# Datenbanken II B: DBMS-Implementierung

# Chapter 1: Introduction

Prof. Dr. Stefan Brass

Martin-Luther-Universität Halle-Wittenberg

Wintersemester 2019/20

`http://www.informatik.uni-halle.de/~brass/dbi19/`

## Objectives

After completing this chapter, you should be able to:

- explain some functions of a DBMS.
- give an overview of the architecture of a DBMS.

  Name some important components.

- enumerate tasks of the database administrator.
- explain the structure of the Oracle Data Dictionary.
- formulate queries against data dictionary tables.
- name at least three tables of the Oracle data dictionary and their most important columns.

## Inhalt

## Database Services (1)

- The main task of a database management system (DBMS) is to manage persistently stored data.

  Persistent means that the data needs to be remembered longer than a single program execution, it even has to survive a shutdown of the OS. Today this basically means it must be stored on disk.

- The data is shared between many users and many application programs.
- The DBMS acts as a server: It receives requests (queries and updates) over the network and sends responses (answers, results) back.

  The DBMS is usually a set of background processes ($\sim$ web server).

## Database Services (2)

- Most of the queries and updates are executed by application programs.

  Ad-hoc queries are relatively seldom in practice. In a heavily loaded system, the DBA will forbid that ad-hoc queries are entered during standard working hours.

- Therefore, most of the queries/updates are known in advance, including estimates for their frequency (e.g. 5 per min).

  "Load profile": List of DB commands with frequencies. Of course, the queries/updates are not completely known. They usually contain parameters (e.g. in the form `:X` with a program variable `X`), in place of constants. Values for the parameters are known at runtime when the command is executed.

## Database Services (3)

- The data is an important asset of the company. Hardware/software failures should not lead to a loss or corruption of data.

  Transactions are an important concept to protect the data in the presence of failures (see below).

- Some systems must run 7 days a week, 24 hours a day (very high availability requirements).
- Many users are (usually) working with a DBMS at the same time. This requires synchronization of concurrent accesses, e.g., via locks.

## Database Services (4)

- Not every user is allowed to do everything: The DBMS must manage access rights for each user.

  It might also have to identify (authenticate) the users, if this service is not taken from the operating system.

- Other database services are, e.g.:
  - Integrity enforcement,
  - views,
  - stored procedures, triggers,
  - system catalog (Data Dictionary).

## Physical Data Independence (1)

- Application programs depend only on the logical structure of the data (e.g. tables and columns).
- Queries and updates in SQL do not depend on the way the data is stored, e.g.
  - how the tables are distributed over disks,
  - which access paths / indexes exist,

    An index over attribute $A$ of relation $R$ is a data structure that permits efficient access to the tuples of $R$ with a given value for $A$.
  - how free-space management parameters are set.

    These determine where on the disk a new row is stored. E.g. it might be possible that rows are stored clustered by an attribute (rows with the same value are stored near to each other).

## Physical Data Independence (2)

- SQL is a declarative query language:
  - An SQL query specifies only what information is sought,
  - but does not prescribe any particular method how to compute this information.
- Declarative languages often allow simpler/shorter formulations.

  The user does not have to think about efficient execution.
- Physical storage details are abstracted away on the SQL level.

## Physical Data Independence (3)

- The DBMS automatically translates the given SQL query into a query evaluation plan (QEP) which is then executed to compute the result of the query.

  The translation is done by the query optimizer. A QEP is a program for the execution engine in the DBMS. QEPs is also called access plans or execution plans. They are similar to relational algebra expressions.

- The physical parameters (indexes etc.) do influence the performance of query evaluation (and the required disk space).

  Certain QEPs depend on the existence of an index.

## Physical Data Independence (4)

- The task of physical DB design is to find good settings for the physical parameters, e.g. to select indexes.
- The physical design needs to be modified from time to time because
  - the size of the database objects changes,
  - the invocation frequency of programs changes,
  - new applications are developed.
- Because of physical data independence, application programs are not affected by this modification.

## Performance Tuning

- The goal of performance tuning is to meet given performance requirements. Techniques are:
  - Modifying the physical design.
  - Changing parameters of the DBMS or the OS.

    E.g. sizes of certain main-memory areas (buffer cache).
  - Extending the given hardware.

    Buying e.g. more main memory or additional disks.
  - Changing the application programs.
  - Changing logical design (e.g. denormalization).
  - Changing business rules (requirements).

## Transactions (1)

- A transaction is a sequence of DB commands, particularly updates, which the DBMS treats as a unit.

