admin\_id INT PRIMARY KEY, |
| first\_name VARCHAR(50), |
| last\_name VARCHAR(50), |
| email VARCHAR(100), |
| password VARCHAR(100), |
| designation VARCHAR(100), |
| college\_id INT, |
| FOREIGN KEY (college\_id) REFERENCES College(college\_id) |
| ); |
|  |
| -- CLUB\_OR\_SOCIETY Table |
| CREATE TABLE Club ( |
| club\_id INT PRIMARY KEY, |
| name VARCHAR(100), |
| email VARCHAR(100), |
| Category VARCHAR(100), |
| secretary\_name VARCHAR(100),  secretary\_id INT, |
| college\_id INT, |
| FOREIGN KEY (college\_id) REFERENCES College(college\_id),  FOREIGN KEY (secretary\_id)REFERENCES USER(user\_id) |
| ); |
|  |
| -- EVENT Table |
| CREATE TABLE Event ( |
| event\_id INT PRIMARY KEY, |
| name VARCHAR(100), |
| type\_of\_event VARCHAR(100), |
| date DATE, |
| location VARCHAR(100), |
| status VARCHAR(50), |
| organised\_BY INT, |
| max\_num\_of\_participants INT, |
| FOREIGN KEY (organised\_BY) REFERENCES Club(club\_id) |
| ); |
|  |
| -- Event Scheduler for Past Events |
| SET GLOBAL event\_scheduler = ON; |
| DELIMITER $$ |
|  |
| CREATE EVENT move\_old\_events\_to\_past |
| ON SCHEDULE EVERY 1 DAY |
| DO |
| BEGIN |
| INSERT INTO Past\_Event (event\_id, name, type\_of\_event, date, location, status) |
| SELECT event\_id, name, type\_of\_event, date, location, status |
| FROM Event |
| WHERE date < CURDATE() - INTERVAL 2 DAY; |
|  |
| DELETE FROM Event |
| WHERE date < CURDATE() - INTERVAL 2 DAY; |
| END$$ |
|  |
| DELIMITER ; |
|  |
| -- COMPETITION Table |
| CREATE TABLE Competition ( |
| comp\_id INT auto\_increment PRIMARY KEY, |
| name VARCHAR(100), |
| type\_of\_comp VARCHAR(100), |
| date DATE, |
| venue VARCHAR(100), |
| event\_id INT, |
| FOREIGN KEY (event\_id) REFERENCES Event(event\_id) |
| ); |
|  |
| -- TRANSACTION Table |
| CREATE TABLE Transaction ( |
| trans\_id INT AUTO\_INCREMENT PRIMARY KEY, |
| amount DECIMAL(10, 2), |
| description TEXT, |
| trans\_type VARCHAR(50), |
| transferred\_to INT, |
| FOREIGN KEY (transferred\_to) REFERENCES Club(club\_id) |
| ); |
|  |
| -- REGISTERS Table |
| CREATE TABLE Registers ( |
| reg\_id INT AUTO\_INCREMENT PRIMARY KEY, |
| user\_id INT, |
| event\_id INT, |
| UNIQUE (user\_id, event\_id), |
| FOREIGN KEY (user\_id) REFERENCES User(user\_id), |
| FOREIGN KEY (event\_id) REFERENCES Event(event\_id) |
| ); |
|  |
| -- REQUESTS\_APPROVAL Table |
| CREATE TABLE Requests\_Approval ( |
| request\_id INT PRIMARY KEY, |
| club\_id INT, |
| source VARCHAR(100), |
| status VARCHAR(50), |
| approved\_by INT, |
| rejected\_by INT, |
| FOREIGN KEY (club\_id) REFERENCES Club(club\_id), |
| FOREIGN KEY (approved\_by) REFERENCES Super\_Admin(admin\_id), |
| FOREIGN KEY (rejected\_by) REFERENCES Super\_Admin(admin\_id) |
| ); |
|  |
| -- FEEDBACK Table |
| CREATE TABLE Feedback ( |
| feedback\_id INT PRIMARY KEY, |
| event\_id INT, |
| user\_id INT, |
| time TIMESTAMP, |
| rating INT CHECK (rating BETWEEN 1 AND 5), |
| comment TEXT, |
| FOREIGN KEY (event\_id) REFERENCES Event(event\_id), |
| FOREIGN KEY (user\_id) REFERENCES User(user\_id) |
| ); |

**Why the Univent Schema is in 2NF**

**What is Second Normal Form (2NF)?**

A table is in **2NF** if:

1. It is already in **First Normal Form (1NF)** (i.e., atomic values and unique rows).
2. It has **no partial dependency**—which means that all non-prime (non-key) attributes are fully functionally dependent on the **entire primary key**, not just part of it.

