UNIT 4 DATABASE SYSTEM CATALOGUE

Structure **Page Nos.**

4.0 INTRODUCTION

The *system catalogue* is a collection of tables and views that contain important information about a database. It is the place where a relational database management system stores schema metadata, such as information about tables and columns, and internal bookkeeping information. A system catalogue is available for each database. Information in the system catalogue defines the structure of the database. For example, the *DDL* (data dictionary language) for all tables in the database is stored in the system catalogue. Most system catalogues are copied from the template database during database creation, and are thereafter database-specific. A few catalogues are physically shared across all databases in an installation; these are marked in the descriptions of the individual catalogues.

The system catalogue for a database is actually part of the database. Within the database are objects, such as tables, indexes, and views. The system catalogue is basically a group of objects that contain information that defines other objects in the database, the structure of the database itself, and various other significant information.

The system catalogue may be divided into logical groups of objects to provide tables that are accessible by not only the database administrator, but by any other database user as well. A user typically queries the system catalogue to acquire information on the user's own objects and privileges, whereas the *DBA* needs to be able to inquire about any structure or event within the database. In some implementations, there are system catalogue objects that are accessible only to the database administrator.

The terms system catalogue and data dictionary have been used interchangeably in most situations. We will make a distinction in the two terms in the context of data dictionary system, which is an implementation of the data dictionary or system catalogue. This unit provides more information on the system catalogue or data dictionary.

4.1 OBJECTIVES

After going through this unit, you should be able to:

- define the system catalogue and its content;
- describe the use of catalogue in a commercial DBMS;
- define the data dictionary system and its advantages and disadvantages, and
- define the role of catalogue in system administration.

4.2 CATALOGUE FOR RELATIONAL DATABASE MANAGEMENT SYSTEM

In database management systems, a file defines the basic organisation of a database. A data dictionary contains a list of all the files in the database, the number of records in each file, and the names and types of each field. Most database management systems keep the data dictionary hidden from users to prevent them from accidentally destroying its contents.

The information stored in a catalogue of an RDBMS includes:

- the relation names,
- attribute names,
- attribute domains (data types),
- descriptions of constraints (primary keys, secondary keys, foreign keys, NULL/ NOT NULL, and other types of constraints),
- views, and
- storage structures and indexes (index name, attributes on which index is defined, type of index etc).

Security and authorisation information is also kept in the catalogue, which describes:

- authorised user names and passwords,
- each user's privilege to access specific database relations and views,
- the creator and owner of each relation. The privileges are granted using GRANT command. A listing of such commands is given in *Figure 1*.

The system catalogue can also be used to store some statistical and descriptive information about relations. Some such information can be:

- number of tuples in each relation,
- the different attribute values,
- storage and access methods used in relation.

All such information finds its use in query processing.

In relational DBMSs the catalogue is stored as relations. The DBMS software is used for querying, updating, and maintaining the catalogue. This allows DBMS routines (as well as users) to access. The information stored in the catalogue can be accessed by the DBMS routines as well as users upon authorisation with the help of the query language such as SQL.

Let us show a catalogue structure for base relation with the help of an example. *Figure 1* shows a relational schema, and *Figure 2* shows its catalogue. The catalogue here has stored - relation names, attribute names, attribute type and primary key as well as foreign key information.

Student
(RollNo, Name, Address, PhoneNo, DOB, email, CourseNo, DNo)

Course
(CourseNo, CourseTitle, Professor)

Dept
(DeptNo, DeptName, Location)

Grade
(RollNo, CourseNo, Grade)

Figure 1: A sample relation

The description of the relational database schema in the *Figure 1* is shown as the tuples (contents) of the catalogue relation in *Figure 2.* This entry is called as CAT_ENTRY. All relation names should be unique and all attribute names *within a particular relation* should also unique. Another catalogue relation can store information such as tuple size, current number of tuples, number of indexes, and creator name for each relation.