  E.g. a transfer of 50 dollars from account 1 to account 2 consists of
  (1) checking the current balance/credit limit of account 1,
  (2) decreasing the balance of 1 by 50 (debit),
  (3) increasing the balance of 2 by 50 (credit).

- Transactions have the ACID-Properties:
  Atomicity, Consistency, Isolation, Durability.

- The successful end of a transaction is marked with the SQL command `COMMIT`.

  One can explicitly declare the unsuccessful end with the command `ROLLBACK`. Implicitly this happens when the program crashes.

## Transactions (2)

Atomicity:

- DBMS guarantee that each transaction is either executed in total, or not at all.

  If the transaction cannot be executed until the end (e.g. because of a power failure or system crash), the DB state before the transaction has begun will be restored when the DBMS is started the next time.

- As long as the transaction has not been declared as complete all changes can be undone (`ROLLBACK`).

- I.e. the DBMS needs to log changes / remember old versions until the transaction is finished.

  This is the purpose of the rollback segments in Oracle.

## Transactions (3)

Durability:

- When a DBMS acknowledges the `COMMIT`, it guarantees that the changes are durable.
- The changes are stored on disk — they are not lost even if there is a power failure one second later.

  In operating systems, one often cannot be sure whether the data is on disk or still in a buffer. In a typical DBMS (like Oracle), the changes are not immediately written to the right place on disk, but to sequential file, the redo log (for efficiency reasons).

- Larger DBMS have powerful backup and recovery mechanisms: Even if a disk fails, no data is lost.

  With OS utilities, typically only one backup per day is created.

## Transactions (4)

Atomicity and Durability Together:

- There is one point in time that lies between
  - the user telling the system that the transaction is complete (`COMMIT`), and
  - the system telling the user that this command was successfully processed,

  when all changes become effective.

  If the system crashes before this point, the database state is not changed. If the system crashes after this point, the database state contains all changes that the transaction has executed. In a typical DBMS, this is the point when the entry for the `COMMIT` is completely contained in the redo log.

## Transactions (5)

Isolation:

- Different processes can access the database concurrently. Without control, this could have strange effects including lost updates.

  DBMS try to create the impression that every transaction runs in isolation, i.e. has exclusive access to the complete DB. The theoretical goal of "Serializability" is not quite reached in practice, some support from the programmer is needed.

- Usually, a DBMS manages locks on database objects (tables, rows, table entries) for this purpose.

  Different types of locks (e.g. shared, exclusive) are used. For the most part, lock management is done automatically by the DBMS. Locks are typically kept until the end of the transaction.

## Transactions (6)

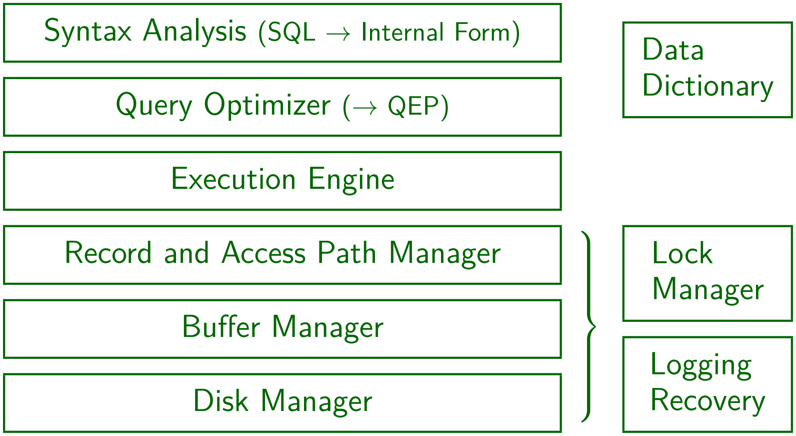
Consistency:

- User and system can be sure that the current state is the result of a sequence of completely executed transactions.

  E.g., it is not possible that a tuple was entered into a table, but not into the index on this table because of a power failure in between.

- The user must ensure that each transaction, if applied fully and in isolation to a consistent state, will produce a consistent state.
- Modern DBMS offer some support for this: Declarative constraints, triggers, stored procedures.

## DBMS Architecture: Example



## Inhalt

## DB Administrator Tasks (1)

- The DBA ensures that the DBMS keeps running:
  - Monitors available disk space, installs new disks.

    If disks fill up, the system will come to a standstill.
  - Ensures that backup copies are made.
    Does the recovery after disk failures etc.
  - Kills sessions of users who locked an important object and then went to lunch.
  - Monitors performance, ensures that users do not monopolize the system.

