

The System Catalog

Every database system must have a meta-database of information on the schemata which it contains. This includes, for each schema, at least the following:

- The names of the relations in the schema.
- The names of the columns of each relation.
- The data type of each column.
- The integrity constraints on the relations.
- Information about indices on the relations.
- The *access privileges* for the elements of the schema.

This database is often called the *system catalog*.

In a relational database system, the catalog itself often consists of relations.

- Figure 17.2 from the 3rd edition of the textbook by Elmasri and Navathe illustrates a basic catalog relation for the Company database of the textbook.
- Figure 17.3 of that same edition shows some alternatives.

Note: Oracle calls the system catalog the *data dictionary*.

Access to the System Catalog

There are a number of distinct ways in which one may access the information contained in the system catalog. Not all systems support all of these modes.

Via SQL:

- The SQL standard requires that an SQL-environment contain an *Information Schema* with the unqualified name `INFORMATION_SCHEMA`. It does not seem that many SQL implementations follow this part of the standard very closely, so one should not depend upon it.
- Some vendors provide a proprietary extension to their SQL which provides access to the system catalog.
 - Oracle SQL*Plus.
 - `pg_` relations of PostgreSQL.
 - This approach has the disadvantage that it is vendor specific; it is not portable across implementations of SQL.
 - This approach has the advantage that it allows vendor-specific features (e.g., special data types, object-relational extensions) to be supported.

Via ODBC:

- The ODBC standard provides API calls which permit one to access much of the information in the system catalog.
 - Since compliance to the ODBC standard is much more common than compliance to the SQL standard, this mode may be depended upon.

Via special interfaces:

- Some systems provide access to the system catalog via an interactive interface.
 - Microsoft Access
 - The *Enterprise Manager* of Oracle.

Basic Features of the PostgreSQL System Catalog

- PostgreSQL has a large number of relations which it calls *system catalogs*.
- The following is taken from the PostgreSQL 7.4 documentation.

Table 43-1. System Catalogs

Catalog Name	Purpose
<u>pg_aggregate</u>	aggregate functions
<u>pg_am</u>	index access methods
<u>pg_amop</u>	access method operators
<u>pg_amproc</u>	access method support procedures
<u>pg_attrdef</u>	column default values
<u>pg_attribute</u>	table columns ("attributes")
<u>pg_cast</u>	casts (data type conversions)
<u>pg_class</u>	tables, indexes, sequences ("relations")
<u>pg_constraint</u>	check constraints, unique constraints, primary key constraints, foreign key constraints
<u>pg_conversion</u>	encoding conversion information
<u>pg_database</u>	databases within this database cluster
<u>pg_depend</u>	dependencies between database

Catalog Name	Purpose
	objects
<u>pg_description</u>	descriptions or comments on database objects
<u>pg_group</u>	groups of database users
<u>pg_index</u>	additional index information
<u>pg_inherits</u>	table inheritance hierarchy
<u>pg_language</u>	languages for writing functions
<u>pg_largeobject</u>	large objects
<u>pg_listener</u>	asynchronous notification support
<u>pg_namespace</u>	schemas
<u>pg_opclass</u>	index access method operator classes
<u>pg_operator</u>	operators
<u>pg_proc</u>	functions and procedures
<u>pg_rewrite</u>	query rewrite rules
<u>pg_shadow</u>	database users
<u>pg_statistic</u>	planner statistics
<u>pg_trigger</u>	triggers
<u>pg_type</u>	data types

- These relations can be viewed in the usual way from the SQL interface.

```
test=> \d pg_database
```

```
Table "pg_catalog.pg_database"
```

Column	Type	Modifiers
datname	name	not null
datdba	integer	not null
encoding	integer	not null
datistemplate	boolean	not null
datallowconn	boolean	not null
datlastsysoid	oid	not null
datvacuumxid	xid	not null
datfrozenxid	xid	not null
datpath	text	not null
datconfig	text[]	
datacl	aclitem[]	

```
Indexes:
```

```
"pg_database_datname_index" unique,  
btree (datname)
```

```
"pg_database_oid_index" unique, btree  
(oid)
```

```
test=>
```

