SQL Constraints

A "CONSTRAINT" is a rule applied to a TABLE or COLUMN that allows the database software to maintain data integrity within the database.

The database software will not allow the restrictions of the constraint to be violated.

Constraints can exist at the TABLE level and/or at the COLUMN level.

The following constraints are available in PostgreSQL, but are not supported by bit.io.

NOTE: See Section 13 of the PostgreSQL Tutorial (on Constraints)
https://www.postgresqltutorial.com/

SQL Constraints

Table level constraints:

- Primary Key
- Foreign Key

Column level constraints:

- Not Null
- Check
- Default
- Unique
- Primary Key

SQL Column Constraints

```
CREATE TABLE "alanparadise/nw"."items"
  (
    itemID      INT          NOT NULL PRIMARY KEY,
    itemcode    VARCHAR(5)   UNIQUE,
    itemname    VARCHAR(40)  NOT NULL DEFAULT ' ',
    quantity    INT          NOT NULL DEFAULT 0,
    price       REAL         NOT NULL DEFAULT 0
  );
```

The UNIQUE constraint prevents inserting a row with a duplicate value

The PRIMARY KEY constraint includes a UNIQUE constraint

SQL Column Constraints

```
CREATE TABLE "alanparadise/nw"."items"
  (
    itemID      INT           NOT NULL PRIMARY KEY,
    itemcode    VARCHAR(5)    UNIQUE,
    itemname    VARCHAR(40)   NOT NULL DEFAULT ' ',
    quantity    INT           NOT NULL DEFAULT 0,
    price       DECIMAL(9,2)  CHECK price < 1000
  );
```

The CHECK constraint includes a condition on the column

SQL Table Constraints

The PRIMARY KEY and FOREIGN KEY constraints (as table-level constraints)

```
CREATE TABLE "alanparadise/nw"."items"
  (
    itemID      INT            NOT NULL,
    supplierid  INT            NOT NULL,
    itemcode    VARCHAR(5)     UNIQUE,
    itemname    VARCHAR(40)    NOT NULL DEFAULT ' ',
    quantity    INT            NOT NULL DEFAULT 0,
    price       DECIMAL(9,2)   CHECK price < 1000
    PRIMARY KEY (itemID)
    CONSTRAINT fk_supplier FOREIGN KEY(supplierid) REFERENCES supplier(supplierid)
  );
```

SQL Table Constraints

A concatenated or composite Primary Key can be set at the table level

```
CREATE TABLE "alanparadise/nw"."items"
  (
    itemID     INT           NOT NULL,
   supplierid INT           NOT NULL,
    itemcode   VARCHAR(5)    UNIQUE,
    itemname   VARCHAR(40)   NOT NULL DEFAULT ' ',
    quantity   INT           NOT NULL DEFAULT 0,
    price      DECIMAL(9,2) CHECK price < 1000
   PRIMARY KEY (itemID, supplierID)
    );
```

SQL Table Constraints

By giving a table-level constraint a name (fk_supplier), the constraint is modifiable by an ALTER

```
CREATE TABLE "alanparadise/nw"."items"
  (
    itemID      INT           NOT NULL,
    supplierid INT        NOT NULL,
    itemcode    VARCHAR(5)    UNIQUE,
    itemname    VARCHAR(40)   NOT NULL DEFAULT ' ',
    quantity    INT           NOT NULL DEFAULT 0,
    price       DECIMAL(9,2) CHECK price < 1000
    PRIMARY KEY (itemID)
    CONSTRAINT fk_supplier FOREIGN KEY(supplierid)
            REFERENCES supplier(supplierid)
  );
```

SQL Table Constraints

A table-level foreign key constraint can indicate how to handle the situation when the foreign key in the PARENT table is deleted or updated.

```
CONSTRAINT fk_supplier FOREIGN KEY(supplierid)
           REFERENCES supplier(supplierid)
       ON UPDATE <update action>
       ON DELETE <delete action>
```

SQL Table Constraints

```
CONSTRAINT fk_supplier FOREIGN KEY(supplierid)
           REFERENCES supplier(supplierid)
        ON UPDATE <update action>
        ON DELETE <delete action>
```

ACTION may be:

CASCADE	The change to the parent is cascaded to all affected child rows
NO ACTION	The change to the parent is prohibited
SET NULL	The foreign key column in the child is set to NULL
SET DEFAULT	The foreign key column in the child is set to its default value
RESTRICT	The change to the parent is prohibited