    He/she administers quotas (disk, CPU) and usage rules.

## DB Administrator Tasks (2)

- The DBA ensures security and confidentiality of the data (some companies have a special security administrator):
    - The DBA creates new user accounts.

      And deletes or locks unused accounts.
    - The DBA manages access rights to DB objects.
    - The DBA might do auditing (who did what) and analyzes the collected data for suspicious events.
- The DBA sometimes needs powerful privileges for the OS. He/she can damage everything.

## DB Administrator Tasks (3)

- The DBA tries to ensure data correctness.
- The DBA does performance tuning.

  Sometimes, external experts are asked to do this.
- The DBA should be an expert for
  - the DBMS software,

    Oracle 8 had $\geq$ 95 volumes ($=$ 1.70 m) of documentation.
  - the database schema (or schemas).

  In addition, he/she should know the users of the system.

## DB Administrator Tasks (4)

- The DBA installs new versions of the DBMS software.
- The DBA is the technical contact point for the DBMS vendor.

  E.g., for support, information about security holes.

- The DBA is responsible for keeping the terms of the software licence agreement.
- In all new developments of application programs for this database the DBA has a say.

# Inhalt

## Data Dictionaries (1)

- Most DBMS have a large collection of system tables, called the “data dictionary” (system catalog).
- In Oracle, it contains the following information:
  - Tables, views, etc. (database schema).
  - Comments on tables/columns (documentation).
  - Database Users, Access Rights, Auditing Data.
  - Indexes, Physical Storage Parameters, File space usage (e.g. allocation of disk blocks for tables).
  - Statistical Information, Performance Data.

## Data Dictionaries (2)

- Data dictionaries are very system-dependent.

  The tables Oracle uses are completely different from those used in DB2 or SQL Server. The data dictionary even changed substantially between different versions of Oracle. However, the SQL-92 standard proposes an information schema with some standard views (currently only implemented in SQL Server).

- The data dictionary is an important tool for the DBA!

  All information that the DBMS has should be available in system tables. Therefore, to understand all information in the data dictionary is to understand the system. A seasoned DBA should know many of the data dictionary tables (at least 50). The Oracle certification exams also contain exercises that ask for data dictionary tables.

## Data Dictionaries (3)

- Of course, modern DBMS also have a graphical user interface to the system data, e.g. the Oracle Enterprise Manager.

  Or: `http://dbs.informatik.uni-hannover.de/ftp/software/oddis/`

- However, such tools fetch their information from the data dictionary.

- In addition, they support only browsing of the data. More complicated evaluations still must be done using SQL queries (possibly in scripts) to the data dictionary.
  E.g. find disks that are nearly full.

## Example: Catalog (1)

- Names of schema objects (e.g. tables, columns) are now stored as data in the database.
- Such data is called "meta-data" (data about data).
- In this way, queries to data and meta-data can be formalized in the same language.

  A general query language like SQL is much more powerful than a specialized set of commands like "`describe`" in Oracle SQL*Plus.

- E.g., this query lists all tables of the current user:

```
SELECT TABLE_NAME FROM CAT
```

- `CAT` is a table (view) from the data dictionary.

## Example: Catalog (2)

- E.g., for the guest user SCOTT, CAT looks as follows:

| CAT | |
|---|---|
| TABLE_NAME | TABLE_TYPE |
| DEPT | TABLE |
| EMP | TABLE |
| ⋮ | ⋮ |

- CAT lists also views, sequences, synonyms.

  CAT lists all table-like objects (TABLE_TYPE shows the exact type). CAT is not listed because it is not owned by SCOTT. Sequencs are generators for unique numbers, for synonyms see below. Both exist only in Oracle.

- Note that table names etc. are stored in uppercase!

  While normally, case is not important for table and column names, it is important for string data.

## Example: Catalog (3)

- All table-like objects accessible by the current user are listed in the data dictionary view "ALL_CATALOG":

| ALL_CATALOG | | |
|---|---|---|
| OWNER | TABLE_NAME | TABLE_TYPE |
| SCOTT | BONUS | TABLE |
| SCOTT | DEPT | TABLE |
| ⋮ | ⋮ | ⋮ |
| SYS | USER_CATALOG | VIEW |
| PUBLIC | USER_CATALOG | SYNONYM |
| PUBLIC | CAT | SYNONYM |
| SYS | ALL_CATALOG | VIEW |
| PUBLIC | ALL_CATALOG | SYNONYM |
| ⋮ | ⋮ | ⋮ |

## Users and Schemas in Oracle

- In Oracle, database objects (like tables) are globally identified by their owner and their name.

