Database creation.

DDL

Database creation

- `CREATE DATABASE` — create a new database

Database creation

- `CREATE DATABASE` — create a new database
- To create a database, you must be a superuser or have the special `CREATEDB` privilege.

Database creation

```
CREATE DATABASE name
    [ [ WITH ] [ OWNER [=] user_name ]
           [ TEMPLATE [=] template ]
           [ ENCODING [=] encoding ]
           [ LC_COLLATE [=] lc_collate ]
           [ LC_CTYPE [=] lc_ctype ]
           [ TABLESPACE [=] tablespace_name ]
           [ ALLOW_CONNECTIONS [=] allowconn ]
           [ CONNECTION LIMIT [=] connlimit ]
           [ IS_TEMPLATE [=] istemplate ] ]
```

Parameters

- *name* - The name of a database to create.

Parameters

- *name* - The name of a database to create.
- *user_name* - The role name of the user who will own the new database, or DEFAULT to use the default

Parameters

- *name* - The name of a database to create.
- *user_name* - The role name of the user who will own the new database, or DEFAULT to use the default
- *template* - The name of the template from which to create the new database, or DEFAULT to use the default

Parameters

- *encoding* - Character set encoding to use in the new database

Parameters

- *encoding* - Character set encoding to use in the new database
- *lc_collate - Collation order to use in the new database*

Parameters

- *encoding* - Character set encoding to use in the new database
- *lc_collate - Collation order to use in the new database*
- *lc_ctype - Character classification to use in the new database*

Parameters

- *tablespace_name* - The name of the tablespace that will be associated with the new database

Parameters

- *tablespace_name* - The name of the tablespace that will be associated with the new database

- *allow_conn* - If false then no one can connect to this database

Parameters

- *tablespace_name* - The name of the tablespace that will be associated with the new database

- *allow_conn* - If false then no one can connect to this database

- *conn_limit* - How many concurrent connections can be made to this database. -1 (the default) means no limit.

Parameters

- *tablespace_name* - The name of the tablespace that will be associated with the new database

- *allow_conn* - If false then no one can connect to this database

- *conn_limit* - How many concurrent connections can be made to this database. -1 (the default) means no limit.

- *istemplate* - If true, then this database can be cloned by any user with CREATEDB privileges; if false (the default), then only superusers or the owner of the database can clone it.

Examples

```
CREATE DATABASE lusiadas;
```

```
CREATE DATABASE sales
    OWNER salesapp
    TABLESPACE salesspace;
```

Examples

```
CREATE DATABASE music
    LC_COLLATE 'sv_SE.utf8'
    LC_CTYPE 'sv_SE.utf8'
    TEMPLATE template0;
```

```
CREATE DATABASE music2
    LC_COLLATE 'sv_SE.iso885915'
    LC_CTYPE 'sv_SE.iso885915'
    ENCODING LATIN9
    TEMPLATE template0;
```

TABLESPACE

```
CREATE TABLESPACE tablespace_name
    [ OWNER { new_owner | CURRENT_USER | SESSION_USER } ]
LOCATION 'directory'
[ WITH ( tablespace_option = value [, ... ] ) ]
```

Examples

```
CREATE TABLESPACE fastspace
    LOCATION '/ssd1/postgresql/data';
```

```
CREATE TABLESPACE indexspace
    OWNER genevieve
    LOCATION '/data/indexes';
```

Deleting database

- DROP DATABASE -- remove a database

Deleting database

- DROP DATABASE -- remove a database
- It removes the catalog entries for the database and deletes the directory containing the data.

Deleting database

- DROP DATABASE -- remove a database
- It removes the catalog entries for the database and deletes the directory containing the data.
- It can only be executed by the database owner.

Deleting database

```
DROP DATABASE [ IF EXISTS ] name
```

DDL

- Data Definition Language - is a standard for commands that define the different structures in a database.

DDL

- Data Definition Language - is a standard for commands that define the different structures in a database.
- DDL statements create, modify, and remove database objects such as tables, indexes, and users.

DDL

- Data Definition Language - is a standard for commands that define the different structures in a database.

- DDL statements create, modify, and remove database objects such as tables, indexes, and users.

- Common DDL statements are `CREATE`, `ALTER`, and `DROP`.

Table Basics

- It consists of rows and columns

Table Basics

- It consists of rows and columns
- The number and order of the columns is fixed, and each column has a name

Table Basics

- It consists of rows and columns
- The number and order of the columns is fixed, and each column has a name
- The number of rows is variable

Table Basics

- It consists of rows and columns
- The number and order of the columns is fixed, and each column has a name
- The number of rows is variable
- Each column has a data type

Table creation

- CREATE TABLE — define a new table

Table creation

- CREATE TABLE — define a new table
- CREATE TABLE will create a new, initially empty table in the current database.