Ordinary relations can also be examined in this way.

```
test=> \d
```

```
          List of relations
 Schema | Name      | Type  | Owner
-----+-----+-----+-----
 public | airline   | table | hegner
 public | airport   | table | hegner
 public | flight    | table | hegner
 public | schedule  | table | hegner
 public | ticket    | table | hegner
(5 rows)
```

```
test=>
```

```
test=> \d airline
```

```
          Table "public.airline"
 Column | Type              | Modifiers
-----+-----+-----
 name   | character varying(15) | not null
 website | character varying(25) | not null
Indexes:
    "pkey_airline" primary key, btree (name)
```

- To view the entire system catalog `pg_catalog`:

```
test-> \dS
```

List of relations(code)			
Schema	Name	Type	Owner
pg_catalog	pg_aggregate	table	postgres
pg_catalog	pg_am	table	postgres
pg_catalog	pg_amop	table	postgres
pg_catalog	pg_amproc	table	postgres
pg_catalog	pg_attrdef	table	postgres
pg_catalog	pg_attribute	table	postgres
pg_catalog	pg_cast	table	postgres
pg_catalog	pg_class	table	postgres
pg_catalog	pg_constraint	table	postgres
pg_catalog	pg_conversion	table	postgres
pg_catalog	pg_database	table	postgres
pg_catalog	pg_depend	table	postgres
pg_catalog	pg_description	table	postgres
pg_catalog	pg_group	table	postgres
pg_catalog	pg_index	table	postgres
pg_catalog	pg_indexes	view	postgres
pg_catalog	pg_inherits	table	postgres
pg_catalog	pg_language	table	postgres
pg_catalog	pg_largeobject	table	postgres
pg_catalog	pg_listener	table	postgres
pg_catalog	pg_locks	view	postgres
pg_catalog	pg_namespace	table	postgres
pg_catalog	pg_opclass	table	postgresid,
pg_catalog	pg_operator	table	postgresid,
pg_catalog	pg_proc	table	postgres
pg_catalog	pg_rewrite	table	postgres
pg_catalog	pg_rules	view	postgres
pg_catalog	pg_settings	view	postgres
pg_catalog	pg_shadow	table	postgres
pg_catalog	pg_stat_activity	view	postgres
pg_catalog	pg_stat_all_indexes	view	postgres
pg_catalog	pg_stat_all_tables	view	
pg_catalog	pg_stat_database	view	postgres
pg_catalog	pg_stat_sys_indexes	view	postgres
pg_catalog	pg_stat_sys_tables	view	postgres
pg_catalog	pg_stat_user_indexes	view	postgres
pg_catalog	pg_stat_user_tables	view	postgres
pg_catalog	pg_statio_all_indexes	view	postgres
pg_catalog	pg_statio_all_sequences	view	postgres
pg_catalog	pg_statio_all_tables	view	postgres

pg_catalog	pg_statio_sys_indexes	view	postgres
pg_catalog	pg_statio_sys_sequences	view	postgres
pg_catalog	pg_statio_sys_tables	view	postgres
pg_catalog	pg_statio_user_indexes	view	postgres
pg_catalog	pg_statio_user_sequences	view	postgres
pg_catalog	pg_statio_user_tables	view	postgres
pg_catalog	pg_statistic	table	postgres
pg_catalog	pg_stats	view	postgres
pg_catalog	pg_tables	view	postgres
pg_catalog	pg_trigger	table	postgres
pg_catalog	pg_type	table	postgres
pg_catalog	pg_user	view	postgres
pg_catalog	pg_views	view	postgres
pg_catalog	pg_xactlock	special	postgres

(54 rows)

- These can be viewed in more detail using *qualified names*.

```
test=> \d pg_catalog.pg_tables
```

```
View "pg_catalog.pg_tables"
```

```
Column      | Type      | Modifiers
```

```
-----+-----+-----
```

Column	Type	Modifiers
schemaname	name	
tablename	name	
tableowner	name	
hasindexes	boolean	
hasrules	boolean	
hastriggers	boolean	

```
View definition:
```

```
SELECT n.nspname AS schemaname, c.relname AS
tablename, pg_get_userbyid(c.relowner) AS tableowner,
c.relhasindex AS hasindexes, c.relhasrules AS
hasrules, c.reltriggers > 0 AS hastriggers
FROM pg_class c
LEFT JOIN pg_namespace n ON n.oid = c.relnamespace
WHERE c.relkind = 'r'::"char";
```

```
test=> select * from pg_catalog.pg_tables where
tableowner='hegner';
```

```
schemaname | tablename | tableowner | hasindexes |
hasrules | hastriggers
```

```
-----+-----+-----+-----+-----+-----
```

schemaname	tablename	tableowner	hasindexes	hasrules	hastriggers
public	airline	hegner	t	f	t
public	airport	hegner	t	f	t
public	flight	hegner	t	f	t
public	ticket	hegner	t	f	t
public	schedule	hegner	t	f	t

```
(5 rows)
```

Basic Features of the Oracle SQL*Plus Approach

At the most basic level, Oracle provides the `DESCRIBE` command within its extension of SQL.

