1. Introduction: Basics of Database Administration

DBMS Architecture: Example

Syntax Analysis (SQL $\rightarrow$ Internal Form)
Query Optimizer ($\rightarrow$ QEP)
Execution Engine
Record and Access Path Manager
Buffer Manager
Disk Manager

Data Dictionary

Lock Manager
Logging Recovery

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.
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:
  
  ```sql
  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:

<table>
<thead>
<tr>
<th>TABLE_NAME</th>
<th>TABLE_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPT</td>
<td>TABLE</td>
</tr>
<tr>
<td>EMP</td>
<td>TABLE</td>
</tr>
</tbody>
</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. Sequences 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":

<table>
<thead>
<tr>
<th>OWNER</th>
<th>TABLE_NAME</th>
<th>TABLE_TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCOTT</td>
<td>BONUS</td>
<td>TABLE</td>
</tr>
<tr>
<td>SCOTT</td>
<td>DEPT</td>
<td>TABLE</td>
</tr>
<tr>
<td>SYS</td>
<td>USER_CATALOG</td>
<td>VIEW</td>
</tr>
<tr>
<td>PUBLIC</td>
<td>USER_CATALOG</td>
<td>SYNONYM</td>
</tr>
<tr>
<td>PUBLIC</td>
<td>CAT</td>
<td>SYNONYM</td>
</tr>
<tr>
<td>SYS</td>
<td>ALL_CATALOG</td>
<td>VIEW</td>
</tr>
<tr>
<td>PUBLIC</td>
<td>ALL_CATALOG</td>
<td>SYNONYM</td>
</tr>
</tbody>
</table>

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
  
  ```sql
  SELECT * FROM SCOTT.DEPT
  ```
1. Introduction: Basics of Database Administration

**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
  ```sql
  SELECT * FROM SYS.ALL_CATALOG WHERE OWNER = 'SCOTT'
  ```
- However, Oracle has introduced synonyms (abbreviations, macros) to simplify this. E.g., try
  ```sql
  CREATE SYNONYM DEPARTMENTS FOR SCOTT.DEPT
  ```
  Then "SELECT * FROM DEPARTMENTS" means actually
  ```sql
  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`.

**Data Dictionary (1)**

- `DICT` lists all data dictionary tables/views:

<table>
<thead>
<tr>
<th>TABLE_NAME</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL_CATALOG</td>
<td>All tables, views, synonyms, sequences accessible to the user Tables, Views, Synonyms and Sequences owned by the user Description of data dictionary tables and views Description of columns in data dictionary tables and views Synonym for DICTIONARY !</td>
</tr>
</tbody>
</table>
1. Introduction: Basics of Database Administration

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.

Stefan Brass: Datenbanken II Universit"at Halle, 2005

Data Dictionary (3)

- E.g. this query prints all data dictionary objects containing "CAT" in their name:
  ```sql
  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):
  ```sql
  COLUMN TABLE_NAME FORMAT A25
  COLUMN COMMENTS FORMAT A50 WORD WRAP
  SET PAGESIZE 100
  ```

Stefan Brass: Datenbanken II Universit"at Halle, 2005

Data Dictionary (4)

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

```
<table>
<thead>
<tr>
<th>TABLE_NAME</th>
<th>COLUMN_NAME</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DICT</td>
<td>TABLE_NAME</td>
<td>Name of the object</td>
</tr>
<tr>
<td>DICT</td>
<td>COMMENTS</td>
<td>Text comment on the object</td>
</tr>
<tr>
<td>DICT_COLUMNS</td>
<td>TABLE_NAME</td>
<td>Name of the object that contains the column</td>
</tr>
<tr>
<td>DICT_COLUMNS</td>
<td>COLUMN_NAME</td>
<td>Name of the column</td>
</tr>
<tr>
<td>DICT_COLUMNS</td>
<td>COMMENTS</td>
<td>Text comment on the object</td>
</tr>
</tbody>
</table>
```

It has 13375 entries for the DBA, 10554 for normal users.

Stefan Brass: Datenbanken II Universit"at Halle, 2005

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:

```
<table>
<thead>
<tr>
<th>OBJECT_NAME</th>
<th>OBJECT_TYPE</th>
<th>CREATED</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPT</td>
<td>TABLE</td>
<td>29-JAN-98</td>
</tr>
<tr>
<td>PK_DEPT</td>
<td>INDEX</td>
<td>29-JAN-98</td>
</tr>
<tr>
<td>EMP</td>
<td>TABLE</td>
<td>29-JAN-98</td>
</tr>
<tr>
<td>PK_EMP</td>
<td>INDEX</td>
<td>29-JAN-98</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Stefan Brass: Datenbanken II Universit"at Halle, 2005
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?
  
