# DATABASE DIRECTORY (DD)

(aka *Data Dictionary, System Catalog*)

Each DDL/DCL command issued causes information to be stored (e.g. the description of a table or view, or a user's permissions on a table or view). Also, every user-issued command must be validated prior to execution (so that it doesn't violate security restrictions) – such validation requires access to previously stored information.

The *Database Directory* (DD) is the repository of all information regarding a database. It is here that all information is stored as a result of user commands and it is from here that information is accessed for validation. Effectively, the DD contains metadata (data about data) that describes the objects of relevance within a database system: tables, views, users, permissions, etc.

Because we are using a relational database management system, it is most convenient to maintain the DD information in a set of relations (tables). These tables can then be read by users using standard SQL SELECT statements [Note, however, that users only have SELECT permission on the DD tables – only the DBMS software itself has the capability of modifying them]. In this document, we outline a typical[1] DD structure that might be employed in a DBMS.

## A. DD Table Structure, Population and Accessibility to Users

**SYSTABAUTH**

| GRANTEE | TNAME | CREATOR | GRANTOR | DATE | A | D | N | I | S | U |
|---|---|---|---|---|---|---|---|---|---|---|
| Public | staff | manager | manager | 17-03-01 | | | | | Y | |
| Public | branch | manager | manager | 17-03-01 | | | | | Y | |
| Scott | viewing | Murphy_P | Murphy_P | 20-04-01 | | | | | Y | |
| Scott | houses | Murphy_P | Jones_K | 01-06-01 | | | | Y | G | |
| Scott | stable | Scott | Scott | 01-01-02 | G | G | G | G | G | G |
| Scott | sview | Scott | Scott | 01-01-02 | G | G | G | G | G | G |
| | | | | | | | | | | |
| | | | | | | | | | | |

*Role*: Contains information on a user's authorization privileges on tables/views that he/she has created or on which he/she has been granted access

*Accessible To*: Each user, on a "private copy" basis

*Modified By:* CREATE TABLE, DROP TABLE, CREATE VIEW, DROP VIEW, GRANT(1), REVOKE(1)

**CATALOG**

| TNAME | CREATOR | TABLETYPE |
|---|---|---|
| staff | manager | table |
| branch | manager | table |
| viewing | Murphy_P | table |
| houses | Murphy_P | view |
| stable | Scott | table |
| sview | Scott | view |
| | | |

[1] The DD structure described here is similar to that used in the Oracle DBMS. This one was chosen because of its clarity of objectives.

**SYSCATALOG**

| TNAME | CREATOR | TABLETYPE |
|---|---|---|
| staff | manager | table |
| branch | manager | table |
| viewing | Murphy_P | table |
| houses | Murphy_P | view |
| stable | Scott | table |
| sview | Scott | view |
| columns | System | view |
| catalog | System | view |
| systabauth | System | view |
| | | |
| | | |
| | | |

*Role*: Contains information on the tables/views accessible to a user

*Accessible To*: Each user, on a "private copy" basis

*Modified By:* same as SYSTABAUTH

Note that SYSCATALOG is effectively a superset of CATALOG, containing information on system (DD) tables as well as user-level tables. In general, several DD tables have counterparts with the prefix "SYS" on their name: the implication is that the latter also includes descriptions of system tables.

**COLUMNS**

| TNAME | CNAME | COLNO | COLTYPE | WIDTH | SCALE | NULLS |
|---|---|---|---|---|---|---|
| staff | Sno | 1 | char | 4 | | not null |
| staff | Fname | 2 | char | 15 | | |
| staff | Lname | 3 | char | 15 | | |
| staff | Address | 4 | char | 56 | | |
| staff | Tel_No | 5 | char | 12 | | |
| staff | Position | 6 | char | 14 | | |
| staff | Sex | 7 | char | 1 | | |
| staff | DOB | 8 | date | | | |
| staff | Salary | 9 | number | 6 | 0 | |
| staff | NIN | 10 | char | 10 | | not null |
| staff | Bno | 11 | char | 5 | | |
| | | | | | | |
| | | | | | | |

*Role*: Contains a description of all tables/views accessible to a user
(another table, called COL, contains a description of tables/views created by a user – i.e. it is a subset of COLUMNS)

*Accessible To*: Each user, on a "private copy" basis

*Modified By:* same as SYSTABAUTH

**VIEWS**

| VIEWNAME | VIEWTEXT |
|---|---|
| sview | Select sno, fname, lname from staff where address like '%London%' |
| | |
| | |

*Role*: Contains the definition of all views created by a user

*Accessible To*: Each user, on a "private copy" basis

*Modified By:* CREATE VIEW, DROP VIEW, DROP TABLE, GRANT(2), REVOKE(2)

**SYSYSERAUTH**

| USERNO | USERNAME | PASSWORD | DATE | C | D | R |
|---|---|---|---|---|---|---|
| 0 | Public | | 22-01-95 | | | |
| 1 | System | sysmgr | 22-01-95 | Y | Y | Y |
| 2 | Scott | tiger | 17-03-95 | Y | | Y |
| 3 | Murphy_P | qwerty | 01-02-98 | Y | | Y |
| 4 | Jones_K | zxcvbn | 06-08-00 | Y | | |
| | | | | | | |

*Role*: Describes all users of database, along with their access privileges

*Accessible To*: Users with DBA privilege only

*Modified By:* GRANT(2), REVOKE(2)

**DTAB**

| DD TABLE | DESCRIPTION |
|---|---|
| catalog | Tables and views accessible to user (excluding data dictionary) |
| col | Columns in tables created by user |
| columns | Columns in tables accessible to user (excluding data dictionary) |
| dtab | Description of tables and views in data dictionary |
| | |

*Role*: Acts as a master index of the DD contents

*Accessible To*: All users

*Modified By:* Not applicable

## B. DD Table Definitions

The Database Directory is presented to the users as a collection of tables which, taken together, describes the users, user tables and user permissions applicable within a database. As is clear from the above examples, there may be considerable overlap in content between the various DD tables. Also, it would appear that there may be multiple copies of some of the tables – so that each user has a "private copy" of the table, containing the information relevant to him/her. If this is so, then there may be much redundancy: for example all users with access to a table would have its description included in their private version of COLUMNS.

