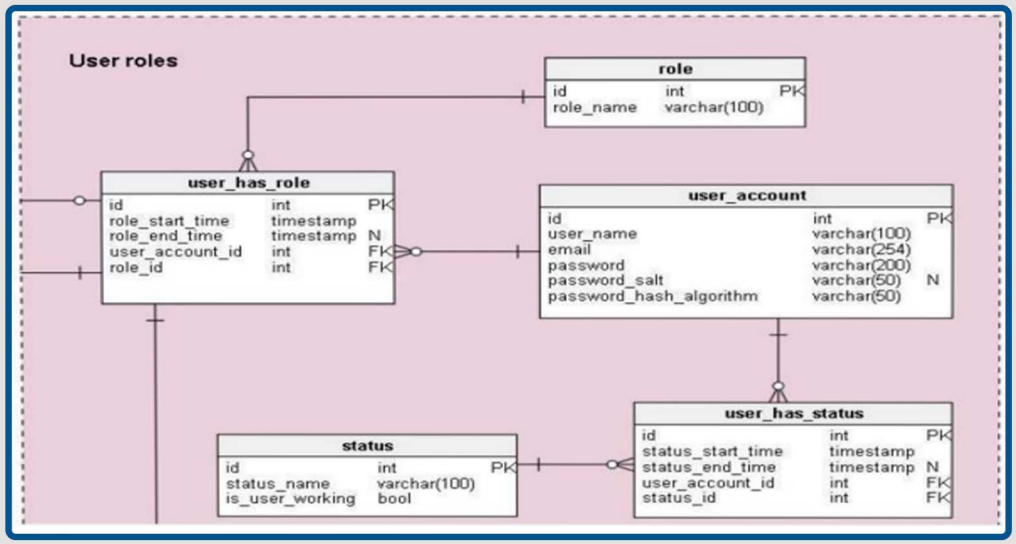
Relational Database Design

(transforming relationships into tables)

Presented By

Nibedita Sarkar

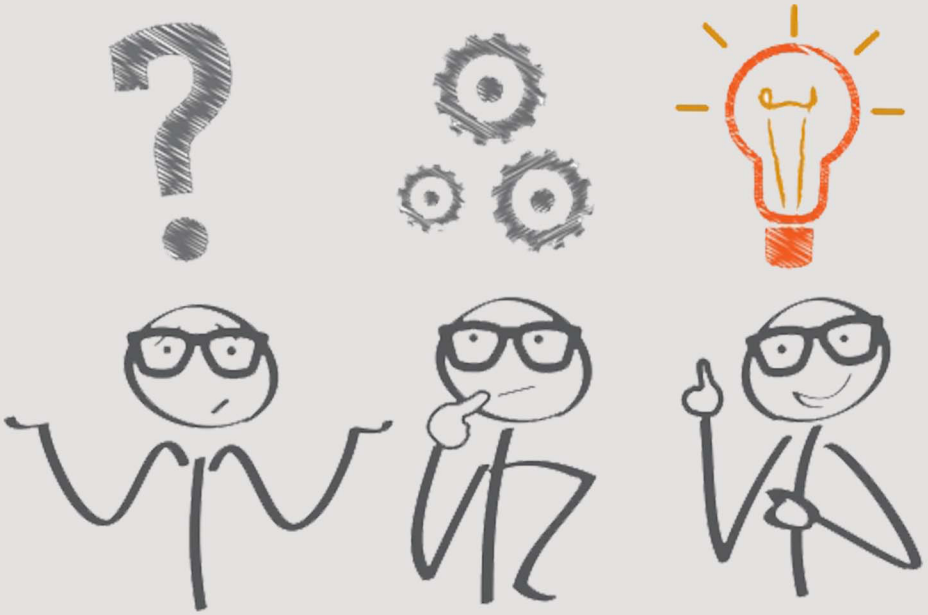
Importance of Converting Relational Database Designs into Tables in SQL Server



- Organizing relational designs into SQL Server tables is important for managing data efficiently.
- Tables provide structure and organization, making it easier to find, use, and analyze data.
- Separate tables for relationships prevent data duplication and ensure accuracy.
- SQL Server offers powerful querying and performance optimization for managing relational databases.
- Converting relational designs to tables enables scalability and easy data modification.