Relation_Name	Attribute	Attr_Type	PK	FK	FK_REL
Student	RollNo	INTEGER(10)	Yes	No	
Student	Name	VARCHAR2(30)	No	No	
Student	Address	VARCHAR2(50)	No	No	
Student	PhoneNo	VARCHAR2(10)	No	No	
Student	DOB	Date/Time	No	No	
Student	email	VARCHAR2(30)	No	No	
Student	CourseNo	INTEGER(10)	No	Yes	Course
Student	DNo	INTEGER(10)	No	Yes	Dept
Course	CourseNo	INTEGER(10)	Yes	No	
Course	CourseTitle	VARCHAR2(10)	No	No	
Course	Professor	VARCHAR2(30)	No	No	
Dept	DeptNo	INTEGER(10)	Yes	No	
Dept	DeptName	VARCHAR2(20)	No	No	
Dept	Location	VARCHAR2(30)	No	No	
Grade	RollNo	INTEGER(10)	Yes	Yes	
Grade	CourseNo	INTEGER(10)	Yes	Yes	
Grade	Grade	VARCHAR(2)	No	No	

Figure 2: One sample catalogue table

Data dictionaries also include data on the secondary keys, indexes and views. The above could also be extended to the secondary key, index as well as view information by defining the secondary key, indexes and views. Data dictionaries do not contain any actual data from the database, it contains only book-keeping information for managing it. **Without a data dictionary, however, a database management system cannot access data from the database.**

The Database Library is built on a *Data Dictionary*, which provides a complete description of record layouts and indexes of the database, for validation and efficient data access. The data dictionary can be used for automated database creation, including building tables, indexes, and referential constraints, and granting access rights to individual users and groups. The database dictionary supports the concept of Attached Objects, which allow database records to include compressed BLOBs (Binary Large Objects) containing images, texts, sounds, video, documents, spreadsheets, or programmer-defined data types.

The data dictionary stores useful metadata, such as field descriptions, in a format that is independent of the underlying database system. Some of the functions served by the Data Dictionary include:

- ensuring efficient data access, especially with regard to the utilisation of indexes,
- partitioning the database into both logical and physical regions,
- specifying validation criteria and referential constraints to be automatically enforced,
- supplying pre-defined record types for Rich Client features, such as security and administration facilities, attached objects, and distributed processing (i.e., grid and cluster supercomputing).

☞ Check Your Progress 1

State whether the following are True or False:

1) A System Catalogue does not store the information on the relationships between two entities. ☐

2) Integrity constraints are applied to relations only and are not stored separately in the Data Dictionary. ☐

3) The system catalogue is a collection of tables and views that contain important information about a database. ☐

4) The information stored in the catalogue cannot be accessed by the users. ☐

5) Structured Query Language can be used to access information from the system catalogues. ☐

4.3 DATA DICTIONARY AND DATA REPOSITORY SYSTEM

The terms data dictionary and data repository are used to indicate a more general software utility than a catalogue. *A* **catalogue** is closely coupled with the DBMS software; it provides the information stored in it to users and the DBA, but it is *mainly* accessed by the various software modules of the DBMS itself, such as DDL and DML compilers, the query optimiser, the transaction processor, report generators, and the constraint enforcer. On the other hand, a Data Dictionary is a data structure that stores meta-data, i.e., data about data. The software package for a *stand-alone* **data dictionary** or **data repository** may interact with the software modules of the DBMS, but it is *mainly* used by the designers, users, and administrators of a computer system for information resource management. These systems are used to maintain information on system hardware and software configurations, documentation, applications, and users, as well as other information relevant to system administration.

If a data dictionary system is used *only* by designers, users, and administrators, and not by the DBMS software, it is called a **passive data dictionary**; otherwise, it is called an **active data dictionary** or **data directory**. An active data dictionary is automatically updated as changes occur in the database. A passive data dictionary must be manually updated.

The data dictionary consists of record types (tables) created in the database by system-generated command files, tailored for each supported back-end DBMS. Command files contain SQL statements for CREATE TABLE, CREATE UNIQUE

INDEX, ALTER TABLE (for referential integrity), etc., using the specific SQL statement required by that type of database.

4.3.1 Data Dictionary Features

A comprehensive data dictionary product will include:

- support for standard entity types (elements, records, files, reports, programs, systems, screens, users, terminals, etc.), and their various characteristics (e.g., for elements, the dictionary might maintain Business name, Business definition, name, Data type, Size, Format, Range(s), Validation criteria, etc.)

