Constraints in SQL

Constraints in SQL

In SQL we can:

- limit values in a given column
- Add a condition for several columns at once
- forbid NULL
- Restrict duplicates
- maintain referential integrity between tables (foreign keys)
- http://www.postgresql.org/docs/9.4/interactive/ddl-constraints.html

CHECK

- Allows to narrow down data types (e.g. add a range)
  - `price numeric CHECK (price > 0)`
  - `price numeric CONSTRAINT positive_price`
    `CHECK (price > 0)`
- Allows to add links between columns in a single table
  - `CONSTRAINT valid_discount CHECK (price > discounted_price)`
- Naming the constraint makes it easier to debug and report bugs

CHECK

```
CREATE TABLE products (
    product_no integer,
    name text,
    price numeric CHECK (price > 0),
    discounted_price numeric CHECK (discounted_price > 0),
    CHECK (price > discounted_price)
);
```

NOT NULL

```
name text NOT NULL
```

is equivalent to a special case of CHECK:

```
CHECK (name IS NOT NULL)
```

The first version is faster, but it can't be named

UNIQUE

Unique keys: There can't be two rows with the same value in a table

(or the tuple in the given columns)

- `product_no integer UNIQUE,`
- `product_no integer CONSTRAINT must_be_different UNIQUE,`
-
```
CREATE TABLE products (
    product_no integer,
    name text,
    price numeric,
    UNIQUE (product_no)
);
```

UNIQUE

PostgreSQL:

- Adding a unique constraint will automatically create a unique btree index on the column or group of columns used in the constraint.
- A uniqueness constraint on only some rows can be enforced by creating a partial index
- http://www.postgresql.org/docs/9.4/interactive/indexes-partial.html

PRIMARY KEY

- PRIMARY KEY means UNIQUE NOT NULL
- Single primary key for a single table
- useful for client applications (e.g. when UPDATE, it is necessary to clearly identify which row is changing)
- examples:

```
product_no integer PRIMARY KEY
```

```
PRIMARY KEY (a, c)
```

FOREIGN KEY

- specifies that the value in a given column (or group of columns) is a reference to an existing value in some other table, i.e. ensures referential integrity
- Referenced columns must have a unique key created: UNIQUE or PRIMARY KEY
- strongly recommended for use when decomposing a large relation to a small
-
```
CREATE TABLE t1 (
        a integer PRIMARY KEY, b integer, c integer,
        FOREIGN KEY (b, c) REFERENCES other_table (x, y));
```

FOREIGN KEY

```
CREATE TABLE products (
    product_no integer PRIMARY KEY,
    name text, price numeric);
CREATE TABLE orders (
    order_id integer PRIMARY KEY);
CREATE TABLE order_items (
    product_no integer REFERENCES products ON DELETE RESTRICT,
    order_id integer REFERENCES orders ON DELETE CASCADE,
    quantity integer, PRIMARY KEY (product_no, order_id));
```

FOREIGN KEY

```
ON DELETE
    CASCADE -- deletes the row that referred to the deleted row
    RESTRICT -- deletes nothing, the DELETE operation ends with an error
    SET NULL -- sets the reference to NULL (fails for NOT NULL columns)
    SET DEFAULT -- sets the reference value to default
                (fails if an incorrect link is created in this way)
    NO ACTION -- this is the default, DELETE will end with an error
```

FOREIGN KEY

```
ON UPDATE
    CASCADE -- changes the reference value to a new referenced value
               (preserves the link between the lines)
    RESTRICT -- prevents a change in the referenced table, UPDATE fails
    SET NULL -- sets the reference to NULL (the binding expires)
    SET DEFAULT -- sets the reference value to default
                   (fails if an incorrect link is created in this way)
    NO ACTION -- this is the default, the binding breaks
```

FOREIGN KEY

- When is the validity of the foreign key evaluated?

```
DEFERRED -- only when committing a transaction

IMMEDIATE -- immediately after the operation
```

- Options for Defining a Foreign Key:

```
NOT DEFERRABLE -- Always immediate evaluation

DEFERRABLE INITIALLY IMMEDIATE -- as part of the transaction

      you can choose when to evaluate, the default is immediately
DEFERRABLE INITIALLY DEFERRED -- as part of the transaction

      you can choose when to evaluate, the default is commit
```

http://www.postgresql.org/docs/9.4/static/sql-set-constraints.html