Part 1: Introduction

References:

- Ramez Elmasri, Shamkant B. Navathe: Fundamentals of Database Systems, 3rd Edition, 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 Edition. 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
- Michael J. Corey, Michael Abbey, Daniel J. Dechichio, Ian Abramson: Oracle8 Tuning. Osborne/ORACLE Press, 1998, ISBN 0-07-882390-0, 608 pages, ca. $44.99.
- 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.
- Mark Gurry, Peter Corrigan: Oracle Performance Tuning, 2nd Edition (with disk). O'Reilly & Asociates, December 1996, ISBN 1565922379, 964 pages.
- 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.
- Oracle 8i Designing and Tuning for Performance, Release 2 (8.1.6), Oracle Corporation, 1999, Part No. A76992-01.

Objectives

After completing this chapter, you should be able to:

- enumerate tasks of the database administrator.
- create users in Oracle.
- explain the structure of the Oracle Data Dictionary.
- formulate queries against data dictionary tables.
- enumerate processes, files, memory structures of the Oracle architecture.
- start and stop the Oracle server, enumerate different system states of the server.

Overview

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 (∼ 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).

Phys. 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).

Phys. 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.

Phys. 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.

Phys. 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)

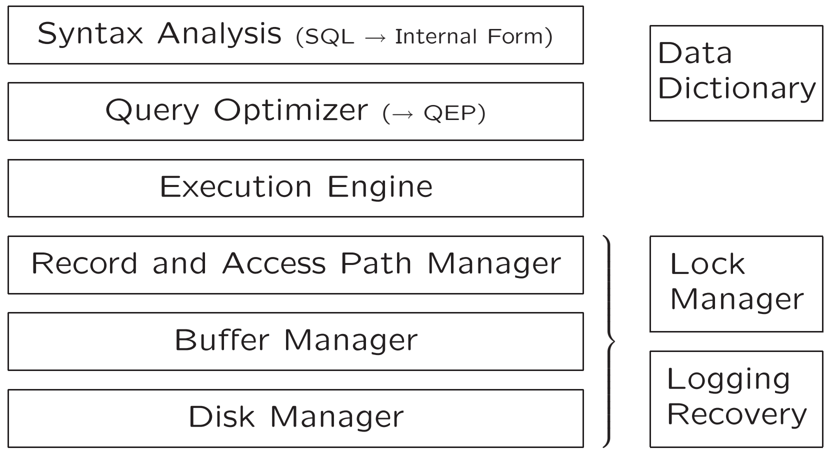
Consistency:

- 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



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.

Overview

1. Database Services, Tasks of the DBA
2. The Oracle Data Dictionary
3. Creating Users in Oracle
4. Oracle Architecture
5. Starting/Stopping Oracle

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://www-db.informatik.uni-hannover.de/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 OWNER='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 8i Reference.

  Note that there is an "Oracle 8i Reference" and an "Oracle 8i SQL Reference". Chapter 2 is about Static Data Dictionary Views, Chapter 3 about Dynamic Performance Views. For an introduction, see Chapter 2 of the Oracle 8i 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 8.1.6, it has 305 rows when queried as normal user, and 764 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, 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 8285 entries for the DBA, 6681 for normal users.

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. Sequence 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.

Overview

Object Privileges (1)

- Access rights in standard SQL are a set of triples: Who can execute which command on which table?

  ```
  GRANT  SELECT ON EMP  TO   SMITH
  REVOKE INSERT ON DEPT FROM MILLER
  ```

  Actually, they are quadruples: Who has given whom what right on which database object? This is important when rights are revoked.

- Rights can also be granted "`TO PUBLIC`".

  All users, including users created in future.

- Rights can be given "`WITH GRANT OPTION`".

  Then the grantee can grant the right to further users. The owner of a table (the user who created it) automatically holds all rights on it `WITH GRANT OPTION`. For views this is more complicated.

Object Privileges (2)

- USER_TAB_PRIVS: Grants on objects for which the current user is owner, grantor, or grantee.

USER_TAB_PRIVS					
GRANTEE	OWNER	TABLE_NAME	GRANTOR	PRIVILEGE	GRANTABLE
PUBLIC	SCOTT	DEPT	SCOTT	SELECT	N
SMITH	SCOTT	EMP	SCOTT	SELECT	Y
MILLER	SCOTT	EMP	SMITH	SELECT	N
SMITH	SCOTT	EMP	SCOTT	INSERT	N

I.e. all users have read access to the table DEPT. SMITH got read access to EMP WITH GRANT OPTION, and has given the right to MILLER. In addition, SMITH can append rows to EMP.
In contrast to other data dictionary tables, the prefix USER here does not mean that only tables owned by the current user are listed.

Object Privileges (3)

- Columns of USER_TAB_PRIVS:
  - ◇ OWNER and TABLE_NAME identify the table.
  - ◇ GRANTEE: The user who got the privilege.
  - ◇ GRANTOR: The user who gave the privilege.

    Because of the grant option, not only the owner can be grantor.
  - ◇ PRIVILEGE: The right, e.g. 'SELECT'.
  - ◇ GRANTABLE: 'YES' if right includes grant option.
- In this way, the SQL GRANT commands are stored in the data dictionary.

Object Privileges (4)

- USER_TAB_PRIVS_MADE is the subset of USER_TAB_PRIVS with OWNER=USER.
- USER_TAB_PRIVS_RECD is the subset of USER_TAB_PRIVS with GRANTEE=USER.
- The user might also have access to database objects because of grants to PUBLIC, which are not listed in these tables (but see ALL_TAB_PRIVS).

  Unless, of course, they are made by the current user or refer to tables of the current user. Otherwise, the name of the current user is neither OWNER, nor GRANTOR, nor GRANTEE, therefore the grant is not shown.

Object Privileges (5)

- The INSERT and UPDATE right can be given selectively for certain columns.

  An insert right for only part of the columns means that the other columns cannot be explicitly specified, and thus get their declared default value (or null).
- USER_COL_PRIVS: Grants that refer to single columns.

  USER_COL_PRIVS looks like USER_TAB_PRIVS, but has the additional column COLUMN_NAME.
- Grants for whole tables are not repeated here.
- USER_COL_PRIVS_MADE, USER_COL_PRIVS_RECD: Subsets with current user as owner/grantee (as above).

