SQL S8 Creating Databases and Tables

My Course Notes and Code

SQL Data Types

- **Boolean**
    - True or False
- **Character**
    - char, varchar, text
- **Numeric**
    - integer, floating-point
- **Temporal**
    - Date, time, timestamp, interval
- **UUID**
- **Array**
- **JSON**
- **Hstore** key-value pairs
- Network addresses (**URL**), geometric data
- **SERIAL**
    - Sequence of integers, often used as the primary key in a teble
    - Worth noting: If a row is removed, the collumn will not adjust - e.g., 1, 2, 3, 6, 7, 8...
    - **Subtypes**:
        - smallserial
        - serial
        - bigserial

https://www.postgresql.org/docs/current/datatype.html

Primary Key

A column *or* group of columns used to *uniquelly* identify each row in the table

- **Unique** = distinct for every row
- **Non null** = there must be an entry

Foreign Key

A column or group of columns used to uniquelly identify rows in another table

Foreing keys are defined in **child tables**, which reference *primary keys* of **parent tables**

- **Child table** = referencing table

- **Parent table** = referenced table

A table can have multiple foreign keys.

Constraints

Constraints are rules imposed on data columns on table. They prevent entering invalid data into the database. They enable *accuracy*, and *reliability* of the data in the database.

- They can are used to define a column as being primary key, or attaching a foreign key relationship to another table

1. Column Constraints
2. Table Constraints

Most commonly used Column constraints:

- NOT NULL
- UNIQUE
- PRIMARY KEY
- FOREIGN KEY
- CHECK - ensures that all values in a column satisfy certain conditions
- EXCLUSION - ... not all comparisons return TRUE
- REFERENCES table(col)

Most commonly used Table constraints:

- CHECK (condition)
- REFERENCES - the values must exist in another column
- UNIQUE (column_list)
- PRIMARY KEY (column_list)

Creating a Table

Full General Syntax:

```
CREATE TABLE table_name(
        column_name TYPE column_constraint,
        column_name TYPE column_constrain,
        table_constraint table_constraint
) INHERITS existing_table_name;
```

Example Simple Syntax:

```
CREATE TABLE players(
        player_id SERIAL PRIMARY KEY,
        age SMALLINT NOT NULL
);
```

INSERT

```
INSERT INTO table(coulmn1, column2, ...)
VALUES
        (value1, value2, ...),
        (value1, value2, ...),
        ...;
```

INSERT values from another table:

```
INSERT INTO table(coulmn1, column2, ...)
SELECT column1, column2, ...
FROM another_table
WHERE condition;
```

- Inserted rows need to match, including constraints
- SERIAL columns do not need to be provided a value

UPDATE

One can also:

- Update everything without WHERE condition
- Update based on another column
- Update using another table's values ('UPDATE join'):

  ```
  UPDATE tableA
  SET original_col = TableB.new_col
  FROM tableB
  WHERE tableA.id = TableB.id
  ```

- Return affected rows across multiple columns:

  ```
  UPDATE ...
  SET ...
  WHERE ...
  RETURNING col1, col2;
  ```

DELETE

DELETE clause removes rows from a table.

```
DELETE FROM table
WHERE row_id = 1;
```

Rows can be deleted based on their presence in other tables:

```
DELETE FROM tableA
USING tableB
WHERE tableA.id = tableB.id;
```

Deleting all rows from a table:

```
DELETE FROM table;
```

A `RETURNING` call can also be added to the command, basically to return rows which were deleted.

ALTER

- Adding, dropping, renaming columns
- Changing columns' data types
- Setting `DEFAULT` values for a column
- Adding `CHECK` constraints
- Renaming table

General Syntax:

```
ALTER TABLE table_name
action;
```

Adding columns:

```
ALTER TABLE table_name
ADD COLUMN new_col TYPE;
```

Removing columns:

```
ALTER TABLE table_name
DROP COLUMN col_name;
```

Altering constraints:

```
ALTER TABLE table_name
ALTER COLUMN col_name
SET DEFAULT value;
-- DROP DEFAULT
-- SET NOT NULL
-- DROP NOT NULL
-- ADD CONSTRAINT constraint_name
```

TIP: Consult PostgreSQL documentation for learning more about the `ALTER` clause.

DROP

Complete removal of a column in a table.

