Lecture 2 - Tables / Insert / Select

Feb 20, 2020

SQL databases are composed of a set of tables.

A table is a set of columns, with names and types - like a struct or record in other languages.

There is a set of rows - that act kind of like an un-ordered array of columns.

SQL normally stores this as a row-order store.

Some SQL-ish databases like Cassandra store this as a colum-store.

```
$ psql
pg=# \i create-tables.sql
pg=# \q
$
```

the file

```
-- \c l02
create table vote_by_county (
    id                serial primary key,
    year              int default 2021,
    state             text default '--',        -- irritatingly all upper case.
    state_uc          text default '--',
    state_po          varchar(2) default '--',     -- Incorrectly Named Column!
    county_name       text default '--',        -- irritatingly all upper case.
    county_name_uc    text default '--',
    county_fips       int default 0,
    office            text default 'unk',
    candidate         text default 'unk',
    candidate_uc      text default 'unk',
    party             text default 'unk',
    candidatevotes    int default 0,
    totalvotes        int default 0,
    version           int,
    vote_mode         text
);
```

Output

```
CREATE TABLE
```

Let's just insert a few rows to see how insert works:

```
-- \c l02

insert into vote_by_county  ( year, state, county_name, version ) values
    ( 2022, 'Wyoming', 'Albeny',   1 );
insert into vote_by_county  ( year, state, county_name, version ) values
    ( 2022, 'Wyoming', 'Big Horn', 2 );
insert into vote_by_county  ( year, state, county_name, version ) values
    ( 2022, 'Wyoming', 'Carbon',   8 );
```

Output

```
INSERT 0 1
INSERT 0 1
INSERT 0 1
```

Now we can get the data back:

```
select year, state, county_name, version
        from vote_by_county ;
```

Data Types for tables

The data types used in our example:

Type	Description
serial	an integer that is generated (if null) by the database.
int	an integer (bigint, numeric etc)
text	a string from NULL, 0 length to `2**33` bytes in length.
varchar(maxlen)	A 'text' field with a limited length.
char(maxlen)	a right blank-padded fixed length string.
char varying(maxlen)	same as varchar()

`not null` disallow NULL values.

`default Value` if insert skips this value then default will be used. This is different than the value NULL.

`primary key` means that .

There are a bunch of other data types in PostgreSQL (MySQL/MariaDB has a bunch too, Oracle has a bunch, MS Sql Server too).

Name	Aliases	Description
`bigint`	`int8`	signed eight-byte integer
`bigserial`	`serial8`	autoincrementing eight-byte integer

Name	Aliases	Description
`bit [ (n) ]`		fixed-length bit string
`bit varying [ (n) ]`	`varbit [ (n) ]`	variable-length bit string
`boolean`	`bool`	logical Boolean (true/false)
`box`		rectangular box on a plane
`bytea`		binary data (“byte array”)
`character [ (n) ]`	`char [ (n) ]`	fixed-length character string
`character varying [ (n) ]`	`varchar [ (n) ]`	variable-length character string
`cidr`		IPv4 or IPv6 network address
`circle`		circle on a plane
`date`		calendar date (year, month, day)
`double precision`	`float8`	double precision floating-point number (8 bytes)
`inet`		IPv4 or IPv6 host address
`integer`	`int`, `int4`	signed four-byte integer
`interval [ fields ] [ (p) ]`		time span
`json`		textual JSON data
`jsonb`		binary JSON data, decomposed
`line`		infinite line on a plane
`lseg`		line segment on a plane
`macaddr`		MAC (Media Access Control) address
`money`		currency amount
`numeric [ (p, s) ]`	`decimal [ (p, s) ]`	exact numeric of selectable precision
`path`		geometric path on a plane
`pg_lsn`		PostgreSQL Log Sequence Number
`point`		geometric point on a plane
`polygon`		closed geometric path on a plane
`real`	`float4`	single precision floating-point number (4 bytes)
`smallint`	`int2`	signed two-byte integer
`smallserial`	`serial2`	autoincrementing two-byte integer
`serial`	`serial4`	autoincrementing four-byte integer
`text`		variable-length character string
`time [ (p) ] [ without time zone ]`		time of day (no time zone)
`time [ (p) ] with time zone`	`timetz`	time of day, including time zone

Name	Aliases	Description
`timestamp [ (p) ] [ without time zone ]`		date and time (no time zone)
`timestamp [ (p) ] with time zone`	`timestamptz`	date and time, including time zone
`tsquery`		text search query
`tsvector`		text search document
`txid_snapshot`		user-level transaction ID snapshot
`uuid`		universally unique identifier
`xml`		XML data