System Privileges (1)

- Commands like "`CREATE TABLE`" cannot be restricted with this standard security model.

  A user who is only supposed to enter data does not need to create new tables. For a secure system, every user should only be able to execute the commands he/she is supposed to execute.

- Therefore, Oracle has also "system privileges".

  Every major DBMS vendor has a to the problem (all different).

- In contrast to "object privileges", these refer to the execution of specific commands, not to DB objects.
- E.g. one needs the system privilege "`CREATE TABLE`" in order to be able to execute this command.

System Privileges (2)

- In order to log into Oracle, one needs the system privilege "`CREATE SESSION`".

  An account can be locked by not granting (or revoking) this privilege. It is still possible to access tables, views, etc. under this account via synonyms or "⟨User⟩.⟨Table⟩" (if one has the necessary access rights).

- Many system privileges are only for DBAs, e.g.:
  - ◇ "`SELECT ANY TABLE`" (read access to all tables),
  - ◇ "`DROP ANY TABLE`" (delete data of arbitrary users),
  - ◇ "`CREATE USER`" (create a new user).

System Privileges (3)

- Since the usual privileges of a DBA are separated into different system privileges, it is possible to have several DBAs with different responsibilities.

  Of course, one can still have one DBA with all privileges.

- There are currently more than 90 different system privileges.

  Basically, every administration command corresponds to a system privilege. Different kinds of `CREATE` commands also correspond to system privileges (since these commands could not be restricted otherwise). Most commands also have an `ANY`-version as a system privilege (allows one to apply the command to objects of any user). `CREATE ANY TABLE`: create tables in any schema.

System Privileges (4)

- If a user has a system privilege "`WITH ADMIN OPTION`", he/she can give it to other users:

  ```
  GRANT CREATE TABLE TO SCOTT
  ```

  Adding "`WITH ADMIN OPTION`" gives `SCOTT` the right to grant "`CREATE TABLE`", too.

- When a system privilege is revoked from a user A who had it "`WITH ADMIN OPTION`", privileges are not recursively revoked from users B who got it from A.

  This might be the reason why it was not called "`GRANT OPTION`". But it is very similar ("`GRANT OPTION`" can be used only for object privileges).

System Privileges (5)

- SYSTEM_PRIVILEGE_MAP: List of all system privileges.

SYSTEM_PRIVILEGE_MAP	
PRIVILEGE	NAME
⋮	⋮
-5	CREATE SESSION
⋮	⋮
-40	CREATE TABLE
⋮	⋮
-47	SELECT ANY TABLE
⋮	⋮

System Privileges (6)

- USER_SYS_PRIVS: System privileges granted to the current user or to PUBLIC.

  Columns are: USERNAME (always the name of the current user, not very useful), PRIVILEGE (name of the system privilege, no join with SYSTEM_PRIVILEGE_MAP necessary), ADMIN_OPTION (similar to grant option for object privileges).

- DBA_SYS_PRIVS: System privileges for each user.

  For DBA only. It has the columns GRANTEE, PRIVILEGE, ADMIN_OPTION.

- Only directly granted privileges are listed.

  Additional system privileges might have been granted via roles (see below). Therefore, USER_SYS_PRIVS is often empty, although the user actually has many system privileges.

Roles (1)

- It is difficult to grant privileges to many users one by one. In one way or another, all modern DBMS support groups of users with similar privileges.
- Oracle has the concept of "roles", which are sets of privileges that can be granted to users:

  ```
  CREATE ROLE MANAGEMENT
  ```

  This command requires DBA rights (system privilege "CREATE ROLE").

- Access rights are granted to a role in the same way as they are granted to a user:

  ```
  GRANT SELECT ON EMP TO MANAGEMENT
  ```


Roles (2)

- Roles can be granted to users (by their owner or users who got them WITH ADMIN OPTION):

  ```
  GRANT MANAGEMENT TO JIM, MARY
  ```

- When a user A is granted a role R, A receives all privileges that were or will be granted to R.

  But if "MANAGEMENT" is not one of the default roles of these users, which are automatically activated when they log in, they must explicity execute "SET ROLE MANAGEMENT" in every session in which they want to use these privileges. (This is not enforced in Oracle 8.0.) Roles can be protected by passwords. Then SET ROLE requires a password.

- If role A is granted to role B, B includes all rights of A. Thus, B is more powerful than A.

Roles (3)

- Several roles are predefined in Oracle 8, e.g.
  - ◇ CONNECT: Basic usage rights.

    This corresponds to the system privileges: CREATE SESSION, ALTER SESSION, CREATE DATABASE LINK, CREATE SYNONYM, CREATE TABLE, CREATE CLUSTER, CREATE VIEW, CREATE SEQUENCE.
  - ◇ RESOURCE: Rights for advanced users.

    This includes e.g. CREATE TABLE, CREATE PROCEDURE, CREATE TRIGGER. Students in this course were granted CONNECT and RESOURCE (but UNLIMITED TABLESPACE was revoked).
  - ◇ DBA: Right to do everything.
- In older Oracle versions, users were classified into these three types.

Roles (4)

- DBA_ROLES: List of all roles defined in the system.

  It has the columns ROLE, PASSWORD_REQUIRED. Only the DBA can create roles, and only the DBA can see the list of all roles.
- USER_ROLE_PRIVS: Roles granted to the current user.

  Roles granted to PUBLIC are also listed: All users have the rights included in such roles. Columns are: USERNAME, GRANTED_ROLE, ADMIN_OPTION, DEFAULT_ROLE, OS_GRANTED.
- DBA_ROLE_PRIVS: Which roles are granted to which user? Also role-to-role grants are shown.

  Columns: GRANTEE, GRANTED_ROLE, ADMIN_OPTION, DEFAULT_ROLE. GRANTEE can be a user or another role.

Roles (5)

- The following tables/views list the access rights included in roles accessible to the current user:
  - ◇ ROLE_ROLE_PRIVS: Roles implied by a role.

    Columns are: ROLE, GRANTED_ROLE, ADMIN_OPTION.
    All rights in GRANTED_ROLE are included in ROLE.
  - ◇ ROLE_SYS_PRIVS: System privileges in a role.