DESCRIBE Works_On

might return something like:

Name	Null?	Type
ESSN	No	Char(9)
PNO	No	Int
HOURS	No	Decimal(3,1)

- This feature can only be used to obtain the most basic information about the relations.
- It may also be used to obtain information about aspects of the schema which are specific to Oracle, such as information about methods and procedures.

Oracle Data Dictionary Views:

- Each view consists of a single relation.
- There are three classes of views:
 - USER – attributes of objects owned by the current user.
 - ALL – attributes of objects accessible by the current user.
 - DBA – attributes of all objects
- Notes:
 - DBA attributes are available only to those with administrator privileges.
 - That which is available under USER and ALL is user dependent.

The following are the tables in the USER views.

USER View Name
USER_TABLES
USER_TAB_COLUMNS
USER_CONSTRAINTS
USER_CONS_COLUMNS
USER_INDICES
USER_IND_COLUMNS
USER_SYNONYMS
USER_TRIGGERS
USER_TAB_PRIVS

There is a corresponding table in each of the views ALL and DBA:

ALL View Name	DBA View Name
ALL_TABLES	DBA_TABLES
ALL_TAB_COLUMNS	DBA_TAB_COLUMNS
ALL_CONSTRAINTS	DBA_CONSTRAINTS
ALL_CONS_COLUMNS	DBA_CONS_COLUMNS
ALL_INDICES	DBA_INDICES
ALL_IND_COLUMNS	DBA_IND_COLUMNS
ALL_SYNONYMS	DBA_SYNONYMS
ALL_TRIGGERS	DBA_TRIGGERS
ALL_TAB_PRIVS	DBA_TAB_PRIVS

- Some of these tables are huge. For example, the ALL_TABLES view table has 37 columns.
- The full Oracle system actually has a much larger set of *static data dictionary views*.

The key feature of these data dictionary views is that they may be queried, just as may ordinary relations.

Example:

```
SELECT Table_Name, Column_Name, Data_Type
FROM ALL_TAB_COLUMNS
WHERE (Table_Name = 'Employee') OR
      (Table_Name = 'Department');
```

might yield:

Table_Name	Column_Name	Data_Type
Employee	Fname	Varchar(5)
Employee	Minit	Char
Employee	Lname	Varchar(15)
Employee	SSN	Char(9)
Employee	Bdate	Date
Employee	Address	Varchar(30)
Employee	Sex	Char
Employee	Salary	Decimal(10,2)
Employee	Superssn	Char(9)
Employee	DNO	Int
Department	Dname	Varchar(15)
Department	Dnumber	Int
Department	Mgrssn	Char(9)
Department	Mgrstartdate	Date

provided that the company database were accessible to the user issuing the query.

These tables also contain statistical information which can be extracted:

First, tell the system to update the statistics:

```
ANALYZE TABLE Employee  
COMPUTE STATISTICS;
```

Now issue a query to retrieve them:

```
SELECT Table_Name, Num_Rows, Blocks  
FROM ALL_TABLES
```

Basic Features of the ODBC Approach

- The ODBC standard provides a collection of API functions for querying the system catalog.
- They are more limited in overall scope than the Oracle collection, because they are not specific to a particular DBMS.

The principal ones are the following:

`SQLTables ()`
`SQLTablePrivileges ()`
`SQLColumns ()`
`SQLColumnPrivileges ()`
`SQLSpecialColumns ()`
`SQLStatistics ()`
`SQLPrimaryKeys ()`
`SQLForeignKeys ()`
`SQLProcedures ()`
`SQLProcedureColumns ()`

Use is relatively straightforward, but involves the detail which is typically associated with ODBC, and will not be elaborated in these slides.