>>

Catalogs

- Database Objects
- Catalogs
- Representing Databases
- Representing Tables

^ >>

❖ Database Objects

RDBMSs manage different kinds of objects

- databases, schemas, tablespaces
- relations/tables, attributes, tuples/records
- constraints, assertions
- views, stored procedures, triggers, rules

Many objects have names (and, in PostgreSQL, some have OIDs).

How are the different types of objects represented?

How do we go from a name (or OID) to bytes stored on disk?

<< ^ >>

❖ Catalogs

Consider what information the RDBMS needs about relations:

- name, owner, primary key of each relation
- name, data type, constraints for each attribute
- authorisation for operations on each relation

Similarly for other DBMS objects (e.g. views, functions, triggers, ...)

This information is stored in the system catalog tables

Standard for catalogs in SQL:2003:
`INFORMATION_SCHEMA`

<< ^ >>

❖ Catalogs (cont)

The catalog is affected by several types of SQL operations:

- **`create`** *Object* **`as`** *Definition*
- **`drop`** *Object* ...
- **`alter`** *Object Changes*
- **`grant`** *Privilege* **`on`** *Object*

where *Object* is one of table, view, function, trigger, schema, ...

E.g. **`drop table Groups;`** produces something like

```
delete from Tables
where  schema = 'public' and name = 'groups';
```

<< ^ >>

❖ Catalogs (cont)

In PostgreSQL, the system catalog is available to users via:

- special commands in the **`psql`** shell (e.g. **`\d`**)
- SQL standard **`information_schema`**

  e.g. **`select * from information_schema.tables;`**

The low-level representation is available to sysadmins via:

- a global schema called **`pg_catalog`**
- a set of tables/views in that schema (e.g. **`pg_tables`**)

<< ^ >>

❖ Catalogs (cont)

You can explore the PostgreSQl catalog via **`psql`** commands

- **`\d`** gives a list of all tables and views
- **`\d Table`** gives a schema for ***`Table`***
- **`\df`** gives a list of user-defined functions
- **`\df+ Function`** gives details of ***`Function`***
- **`\ef Function`** allows you to edit ***`Function`***
- **`\dv`** gives a list of user-defined views
- **`\d+ View`** gives definition of ***`View`***

You can also explore via SQL on the catalog tables

<< ^ >>

❖ Catalogs (cont)

A PostgreSQL installation (cluster) typically has many DBs

Some catalog information is global, e.g.

- catalog tables defining: databases, users, ...
- one copy of each such table for the whole PostgreSQL installation
- shared by all databases in the cluster (in **`PGDATA/pg_global`**)

Other catalog information is local to each database, e.g

- schemas, tables, attributes, functions, types, ...
- separate copy of each "local" table in each database
- a copy of many "global" tables is made on database creation

<< ^ >>

❖ Catalogs (cont)

Side-note: PostgreSQL tuples contain

- owner-specified attributes (from **`create table`**)
- system-defined attributes

`oid`	unique identifying number for tuple (optional)
`tableoid`	which table this tuple belongs to
`xmin/xmax`	which transaction created/deleted tuple (for MVCC)

OIDs are used as primary keys in many of the catalog tables.

<< ^ >>

❖ Representing Databases

Above the level of individual DB schemata, we have:

- databases ... represented by **`pg_database`**
- schemas ... represented by **`pg_namespace`**
- table spaces ... represented by **`pg_tablespace`**

These tables are global to each PostgreSQL cluster.

Keys are names (strings) and must be unique within cluster.

<< ^ >>

❖ Representing Databases (cont)

pg_database contains information about databases:

- **oid, datname, datdba, datacl[], encoding, ...**

pg_namespace contains information about schemata:

- **oid, nspname, nspowner, nspacl[]**

pg_tablespace contains information about tablespaces:

- **oid, spcname, spcowner, spcacl[]**

PostgreSQL represents access via array of access items:

```
Role=Privileges/Grantor
```

where *Privileges* is a string enumerating privileges, e.g.

```
jas=arwdRxt/jas,fred=r/jas,joe=rwad/jas
```

<< ^ >>

❖ Representing Tables

Representing one table needs tuples in several catalog tables.

Due to O-O heritage, base table for tables is called **`pg_class`**.

The **`pg_class`** table also handles other "table-like" objects:

- views ... represents attributes/domains of view
- composite (tuple) types ... from **`CREATE TYPE AS`**
- sequences, indexes (top-level defn), other "special" objects

All tuples in **`pg_class`** have an OID, used as primary key.

Some fields from the **`pg_class`** table:

- **`oid, relname, relnamespace, reltype, relowner`**
- **`relkind, reltuples, relnatts, relhaspkey, relacl, ...`**

<< ^

❖ Representing Tables (cont)

Details of catalog tables representing database tables

pg_class holds core information about tables

- **relname, relnamespace, reltype, relowner, ...**
- **relkind, relnatts, relhaspkey, relacl[], ...**

pg_attribute contains information about attributes

- **attrelid, attname, atttypid, attnum, ...**

pg_type contains information about types

- **typname, typnamespace, typowner, typlen, ...**
- **typtype, typrelid, typinput, typoutput, ...**

Produced: 15 Feb 2021