One very good example of view application is the set of tables that contain information about the objects in the database, collectively known as the

**Data Dictionary**

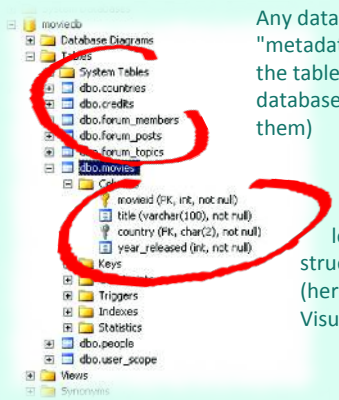
or sometime called the

**Catalog**

They are using all the features we have seen (you only have privileges to read views and only see what is relevant to your account)

## One catalog per database

You always have ONE catalog per database. A database is an independent unit and you can have foreign keys only within (inside) one database; however, you can have several schemas in a database, and you can reference tables in another schema. There may also be metadata such as user accounts that is shared among databases. Most DBMS products can manage several databases at once; other than SQLite, the exception is MySQL that only has ONE catalog. What MySQL calls a *database* is actually a schema.




Any database stores "metadata" that describes the tables in your database (and not only them)

All client tools use this information to let you browse the structure of your tables (here it's SQL Server, Visual Studio)

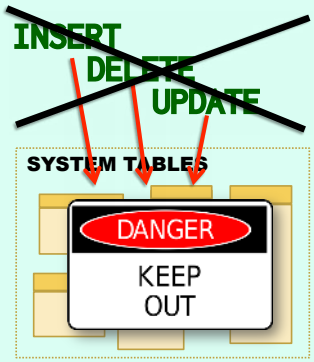
**CREATE**

**DROP**

**ALTER**

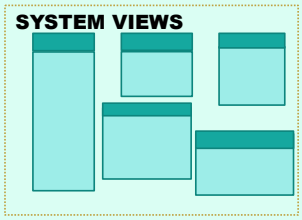
**GRANT**

**REVOKE**




Whenever you are issuing DDL commands, you are actually modifiying system tables. They must NEVER be directly changed.

## SYSTEM VIEWS



SQLite sqlite_master

Read access to these tables is provided through system views.

information_schema

sysibm

+ views in schema sys
+ views in schema syscat
+ pg_... views

## INFORMATION_SCHEMA.TABLES

In these views you only see what YOU are allowed to see. Only administrators see everything.

## SYSCAT.TABLES

## USER_TABLES / ALL_TABLES

Those you have created

Those you can query

*standard*

INFORMATION_SCHEMA

= minimum ...

The "SQL standard" defines a schema for the catalog that several DBMS vendors try to implement (Oracle, so far, doesn't follow it, perhaps because Oracle has no schemas independent from user accounts, and DB2 doesn't call it INFORMATION_SCHEMA). However, you only find minimum information in INFORMATION_SCHEMA. Some products have views to describe triggers, others haven't, for instance. Other than a small common set, many columns may alos be different simply because implementations are different.

```
moviedb=> select table_name,
moviedb->        column_name,
moviedb->        ordinal_position,
moviedb->        data_type
moviedb-> from information_schema.columns
moviedb-> where table_name = 'movies';

 table_name |  column_name  | ordinal_position |     data_type
------------+---------------+------------------+-------------------
 movies     | year_released |                4 | integer
 movies     | country       |                3 | character
 movies     | title         |                2 | character varying
 movies     | movieid       |                1 | integer
(4 rows)
```

There are usually simpler commands to display the structure of a table, but these commands execute nothing more than this type of query. Everything is pulled out of the data dictionary. This MySQL query gives the same result on PostgreSQL (very rare!).

# INFORMATION

As a developer, you can get from the data dictionary some information that is hard to get elsewhere (constraints, for instance).
Database administrators use them a lot for scripting, because the data dictionary always reflects the current state of a database.

## SCRIPTING (DBA)

```
select 'drop table ' ||
       table_name || ';'
from information_schema.tables
where table_name like 'TMP%'
  and ...
```

DBAs often use queries on the catalog to generate other SQL queries; it can be done in a script, or in a procedure with a cursor (a case when cursors are mandatory). They sometimes generate other commands, such as shell script, for instance for backing up database files (the name of which can be found in some remote corners of the catalog).

**Most important tables for developers in INFORMATION_SCHEMA (PostgreSQL version)**

TABLES
COLUMNS
} Also views

TABLE_CONSTRAINTS — Tables with constraint defined

CONSTRAINT_COLUMN_USAGE
KEY_COLUMN_USAGE
} Columns used in a constraint

CHECK_CONSTRAINTS

SEQUENCES ROUTINES TRIGGERS VIEWS

Those are all views you should be aware of.

VIEW_TABLE_USAGE

**Most important tables for developers in INFORMATION_SCHEMA (PostgreSQL version)**

**Nothing** on indexes <u>not</u> associated with PK or UNIQUE constraint!

Must look into **pg_catalog**

**pg_index**
**pg_class**
**pg_attribute**

Most but not all information relevant to a developer can be found in INFORMATION_SCHEMA.
Performance indexes unrelated to constraints are to be found elsewhere, in pg_catalog in PostgreSQL.
pg_catalog is on the wild side, full of numerical identifiers and columns that contain arrays (not very normalized ...)

**Stuff mostly for DB administrators**

Many catalog views are also mostly of interest to database administrators.

*Roles and privileges*

| | |
|---|---|
| `pg_class` | relates tables to files on disk |
| `pg_statistic` | estimates used by the optimizer |
| `pg_settings` | database parameters |

*and so forth...*