Table creation

- CREATE TABLE — define a new table
- CREATE TABLE will create a new, initially empty table in the current database.
- The table will be owned by the user issuing the command.

Table creation

```
CREATE TABLE [ IF NOT EXISTS ] table_name ( [
  { column_name data_type [ COLLATE collation ] [ column_constraint [ ... ] ]
    | table_constraint
    | LIKE source_table [ like_option ... ] }
    [, ... ]
] )
[ INHERITS ( parent_table [, ... ] ) ]
[ TABLESPACE tablespace_name ]
```

Parameters

- IF NOT EXISTS - Do not throw an error if a relation with the same name already exists.

Parameters

- IF NOT EXISTS - Do not throw an error if a relation with the same name already exists.

- *table_name* - The name (optionally schema-qualified) of the table to be created.

Parameters

- IF NOT EXISTS - Do not throw an error if a relation with the same name already exists.

- *table_name* - The name (optionally schema-qualified) of the table to be created.

- *column_name* - The name of a column to be created in the new table.

Parameters

- IF NOT EXISTS - Do not throw an error if a relation with the same name already exists.

- *table_name* - The name (optionally schema-qualified) of the table to be created.

- *column_name* - The name of a column to be created in the new table.

- *data_type* - The data type of the column. This can include array specifiers.

Parameters

- COLLATE *collation - The COLLATE clause assigns a collation to the column (which must be of a collatable data type).*

Parameters

- COLLATE *collation - The COLLATE clause assigns a collation to the column (which must be of a collatable data type).*

- INHERITS (*parent_table* [, ...]) - The optional INHERITS clause specifies a list of tables from which the new table automatically inherits all columns. Parent tables can be plain tables or foreign tables.

Parameters

- COLLATE *collation - The COLLATE clause assigns a collation to the column (which must be of a collatable data type).*

- INHERITS (*parent_table* [, ...]) - The optional INHERITS clause specifies a list of tables from which the new table automatically inherits all columns. Parent tables can be plain tables or foreign tables.

- LIKE *source_table* [*like_option* ...] - The LIKE clause specifies a table from which the new table automatically copies all column names, their data types, and their not-null constraints.

Parameters

- CONSTRAINT *constraint_name* - An optional name for a column or table constraint.

Parameters

- CONSTRAINT *constraint_name* - An optional name for a column or table constraint.

- TABLESPACE *tablespace_name* -The tablespace_name is the name of the tablespace in which the new table is to be created.

Examples

```
CREATE TABLE films (
    code        char(5) CONSTRAINT firstkey PRIMARY KEY,
    title       varchar(40) NOT NULL,
    did         integer NOT NULL,
    date_prod   date,
    kind        varchar(10),
    len         interval hour to minute
);
```

Examples

```
CREATE TABLE films (
    code          char(5),
    title         varchar(40),
    did           integer,
    date_prod     date,
    kind          varchar(10),
    len           interval hour to minute,
    CONSTRAINT production UNIQUE(date_prod)
);
```

Examples

```
CREATE TABLE array_int (
    vector  int[][]
);
```

Examples

```
CREATE TABLE distributors (
    did     integer,
    name    varchar(40),
    PRIMARY KEY(did)
);
```

```
CREATE TABLE distributors (
    did     integer PRIMARY KEY,
    name    varchar(40)
);
```

Default values

```
CREATE TABLE products (
    product_no integer,
    name text,
    price numeric DEFAULT 9.99
);
```

Modifying tables

- ALTER TABLE — change the definition of an existing table

Modifying tables

- ALTER TABLE — change the definition of an existing table

```
ALTER TABLE [ IF EXISTS ] [ ONLY ] name [ * ]
    action [, ... ]
```

Parameters

- ADD COLUMN - adds a new column to the table, using the same syntax as CREATE TABLE

Parameters

- ADD COLUMN - adds a new column to the table, using the same syntax as CREATE TABLE
- DROP COLUMN - drops a column from a table. Indexes and table constraints involving the column will be automatically dropped as well.

Parameters

- ADD COLUMN - adds a new column to the table, using the same syntax as CREATE TABLE

- DROP COLUMN - drops a column from a table. Indexes and table constraints involving the column will be automatically dropped as well.

- SET DATA TYPE/TYPE - changes the type of a column of a table.

Parameters

- SET/DROP DEFAULT - These forms set or remove the default value for a column.