- support for user-designed entity types (this is often called the "extensibility" feature); this facility is often exploited in support of data modelling, to record and cross-reference entities, relationships, data flows, data stores, processes, etc.

- the ability to distinguish between versions of entities (e.g., test and production)
- enforcement of in-house standards and conventions.

- comprehensive reporting facilities, including both "canned" reports and a reporting language for user-designed reports; typical reports include:

    - detail reports of entities
    - summary reports of entities
    - component reports (e.g., record-element structures)
    - cross-reference reports (e.g., element keyword indexes)
    - where-used reports (e.g., element-record-program cross-references).

- a query facility, both for administrators and casual users, which includes the ability to perform generic searches on business definitions, user descriptions, synonyms, etc.

- language interfaces, to allow, for example, standard record layouts to be automatically incorporated into programs during the compile process.

- automated input facilities (e.g., to load record descriptions from a copy library).

- security features

- adequate performance tuning abilities

- support for DBMS administration, such as automatic generation of DDL (Data Definition Language).

4.3.2 Data Dictionary Benefits

The benefits of a fully utilised data dictionary are substantial. A data dictionary has the potential to:

- facilitate data sharing by

    - enabling database classes to automatically handle multi-user coordination, buffer layouts, data validation, and performance optimisations,
    - improving the ease of understanding of data definitions,
    - ensuring that there is a single authoritative source of reference for all users
- facilitate application integration by identifying data redundancies,
- reduce development lead times by

  - simplifying documentation
  - automating programming activities.
- reduce maintenance effort by identifying the impact of change as it affects:
  - users,
  - data base administrators,
  - programmers.
- improve the quality of application software by enforcing standards in the development process
- ensure application system longevity by maintaining documentation beyond project completions
- data dictionary information created under one database system can easily be used to generate the same database layout on any of the other database systems BFC supports (Oracle, MS SQL Server, Access, DB2, Sybase, SQL Anywhere, etc.)

These benefits are maximised by a *fully utilised* data dictionary. As the next section will show, our environment is such that not all of these benefits are immediately available to us.

4.3.3 Disadvantages of Data Dictionary

A DDS is a useful management tool, but it also has several disadvantages.
It needs careful planning. We would need to define the exact requirements designing its contents, testing, implementation and evaluation. The cost of a DDS includes not only the initial price of its installation and any hardware requirements, but also the cost of collecting the information entering it into the DDS, keeping it up-to-date and enforcing standards. The use of a DDS requires management commitment, which is not easy to achieve, particularly where the benefits are intangible and long term.

4.4 SYSTEM CATALOGUE IN A COMMERCIAL DATABASE MANAGEMENT SYSTEM

In this section we will discuss the system catalogue in the context of one of the commercial DBMS – Oracle. Although this section discusses the features of this commercial DBMS, yet we have tried to keep the details as general as possible. The collection of metadata is called the **data dictionary**. The metadata is information about schema objects, such as **Tables,** indexes, views, triggers and more. A data dictionary contains:

- the definitions of all schema objects in the database (tables, views, indexes, clusters, synonyms, sequences, procedures, functions, packages, triggers, and so on),
- amount of space has been allocated for, and is currently used by, the schema objects,
- default values for columns,
- integrity constraint information,
- the names of database users,
- privileges and roles each user has been granted,
- auditing information, such as who has accessed or updated various schema objects,
- other general database information.

Not only is the data dictionary central to every database, it is an important tool for all users, from end users to application designers and database administrators. SQL statements are used to access the data dictionary. Because the data dictionary is read-only, only SELECT statements can be used against its tables and views.

The data dictionary is structured in tables and views, just like other database data. All the data dictionary tables and views for a given database are stored in that database's SYSTEM table space. Access to the data dictionary is allowed through numerous views, which may be divided into three categories: USER, ALL, and DBA.

4.4.1 Views with the Prefix USER

The views most likely to be of interest to typical database users are those with the prefix USER. These views are as follows:

- refer to the user's own private environment in the database, including information about schema objects created by the user, grants made by the user, and so on,
- display only rows pertinent to the user,
- have columns identical to the other views, except that the column OWNER is implied,
- return a subset of the information in the ALL views,
- can have abbreviated PUBLIC synonyms for the sake of convenience.

