**Subject:** Julio Pochet - Foreign Keys in SQL

**Understanding Foreign Keys**

A **foreign key** is a column in one table that links to a **primary key** in another table. It’s like a bridge that connects related data, ensuring everything stays organized and consistent. Without foreign keys, databases could become a mess, with orphaned records and broken relationships.

**Real-Life Example**

Imagine we have two tables: Customers and Orders. Each order must belong to a valid customer, so we use a foreign key to connect them:

CREATE TABLE Customers (

customer\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50)

);

CREATE TABLE Orders (

order\_id INT PRIMARY KEY,

customer\_id INT,

order\_date DATE,

FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)

);

Here, the customer\_id in the Orders table is a **foreign key** that references the customer\_id in the Customers table. This makes sure we don’t have orders linked to customers that don’t exist.

**Challenges with Foreign Keys**

🔹 **Accidental Deletion:** What happens to customers' orders if they are deleted? Without the right settings, they might get deleted, too, which isn’t always what we want.

🔹 **Slower Performance:** Every time we insert or update a record, the database has to check the foreign key constraints, which can slow things down in large databases.

🔹 **Data Integrity Issues:** If foreign keys aren’t set up correctly, you could end up with broken links or orphaned records.

**Final Thoughts**

Foreign keys are super useful for keeping databases structured, but they must be used carefully. Sometimes, developers skip them instead of handling relationships at the application level.