    Columns are: ROLE, PRIVILEGE, ADMIN_OPTION.
  - ◇ ROLE_TAB_PRIVS: Table privileges granted to roles.

    Columns are: ROLE, OWNER, TABLE_NAME, COLUMN_NAME (null if right for entire table), PRIVILEGE, GRANTABLE.

Creating Users (1)

User Authentication:

- Oracle can perform the user authentication itself. One must specify a user name and a password:

```
CREATE USER BRASS IDENTIFIED BY ABC_78
```

  Passwords have the same syntax as table names: They are not case-sensitive and "..." is needed to include special characters.
- Oracle can also rely on the authentication done by the operating system or a network service:

```
CREATE USER OPS$BRASS IDENTIFIED EXTERNALLY
```

  So when the UNIX user BRASS logs into Oracle (with empty username/password), he becomes the Oracle user OPS$BRASS.

Creating Users (2)

- A user created as explained above has no rights, not even the system privilege to connect to the DB.
- The necessary privileges can be given e.g. with:

```
GRANT CONNECT, RESOURCE TO BRASS
```

- After the GRANT, these roles can be made default roles, so that they are automatically activated when the user logs in:

```
ALTER USER BRASS DEFAULT ROLE ALL
```

It seems that roles without a password automatically become default roles (?). So this command might not be necessary.

Tablespaces and Quotas (1)

- A tablespace is a database file or a collection of DB files (storage space, container for tables).
- All tablespaces are listed in the system catalog table DBA_TABLESPACES.

  E.g. use "SELECT TABLESPACE_NAME FROM DBA_TABLESPACES" to list all tablespaces. This query must be executed by a DBA. All users have read access to USER_TABLESPACES (tablespaces that are accessible by the current user). The files for each tablespace are listed in DBA_DATA_FILES. It has e.g. the columns FILE_NAME, FILE_ID, TABLESPACE_NAME, BYTES. See also DBA_FREE_SPACE/USER_FREE_SPACE and DBA_FREE_SPACE_COALESCED.

- The tablespace "SYSTEM" contains e.g. the data dictionary (collection of system tables).

Tablespaces and Quotas (2)

- ```
  CREATE USER BRASS IDENTIFIED BY MY_PASSWORD
              DEFAULT TABLESPACE USER_DATA
              TEMPORARY TABLESPACE TEMPORARY_DATA
              QUOTA 2M ON USER_DATA
              QUOTA UNLIMITED ON TEMPORARY_DATA
  ```
- A tablespace can be defined when a table is created.

  Otherwise it is stored in the user's DEFAULT TABLESPACE (which is SYSTEM if it is not set in the CREATE USER).

- Without quota (and "UNLIMITED TABLESPACE"), the user cannot create tables on the tablespace.

  Use: REVOKE UNLIMITED TABLESPACE FROM BRASS

Data Dictionary: Users (1)

- ALL_USERS: List of all users, accessible by all users:
  - ◇ USERNAME: Name of the Oracle account.
  - ◇ USER_ID: Internal number of the account.
  - ◇ CREATED: Date/time when account was created.

