# PostgreSQL

**Fanavaran Anisa**

**Iran Linux House**

Linux & Open Source Training Center

www.anisa.co.ir

## Section 2 : DDL & PSQL Part 2

## Sequence

```
CREATE SEQUENCE [ IF NOT EXISTS ] sequence_name
    [ AS { SMALLINT | INT | BIGINT } ]
    [ INCREMENT [ BY ] increment ]
    [ MINVALUE minvalue | NO MINVALUE ]
    [ MAXVALUE maxvalue | NO MAXVALUE ]
    [ START [ WITH ] start ]
    [ CACHE cache ]
    [ [ NO ] CYCLE ]
    [ OWNED BY { table_name.column_name | NONE } ]
```

```
CREATE SEQUENCE mysequence
INCREMENT 5 START 100;
```

SELECT nextval('mysequence');

```
CREATE SEQUENCE three
INCREMENT -1
MINVALUE 1
MAXVALUE 3
START 3
CYCLE;
```

## Sequence – A Challenging Sample

```
CREATE TABLE order_details(
    order_id SERIAL,     item_id INT NOT NULL,
    item_text VARCHAR NOT NULL,
    price DEC(10,2) NOT NULL,
    PRIMARY KEY(order_id, item_id)
);
```

```
Tip: PostgreSQL implicitly creates a sequence named
<table_name>_<column_name>_seq. In this case, it would be
order_details_order_id_seq.
```

```
CREATE SEQUENCE order_item_id
START 10
INCREMENT 10
MINVALUE 10
OWNED BY order_details.item_id;

INSERT INTO  order_details(order_id, item_id, item_text, price)
VALUES
    (100, nextval('order_item_id'),'DVD Player',100),
    (100, nextval('order_item_id'),'Android TV',550),
    (100, nextval('order_item_id'),'Speaker',250);
```

## Sequence – A Challenging Sample -> Simplified!

```
CREATE SEQUENCE order_id_seq
START 10
INCREMENT 10
MINVALUE 10
OWNED BY order_details.order_id;


CREATE TABLE order_details (
    order_id INT DEFAULT nextval('order_id_seq')
PRIMARY KEY,
    item_id INT NOT NULL,
    item_text VARCHAR NOT NULL,
    price DEC(10,2) NOT NULL
);
```

## Serial

```
CREATE TABLE table_name(
    id SERIAL
);
```

is equivalent to the following statements:

```
CREATE SEQUENCE table_name_id_seq;

CREATE TABLE table_name (
    id integer NOT NULL DEFAULT nextval('table_name_id_seq')
);

ALTER SEQUENCE table_name_id_seq
OWNED BY table_name.id;
```

## PostgreSQL Identity Column

```
column_name type GENERATED { ALWAYS | BY DEFAULT } AS
IDENTITY[ ( sequence_option ) ]
```

- The type can be SMALLINT, INT, or BIGINT.
- The GENERATED ALWAYS instructs PostgreSQL to always generate a value for the identity column. If you attempt to insert (or update) values into the GENERATED ALWAYS AS IDENTITY column, PostgreSQL will issue an error.
- The GENERATED BY DEFAULT also instructs PostgreSQL to generate a value for the identity column. However, if you supply a value for insert or update, PostgreSQL will use that value to insert into the identity column instead of using the system-generated value.

## PostgreSQL Identity Column

```
CREATE TABLE color (
    color_id INT GENERATED ALWAYS AS IDENTITY,
    color_name VARCHAR NOT NULL
);

INSERT INTO color(color_name)
VALUES ('Red');


INSERT INTO color (color_id, color_name)
VALUES (2, 'Green');

[Err] ERROR:  cannot insert into column "color_id"
DETAIL:  Column "color_id" is an identity column defined
as GENERATED ALWAYS.
HINT:  Use OVERRIDING SYSTEM VALUE to override.
```

## PostgreSQL Identity Column

```
CREATE TABLE color (
    color_id INT GENERATED BY DEFAULT AS IDENTITY
    (START WITH 10 INCREMENT BY 10),
    color_name VARCHAR NOT NULL
);
```

### Adding an identity column to an existing table

```
ALTER TABLE table_name ALTER COLUMN column_name
ADD GENERATED { ALWAYS | BY DEFAULT } AS IDENTITY { (
sequence_option ) }
```

## Alter Table

The following illustrates the basic syntax of the ALTER TABLE statement:

**ALTER TABLE table_name action;**

PostgreSQL provides you with many actions:

- Add a column
- Drop a column
- Change the data type of a column
- Rename a column
- Set a default value for the column.
- Add a constraint to a column.
- Rename a table

## Alter Table

The following illustrates the basic syntax of the ALTER TABLE statement:

**ALTER TABLE table_name action;**

PostgreSQL provides you with many actions:

- Add a column
- Drop a column
- Change the data type of a column
- Rename a column
- Set a default value for the column.
- Add a constraint to a column.
- Rename a table

## Alter Table

```
CREATE TABLE assets (
    id serial PRIMARY KEY,
    name TEXT NOT NULL,
    asset_no VARCHAR NOT NULL,
    description TEXT,
    location TEXT,
    acquired_date DATE NOT NULL
);

ALTER TABLE assets
    ALTER COLUMN location TYPE VARCHAR,
    ALTER COLUMN description TYPE VARCHAR;


ALTER TABLE assets
ALTER COLUMN asset_no TYPE INT
USING asset_no::integer;
```

## Alter Table

```
ALTER TABLE customers
ADD COLUMN fax VARCHAR,
ADD COLUMN email VARCHAR



ALTER TABLE customers
ADD COLUMN contact_name VARCHAR NOT NULL
###Error

ALTER TABLE customers
ADD COLUMN contact_name VARCHAR;

ALTER TABLE customers
ALTER COLUMN contact_name SET NOT NULL;
```

## Alter Table – Drop Column

```
ALTER TABLE table_name
DROP COLUMN column_name CASCADE;


ALTER TABLE table_name
DROP COLUMN IF EXISTS column_name

ALTER TABLE table_name
DROP COLUMN column_name1,
DROP COLUMN column_name2,
...;

ALTER TABLE books
DROP COLUMN publisher_id CASCADE;
```

## Alter Table – Rename Column/Table

```
ALTER TABLE table_name
RENAME column_name1 TO new_column_name1;

ALTER TABLE table_name
RENAME column_name2 TO new_column_name2;

ALTER TABLE customers
RENAME COLUMN email TO contact_email;


ALTER TABLE vendors RENAME TO suppliers;

ALTER TABLE suppliers
ADD COLUMN group_id INT NOT NULL;

ALTER TABLE suppliers
ADD FOREIGN KEY (group_id) REFERENCES
supplier_groups (id);
```

## Alter Table – Command List

| Command | Description | Sample Usage |
|---|---|---|
| **ALTER TABLE table_name ADD COLUMN** | Adds a new column to an existing table. | ALTER TABLE employees ADD COLUMN birthdate DATE; |
| **ALTER TABLE table_name DROP COLUMN** | Removes a column from an existing table. | ALTER TABLE employees DROP COLUMN salary; |
| **ALTER TABLE table_name ALTER COLUMN** | Modifies the data type or other properties of a column. | ALTER TABLE employees ALTER COLUMN name TYPE VARCHAR(100); |
| **ALTER TABLE table_name RENAME COLUMN** | Renames a column in an existing table. | ALTER TABLE employees RENAME COLUMN name TO full_name; |
| **ALTER TABLE table_name ADD CONSTRAINT** | Adds a constraint to a table. | ALTER TABLE employees ADD CONSTRAINT salary_check CHECK (salary > 0); |
| **ALTER TABLE table_name DROP CONSTRAINT** | Removes a constraint from a table. | ALTER TABLE employees DROP CONSTRAINT salary_check; |

## Alter Table – Command List

| | | |
|---|---|---|
| **ALTER TABLE table_name RENAME TO** | Renames an existing table. | ALTER TABLE old_table_name RENAME TO new_table_name; |
| **ALTER TABLE table_name OWNER TO** | Changes the owner of a table. | ALTER TABLE employees OWNER TO new_owner; |
| **ALTER TABLE table_name SET SCHEMA** | Moves a table to a different schema. | ALTER TABLE employees SET SCHEMA new_schema; |
| **ALTER TABLE table_name ENABLE TRIGGER** | Enables a trigger on a table. | ALTER TABLE employees ENABLE TRIGGER trigger_name; |
| **ALTER TABLE table_name DISABLE TRIGGER** | Disables a trigger on a table. | ALTER TABLE employees DISABLE TRIGGER trigger_name; |
| **ALTER TABLE table_name CLUSTER ON** | Reorders the table based on the specified index. | ALTER TABLE employees CLUSTER ON index_name; |
| **ALTER TABLE table_name SET WITHOUT OIDS** | Disables OID storage for a table. | ALTER TABLE employees SET WITHOUT OIDS; |
| **ALTER TABLE table_name SET WITH OIDS** | Enables OID storage for a table. | ALTER TABLE employees SET WITH OIDS; |
| **ALTER TABLE table_name SET TABLESPACE** | Moves a table to a different tablespace. | ALTER TABLE employees SET TABLESPACE new_tablespace; |
| **ALTER TABLE table_name SET (storage_parameter)** | Sets storage parameters for a table. | ALTER TABLE employees SET (fillfactor = 70); |