Parameters

- SET/DROP DEFAULT - These forms set or remove the default value for a column.
- SET/DROP NOT NULL - These forms change whether a column is marked to allow null values or to reject null values.

Parameters

- SET/DROP DEFAULT - These forms set or remove the default value for a column.

- SET/DROP NOT NULL - These forms change whether a column is marked to allow null values or to reject null values.

- ADD *table_constraint* - This form adds a new constraint to a table using the same syntax as CREATE TABLE.

Examples

```
ALTER TABLE distributors
    ADD COLUMN address varchar(30);
```

```
ALTER TABLE distributors
    DROP COLUMN address RESTRICT;
```

Examples

```
ALTER TABLE distributors
    ALTER COLUMN address TYPE varchar(80),
    ALTER COLUMN name TYPE varchar(100);
```

```
ALTER TABLE distributors
    ALTER COLUMN address SET DATA TYPE varchar(80),
    ALTER COLUMN name SET DATA TYPE varchar(100);
```

Deleting tables

DROP TABLE — removes tables from the database.

Deleting tables

DROP TABLE — removes tables from the database.

Only the table owner, the schema owner, and superuser can drop a table.

Deleting tables

```
DROP TABLE [ IF EXISTS ] name [, ...]
    [ CASCADE | RESTRICT ]
```

Parameters

- CASCADE - Automatically drop objects that depend on the table (such as views), and in turn all objects that depend on those objects

Parameters

- CASCADE - Automatically drop objects that depend on the table (such as views), and in turn all objects that depend on those objects

- RESTRICT - Refuse to drop the table if any objects depend on it. This is the default.

Examples

```
DROP TABLE films, distributors CASCADE;
```

Data types

- PostgreSQL has a rich set of native data types available to users.
- Users can add new types to PostgreSQL using the CREATE TYPE command.

Data types

- Numeric types
- Character types
- Binary Data types
- Date/Time types
- Boolean types
- Arrays

Integer types

- *smallint* (2 bytes) - small-range integer (-32768 to +32767)

Integer types

- *smallint* (2 bytes) - small-range integer (-32768 to +32767)
- *integer* (4 bytes) - typical choice for integer (-2147483648 to +2147483647)

Integer types

- *smallint* (2 bytes) - small-range integer (-32768 to +32767)
- *integer* (4 bytes) - typical choice for integer (-2147483648 to +2147483647)
- bigint (8 bytes) - large-range integer (-9223372036854775808 to +9223372036854775807)

Floating-Point types

- *real* (4 bytes) - variable-precision, inexact (6 decimal digits precision)

Floating-Point types

- *real* (4 bytes) - variable-precision, inexact (6 decimal digits precision)

- *double* (8 bytes) - variable-precision, inexact (15 decimal digits precision)

Serial types

- *smallserial* (2 bytes) - small autoincrementing integer (1 to 32767)

Serial types

- *smallserial* (2 bytes) - small autoincrementing integer (1 to 32767)
- *serial* (4 bytes) - autoincrementing integer (1 to 2147483647)

Character Types

- *varchar(n)* - variable-length string with limit

Character Types

- *varchar(n)* - variable-length string with limit
- *char(n)* - fixed-length string, blank padded

Character Types

- *varchar(n)* - variable-length with limit
- *char(n)* - fixed-length, blank padded
- text - variable unlimited length

Binary Data Types

- *bytea* - variable-length binary string (1 or 4 bytes)

Date/Time Types

- *timestamp [(p)] [without time zone]* (8 bytes)- both date and time (no time zone)

Date/Time Types

- *timestamp [(p)] [without time zone]* (8 bytes)- both date and time (no time zone)
- *timestamp [(p)] with time zone* (8 bytes) - both date and time, with time zone

Date/Time Types

- *timestamp [(p)] [without time zone]* (8 bytes)- both date and time (no time zone)

- *timestamp [(p)] with time zone* (8 bytes) - both date and time, with time zone

- *date* (4 bytes)- date (no time of day)

Date/Time Types

- *time [(p)] [without time zone]* (8 bytes)- time of day (no date)

Date/Time Types

- *time [(p)] [without time zone]* (8 bytes)- time of day (no date)
- *time [(p)] with time zone* (12 bytes) - time of day (no date), with time zone

Date/Time Types

- *time [(p)] [without time zone]* (8 bytes)- time of day (no date)

- *time [(p)] with time zone* (12 bytes) - time of day (no date), with time zone

- *interval [fields] [(p)]* (16 bytes)- time interval

Boolean Types

- *boolean* (1 byte)- state of true or false

Questions?