  ```sql
  SELECT CREATED
  FROM OBJ
  WHERE OBJECT_NAME = 'EMP'
  ```

- To see also the time, select the following:
  
  ```sql
  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:

<table>
<thead>
<tr>
<th>TABLE_NAME</th>
<th>COLUMN_NAME</th>
<th>DATA_TYPE</th>
<th>COLUMN_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPT</td>
<td>DEPTNO</td>
<td>NUMBER</td>
<td>1</td>
</tr>
<tr>
<td>DEPT</td>
<td>DNAME</td>
<td>VARCHAR2</td>
<td>2</td>
</tr>
<tr>
<td>DEPT</td>
<td>LOC</td>
<td>VARCHAR2</td>
<td>3</td>
</tr>
<tr>
<td>EMP</td>
<td>EMPNO</td>
<td>VARCHAR2</td>
<td>1</td>
</tr>
<tr>
<td>EMP</td>
<td>ENAME</td>
<td>VARCHAR2</td>
<td>2</td>
</tr>
<tr>
<td>EMP</td>
<td>DEPTNO</td>
<td>NUMBER</td>
<td>8</td>
</tr>
</tbody>
</table>

  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.

Constraints (1)

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

<table>
<thead>
<tr>
<th>OWNER</th>
<th>CONSTRAINT_NAME</th>
<th>CONSTRAINT_TYPE</th>
<th>TABLE_NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>SCOTT</td>
<td>PK_DEPT</td>
<td>P</td>
<td>DEPT</td>
</tr>
<tr>
<td>SCOTT</td>
<td>SYS_C001293</td>
<td>C</td>
<td>DEPT</td>
</tr>
<tr>
<td>SCOTT</td>
<td>FK_EMP</td>
<td>P</td>
<td>EMP</td>
</tr>
<tr>
<td>SCOTT</td>
<td>FK_DEPTNO</td>
<td>R</td>
<td>EMP</td>
</tr>
</tbody>
</table>

- The columns in a key etc. are listed in the table ```USER_CONS_COLUMNS```.

<table>
<thead>
<tr>
<th>Owner</th>
<th>Constraint_Name</th>
<th>Table_Name</th>
<th>Column_Name</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRASS</td>
<td>PK_STUDENTS</td>
<td>STUDENTS</td>
<td>SID</td>
<td>1</td>
</tr>
<tr>
<td>BRASS</td>
<td>PK_RESULTS</td>
<td>RESULTS</td>
<td>SID</td>
<td>1</td>
</tr>
<tr>
<td>BRASS</td>
<td>PK_RESULTS</td>
<td>RESULTS</td>
<td>CAT</td>
<td>2</td>
</tr>
<tr>
<td>BRASS</td>
<td>PK_RESULTS</td>
<td>RESULTS</td>
<td>ENO</td>
<td>3</td>
</tr>
<tr>
<td>BRASS</td>
<td>FK_RES_STUD</td>
<td>RESULTS</td>
<td>SID</td>
<td>1</td>
</tr>
<tr>
<td>BRASS</td>
<td>FK_RES_EX</td>
<td>RESULTS</td>
<td>CAT</td>
<td>1</td>
</tr>
<tr>
<td>BRASS</td>
<td>FK_RES_EX</td>
<td>RESULTS</td>
<td>ENO</td>
<td>2</td>
</tr>
</tbody>
</table>
**Views**

- **USER_VIEWS** contains the view-defining queries:

<table>
<thead>
<tr>
<th>VIEW_NAME</th>
<th>TEXT_LENGTH</th>
<th>TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SALESMEN</td>
<td>62</td>
<td>SELECT ENAME, SAL+COMM AS SAL FROM EMP WHERE JOB = 'SALESMAN'</td>
</tr>
</tbody>
</table>

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).

**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.

<table>
<thead>
<tr>
<th>USERNAME</th>
<th>USER_ID</th>
<th>CREATED</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYS</td>
<td>0</td>
<td>29-JAN-98</td>
</tr>
<tr>
<td>SYSTEM</td>
<td>5</td>
<td>29-JAN-98</td>
</tr>
<tr>
<td>SCOTT</td>
<td>20</td>
<td>29-JAN-98</td>
</tr>
<tr>
<td>BRASS</td>
<td>24</td>
<td>13-MAY-01</td>
</tr>
</tbody>
</table>

**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 **MAXgetBytes** 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).
1. Introduction: Basics of Database Administration

