How does SQL allow implementation of the entity integrity and referential integrity constraints described in Chapter 3? What about referential triggered actions?

PRIMARY KEY clause

FOREIGN KEY clause

Default SQL action for an integrity violation is to reject the update operation, known as REJECT (p. 186)

Referential triggered actions: SET NULL, CASCADE, and SET DEFAULT

SQL allows the implementation of the entity integrity constraint through the PRIMARY KEY clause. Take a look at the following SQL DDL to create a CUSTOMER relation:

CREATE TABLE customers (

email\_address varchar PRIMARY KEY,

last\_name varchar,

first\_name varchar

);

The PRIMARY KEY clause is read the same as if “UNIQUE NOT NULL” was used after declaring the email\_address column. Declaring that email\_address is a primary key, SQL understands that the value of this attribute, a customer’s email address, in each record of the CUSTOMERS table must be unique and not null, thus supporting the entity integrity constraint in the relation.

To implement referential integrity constraints, SQL allows primary keys from one table to be referenced in another table. Take the example of an ORDERS tables below:

CREATE TABLE orders (

order\_id int PRIMARY KEY,

email\_address varchar REFERENCES customers (email\_address),

…

);

The ORDERS table references the email\_address column in the CUSTOMERS table, so that each order is linked to the customer who submitted it. Using the REFERENCES keyword, the value of the email\_address column for each record in ORDERS will correspond to a value in the email\_address column in CUSTOMERS.

<https://www.postgresql.org/docs/current/ddl-constraints.html#DDL-CONSTRAINTS-PRIMARY-KEYS>

https://www.datacamp.com/tutorial/integrity-constraints-sql