  The owner is the user who created the object.

- Different users can create tables with the same name, these are different tables in Oracle.

  I.e. every user has his/her own database schema. In Oracle, there is a 1:1-mapping between DB schemas and users (accounts). Of course, it is possible that the same person has two separate user accounts.

- If one has read access to the table `DEPT` owned by `SCOTT`, one can access it with

```
SELECT * FROM SCOTT.DEPT
```

## Synonyms in Oracle (1)

- The data dictionary is owned by the user SYS.

  SYS is the most powerful account in Oracle.

- E.g., one can query ALL_CATALOG with

```
SELECT *
FROM   SYS.ALL_CATALOG
WHERE  OWNBER = 'SCOTT'
```

- However, Oracle has introduced synonyms (abbreviations, macros) to simplify this. E.g., try

```
CREATE SYNONYM DEPARTMENTS FOR SCOTT.DEPT
```

  Then "SELECT * FROM DEPARTMENTS" means actually

```
SELECT * FROM SCOTT.DEPT
```

## Synonyms in Oracle (2)

- Normal synonyms are only applicable for commands of the user who created them.
- However, Oracle also has "public synonyms", which are available to all users (who do not have a table etc. of the same name).

  Only a DBA can use "`CREATE PUBLIC SYNONYM`". Public synonyms appear in the data dictionary as synonyms owned by the special user "`PUBLIC`".
- `CAT` and `ALL_CATALOG` are such public synonyms.

  It is possible to create a table called "`ALL_CATALOG`". Then one must use "`SYS.ALL_CATALOG`" in order to access the data dictionary "table".

## Oracle Data Dictionary (1)

- There are three versions of the catalog table:
  - USER_CATALOG: Table-like objects owned by the current user.

    Columns are TABLE_NAME and TABLE_TYPE. The column OWNER (present in the other two versions) would always be the current user.

  - ALL_CATALOG: Table-like objects accessible by the current user.
  - DBA_CATALOG: All table-like objects in the system.

    Of course, DBA_CATALOG is accessible only to DBAs.

- Most data dictionary tables in Oracle exist in three versions with the prefixes USER, ALL, and DBA.

## Oracle Data Dictionary (2)

- The "real" system tables have a rather unreadable format for performance reasons.

  A system can use any data structure for the system data, as long as it offers a relational interface to these data. It does not necessarily have to be the same data structure as used for normal user tables.

- However, Oracle has defined many views to give a more user-friendly (and stable) interface.

  The definitions of the data dictionary views are in:
  `$ORACLE_HOME/rdbms/admin/catalog.sql`

- `USER_CATALOG` etc. are views owned by `SYS`.

  They must be views, because otherwise they could not show every user a different result (the query uses the SQL-function "`USER`").

## Oracle Data Dictionary (3)

- In addition, Oracle has defined public synonyms that simplify the access to these views.
- E.g., `USER_CATALOG` is a public synonym that points to `SYS.USER_CATALOG`.

  The public synonyms are not defined in all Oracle installations (at least for the `DBA`-views). Then one must write, e.g., `SYS.USER_CATALOG`.

- Furthermore, abbreviations have been defined (as public synonyms) for the most important data dictionary tables.
- E.g., `CAT` is a synonym for `SYS.USER_CATALOG`.

## Oracle Data Dictionary (4)

- The data dictionary contains also “Dynamic Performance Views”. Their names start with `V$`.

  In contrast, the above data dictionary views are called the “Static Data Dictionary Views”.

- These “tables” give a relational interface to data structures of the server (in main memory).

- E.g., currently active sessions (users logged into Oracle) are listed in `V$SESSION`.

  Dynamic performance views use the singular form, whereas the static data dictionary views normaly use the plural form (e.g., `USER_TABLES`). This is inconsistent and confusing.

## Oracle Data Dictionary (5)

- The Oracle Data Dictionary is documented in the "Oracle Database Reference".

  Don't mix this up with the "Oracle Database SQL Reference". Part II is about Static Data Dictionary Views, Part III about Dynamic Performance Views. For an introduction, see Chapter 7 of the Oracle Database Concepts Manual. The real system tables are not documented. However, one could analyze the view definitions.

- The data dictionary tables are read-only to ensure consistency.

  E.g. `INSERT` cannot be used on system tables. They can only be changed with specialized commands like `CREATE TABLE`. As `SYS`, it might be possible to update the real system tables, but it is quite likely that this will destroy the entire database.