In fact, no redundancy or overlap is needed: the DD is, in reality, a collection of overlapping views into a small number of base tables. Since individual users have no direct privilege to modify the DD contents, the use of views is appropriate. The DBMS software itself, however, operates directly on the underlying base tables and is thus capable of modifying their contents. The DD is a good example of the power and flexibility possible through the use of views. We use examples below to indicate how these DD views are defined and, in particular, how "private copy" views, and views accessible to DBA users only, can be provided.

We first note that every SQL command submitted by a user is appended with the username of that person (placed in a string variable *:uname*):

```
:uname                    (e.g. 'Scott')


SELECT Fname, Lname
FROM STAFF
WHERE Position = 'Manager'
```

```
:uname                    (e.g. 'Scott')


SELECT TNAME, CREATOR
FROM SYSTABAUTH
WHERE GRANTOR = 'manager'
    AND I IS NOT NULL
```

Now that :uname variable can be built into the view definition of the DD tables, and can thereby constrain the view contents on a per-user basis. For example, taking SYSUSERAUTH:

```
CREATE VIEW SYSTABAUTH AS
SELECT  GRANTEE, TNAME, CREATOR, GRANTOR, DATE, A, D, N, I, S, U
FROM    underlying DD base tables
WHERE   join conditions / other restrictions
        .
        .
  AND   GRANTEE in (:uname, 'Public')
```

The above definition will guarantee that, whenever a user attempts to access SYSTABAUTH, he/she will only see those rows with a GRANTEE column of his/her own username or of 'Public'. In effect, it is a "private copy' directory for that user. The constraint will automatically be applied during the *query modification* process.

Similarly, a DD table such as SYSUSERAUTH can be hidden from users without DBA permission by being defined as follows:

```
CREATE VIEW SYSUSERAUTH AS
SELECT  USERNO, USERNAME, PASSWORD, DATE, C, D, R
FROM   underlying DD base tables
WHERE  join conditions / other restrictions
       .
       .
  AND  'Y' =
  (   SELECT  D
      FROM   underlying DD base tables
      WHERE USERNAME = :uname  )
```

## C. Use of DD for Command Validation

The DD acts as a system-maintained information resource for each user of the system – by navigating through the DD (using SELECT commands), a user may determine which database objects he/she has created or has access to, the access privileges concerned, from whom the privileges came, the object descriptions, etc.

The DBMS software itself also looks up the DD for validation of commands submitted by a user. For example, the SELECT command given below would require several validation steps, many requiring access to the DD contents:

```
:uname
SELECT Fname, Lname, BRANCH.Tel_No, Salary
FROM STAFF, BRANCH
WHERE STAFF.Bno  =  BRANCH.Bno
   AND  Position = 'Manager'
   AND  City = 'London'
```

*Validation Steps:*

1. For each entry in the FROM clause, does user :uname have S privilege? (SYSTABAUTH)
2. For each entry in the FROM clause, is it a view? (CATALOG)
   If so, activate query modifier, which retrieves the view definition (VIEWS),
   and merges it with the user query to produce a final query
3. Is each table in the WHERE & SELECT clauses also referenced in the FROM clause?
4. Is each column entry in the SELECT & WHERE clauses valid and unambiguous? (COLUMNS)
5. Is each condition in the WHERE clause type-compatible? (COLUMNS)

   If all conditions satisfied, then compile and execute command

In fact, every user-issued SQL (DML, DDL & DCL) command requires validation, most of it involving access to the DD tables. For example, the GRANT command below must be checked:

```
:uname
GRANT INSERT
ON STAFF
TO Johnstone
```

*Validation Steps:*

1. Does the table in the ON clause exist? (CATALOG)
2. Does user :uname have 'grant option' for the permission named in the GRANT clause? (SYSTABAUTH)
3. Does the user named in the TO clause exist? (SYSUSERAUTH)

   If all conditions satisfied, then compile and execute command

**Note:**

Each user-submitted command, however simple, will be subjected to several security/validation steps. This can impose considerable overhead – in the sense of effort and time - on command processing by the DBMS. For example, a simple SELECT statement on a single table actually requires a potential search of five separate tables (SYSTABAUTH, CATALOG, VIEWS, COLUMNS and the table itself). In order to reduce this overhead and to speed up processing, database systems employ several techniques to ensure efficiency:

1. The DBMS uses precompiled, and highly optimized, routines for DD access – i.e. SQL queries are not formulated for the validation steps.
2. The DBMS does not use the DD views, but directly accesses the underlying base tables.
3. The underlying DD base tables are multiply indexed.
4. Much of the DD is maintained in main memory during DBMS use (as opposed to disk), reducing access times.