Relational DB & SQL - C11

Basic Concepts of the Relational Model

To do: Go through the activity to the end

Table or Column Constraints

The Optional Column Constraints are `NULL`, `NOT NULL`, `UNIQUE`, `PRIMARY KEY` and `DEFAULT`, used to initialize a value for a new record. The column constraint `NULL` indicates that null values are allowed, which means that a row can be created without a value for this column. The column constraint `NOT NULL` indicates that a value must be supplied when a new row is created.

To illustrate, we will use the SQL statement `CREATE TABLE departments` to create the **departments** table with 7 attributes or fields.

```
1  CREATE TABLE departments
2  (
3      id BIGINT NOT NULL,
4      name VARCHAR(20) NULL,
5      dept_name VARCHAR(20) NULL,
6      seniority VARCHAR(20) NULL,
7      graduation VARCHAR(20) NULL,
8      salary BIGINT NULL,
9      hire_date DATE NULL,
10         CONSTRAINT pk_1 PRIMARY KEY (id)
11  ) ;
```

The first field is id with a field type of `BIGINT`. The user cannot leave this field empty (`NOT NULL`).

Similarly, the second field is name with a field type `VARCHAR` of length 20.

After all the table columns are defined, a table constraint, identified by the word `CONSTRAINT`, is used to create the primary key:

```
CONSTRAINT pk_1 PRIMARY KEY(id)
```

We can use the optional column constraint `IDENTITY` to provide a unique, incremental value for that column. Identity columns are often used with the `PRIMARY KEY` constraints to serve as the unique row identifier for the table. The `IDENTITY` property can be assigned to a column with a tinyint, smallint, int, decimal, or numeric data type. This constraint:

- Generates sequential numbers.
- Does not enforce entity integrity.
- Only one column can have the `IDENTITY` property.

- Must be defined as an integer, numeric or decimal data type.
- Cannot update a column with the `IDENTITY` property.
- Cannot contain `NULL` values.
- Cannot bind defaults and default constraints to the column.

For `IDENTITY[(seed, increment)]`

- **Seed** – the initial value of the identity column
- **Increment** – the value to add to the last increment column

---

The `UNIQUE` constraint prevents duplicate values from being entered into a column.

- Both PK and `UNIQUE` constraints are used to enforce entity integrity.
- Multiple `UNIQUE` constraints can be defined for a table.
- When a `UNIQUE` constraint is added to an existing table, the existing data is always validated.
- A `UNIQUE` constraint can be placed on columns that accept nulls. Only one row can be `NULL`.
- A `UNIQUE` constraint automatically creates a unique index on the selected column.

This is an examle using the `UNIQUE` constraint.

```
CREATE TABLE employee
(
id BIGINT NOT NULL UNIQUE
name VARCHAR(20) NOT NULL
);
```

The `CHECK` constraint restricts values that can be entered into a table.

- It can contain search conditions similar to a `WHERE` clause.
- It can reference columns in the same table.
- The data validation rule for a `CHECK` constraint must evaluate to a boolean expression.
- It can be defined for a column that has a rule bound to it.

This is the general syntax for the `CHECK` constraint:

```
[CONSTRAINT constraint_name]
CHECK [NOT FOR REPLICATION] (expression)
```

```
CREATE TABLE departments
(
    id BIGINT NOT NULL,
    name VARCHAR(20) NULL,
    dept_name VARCHAR(20) NULL,
    seniority VARCHAR(20) NULL,
    graduation VARCHAR(20) NULL,
    salary BIGINT NULL,
    hire_date DATE NULL,
        CONSTRAINT pk_1 PRIMARY KEY (id),
        CHECK (salary BETWEEN 40000 AND 100000)
 ) ;
```

The `DEFAULT` constraint is used to supply a value that is automatically added for a column if the user does not supply one.

- A column can have only one `DEFAULT`.
- The `DEFAULT` constraint cannot be used on columns with a timestamp data type or identity property.
- `DEFAULT` constraints are automatically bound to a column when they are created.

The general syntax for the `DEFAULT` constraint is:

```
[CONSTRAINT constraint_name]
DEFAULT {constant_expression | niladic-function | NULL}
[FOR col_name]

```

Here is an example:

```
ALTER TABLE departments
ADD CONSTRAINT def_dept_name DEFAULT 'HR' FOR dept_name;

```

Previous

Next