## Data Dictionary (1)

- DICT lists all data dictionary tables/views:

| DICT | |
|---|---|
| TABLE_NAME | COMMENTS |
| ALL_CATALOG | All tables, views, synonyms, sequences accessible to the user |
| USER_CATALOG | Tables, Views, Synonyms and Sequences owned by the user |
| DICTIONARY | Description of data dictionary tables and views |
| DICT_COLUMNS | Description of columns in data dictionary tables and views |
| DICT | Synonym for DICTIONARY |
| ⋮ | ⋮ |

## Data Dictionary (2)

- Columns of `DICT` are:
  - `TABLE_NAME`: Name of the table, view, synonym.
  - `COMMENTS`: Short description.
- In Oracle 9.2.0, it has 508 rows when queried as normal user, and 1284 rows when queried as DBA.
- It is difficult to remember all data dictionary tables, but if one only remembers `DICT` and `DICT_COLUMNS`, one has a good chance to find the right table.
- `DICT` and `DICT_COLUMNS` contain meta-meta data.

  The schema of the data dictionary. There are no (meta)$^3$-data tables.

## Data Dictionary (3)

- E.g. this query prints all data dictionary objects containing "`CAT`" in their name:

```
SELECT *
FROM   DICT
WHERE  TABLE_NAME LIKE '%CAT%'
```

- The output in SQL*Plus looks better if the following formatting commands are entered before the query (works only in SQL*Plus, is not part of SQL):

```
COLUMN TABLE_NAME FORMAT A25
COLUMN COMMENTS FORMAT A50 WORD WRAP
SET PAGESIZE 100
```

## Data Dictionary (4)

- DICT_COLUMNS contains information about the single columns of the data dictionary tables (views):

| DICT_COLUMNS | | |
|---|---|---|
| TABLE_NAME | COLUMN_NAME | COMMENTS |
| DICT | TABLE_NAME | Name of the object |
| DICT | COMMENTS | Text comment on the object |
| DICT_COLUMNS | TABLE_NAME | Name of the object that contains the column |
| DICT_COLUMNS | COLUMN_NAME | Name of the column |
| DICT_COLUMNS | COMMENTS | Text comment on the object |
| ⋮ | ⋮ | ⋮ |

It has 13 375 entries for the DBA, 10 554 for normal users.

## Inhalt

## Database Objects (1)

- USER_OBJECTS (synonym OBJ) lists all database objects (tables etc. like in CAT, but also e.g. indexes, procedures, triggers) owned by the current user:

| OBJ | | | | |
|---|---|---|---|---|
| OBJECT_NAME | ··· | OBJECT_TYPE | CREATED | ··· |
| DEPT | ··· | TABLE | 29-JAN-98 | ··· |
| PK_DEPT | ··· | INDEX | 29-JAN-98 | ··· |
| EMP | ··· | TABLE | 29-JAN-98 | ··· |
| PK_EMP | ··· | INDEX | 29-JAN-98 | ··· |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

## Database Objects (2)

- The most important columns of OBJ are:
  - OBJECT_NAME: Name of the table, index, etc.
  - OBJECT_TYPE: E.g. TABLE, INDEX.

    OBJECT_TYPE can be: CLUSTER, FUNCTION, INDEX, LIBRARY, PACKAGE, PACKAGE BODY, PROCEDURE, SEQUENCE, SYNONYM, TABLE, TRIGGER, TYPE, UNDEFINED, VIEW.
  - CREATED: Date/Time when object was created.
  - LAST_DDL_TIME: Last ALTER TABLE, GRANT, etc.
  - TIMESTAMP: Last change of object specification.

    This changes e.g. when a column is added, but it does not change when constraints are added or a grant is made.

## Database Objects (3)

- Columns of `OBJ`, continued:
  - `GENERATED`: Was object name system generated?

    E.g. when the user does not specify a constraint name for a primary key, the name of the corresponding index will be something like "`SYS_C001284`", and this column will contain a "`Y`".

  - `STATUS`: `VALID`, `INVALID`, or `N/A`.

    Normally, it is "`VALID`". But if e.g. a view references a table that is deleted, the view is not automatically deleted, but its status becomes "`INVALID`".

  - `TEMPORARY`: No multi-user sync., no recovery.

    Each process/session can see only the data it has placed itself in the object.

## Database Objects (4)

- Of course, there are also `ALL_OBJECTS`/`DBA_OBJECTS` that list all accessible/all objects of the database.

