1. Foreign Key in MsSQL

In Microsoft SQL Server, a foreign key is a constraint that ensures referential integrity between two tables.

1. Create the parent table with a primary key:

CREATE TABLE parent\_table (

parent\_id INT PRIMARY KEY,

column1 VARCHAR(50),

column2 INT

)

1. Create the child table with a foreign key constraint that references the primary key of the parent table:

CREATE TABLE child\_table (

child\_id INT PRIMARY KEY,

column3 VARCHAR(50),

parent\_id INT, FOREIGN KEY (parent\_id) REFERENCES parent\_table(parent\_id)

)

* The foreign key constraint in the child table specifies that the values in the parent\_id column must exist in the parent\_table's primary key column.
* If you try to insert a value in the child table that does not exist in the parent\_table, you will get an error.

**Simple Examples**

--\_\_\_\_\_\_\_\_ 1. Department Repeted \_\_\_\_\_\_\_\_\_\_\_ 1 chractor tak 1- byte (1bytes=8bits)

----- Employee + Depepartmeent together ---

create table employee1(

id int primary key identity,

name varchar(25),

department varchar(25)

)

insert into employee1 values ('Saqib','Employee'), ('Noman','Employee'), ('ahamd','Employee')

, ('Sajid','HR')

select \* from employee1

--\_\_\_\_\_\_\_\_ 1. Department Id Repeted (int take 1 bit) \_\_\_\_\_\_\_\_\_\_\_

----- Employee --- but department seprated by F.k -----

create table employee3(

id int primary key identity,

name varchar(25),

depId int foreign key References department(id)--department varchar(25)

)

insert into employee3 values ('Asif',1),('Sajid',1),('Arif',1),('Akram',1),('Ahamd',1),('Noman',1),('Billal',1),

('saqib',2),('Subhan' , 2)

select \* from employee3

create table department (

id int primary key identity,

department varchar(25)

)

insert into department values ('Employee'),('Hr'),('Finance')

select \* from department

Multiple Foreign keys Example

Sure, here's an example of using foreign keys to establish relationships between multiple tables in a database:

Suppose we have the following four tables in our database:

1. **users** - stores information about registered users

CREATE TABLE users (

user\_id INT PRIMARY KEY identity,

username VARCHAR(50) UNIQUE,

password VARCHAR(50),

email VARCHAR(50)

);

1. **orders** - stores information about orders placed by users

CREATE TABLE orders (

order\_id INT PRIMARY KEY,

order\_date DATE,

user\_id INT, FOREIGN KEY (user\_id) REFERENCES users(user\_id)

);

1. **order\_items** - stores information about items in each order

CREATE TABLE order\_items (

item\_id INT PRIMARY KEY,

order\_id INT,

product\_id INT,

quantity INT,

constraint cns\_ind\_orderId FOREIGN KEY (order\_id) REFERENCES orders(order\_id),

constraint cns\_ind\_productId FOREIGN KEY (product\_id) REFERENCES products(product\_id)

);

1. **products** - stores information about products available for purchase

CREATE TABLE products (

product\_id INT PRIMARY KEY,

product\_name VARCHAR(50),

price DECIMAL(10,2)

);

In this example, we have established the following relationships using foreign keys:

* The **orders** table has a foreign key **user\_id** that references the **user\_id** column in the **users** table. This establishes a relationship between orders and the users who placed them.
* The **order\_items** table has foreign keys **order\_id** and **product\_id** that reference the **order\_id** and **product\_id** columns in the **orders** and **products** tables, respectively. This establishes relationships between orders, items in those orders, and the products that make up those items.

By using foreign keys to establish these relationships, we can ensure that the data in our database remains consistent and maintain referential integrity between tables. We can also easily query and join data across multiple tables to obtain valuable insights about our users, orders, and products.

2. Why Foreign key – why this ?????

--\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 1st WithOut Foreign key \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

create table users(

id int primary key identity,

name varchar(25),

password varchar(25),

email varchar(25)

)

create table orders(

order\_id int primary key identity,

order\_date date,

userid int

)

select \* from users

select \* from orders

insert into orders values ('2022-7-23',1) -- this 1 user Exist

insert into orders values ('2022-7-23',4) --- this 4 user Noooooot Exist Why its laat him insert the Record

-- With Out Relation --- Table Late To Insert Record

--\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2nd By Foreign key \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

drop table orders

create table orders(

order\_id int primary key identity,

order\_date date,

userId int foreign key references users(id) --userid int

)

select \* from users

select \* from orders

insert into orders values ('2022-7-23',1) -- this 1 user Exist

insert into orders values ('2022-7-23',4) --- this 4 user Not Exist \_\_\_ F.K Will Not Leet Insert record

--- When We Create F.K we Can not Insert Wronge User in Order ---

---- The Benefit of Referential Integrity Constraint

Benefits

Some of the benefits of using foreign keys in a database are:

1. Data consistency: Foreign keys ensure that the data in related tables is consistent. They enforce referential integrity, which means that a record in one table must always have a corresponding record in another table. This prevents orphaned records and data inconsistencies.
2. Data redundancy: Foreign keys help to reduce data redundancy in a database. Without foreign keys, you might need to store the same data in multiple tables, which can lead to inconsistencies and errors. With foreign keys, you can store the data in one table and reference it from other tables as needed.
3. Query performance: Foreign keys can improve query performance by allowing the database to use more efficient join algorithms. Joins are used to combine data from multiple tables, and foreign keys provide a way to identify the relationships between tables. This can help the database engine to optimize queries and reduce the amount of data that needs to be processed.
4. Data integrity: Foreign keys help to ensure data integrity by preventing invalid data from being inserted into related tables. If a foreign key constraint is violated, the database will prevent the insert or update operation from proceeding. This helps to maintain the overall integrity of the database.
5. Simplified database design: Using foreign keys can simplify the database design by reducing the number of tables and columns needed. Instead of duplicating data in multiple tables, you can store the data in one table and reference it as needed from other tables. This can make the database easier to understand and maintain over time.