ALL_USERS

USERNAME	USER_ID	CREATED
SYS	0	29-JAN-98
SYSTEM	5	29-JAN-98
SCOTT	20	29-JAN-98
BRASS	24	13-MAY-01
⋮	⋮	⋮
  ```

Data Dictionary: Users (2)

- DBA_USERS: Full information about all users. Only the DBA can look at this table.

  It has the following columns: USERNAME, USER_ID, PASSWORD (stored in encrypted form), DEFAULT_TABLESPACE, TEMPORARY_TABLESPACE, CREATED, PROFILE, ACCOUNT_STATUS (indicates whether account is locked, expired, or unlocked), LOCK_DATE, EXPIRY_DATE, INITIAL_RSRC_CONSUMER_GROUP, EXTERNAL_NAME.

- USER_USERS: Single row with information about the current user.

  It has the following columns: USERNAME, USER_ID, ACCOUNT_STATUS, LOCK_DATE, EXPIRY_DATE, DEFAULT_TABLESPACE, CREATED, EXTERNAL_NAME.

Data Dictionary: Quotas

- DBA_TS_QUOTAS: How many bytes/blocks on which tablespace are charged to which user, and what is the allowable maximum (quota)?

  Columns of this table are: TABLESPACE_NAME, USERNAME, BYTES, MAX_BYTES, BLOCKS, MAX_BLOCKS. The columns BYTES and MAX_BYTES are derived from the information in blocks.

- USER_TS_QUOTAS: The current and maximal file space usage of the current user.
- All table data is charged to the table owner (even if other users actually inserted the rows).

Some Predefined Users (1)

- SYS: Owner of the system tables (data dictionary).

  Most powerful account. Default password: CHANGE_ON_INSTALL.

- SYSTEM: The default database administrator.

  For most administration tasks. Default password: MANAGER.

- SCOTT: Guest and demonstration account.

  Default password: TIGER. Sometimes there are additional accounts used in tutorials: ADAMS, BLAKE, CLARK, JONES.

- OUTLN: Schema contains information for optimizer.

  Default password: OUTLN.

Some Predefined Users (2)

- DBSNMP: Information for the "intelligent agent".

  It is used for remote administration via the "enterprise manager". Default password: DBSNMP.

- One should check the list of users in the system table ALL_USERS and lock all users that are currently not needed (or change their passwords).

  There is also a table DBA_USERS with more information. The list of users created during the installation can change with new versions. Also, when one installs additional software (e.g. the Oracle application manager), more accounts are created.

- Hackers know all the default passwords!

Changing and Deleting Users

- If a user has forgotten his/her password:

  `ALTER USER BRASS IDENTIFIED BY NEW_PASSWORD`

- A user without tables can be deleted in this way:

  `DROP USER BRASS`

- To delete the user including all his/her data, use:

  `DROP USER BRASS CASCADE`

- The following command ensures that the user can no longer log in, but leaves his/her data untouched:

  `ALTER USER BRASS ACCOUNT LOCK`

External Password File

- Whereas the above passwords are stored in the database (encrypted), there usually is an additional file that contains passwords of administrators who need e.g. to start up the database.

  When the database is not running, passwords stored in the database cannot be accessed. If you use `CONNECT INTERNAL` in the server manager (`svrmgrl`) or `CONNECT SYS AS SYSDBA`, the default password is `ORACLE`. Actually, the `SYS` password in the password file and in the database can be different. The password file is generated by the `orapwd` utility program. Later, every user granted `SYSDBA`/`SYSOPER` rights is also stored in the password file. Instead of using a password file, you can use OS authentication. This depends on the parameter `REMOTE_LOGIN_PASSWORDFILE`.

Other Security Features (1)

- The resource usage of DB users can be restricted by creating a "profile" for them. This defines e.g.
  - ◇ How many concurrent sessions the user can have (number of windows with DB applications).
  - ◇ After what idle time he/she is logged off.
  - ◇ How much CPU time and how many logical reads (disk accesses) is allowed per session/per call.
  - ◇ After what time a password must be changed.
  - ◇ Which function is used to check the password complexity.

Other Security Features (2)

- Oracle also has an `AUDIT` command for defining which user actions are logged in system tables, so that one can later find out who did what.
  - ◇ E.g. all insertions should be logged that were executed (not refused):

    ```
    AUDIT INSERT ON SCOTT.EMP
    BY SESSION WHENEVER SUCCESSFUL;
    ```

    "`BY SESSION`" means that only one record is written for an entire session that did this operation (default). Alternative: "`BY ACCESS`".
  - ◇ E.g. log all unsuccessful login attempts:

    `AUDIT CONNECT WHENEVER NOT SUCCESSFUL;`

Overview

Oracle Files: Data Files (1)

- Oracle normally stores table data in standard operating system files.

  Windows e.g.: `C:\Oracle\ORADATA\orcl\System01.dbf`
  UNIX e.g.: `/ora8/oradata/ifidb/system01.dbf`

- Alternatively, Oracle can store the data on raw devices (direct disk access, not via the OS): Better performance, but more complicated administration.
- The files can only be processed by Oracle (no standard format, the format is also not documented).

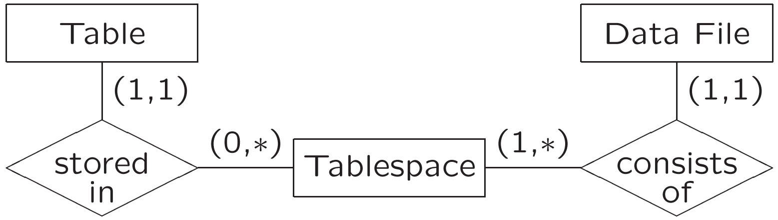
  Every DBMS vendor tries to beat the other vendors in performance benchmarks. Therefore, each vendor uses its own data structures.

Oracle Files: Data Files (2)

- Oracle does not use one file per table or per user.

  Any number of tables, indexes, etc. can be stored in the same file. Simplest case: Entire DB in a single file.

- The relationship between tables and data files is many to many (via tablespaces):



Oracle Files: Data Files (3)

- The data is not encrypted:
  - ◇ Persons who can access the data files can circumvent the Oracle access control.
  - ◇ OS access rights must be used so that only the DBA can access the data files.
- Data files can be autoextensible or have fixed size.

  In order to avoid fragmentation, data files are normally made large when they are created (the DBA can specify any size). Then Oracle manages the free space within them. If Oracle should ever run out of space, it can request more space from the operating system (make the file bigger) if the file was declared as autoextensible.

Oracle Files: Data Files (4)

- The data dictionary view `DBA_DATA_FILES` lists all files for storing table data. Columns are:
  - ◇ `FILE_NAME`: Filename with path.
  - ◇ `FILE_ID`: Numeric file identification.
  - ◇ `TABLESPACE_NAME`: Logical collection of data files.
  - ◇ `BYTES`, `BLOCKS`: Current file size.
  - ◇ `STATUS`: `AVAILABLE` or `INVALID` (not in use).
  - ◇ `RELATIVE_FNO`: File ID used in ROWIDs.
  - ◇ `AUTOEXTENSIBLE`: Oracle can make the file larger.
  - ◇ `MAXBYTES`, `MAXBLOCKS`: Limit for autoextension.
  - ◇ `INCREMENT_BY`: Step size for autoextension.