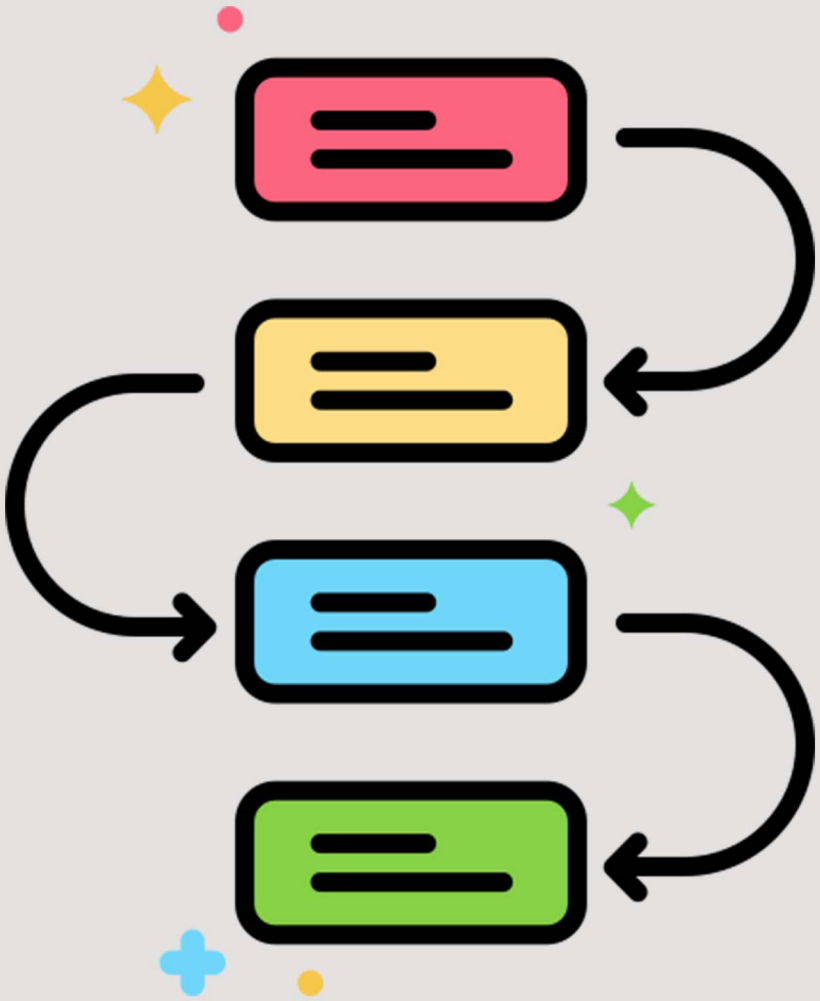
Problem

How to convert a relational design into tables in SQL Server?



Challenge

- Organizing users, roles, user accounts, and statuses into separate tables.
- Need to ensure proper organization and separation of data for efficient management.
- Difficulty in establishing relationships and maintaining data integrity.
- Solution required to convert the relational design into SQL Server tables effectively.



Overview:

This project focuses on converting a relational design into tables and inserting data into them.

Objective:

Transforming the relational design into SQL Server tables for efficient data management.

Importance:

Establishing foreign keys ensures data integrity and maintains relationships between tables.

Key Steps:

Converting design into tables, inserting data, and setting up foreign keys.

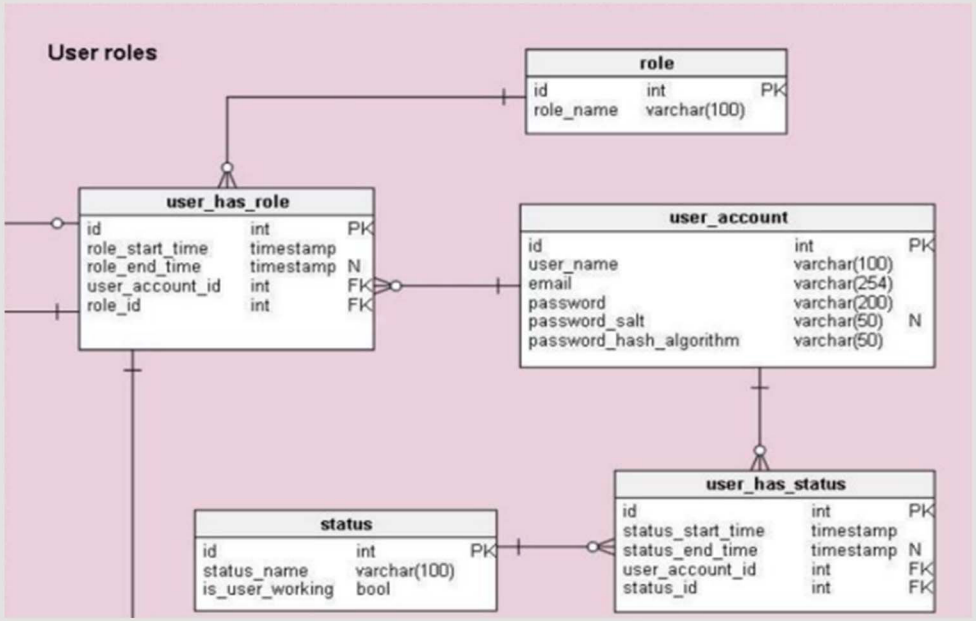
Outcome:

Organized data structure and improved data integrity.

Task to be Performed:

- Define relations/attributes: Identify and define the entities and their attributes for the tables.
- Define primary keys: Determine unique identifiers for each table to ensure data integrity.
- Create foreign keys: Establish relationships between tables using foreign keys.
- Insert data into tables: Populate the tables with at least two rows in each, representing real-world data.
- Delete all data from tables: Clear the tables to start fresh or remove existing vdata.

Table Design:



Role Table:

```
create table role
(
  id int primary key,
  role_name varchar(100)
);
```

User Accounts Table:

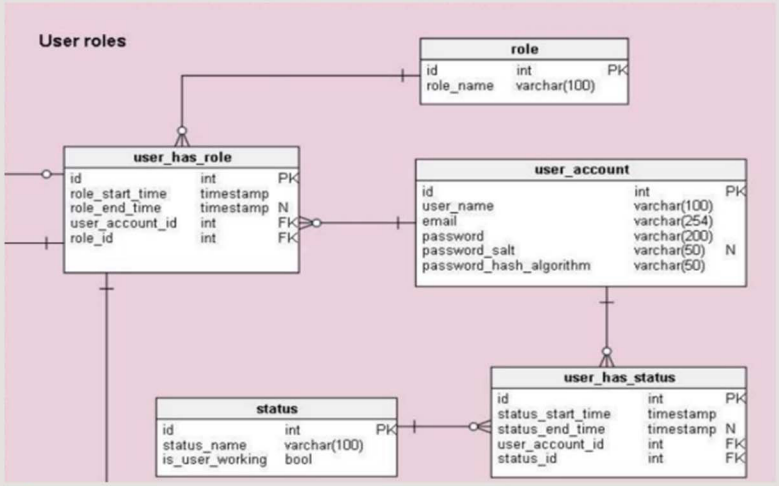
```
create table user_account
(
  id int primary key,
  user_name varchar(100),
  email varchar(254),
  password varchar(200),
  password_salt varchar(50) null,
  password_hash_algorithm varchar(50)
);
```

Status Table:

```
create table status
(
  id int primary key,
  status_name varchar(100),
  is_user_working bit
);
```

User has Role Table:

```
create table user_has_role
(
  id int primary key,
  role_start_time timestamp,
  role_end_time datetime Null,
  user_account_id int,
  role_id int,
  constraint fk_role_id foreign key(role_id) references role(id),
  constraint fk_user_account_id1 foreign key(user_account_id) references user_account(id)
);
```



User has Status Table:

```
create table user_has_status
(
  id int primary key,
  status_start_time datetime,
  status_end_time datetime null,
  user_account_id int,
  status_id int,
  constraint fk_user_account_id foreign key(user_account_id) references user_account(id),
  constraint fk_status_id foreign key(status_id) references status(id)
);
```

Data Insertion :

```
/*Table: role*/
INSERT INTO role (id, role_name)
VALUES (1, 'Admin'), (2, 'User');

/*Table: status*/
INSERT INTO status (id, status_name, is_user_working)
VALUES (1, 'Active', 1), (2, 'Inactive', 0);

/*Table: user_account*/
INSERT INTO user_account (id, user_name, email, password, password_salt, password_hash_algorithm)
VALUES (1, 'JohnDoe', 'john.doe@example.com', 'password123', 'somesalt', 'SHA256'),
    (2, 'JaneSmith', 'jane.smith@example.com', 'pass456', 'anothersalt', 'SHA512');

/*Table: user_has_role*/
INSERT INTO user_has_role (id, role_start_time, role_end_time, user_account_id, role_id)
VALUES (1, DEFAULT, NULL, 1, 1), (2, DEFAULT, NULL, 2, 2);

/*Table: user_has_status*/
INSERT INTO user_has_status (id, status_start_time, status_end_time, user_account_id, status_id)
VALUES (1, CURRENT_TIMESTAMP, NULL, 1, 1), (2, CURRENT_TIMESTAMP, NULL, 2, 2);
```