  These have also a column `OWNER`. All columns: `OWNER`, `OBJECT_NAME`, `SUBOBJECT_NAME`, `OBJECT_ID`, `DATA_OBJECT_ID`, `OBJECT_TYPE`, `CREATED`, `LAST_DDL_TIME`, `TIMESTAMP`, `STATUS`, `TEMPORARY`, `GENERATED`.

- E.g. when was the table "EMP" created?

```
SELECT CREATED
FROM   OBJ
WHERE  OBJECT_NAME = 'EMP'
```

- To see also the time, select the following:

```
TO_CHAR(CREATED, 'DD.MM.YYYY HH24:MI:SS')
```

## Table Columns (1)

- USER_TAB_COLUMNS (synonym COLS) describes the columns of tables owned by the current user:

| COLS | | | | | |
|---|---|---|---|---|---|
| TABLE_NAME | COLUMN_NAME | DATA_TYPE | ··· | COLUM_ID | ··· |
| DEPT | DEPTNO | NUMBER | ··· | 1 | ··· |
| DEPT | DNAME | VARCHAR2 | ··· | 2 | ··· |
| DEPT | LOC | VARCHAR2 | ··· | 3 | ··· |
| EMP | EMPNO | VARCHAR2 | ··· | 1 | ··· |
| EMP | ENAME | VARCHAR2 | ··· | 2 | ··· |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| EMP | DEPTNO | NUMBER | ··· | 8 | ··· |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

In Oracle, NUMERIC is called NUMBER, and VARCHAR2 is currently used instead of VARCHAR. Of course, Oracle understands the SQL-92 type names and internally translates them to its native types.

## Table Columns (2)

- The most important columns of COLS are:
  - TABLE_NAME, COLUMN_NAME: Identify the column.
  - COLUMN_ID: Column position (1,2,. . . ) in table.
  - DATA_TYPE: E.g., CHAR, VARCHAR2, NUMBER, DATE.
  - DATA_PRECISION, DATA_SCALE: For numeric types.

    DATA_PRECISION is the total number of decimal digits, DATA_SCALE the number of digits after the decimal point. For FLOAT, binary digits are counted in DATA_PRECISION, and DATA_SCALE is null.
  - CHAR_COL_DECL_LENGTH: Length of string types.
  - DATA_LENGTH: Maximum column length in bytes.
  - NULLABLE: “N” if “NOT NULL”, “Y” otherwise.

## Table Columns (3)

- E.g., list all columns of the table “DEPT”:

```
SELECT COLUMN_ID, COLUMN_NAME
FROM   COLS
WHERE  TABLE_NAME = 'DEPT'
ORDER  BY COLUMN_ID
```

- In SQL*Plus, the following command shows the columns of a table together with their types:

```
DESCRIBE ⟨Table⟩
```

- As can be expected, there are also `ALL_TAB_COLUMNS` and `DBA_TAB_COLUMNS`.

## Table Columns (4)

- In total, COLS has 25 columns.

  TABLE_NAME, COLUMN_NAME, DATA_TYPE, DATA_TYPE_MOD, DATA_TYPE_OWNER, DATA_LENGTH, DATA_PRECISION, DATA_SCALE, NULLABLE, COLUMN_ID, DEFAULT_LENGTH, DATA_DEFAULT, NUM_DISTINCT, LOW_VALUE, HIGH_VALUE, DENSITY, NUM_NULLS, NUM_BUCKETS, LAST_ANALYZED, SAMPLE_SIZE, CHARACTER_SET_NAME, CHAR_COL_DECT_LENGTH, GLOBAL_STATS, USER_STATS, AVG_COL_LEN.

  The sequence of columns is historically determined: Extensions at the end.

- Especially, COLS also contains statistical information about the columns that is used by the optimizer.

  E.g. NUM_DISTINCT contains the number of distinct column values. But this information is not kept current for performance reasons: Every transaction would need to lock these data. One must use e.g. the command "ANALYZE TABLE DEPT COMPUTE STATISTICS", to create or update the statistical information for the columns of DEPT (see below).

# Constraints (1)

- USER_CONSTRAINTS lists all constraints on tables that are owned by the current user.

| USER_CONSTRAINTS | | | | |
|---|---|---|---|---|
| OWNER | CONSTRAINT_NAME | CONSTRAINT_TYPE | TABLE_NAME | ··· |
| SCOTT | PK_DEPT | P | DEPT | ··· |
| SCOTT | SYS_C001293 | C | DEPT | ··· |
| SCOTT | PK_EMP | P | EMP | ··· |
| SCOTT | FK_DEPTNO | R | EMP | ··· |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

- The columns in a key etc. are listed in the table USER_CONS_COLUMNS, see below.