For example, the following query returns all the objects contained in your schema:

SELECT object_name, object_type FROM USER_OBJECTS;

4.4.2 Views with the Prefix ALL

Views with the prefix ALL refer to the user's overall perspective of the database. These views return information about schema objects to which the user has access through public or explicit grants of privileges and roles, in addition to schema objects that the user owns. For example, the following query returns information on all the objects to which the user has access:

SELECT owner, object_name, object_type FROM ALL_OBJECTS;

4.4.3 Views with the Prefix DBA

Views with the prefix DBA show a global view of the entire database. Synonyms are not created for these views, because DBA views should be queried only by administrators. Therefore, to query DBA views, administrators must prefix the view name with its owner, SYS, as in the following:

SELECT owner, object_name, object_type FROM SYS.DBA_OBJECTS;

4.4.4 SYS, Owner of the Data Dictionary

The user named SYS owns all base tables and user-accessible views of the data dictionary. No general user should *ever* alter (UPDATE, DELETE, or INSERT) any rows or schema objects contained in the SYS schema, because such activity can compromise data integrity. The security administrator must keep a strict watch of this central account. Altering or manipulating the data in the data dictionary tables can permanently and detrimentally affect the operation of a database.

Use of Data Dictionary

The data dictionary in Oracle has three primary uses:

- Oracle accesses the data dictionary to find information about users, schema objects, and storage structures.

- Oracle modifies the data dictionary every time that a data definition language (DDL) statement is issued.
- Any Oracle user can use the data dictionary as a read-only reference for information about the database.

The system catalogue in Oracle contains information on all three levels of database schemas: external which has view definitions, conceptual - which defines base tables, and internal schema - giving the storage and index descriptions. Some of the catalogue views relating to each of the three schema levels are given below. The catalogue (metadata) data can be retrieved through SQL statements just as the user's actual data.

The first query is the example of the conceptual schema information where an object owned by a particular user, 'KARAN' can be obtained:

SELECT * FROM ALL-CATALOGUE WHERE OWNER = 'KARAN';

The result of this query could be as shown in *Figure 3*, which indicates that three base tables are owned by KARAN: STUDENT, COURSE, and DEPT, plus a view STUCRS. The columns are retrieved as shown.

OWNER	TABLENAME	TABLE_TYPE
KARAN	STUDENT	TABLE
KARAN	COURSE	TABLE
KARAN	DEPT	TABLE
KARAN	GRADE	TABLE
KARAN	STUCRS	VIEW

Figure 3: Result of query about particular user

To find some of the information describing the columns of the GRADE table *for* 'KARAN', the following query could be submitted:

SELECT COLUMN-NAME, DATA-TYPE, DATA-LENGTH, NUM-DISTINCT,
LOW-VALUE, HIGH-VALUE

FROM USER-TAB-COLUMNS
WHERE TABLE-NAME = 'GRADE';

The result of this query could be shown as in *Figure 4*. Because the USER- TAB-COLUMNS table of the catalogue has the prefix USER, this query must be submitted by the owner of the DEPT table. The NUM-DISTINCT column specifies the number of distinct values for a given column and LOW-VALUE and HIGH-VALUE show the lowest and highest values for the given column.

COLUMN-NAME	DATA-TYPE	DATA-LENGTH	NUM-DISTINCT	LOW-VALUE	HIGH-VALUE
ROLL NO.	NUMBER	10	50	1234A	9999A
COURSE NO.	NUMBER	10	50	AB012	MG999
GRADE	VARCHAR	2	8	F	A

Figure 4: Result of query on column details on GRADE table

A general listing of the SELECT clause used in Data Dictionary to retrieve the information is given in *Figure 5*. It includes the various entity, data type, primary key (pk), foreign key (fk), alternate key (ak), and attribute definition for the retrieval.

Listing: The Dictionary SELECTs

```
select table_name as table_name,
  as column_name,
  entity_definition as entity_definition,
  entity_note as entity_note,
  as column_datatype,
  as default_value,
  as pk, as fk, as ak,
  as attribute_definition,
  as sample_instance_data,
      query as query
  from entity
  union
  select table_usage,
      column_name,
      column_datatype,
      null_option,
      default_value,
      pk, fk, ak,
      attribute_definition,
  from attribute
  order by 1, 2
```

Figure 5: A general listing for SELECT clauses on data dictionary

At runtime each detail row displays either table data, or column data, but not both. A listing of GRANT commands is given in *Figure 6*. This gives the privileges to the users according to the GRANT commands.