Displaying Table :

```
SELECT * FROM role;
SELECT * FROM status;
SELECT * FROM user_account;
SELECT * FROM user_has_role;
SELECT * FROM user_has_status;
```

Results | Messages

	id	role_name
1	1	Admin
2	2	User

	id	status_name	is_user_working
1	1	Active	1
2	2	Inactive	0

	id	user_name	email	password	password_salt	password_hash_algorithm
1	1	JohnDoe	john.doe@example.com	password123	somesalt	SHA256
2	2	JaneSmith	jane.smith@example.com	pass456	anothersalt	SHA512

	id	role_start_time	role_end_time	user_account_id	role_id
1	1	0x00000000000007D1	NULL	1	1
2	2	0x00000000000007D2	NULL	2	2

	id	status_start_time	status_end_time	user_account_id	status_id
1	1	2023-06-04 15:51:06.823	NULL	1	1
2	2	2023-06-04 15:51:06.823	NULL	2	2

Data Deletion:

- Drop foreign key constraints to ensure smooth data deletion.
- Delete all the data from each table to start with a fresh dataset.

--DROPIING FOREIGN KEY CONSTRAINTS--

```
alter table user_has_role
drop constraint fk_role_id;

alter table user_has_role
drop constraint fk_user_account_id1;

alter table user_has_status
drop constraint fk_user_account_id;

alter table user_has_status
drop constraint fk_status_id;
```

--DELETE ALL DATA FROM EACH TABLE--

```
DELETE FROM role;
DELETE FROM status;
DELETE FROM user_account;
DELETE FROM user_has_role;
DELETE FROM user_has_status;
```

Table After Deleting all the data :

Results | Messages

id	role_name

id	status_name	is_user_working

id	user_name	email	password	password_salt	password_hash_algorithm

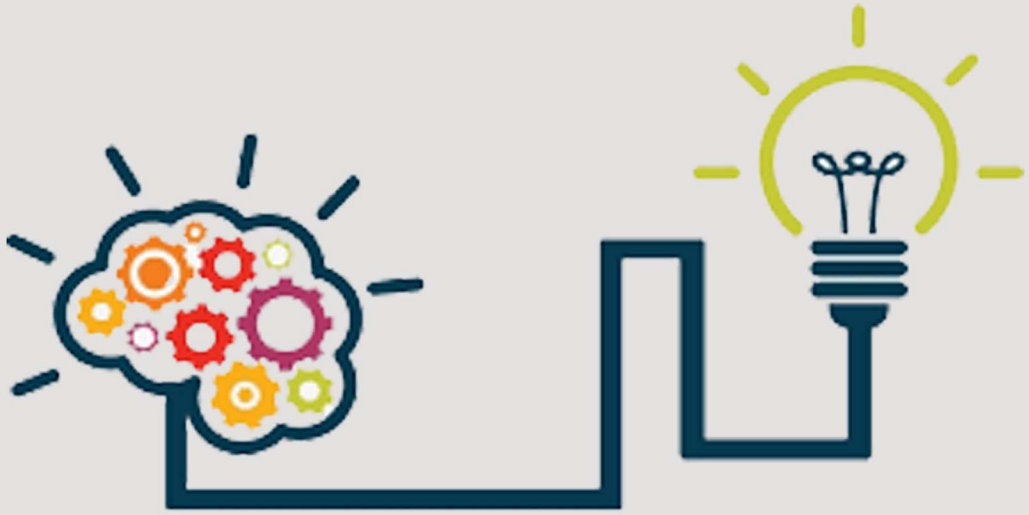
id	role_start_time	role_end_time	user_account_id	role_id

id	status_start_time	status_end_time	user_account_id	status_id

Conclusion :

- Understanding relational database design is essential for organizing and structuring data effectively.
- Creating tables allows us to represent different entities and their attributes in a structured manner.
- Inserting data into tables ensures that the database contains relevant information for analysis and retrieval.
- Considering data integrity through techniques like foreign keys maintains the accuracy and consistency of data.
- Deleting all data from tables, when necessary, helps in proper data management and starting fresh.

By mastering these aspects, we can build robust and efficient databases that facilitate smooth data operations and reliable data analysis.



Presented By

Nibedita Sarkar

https://www.linkedin.com/in/nibedita-sarkar/