**Univent Schema Analysis**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Here’s why each table is in **2NF**: | | | **Table Name** | **Primary Key** | **2NF Justification** | | --- | --- | --- | | **User** | user\_id | All columns like name, age, role, email depend fully on user\_id. | | **College** | college\_id | name and location are atomic and depend on the full key. | | **Super\_Admin** | admin\_id | Attributes depend on admin\_id; college\_id is a foreign key, not part of PK. | | **Club** | club\_id | Attributes like name, email, etc., depend entirely on club\_id. | | **Event** | event\_id | All fields are fully dependent on event\_id. | | **Competition** | comp\_id | All fields depend on comp\_id; event\_id is a FK. | | **Transaction** | trans\_id | Each attribute is fully dependent on trans\_id. | | **Registers** | reg\_id (and unique user\_id, event\_id) | Fully functionally dependent on the primary key. | | **Requests\_Approval** | request\_id | Each attribute depends fully on request\_id. | | **Feedback** | feedback\_id | All attributes are fully functionally dependent. | | |

**Key Observations**

* No table has **composite primary keys** where only part of the key is used to determine a non-key attribute.
* Each table contains **atomic, non-redundant data** with all fields depending **only on their respective primary keys**.
* The **foreign keys** (like college\_id, event\_id, etc.) are used only to **connect related tables**, and don’t violate 2NF principles.

**Conclusion**

The **Univent Database Schema** is fully normalized to **2NF**:

* **No partial dependencies**
* **No data redundancy**
* **Improved data integrity**

This ensures better **efficiency**, **scalability**, and **clean relational structure** for managing event and college-related data.

**Steps to Move from 2NF to 3NF:**

1. **Remove secretary\_name** from the Club table to eliminate the transitive dependency.
2. **Ensure secretary\_id** is retained in the Club table as it correctly references user\_id from the User table.

**SQL Code:**

|  |
| --- |
| -- Remove the secretary\_name column from the Club table |
| ALTER TABLE Club |
| DROP COLUMN secretary\_name; |

**Explanation of the Fix:**

* **Before**: The secretary\_name in the Club table depended on secretary\_id, which in turn depended on user\_id in the User table. This created a transitive dependency.
* **After**: By removing secretary\_name from the Club table, we ensure that all non-key attributes in Club depend directly on the primary key (club\_id), which satisfies the 3NF condition.

### ****Transition from 3NF to BCNF:****

#### ****Overview of BCNF:****

To transition a schema from **3NF** to **BCNF** (Boyce-Codd Normal Form), we must ensure that for every functional dependency in a table, the left-hand side of the dependency (the determinant) is a **superkey**. A **superkey** is any set of attributes that can uniquely identify a record in a table.

#### ****1. Identification of BCNF Violations:****

In our schema, after achieving **3NF**, the next step is to verify if all functional dependencies have **superkeys** on the left-hand side. A table is in **BCNF** if and only if for every non-trivial functional dependency, the determinant is a superkey.

#### ****Step-by-Step Analysis:****

##### **1.** User **Table:**

* **Primary Key**: user\_id
* Functional dependencies:
  + user\_id → (first\_name, last\_name, age, role, email)
* Since user\_id is the primary key, this table satisfies **BCNF**.

##### **2.** College **Table:**

* **Primary Key**: college\_id
* Functional dependencies:
  + college\_id → (name, location)
* Since college\_id is the primary key, this table satisfies **BCNF**.

##### **3.** Super\_Admin **Table:**

* **Primary Key**: admin\_id
* Functional dependencies:
  + admin\_id → (first\_name, last\_name, email, password, designation, college\_id)
* Since admin\_id is the primary key, this table satisfies **BCNF**.

##### **4.** Club **Table:**

* **Primary Key**: club\_id
* Functional dependencies:
  + club\_id → (name, email, category, secretary\_id, college\_id)
  + secretary\_id → secretary\_name (However, this dependency was resolved in 3NF by removing the secretary\_name column from the Club table.)
* As the remaining dependencies in the table are on the primary key (club\_id) and foreign keys referencing other tables, this table satisfies **BCNF**.

##### **5.** Event **Table:**

* **Primary Key**: event\_id
* Functional dependencies:
  + event\_id → (name, type\_of\_event, date, location, status, organised\_BY, max\_num\_of\_participants)
  + organised\_BY → (club\_id) (via foreign key organised\_BY referencing Club(club\_id))
* Since event\_id is the primary key, this table satisfies **BCNF**.

##### **6.** Competition **Table:**

* **Primary Key**: comp\_id
* Functional dependency:
  + comp\_id → (name, type\_of\_comp, date, venue, event\_id)
* Since comp\_id is the primary key, this table satisfies **BCNF**.

##### **7.** Transaction **Table:**

* **Primary Key**: trans\_id
* Functional dependency:
  + trans\_id → (amount, description, trans\_type, transferred\_to)
  + transferred\_to → (club\_id) (via foreign key transferred\_to referencing Club(club\_id))
* Since trans\_id is the primary key, this table satisfies **BCNF**.

##### **8.** Registers **Table:**

* **Primary Key**: reg\_id
* Functional dependencies:
  + reg\_id → (user\_id, event\_id)
  + user\_id → user\_details (since we know user details are already in the User table, this is not problematic.)
* Since reg\_id is the primary key, this table satisfies **BCNF**.

##### **9.** Requests\_Approval **Table:**

* **Primary Key**: request\_id
* Functional dependencies:
  + request\_id → (club\_id, source, status, approved\_by, rejected\_by)
* Since request\_id is the primary key, this table satisfies **BCNF**.

##### **10.** Feedback **Table:**

* **Primary Key**: feedback\_id
* Functional dependency:
  + feedback\_id → (event\_id, user\_id, time, rating, comment)
* Since feedback\_id is the primary key, this table satisfies **BCNF**.

#### ****Conclusion:****

After analyzing all the tables in the schema, we see that **all of them satisfy BCNF**. There are no violations where a non-superkey determines other attributes. Hence, the schema is already in **BCNF**.