Listing: Generating Grant Commands

```
select string ( 'grant all on ',
      table_name, ' to administration;' )
      as "GRANT COMMAND" from entity
union select string ( 'grant delete on ',
      table_name, ' to endusers;' )
from entity where query like '% delete%'
union select string ( 'grant insert on ',
      table_name, ' to endusers;' )
from entity where query like '% insert%'
union select string ( 'grant select on ',
      table_name, ' to endusers;' )
from entity where query like '% select%'
union select string ( 'grant update on ',
      table_name, ' to endusers;' )
from entity where query like '% update%'
order by 1;
```

Figure 6: A generalised listing of GRANT command

In a simple data dictionary each table definition is immediately followed by an alphabetic list of its columns, together with their data types, null versus not null settings, default values and an indication of whether it is part of a primary, foreign and/or alternate key (PK, FK, AK). As with the tables, a concise free-form description appears directly beneath each column name.

Some of these details are specific to the needs of one particular data base application. Their primary importance is to the human reader, not computer programme. Many of these details are available elsewhere; for example, the data types are visible in the database diagram shown in *Figure 7*, and the default values and access privileges may be discovered by querying the system catalogue *(Figure 8)*. The dictionary

serves to gather the important information together in one place and, of course, to provide the table and column descriptions that make a dictionary what it is.

Addr_Info Table Business or any other address Information End Users have SELECT, INSERT and UPDATE rights
Contact_Addr_id INTEGER NOT NULL (PK) (FK) Auto number Automatically generated Number Unique on the defined database
Date/Time Updated TIMESTAMP NOT NULL DEFAULT Date/Time this contact was added or updated
Fax CHAR (12) NOT NULL Fax number of the contact
First_name CHAR(20) Not NULL First Name of the contact
Website CHAR(100) NOT NULL WWW URL of the contact

Figure 7: A sample database diagram with description

The detailed Data dictionary is stored as given in *Figure 8*

Addr_Info
Addr_id:CHAR(6) NOT NULL(FK)
Contact_Addr_id INTEGER NOT NULL(PK)
First_name CHAR (20) NOT NULL
Last_name CHAR (20) NOT NULL
Title CHAR (10) NOT NULL
Addr1 CHAR (50) NOT NULL
Addr2 CHAR (20) NOT NULL
City CHAR (20) NOT NULL
State CHAR (20) NOT NULL (FK)
Pin_Code CHAR (6) NOT NULL
PhoneNo CHAR (12) NOT NULL
Fax CHAR (12) NOT NULL
Email CHAR (50) NOT NULL
Website CHAR (100)
Date/Time updated TIMESTAMP NOT NULL

Figure 8: A detailed data dictionary

The main reason to store dictionary information in the database diagram itself is the ease of data entry. It is quite hard to ensure correctness and completeness if the dictionary data is stored separately from the diagram.

The text stored in the query column has been written to be both readable by human beings and usable by the SQL command in *Figure 7*. Precise positive statements are used, such as “End users have select, insert and update rights” rather than vague or negative sentences like “End users can’t remove any rows from this table”.

A data dictionary is a valuable tool if it contains practical and useful information, and if it is kept up to date and accurate. That implies it must reflect the physical reality of the database rather than simple logical view, and that it must be easy to update and generate.

☞ Check Your Progress 2

1) List the various types of views through which access to the data dictionary is allowed.

……

……

……

……

2) What are the uses of Data Dictionary in Oracle?

……

……

……

……

3) What is the difference between active and passive Data Dictionary?

……

……

……

……

4.5 CATALOGUE IN DISTRIBUTED DATABASE AND OBJECT ORIENTED DATABASE SYSTEMS

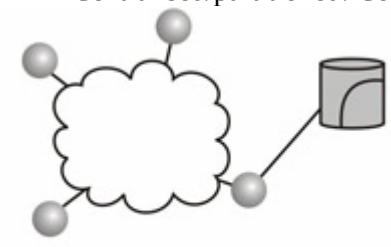
The applications of the database systems over the time were required to support distributed nature of organisation and object oriented system design. Let us discuss the issues of data dictionary in the context of these advanced database systems.