Oracle Files: Tempfiles (1)

- When large tables/query results must be sorted, Oracle needs temporary space on the disk.

  Small sorts are done in main memory.
- This is allocated in temporary segments.

  Temporary segments are also used for temporary tables.
- At the beginning, temporary segments were allocated in normal tablespaces, then Oracle introduced temporary tablespaces, and finally (8i) temporary datafiles (tempfiles) for temporary tablespaces.

Oracle Files: Tempfiles (2)

- Each variant of temporary data management is more efficient than the previous one, but all old variants are still supported.
- No backup copies of tempfiles are ever needed.

  Remember that the correctness of all information in this course is not guaranteed. You cannot sue me or my university for errors.
- Tempfiles are listed in
  - ◇ `DBA_TEMP_FILES`
  - ◇ `V$TEMPFILE`.
- Tempfiles typically have the extension “`.tmp`”.

Oracle Files: Control Files (1)

- When Oracle starts, it reads the names and locations of the datafiles from a “control file”.

  Windows e.g.: `C:\Oracle\ORADATA\orcl\Control01.ctl`.
  UNIX e.g.: `/ora8/oradata/ifidb/control01.ctl`.
- The control file contains also backup and recovery information.
- For safety reasons, there should normally be more than one control file (on different disks).

  If all copies of the controlfile are lost, the DBA is in big trouble.

Oracle Files: Control Files (2)

- `V$CONTROLFILE`: List of control files. Columns are:
  - ◇ `STATUS`: Normally null. Can be `INVALID`.
  - ◇ `NAME`: Path and name of the control file.
- Information from the control file is shown in, e.g.:
  - ◇ `V$DATAFILE`,
  - ◇ `V$DATABASE`,
  - ◇ `V$LOG`,
  - ◇ `V$LOG_HISTORY`.

Oracle Files: Redo Log (1)

- When data file blocks are updated, they are not immediately written back to the disk.

  A block is the unit of exchange between disk and main memory: E.g., the system reads and writes always 8 KB. For performance reasons, disk blocks from the data files are kept for some time in a main memory buffer, even when they were modified (delayed/lazy writing).

- However, all changes to the data files are logged in the redo log files.

  It is faster to write only the new/modified data to a sequential log file than to write all parts of the data files that are affected by the change. Sooner or later the data file must be written, but this can happen in the background when there is time. It is possible that the same block is changed several times before to is written to the disk.

Oracle Files: Redo Log (2)

- The redo log files are needed for transaction processing (Recovery after a system crash.)

  Windows e.g.: `C:\Oracle\ORADATA\orcl\Redo01.log`.
  UNIX e.g. `/ora8/oradata/ifidb/redo01.log`

- Since the redo log files are so important for recovery, one usually has two copies of every log file.

  Of course, they should be on different disks.

- The log files that are copies of each other are called a log file group.

  Oracle automatically writes the same information to all files in a log file group. In contrast, Oracle does not manage copies of data files.

Oracle Files: Redo Log (3)

- When the changes that are documented in the log file are reflected in the data files it can be overwritten with new changes.

  However, if one wants to be protected against loss of data files (e.g., because of a disk failure), one needs to keep all redo log files since the last backup of the data files. This is done by copying redo log files to an "archive destination" (a tape or a slower disk) before they are overwritten. If Oracle is put into "`ARCHIVELOG`" mode (this is not the default), it automatically ensures that the redo log files are archived before they are overwritten. In order to distinguish the main redo log files from their archived copies, they are called the "online log". More information will be given later (Backup&Recovery).

Oracle Files: Redo Log (4)

- Redo log files are reused in a cyclic way, e.g. the output first goes to group 1, then to group 2, then to group 3, and then again to group 1.

  Every Oracle instance needs at least two log files.

Group 1	Group 2	Group 3
File: /disk1 /redo01.log	File: /disk1 /redo02.log	File: /disk1 /redo03.log
File: /disk2 /redo01.log	File: /disk2 /redo02.log	File: /disk2 /redo03.log

Oracle Files: Redo Log (5)

- `V$LOGFILE`: List of all log files.
  - ◇ `GROUP#`: Logfiles with the same group number are copies of each other (for safety reasons).
  - ◇ `STATUS`: Usually null.

    It can be also be `INVALID` (file is inaccessible), `STALE` (file contens are incomplete), `DELETED` (file is no longer used).
  - ◇ `MEMBER`: File name of the log file.
- `V$LOG` contains information about the log file groups and their contents.

  Columns: `GROUP#`, `THREAD#`, `SEQUENCE#`, `BYTES`, `MEMBERS`, `ARCHIVED`, `STATUS`, `FIRST_CHANGE#`, `FIRST_TIME`.

Oracle Files: Parameters (1)

- When Oracle is started, it reads an initialization parameter file.

  Windows e.g.: `C:\Oracle\Admin\orcl\pfile\init.ora`
  UNIX e.g.: `/ora8/product/8.1.6/admin/ifidb/pfile/initifi.ora`
  Note that it might point to/include another file (`PFILE=...`).
- It contains the settings of important tuning parameters, as well as the location of the control files.
- Traditionally, the initialization parameter file was a standard ASCII file that could be viewed and modified with any text editor.

Oracle Files: Parameters (2)

- Beginning with Oracle 9i, "Server Parameter Files" are used that are binary files and can be changed only by Oracle.

  Traditional text files are still supported and it is possible to map in both directions with `CREATE SPFILE` and `CREATE PFILE`. Parameters are changed with `ALTER SYSTEM` (some also with `ALTER SESSION`). The advantage of server parameter files is that Oracle can automatically modify for `ALTER SYSTEM` commands.
- Some parameters can be set only when the DBMS starts up, other parameters can be modified while the DBMS is running, and some parameters can be set separately for each user session.

Oracle Files: Parameters (3)

- `V$PARAMETER`: Settings of the initialization parameters that are in effect for the current session.

  Columns: `NUM`, `NAME`, `TYPE`, `VALUE`, `ISDEFAULT`, `ISSES_MODIFIABLE` (i.e. this parameter can be set separately for each session), `ISSYS_MODIFIABLE` (i.e. this parameter can be modified while the DBMS is running), `ISMODIFIED`, `ISADJUSTED`, `DESCRIPTION`, `UPDATE_COMMENT`. There is also a view `V$PARAMTER2` that displays list-valued parameters differently (one row per list element), and `V$SYSTEM_PARAMETER/...2` that contain the global values (inherited by each session when it starts).

- E.g. location of the control files:

```
SELECT VALUE FROM V$PARAMETER
WHERE  NAME = 'control_files'
```

- SQL*Plus has a command `SHOW PARAMETER` X.

Oracle Files: ALERT File (1)

- The alert file of the DB contains information about:
  - ◇ each time the server is started or stopped,

    At startup, the size of the shared memory areas and the started background processes are shown.

  - ◇ important administrative operations,

    For instance, adding files to the database.

  - ◇ errors.

    E.g., the archive log disk is full, therefore soon no more updates will be possible (when the online log files are used up).

Oracle Files: ALERT File (2)

- The DBA should carefully monitor the alert file.

  Windows, e.g.: `C:\Oracle\Admin\orcl\Bdump\orclALRT.log`
  UNIX, e.g.: `/ora8/product/8.1.6/admin/ifidb/bdump/alert_ifi.log`

- Some DBAs always have a window open that shows the last lines of the alert file.

  Of course, the Oracle Enterprise Manager also has a window that shows the current status of the database. One can also specify that when certain events happen, an email or SMS is automatically sent.

- In addition, each Oracle process has a trace file.

  Server processes only if the parameter `SQL_TRACE` is set to `TRUE`.

Oracle Architecture (1)

- Clients (such as SQL*Plus or an application program written in Pro*C) connect over the network to the DB server.
- The server part of Oracle consists of several processes and a shared memory area (SGA = "System Global Area" or "Shared Global Area").
- This server part is called an Oracle instance.

  Normally, an instance is synonymous to database. But with "Oracle real application clusters" it is also possible that several instances (using different CPUs) access the same database. Of course, it is possible to have several Oracle instances running on the same machine (managing different databases).

Oracle Architecture (2)

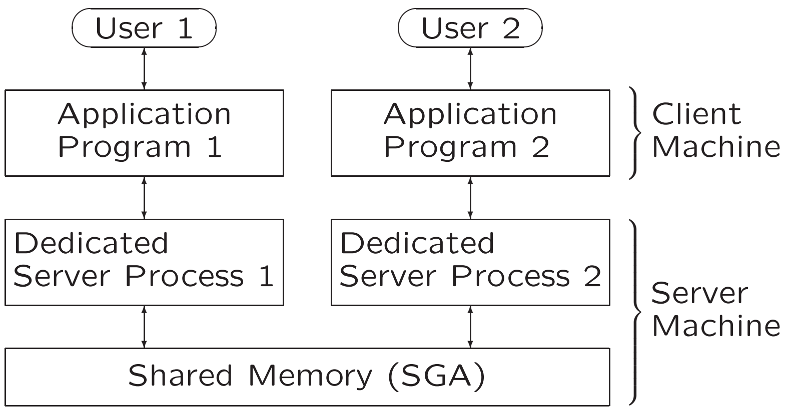
- Oracle supports two architectural variants:
  - ◇ Dedicated server architecture: One server process per client (classical/old architecture).

    The process is started when the client connects to Oracle and terminated when it logs off. Problem: The process is idle most of the time, but still binds resources (memory).
  - ◇ Multithreaded server architecture (MTS): There is a pool of server processes and a dispatcher which sends client requests to one of the servers.

    This is advantageous when a large number of concurrent clients must be served.

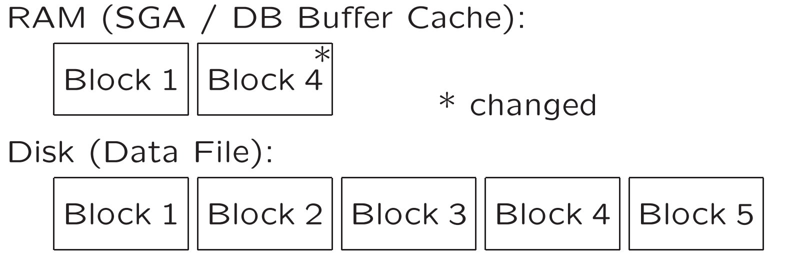
Oracle Architecture (3)

Dedicated Server Architecture:



Oracle Architecture (4)

- Part of the SGA is the DB buffer cache. It stores recently accessed DB blocks (containing e.g. table data).



- Accesses to RAM are much faster than accesses to disk, but the RAM is usually smaller (and volatile).

Oracle Architecture (5)

- When a server process needs a block from a datafile, it first checks whether the block is already in the DB buffer cache.
- If the block is already in the buffer cache, the server process can directly access the data there.

  Unless it is locked, of course.
- If not, the server process allocates a free buffer and reads the block into that buffer.

  The block then remains some time in the buffer, in case it is needed again. Buffering is explained in more detail in Chapter 2.

Oracle Architecture (6)

- Even if the server process changes the block in the buffer, it does not write the block back to the disk.
- The "DB writer" background processes (DBW0, DBW1, . . .) save changed blocks from the DB buffer cache periodically back to the disk.

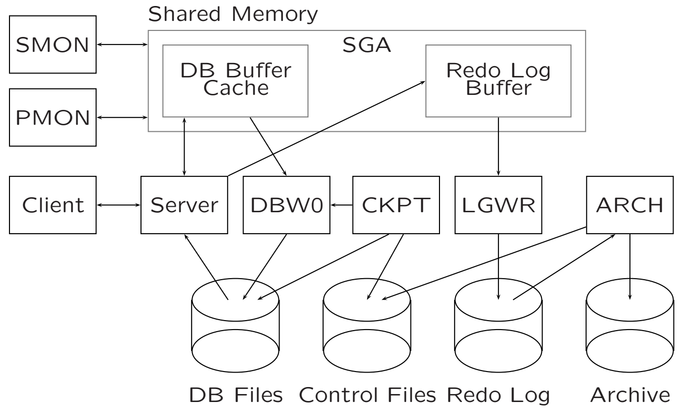
  Because of the delayed writing, it might be that the block is changed many times before it is finally written back.
- Oracle uses a set of 5 or more background processes (besides the server processes).

Oracle Architecture (7)

- Another part of the SGA is the Redo Log Buffer. It stores a transcript of all changes to DB blocks.

  This information is needed e.g. if the system should crash before the DB Writer saved a block back to disk. The Log entries are written to disk at commit time (or earlier).
- The log writer process (LGWR) saves the entries from the log buffer to the current redo log file.

Oracle Architecture (8)



Oracle Architecture (9)

More Background Processes:

- Checkpoint Process (`CKPT`)

  CKPT ensures that from time to time all changed blocks are written back to disk so that older log files are only needed in case of disk failures, but not for "normal" system crashes.
  It calls the DB writer process and updates the controlfiles. A checkpoint is set (minimally) every time a log file becomes full. One can configure more frequent checkpoints (`init.ora`).
- Archiver Process (`ARCH`)

  ARCH writes full redo log files to tape storage or another archive location (if the DB runs in ARCHIVELOG mode). If the data files should be damaged, all redo log files generated since the last backup are needed.

Oracle Architecture (10)

Oracle Background Processes, Continued:

- System Monitor Process (`SMON`)

  SMON performs recovery after a system crash and does some clean-up tasks regularly. E.g. it merges contiguous free extents etc.

- Process Monitor Process (`PMON`)

  PMON performs clean-up tasks when a user process fails (e.g. remove its locks).

Oracle Instance:

- An Oracle instance consists of an SGA (system global area) and a set of background processes.

Oracle Architecture (11)

Main Networking Process:

- Listener

  The network listener process accepts connections from clients and creates dedicated server processes. It can also manage pre-spawned server processes.

Processes for Multithreaded Servers:

- Dispatcher Processes (`Dnnn`)

  In the multithreaded configuration, these processes distribute client requests over shared DB servers.

- Shared Server Processes (`Snnn`)

  DB Server for the multithreaded configuration.

Oracle Architecture (12)

Processes for Distributed Databases:

- Recoverer Process (`RECO`)

  This process connects to other nodes in a distributed DB to resolve in-doubt transactions (no final COMMIT/ABORT).

- Job Queue Processes (`SNPn`)

  These processes automatically update table snapshots in a distributed database.

Processes for Oracle Parallel Server:

- Lock Process (`LCK0`)

  Manages locks between different Oracle instances (each instance has its own SGA and background processes).

Oracle Architecture (13)

Memory Structures:

- Besides the DB buffer cache and the redo log buffer, the SGA (System Global Area) contains also the "Shared Pool".
- The shared pool contains e.g.
  - ◇ the "row cache" (for data dictionary entries).

    It seems that caching single rows is more efficient for the data dictionary than caching entire blocks.

  - ◇ the library cache.

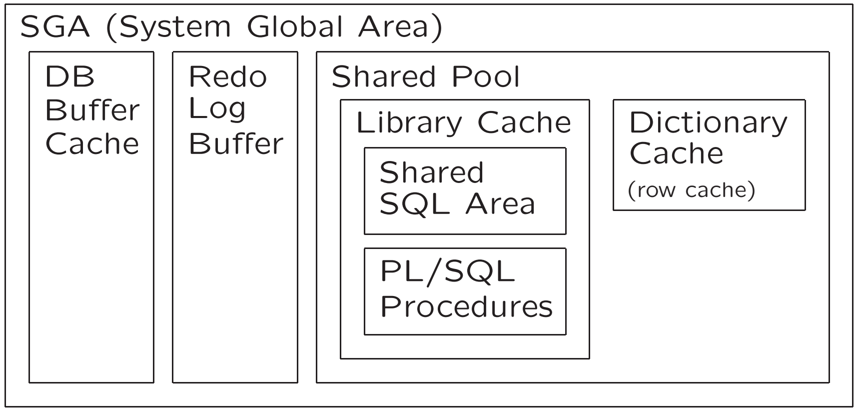
Oracle Architecture (14)

Memory Structures, continued:

- The library cache contains e.g.
  - ◇ the Shared SQL Area (here recently executed SQL statements together with the query evaluation plans are stored).

    This is a cache for query evaluation plans: If the same SQL statement (e.g., from an application program) is executed again and again (possibly with different parameter values), the (relatively expensive) query optimization does not have to be repeated.
  - ◇ Stored procedures/packages in compiled form.

Oracle Architecture (15)



Oracle Architecture (16)

Program Global Area (PGA):

- The PGA is memory that is allocated inside the dedicated server process (i.e. not shared).
- It contains e.g.
  - ◇ Stack area (session-specific variables, arrays, ...)
  - ◇ Private SQL areas (bind information, runtime buffers, etc.)
- The private SQL areas contain also the sort areas.

  Sorting can run faster with more memory. Only the retained portion of the sort area is part of the private SQL area.

Oracle Architecture (17)

Remark about Multithreaded Server:

- In the multithreaded server configuration, the private SQL areas are allocated in the SGA (in the shared pool).

  That means the size of the SGA must be increased when changing from the dedicated server configuration to the multithreaded server configuartion.

Oracle Architecture (18)

Related Information in the Data Dictionary:

- V$PROCESS: List of the Oracle processes.
- V$BGPROCESS: Description of background processes.
- V$BUFFER_POOL: Size of DB buffer cache.

  Oracle can have different "buffer pools" with different replacement strategies. "BUFFERS" is the number of buffer frames. The buffer pools can be segmented into multiple sets for multiprocessor systems. Buffering is explained in more detail in the next chapter.

Oracle Architecture (19)

Related Information in the Data Dictionary, continued:

- V$RESOURCE_LIMIT: Sizes of some arrays in the SGA.

  E.g. the SGA contains an array for the currently active transactions. V$RESOURCE_LIMIT also reports the use of these resources, e.g. how many transactions are currently active or were ever active concurrently (since instance startup).

- V$SGA: Size of the components of the SGA.

  E.g. it contains the size of the DB buffer pool in bytes.

- V$SGASTAT: Detailed information about the size of memory structures, listing components of the shared pool individually.

Initialization Parameters

- DB_BLOCK_SIZE: Size (in bytes) of a single DB block.
- DB_BLOCK_BUFFERS: Number of buffer frames.

  So the total memory needed is DB_BLOCK_BUFFERS * DB_BLOCK_SIZE.

- LOG_BUFFER: Size of the redo log buffer (in bytes).
- SHARED_POOL_SIZE: Size (in bytes) of the cache for data dictionary rows, query evaluation plans, etc.
- DB_FILE_MULTIBLOCK_READ_COUNT: Number of blocks that are read in one OS call during full table scans.
- There are more than 200 such parameters.

Overview

Controlling Oracle (1)

- The DB server can be running (i.e. executing queries and updates) or not running (shut down).

  There are also several intermediate/restricted states, which are necessary for DB creation or recovery, or certain administrative operations.

- One of the tasks of the DBA is to control the availability of the database.
- For adminitrative operations (such as DB startup), the DBA should log into Oracle as follows:

```
UNIX> sqlplus /nolog
SQL> CONNECT SYS AS SYSDBA
```


Controlling Oracle (2)

- "`/nolog`" means that no connection to a database is opened (e.g. when the database is not yet started).

  This might not be necessary, since when one logs in "`AS SYSDBA`", one gets only a warning when the database is not yet started.

- "`AS SYSDBA`" gives special administrative rights, e.g. the right to startup the database or shut it down.

  "`SYSDBA`" is a system privilege in Oracle, but it is can also be viewed as a special type of connection to the database. In other words, it is a privilege that must be explicitly activated. While `SYS` can only log in `AS SYSDBA`, other users who were granted the `SYSDBA` right can choose whether they log in `AS SYSDBA` or as a normal user. There is also a weaker right called `SYSOPER`, which permits to start up or shut down the database, but does not permit to view all data in the database.

Controlling Oracle (3)

- As long as the DBMS does not run, the passwords for `SYS` etc. stored in the DB cannot be accessed.
- There are two possibilities to authenticate DBAs:
  - ◇ OS Authentication: Members of an operating system group called `dba` may start/stop the DB.

    E.g. the OS user who owns the Oracle programs and files (usually `oracle`). On other systems, only the OS administrator (`root`) may start/stop the DB.

  - ◇ Password File: There is a password file in addition to the passwords stored in the database.

    It contains the passwords of all users who have been granted the `SYSDBA` or `SYSOPER` system privileges.

Controlling Oracle (4)

- The password file is e.g. stored in (OS dependent):