## Constraints (2)

- Columns of `USER_CONSTRAINTS` (slide 1/4):
  - `OWNER`: Owner of constraint definition.

    This seems to be always the same as the owner of the table. Even if user `A` gives the `ALTER` right on a table to user `B`, and user `B` adds a constraint, still `A` is listed as owner. Also, even `ALL_CONSTRAINTS` has not two owner columns (one for the table and one for the constraint).

  - `CONSTRAINT_NAME`: Name of the constraint.
  - `CONSTRAINT_TYPE`: E.g. "`P`" for primary key.

    The complete list of type codes is: `C` for a check constraint (includes `NOT NULL`), `P` for primary key, `U` for unique constraint, `R` for a foreign key, `V` for "with check option" in a view declaration, `O` for "with read only" in a view declaration.

## Constraints (3)

- Columns of USER_CONSTRAINTS (slide 2/4):
  - TABLE_NAME: Table on which constraint is defined.
  - SEARCH_CONDITION: Text of the CHECK-condition.

    NOT NULL constraints have "*A* IS NOT NULL".
  - R_OWNER and R_CONSTRAINT_NAME: Referenced key constraint (for foreign key constraints).

    I.e. in order to print the referenced table of a foreign key constraint, one needs to consider two rows in USER_CONSTRAINTS: One row (X) for the foreign key, and one (Y) for the referenced key.

    Y.TABLE_NAME is the result. Join condition: X.R_OWNER = Y.OWNER AND X.R_CONSTRAINT_NAME = Y.CONSTRAINT_NAME.
  - DELETE_RULE: CASCADE or NO ACTION.

## Constraints (4)

- Columns of `USER_CONSTRAINTS` (slide 3/4):
  - `STATUS`: `ENABLED` or `DISABLED`.

    `DISABLED` means that the constraint is not checked for new or modified rows. I.e. it is still in the data dictionary, but is currently not applied. `RELY` (see below) together with `ENABLED` would also mean that the constraint is not checked.

  - `DEFERRABLE`: `DEFERRABLE` or `NOT DEFERRABLE`.

    For a deferrable constraint, the user can choose to check it at the end of the transaction instead immediately after every statement.

  - `DEFERRED`: `IMMEDIATE` or `DEFERRED`.

    Default setting for checking this constraint (until the user specifies something different for his transaction: `SET CONSTRAINTS`). For a `NOT DEFERRABLE` constraint, it can only be `IMMEDIATE`.

## Constraints (5)

- Columns of `USER_CONSTRAINTS` (slide 4/4):
  - `VALIDATED`: `VALIDATED` or `NOT VALIDATED`.

    `VALIDATED` means that all data obeys the constraint. It is possible to change the constraint from `DISABLED` to `ENABLED` without validating the existing data.
  - `GENERATED`: `USER NAME` or `GENERATED NAME`.
  - `BAD`: Possible Year-2000 problem.
  - `RELY` (new in 8i): System assumes that the constraint is satisfied without actually checking it.

    This might be important for query optimization.
  - `LAST_CHANGE`: When was it enabled/disabled?

## Constraints (6)

- USER_CONS_COLUMNS: Columns of a key or foreign key, or referenced in CHECK/NOT NULL constraints.

| USER_CONS_COLUMNS | | | | |
|---|---|---|---|---|
| OWNER | CONSTRAINT_NAME | TABLE_NAME | COLUMN_NAME | POSITION |
| BRASS | PK_STUDENTS | STUDENTS | SID | 1 |
| BRASS | PK_RESULTS | RESULTS | SID | 1 |
| BRASS | PK_RESULTS | RESULTS | CAT | 2 |
| BRASS | PK_RESULTS | RESULTS | ENO | 3 |
| BRASS | FK_RES_STUD | RESULTS | SID | 1 |
| BRASS | FK_RES_EX | RESULTS | CAT | 1 |
| BRASS | FK_RES_EX | RESULTS | ENO | 2 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

## Constraints (7)

- Columns of `USER_CONS_COLUMNS`:
  - `OWNER`, `CONSTRAINT_NAME`: Identify the constraint.
  - `TABLE_NAME`: Table on which constraint is defined.

    Redundant: Same as in `USER_CONSTRAINTS`.
  - `COLUMN_NAME`: Column that participates in key, foreign key, or `CHECK`-constraint (includes `NOT NULL`).
  - `POSITION`: Sequence number of column in key.