4.5.1 Catalogue in Distributed Database Systems

The data dictionary stores useful metadata on database relations. In a Distributed database systems information on locations, fragmentations and replications is also added to the catalogues.

The distributed database catalogue entries must specify site(s) at which data is being stored in addition to data in a system catalogue in a centralised DBMS. Because of data partitioning and replication, this extra information is needed. There are a number of approaches to implementing a distributed database catalogue.

- Centralised: Keep one master copy of the catalogue,
- Fully replicated: Keep one copy of the catalogue at each site,
- Partitioned: Partition and replicate the catalogue as usage patterns demand,
- Centralised/partitioned: Combination of the above.



(a) Centralised

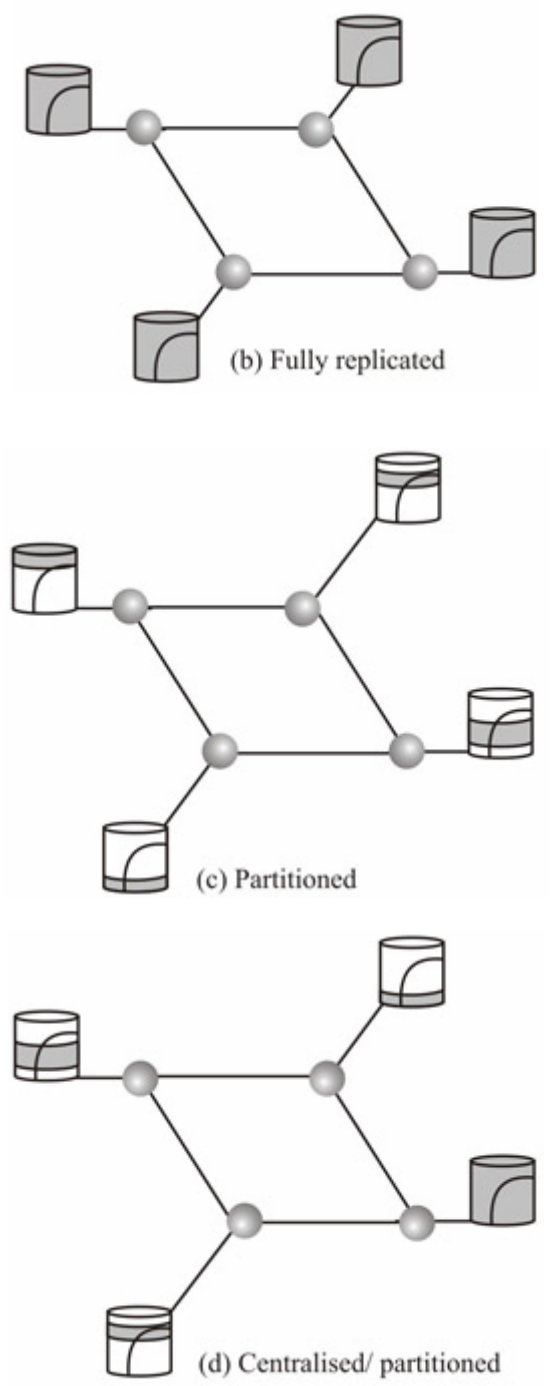


Figure 9: A distributed system catalogue

Catalogues are generally updated during creation and/or deletion of relations and modification of attributes. Certain characteristics of the data affect the access and query evaluation. For example, in some distributed catalogues such as R* information of locally stored objects including the fragments and replicates is stored on local catalogues. The object storage sites of any object are maintained at the place where the object was first created and the new sites where the object is moved is updated. This is a partitioned scheme where the catalogue is partitioned and replicated as per the demands of usage patterns.

In some distributed systems local and global relations are different. Global relations can be accessed from all sites. A catalogue containing information of a global nature is maintained at all sites. This is a fully replicated or redundant scheme where - one copy of the catalogue is kept at each site.

Distributed catalogue also store the information on local-to-global name mappings, file organisation and access methods, general as well as integrity constraints details and database statistics.