```
C:\orawin95\database\pwdorcl.ora
$ORACLE_HOME/dbs/orapw$ORACLE_SID
```

- The program "`orapwd`" creates the password file:

```
orapwd FILE=orapworcl PASSWORD=nina ENTRIES=5
```

- `PASSWORD` is the initial password for `SYS`.
- `ENTRIES` is the maximal number of users with `SYSDBA` or `SYSOPER` rights (i.e. with passwords in this file).

  The file has a fixed size. Whenever a user is granted `SYSDBA` or `SYSOPER`, an entry in the password file is created.

Controlling Oracle (5)

- If one forgot the password: Delete or rename the password file and recreate it.

  The DB must be restarted. Afterwords one can log in `AS SYSDBA` and change the password in the DB (the two `SYS` passwords do not agree).

- A password file is used if the initialization parameter `REMOTE_LOGIN_PASSWORD_FILE` is set to `EXCLUSIVE`.

  It is `NONE` by default which means OS authentication.

- On Windows, a DBA password is stored in the registry. `DBA_AUTHORIZATION` in `HKEY_LOCAL_MACHINE/SOFTWARE/ORACLE`.

  This is used for the automatic start and stop of the DBMS.

Controlling Oracle (6)

- All users who connect `AS SYSDBA` are mapped to `SYS` when they access the DB.

  It might be a surprise that one's own tables are not available when one works in this mode. Users who work `AS SYSOPER` are mapped to `PUBLIC`.

- The DBA usually works while logged into the server machine.

  Of course, he/she can use `ssh` to log into the server from somewhere else. It is possible to do administration remotely (with SQL*Plus running on the client), but normally the network connection is not very secure. Also, unless one uses the new server parameter files, a copy of the initialization parameter file must be available on the client.

Starting Oracle (1)

- Starting Oracle proceeds in three steps. Normally, they are automatically done one after the other, but it is possible to to stop in an intermediate state:
  - ◇ The instance is started.

    The parmeter file is read, the SGA is allocated, the background processes are started. In this state a new DB can be created.

  - ◇ The database is mounted.

    The control file is read, but the data files are not yet open. E.g. data files can be renamed in this state.

  - ◇ The database is open.

    The datafiles and log files are open, and the database is available for normal operations.

Starting Oracle (2)

- The commands for starting Oracle in one of the above states are:
  - ◇ `STARTUP NOMOUNT`: Only processes started.
  - ◇ `STARTUP MOUNT`: Only controlfiles open.
  - ◇ `STARTUP` or `STARTUP OPEN`: Full startup.
- It is possible to add the keyword `FORCE`, this e.g. kills processes remaining from a previous instance.
- `STARTUP RESTRICT` opens the database, but allows only users with the `RESTRICTED SESSION` privilege to connect (e.g. only DBAs).

Starting Oracle (3)

- One can explicitly specify the parameter file and/or the DB name:

  `STARTUP OPEN ifi PFILE=initifi.ora`

  The default `PFILE` is `$ORACLE_HOME/dbs/init$ORACLE_SID.ora` on UNIX, and `%ORACLE_HOME%\database\initORCL.ora` on Windows. However, in starting in Oracle 9i, the default is to use a server parameter file. If one uses a traditional text parameter file, one must specify `PFILE=...`.

- A database in mounted state can be made available for users with

  `ALTER DATABASE OPEN`

Shutting Oracle Down (1)

- The shutdown proceeds in the same three steps as the startup (in inverse order):
  - ◇ The database is closed.

    The contents of the DB buffer cache and the redo log buffer is written to disk and the files are closed. The control files remain open.
  - ◇ The database is dismounted.

    The control files are closed.
  - ◇ The instance is shut down.

    The background processes are terminated, the SGA memory is given back to the operating system.

Shutting Oracle Down (2)

- There are four different shutdown modes:
  - ◇ `SHUTDOWN NORMAL`

    The shutdown waits until all users logged off from Oracle. No new users can log into Oracle.
  - ◇ `SHUTDOWN TRANSACTIONAL`

    The shutdown waits until all active transactions are finished. No new transactions can be started.
  - ◇ `SHUTDOWN IMMEDIATE`

    All active transactions are rolled back. Then all buffers are written and the shutdown proceeds normally.
  - ◇ `SHUTDOWN ABORT`

    The Oracle processes are killed. Recovery will be needed.

Killing Sessions (1)

- All currently active sessions (i.e. users that are logged in) are listed in `V$SESSION`. Columns are, e.g.:
  - ◇ `SADDR`: Memory address of session data in DBMS.
  - ◇ `SID`: Session identifier.
  - ◇ `SERIAL#`: Serial number of the session.

    The `SID` is reused for another session when one user logs out and the next logs in. The serial number is added to make it unique.
  - ◇ `USERNAME`: Oracle user name.
  - ◇ `OSUSER`: Operating system user name.
  - ◇ `PROGRAM`: Operating system program name.

Killing Sessions (2)

- Before the DBA shuts down the system or kills a session, he/she might check whether transactions are currently running.

  A transaction is started at the first INSERT/UPDATE/DELETE and ends with ROLLBACK or COMMIT. I.e. if the session should be killed, changes are rolled back (a user will have to enter data again).

- V$TRANSACTION lists the currently active transactions. Attributes are, e.g.,:
  - SES_ADDR: Memory address of the session data.
  - START_TIME: Time when the transaction started.

Killing Sessions (3)

- Exercise: Write a query to list the users who have currently active transactions.
- V$TRANSACTION_ENQUEUE lists the locks hold by transactions.

  One attribute is the SID of the session that holds the lock. CTIME is the time since the transaction holds the lock (in seconds). BLOCK is 1 if another transaction waits for the lock. V$LOCK lists all locks and lock requests (including system locks: The list is quite long).

- The DBA can kill a session with the command:

```
ALTER SYSTEM KILL SESSION '8,5948'
```

  8 is the SID and 5948 is the SERIAL# of the session.