- In PostgreSQL it also automatically removes all of its indexes and constraints.
- Views, triggers, stored procedures (dependencies associated a column) - the `DROP` clause won't remove the columns used here.
    - Unless we use the additional `CASCADE` clause

General Syntax:

```
ALTER TABLE table_name
DROP COLUMN col_name;
```

Remove all dependencies:

```
ALTER TABLE table_name
DROP COLUMN col_name CASCADE;
```

Checking for whether the column exists, to avoid error:

```
ALTER TABLE table_name
DROP COLUMN IF EXISTS col_name
```

Drop multiple columns:

```
ALTER TABLE table_name
DROP COLUMN col_one,
DROP COLUMN col_two;
```

CHECK constraint

Allowes us to create more customised constraints.

- E.G., all integers inserted in a column should fall below a certain treshold.

General Syntax:

```
CREATE TABLE example(
	ex_id SERIAL PRIMARY KEY,
	age SMALLINT CHECK(age > 21),
	parent_age SMALLINT CHECK(parent_age > age)
);
```

CODE - The entire course segment

```
-- CREATE TABLE ------------------------------------------------------------------------

CREATE TABLE account(
	user_id SERIAL PRIMARY KEY,
	username VARCHAR(50) UNIQUE NOT NULL,
	password VARCHAR(50) NOT NULL,
	email VARCHAR(250) UNIQUE NOT NULL,
	created_on TIMESTAMP NOT NULL,
	last_login TIMESTAMP
);

CREATE TABLE job(
	job_id SERIAL PRIMARY KEY,
	job_name VARCHAR(200) UNIQUE NOT NULL
);

CREATE TABLE account_job(
	user_id INTEGER REFERENCES account(user_id),
	job_id INTEGER REFERENCES job(job_id),
	hire_date TIMESTAMP
);

-- INSERT clause -----------------------------------------------------------------------

SELECT *
FROM account;

INSERT INTO account(username, password, email, created_on)
VALUES
	('Vuk', 'password', 'vuk@mail.com', CURRENT_TIMESTAMP);

SELECT *
```

```sql
FROM account;

INSERT INTO job(job_name)
VALUES
        ('Data Analyst'),
        ('President');

INSERT INTO account_job(user_id, job_id, hire_date)
VALUES
        (1, 1, CURRENT_TIMESTAMP);

SELECT *
FROM account_job;

-- UPDATE clause --------------------------------------------------------------------------

UPDATE account
SET last_login = CURRENT_TIMESTAMP
RETURNING last_login;

UPDATE account
SET last_login = created_on
RETURNING email, created_on, last_login;

SELECT *
FROM account;

UPDATE account_job
SET hire_date = account.created_on
FROM account
WHERE account_job.user_id = account.user_id;

SELECT *
FROM account_job;

-- DELETE clause --------------------------------------------------------------------------

INSERT INTO job(job_name)
VALUES
        ('Cowboy');

SELECT * FROM job;

DELETE FROM job
WHERE job_name = 'Cowboy'
RETURNING job_id, job_name;

-- ALTER clause ---------------------------------------------------------------------------

CREATE TABLE information(
        info_id SERIAL PRIMARY KEY,
        title VARCHAR(500) NOT NULL,
        person VARCHAR(50) NOT NULL UNIQUE
);
```

```sql
SELECT *
FROM information;

ALTER TABLE information
RENAME TO new_info; -- renaming table

ALTER TABLE new_info
RENAME COLUMN person TO people; -- renaming a column

SELECT *
FROM new_info;

ALTER TABLE new_info -- removing a constraint
ALTER COLUMN people DROP NOT NULL; -- alternatively: SET

INSERT INTO new_info(title)
VALUES ('new information');

SELECT *
FROM new_info;

-- DROP clause -----------------------------------------------------------------------------

ALTER TABLE new_info
DROP COLUMN people;

SELECT *
FROM new_info;

ALTER TABLE new_info
DROP COLUMN IF EXISTS people;

-- CHECK constraint ------------------------------------------------------------------------

CREATE TABLE employees(
        emp_id SERIAL PRIMARY KEY,
        first_name VARCHAR(50) NOT NULL,
        last_name VARCHAR(50) NOT NULL,
        birth_date DATE CHECK(birth_date > '1990-01-01'),
        hire_date DATE CHECK(hire_date > birth_date),
        salary INTEGER CHECK (salary > 0)
);

INSERT INTO employees(
        first_name,
        last_name,
        birth_date,
        hire_date,
        salary
)
VALUES(
        'Vukašin',
        'Višković',
        '1996-12-06',
        '2022-05-05',
        100
),
        ('Mike',
         'Oldfield',
         '1980-01-01',
         '2022-01-01',
         100
);
```