    1 for the first column of a composed key or foreign key, 2 for the second, and so on. The column sequence is not necessarily the same as the sequence in the table (although that should be avoided). `POSITION` is null for `CHECK`-constraints.

## Constraints (8)

- E.g. print composed primary key of table "XYZ":

```
SELECT COL.POSITION, COL.COLUMN_NAME
FROM   USER_CONSTRAINTS CON,
       USER_CONS_COLUMNS COL
WHERE  CON.TABLE_NAME = 'XYZ'
AND    CON.CONSTRAINT_TYPE = 'P'
AND    CON.OWNER = COL.OWNER
AND    CON.CONSTRAINT_NAME = COL.CONSTRAINT_NAME
ORDER  BY COL.POSITION
```

- Exercise: Print all tables that contain a foreign key that references table "XYZ".

# Views

- USER_VIEWS contains the view-defining queries:

| USER_VIEWS | | | |
|---|---|---|---|
| VIEW_NAME | TEXT_LENGTH | TEXT | ··· |
| SALESMEN | 62 | SELECT ENAME, SAL+COMM AS SAL FROM EMP WHERE JOB = 'SALESMAN' | ··· |

The column TEXT contains the view-defining query. It has data type LONG (many restrictions, e.g. it cannot be input for string concatenation "||"). In SQL*Plus, use e.g. "SET LONG 10000" to see queries up to 10000 characters. TEXT_LENGTH is the string length of the query.

- COLS: Shows columns also of views.

- USER_DEPENDENCIES: Dependencies of views and procedures on tables etc. (tables used in a view).

## Synonyms

- Suppose user SCOTT creates a synonym with:

```
CREATE SYNONYM PRES FOR BRASS.PRESIDENTS
```

- USER_SYNONYMS (or SYN) list all synonyms that were created by the curent user:

| USER_SYNONYMS | | | |
|---|---|---|---|
| SYNONYM_NAME | TABLE_OWNER | TABLE_NAME | DB_LINK |
| PRES | BRASS | PRESIDENTS | |

- ALL_SYNONYMS lists all accessible synonyms.
- PUBLICSYN lists all public synonyms.

## Comments (1)

- It is possible to store some documentation about tables and columns in the data dictionary:

  ```
  COMMENT ON TABLE ⟨Table⟩ IS '⟨Text⟩'
  COMMENT ON COLUMN ⟨Table⟩.⟨Column⟩ IS '⟨Text⟩'
  ```

  These commands are Oracle-specific.

- USER_TAB_COMMENTS contains comments about own tables and views:

| USER_TAB_COMMENTS | | |
|---|---|---|
| TABLE_NAME | TABLE_TYPE | COMMENTS |
| EMP | TABLE | List of all employees |
| ⋮ | ⋮ | ⋮ |

## Comments (2)

- USER_COL_COMMENTS contains comments about the columns of one's own tables and views:

| USER_COL_COMMENTS | | |
|---|---|---|
| TABLE_NAME | COLUMN_NAME | COMMENTS |
| EMP | EMPNO | Employee number (ID) |
| ⋮ | ⋮ | ⋮ |

- All tables and all columns are listed.
  If no comment was stored, a null value appears in the column "COMMENTS".

  Comments can be up to 4000 characters long.

# References

- Ramez Elmasri, Shamkant B. Navathe: Fundamentals of Database Systems, 3rd Ed. Chapter 17: “Database System Architectures and the System Catalog”, Chapter 10: “Examples of Relational Database Management Systems: Oracle and Microsoft Access”
- Raghu Ramakrishnan, Johannes Gehrke: Database Management Systems, 2nd Ed. McGraw-Hill, 2000, ISBN 0-07-232206-3, 906 pages.
- H. Garcia-Molina, J. D. Ullman, J. Widom: Database System Implementation. Prentice Hall, ISBN 0130402648, 672 pages, ca. $60.00
- Jason S. Couchman: Oracle8i Certified Professional: DBA Certification Exam Guide with CDROM. Osborne/ORACLE Press, ISBN 0-07-213060-1, ca. 1257 pages, ca. $99.99.
- Jim Gray, Andreas Reuter: Transaction Processing: Concepts and Techniques. Morgan Kaufmann Publishers, 1993, ISBN 1-55860-190-2, 1070 pages, ca. $84.95
- Oracle 8i Concepts, Release 2 (8.1.6), Oracle Corporation, 1999, Part No. A76965-01.
- Oracle 8i Administrator’s Guide, Release 2 (8.1.6), Oracle Corporation, 1999, Part No. A76956-01.