Remote catalogue information can be requested by any site and stored for later use. However, this stored catalogue may become inconsistent over time as the remote catalogue gets updated. Any query accessing the stored catalogue yields out-of-date information. Hence, the stored remote catalogue has to be updated from time to time.

4.5.2 Catalogue in Object Oriented Database Systems

The concept of object-oriented databases will be introduced in Unit 1 of Block 3 of this course. An object oriented database systems brings together the features of object-oriented programming and the advantages of database systems under one persistent DBMS interface. Thus, they are very useful in applications with complex interrelationships, complex classes, inheritance etc. However, as far as data dictionary is concerned, it now additionally needs to describe few more classes, objects and their inter-relationships. Thus, a data dictionary for such a system may be more complex from the viewpoint of implementation; however, from the users point of view it has almost similar concepts. This data dictionary should now also store class definitions including the member variables, member functions, inheritances and relationships of various class objects.

4.6 ROLE OF SYSTEM CATALOGUE IN DATABASE ADMINISTRATION

The database administration is a specialised database activity that is performed by a database administrator. The system catalogue has an important role to play in the database administration. Some of the key areas where the system catalogue helps the database administrator are defined below:

Enforcement of Database Integrity: System catalogue is used to store information on keys, constraints, referential integrity, business rules, triggering events etc. on various tables and related objects. Thus, integrity enforcement would necessarily require the use of a data dictionary.

Enforcement of Security: The data dictionary also stores information on various users of the database systems and their access rights. Thus, enforcement of any security policy has to be processed through the data dictionary.

Support for Database System Performance: The data dictionary contains information on the indexes, statistics etc. Such information is very useful for query optimisation. Also such information can be used by the database administrator to suggest changes in the internal schema.

Data dictionary can also support the process of database application development and testing (although this is not a direct relationship to database administration) as they contain the basic documentation while the systems are in the process of being developed.

☞ Check Your Progress 3

1) List the disadvantages of a data dictionary.
……
……
……
……

2) What are the approaches to implementing a distributed database catalogue?
……
……

……

……

3) How are Object Oriented catalogues implemented?

……

……

……

……

4.7 SUMMARY

This unit provides an detailed view of a data dictionary in a DBMS. The data dictionary is one of the most important implementation tools in a database system. The system catalogue provides all the information on various entities, attributes, database statistics etc. It is a very useful tool if implemented actively in a database system. However, active implementation of a data dictionary is costly.

In this unit we have discussed concepts related to data dictionary and its use in oracle by the different types of users. We have also presented information on the data dictionary system and its advantages and disadvantages. We have provided a brief introduction to system catalogue in distributed systems and how data dictionary is useful in system administration.

4.8 SOLUTIONS/ANSWERS

Check Your Progress 1

1) False 2) False 3) True 4) False 5) True

Check Your Progress 2

1) a) Views with the Prefix USER b) Views with the Prefix ALL c) Views with the Prefix DBA

2) The data dictionary in Oracle has three primary uses:

- Oracle accesses the data dictionary to find information about users, schema objects, and storage structures.
- Oracle modifies the data dictionary every time that a data definition language (DDL) statement is issued.

3) If a data dictionary system is used *only* by designers, users, and administrators, not by the DBMS software, it is called a **passive data dictionary**; otherwise, it is called an **active data dictionary** or **data directory**. An active data dictionary is automatically updated as changes occur in the database. A passive data dictionary must be manually updated.

Check Your Progress 3

1) (a) careful planning, (b) defining the exact requirements designing its contents, (c) testing, (d) implementation, (e) evaluation, (f) cost of a DDS includes not only the initial price of its installation and any hardware requirements, but also the cost of collecting the information entering into the DDS, (g) keeping it up-to-date and enforcing standards.

2) a) Centralised: Keeps one master copy of the catalogue
b) Fully replicated: Keeps one copy of the catalogue at each site

c) Partitioned: Partitions and replicates the catalogue as per the demands of the usage pattern
d) Centralised/partitioned: Combination of the above

3) In addition to system catalogues that may be used in simple relational database system, an object oriented system catalogue needs to define complex user defined classes, inheritance hierarchy and complex inter-relationships.