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

Oracle Architecture

SGA (System Global Area)
- DB Buffer Cache
- Redo Log Buffer
- Shared Pool
  - Library Cache
    - Shared SQL Area
    - PL/SQL Procedures
  - Dictionary Cache
    - (row cache)

Shared Memory:
- SMON
- PMON
- DB Buffer Cache
- SGA
- Redo Log Buffer
- Client
- Server
- DBW0
- CKPT
- LGWR
- ARCH

Disk:
- Data Files
- Control File(s)
- Redo Log
- Archive

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.
2. Disks and the Buffer Cache 2-47

Overview

1. Disks
2. RAID Storage
3. Tablespaces in Oracle
4. Storage Hierarchy, The Buffer Manager
5. Disk/Buffer Performance in Oracle

Stefan Brass: Datenbanken II B Universität Halle, 2005

2. Disks and the Buffer Cache 2-48

Tablespaces (1)

• In Oracle, one can use tablespaces to control on which disk(s) a table is stored.
  Tablespaces are physical containers for tables. When a table is created, the tablespace in which the table is stored can be defined.
• Tablespaces are groups of data files. The files can be on the same disk or spread across several disks.
  A tablespace is something like a logical disk.
• Every data file can belong to only one tablespace.
• It is possible to have data files from different tablespaces on the same physical disk.

Stefan Brass: Datenbanken II B Universität Halle, 2005

2. Disks and the Buffer Cache 2-49

Tablespaces (2)

Table
(1, 1)
<created in>
(0, ∗)

Tablespace
(1, ∗)
consists of
(1, 1)

Data File

A data file is spread across several disks only in case of RAID systems.

Stefan Brass: Datenbanken II B Universität Halle, 2005

2. Disks and the Buffer Cache 2-50

Tablespaces (3)

• Example for defining the tablespace for a table:
  CREATE TABLE STUDENTS(STUD_ID NUMERIC(5), ...)
  TABLESPACE USER_DATA;
• Clauses setting physical attributes are specific to a DBMS (in this case Oracle), they are not contained in the SQL standard.
• The data file cannot be specified.
  Actually, the data file cannot be specified only for the first extent (piece of storage) allocated for the table. One can manually allocate additional extents in specified data files in order to stripe a table between different disks.

Stefan Brass: Datenbanken II B Universität Halle, 2005
Managing Tablespaces (1)

- A tablespace is created with a command like the following:
  
  ```sql
  CREATE TABLESPACE USER_DATA
  DATAFILE 'D:\User1.ora' SIZE 20M;
  ```

- Oracle will automatically create the datafile.
  
  Use "SIZE 20M REUSE" if the file exists and can be overwritten (the size is optional in this case).

- A data file can be added to a tablespace with the following command:
  
  ```sql
  ALTER TABLESPACE USER_DATA
  ADD DATAFILE 'D:\User2.ora' SIZE 20M;
  ```

Managing Tablespaces (2)

- It is possible to let Oracle extend the datafile whenever the tablespace becomes full:
  
  ```sql
  CREATE TABLESPACE USER_DATA
  DATAFILE 'D:\User1.ora' SIZE 20M
  AUTOEXTEND ON NEXT 5M MAXSIZE 50M;
  ```

  The file is created with 20 MB size. When it is full, it is increased to 25 MB, 30 MB, and so on until 50 MB. When the 50 MB are used up, further commands that need additional storage (e.g. insertions) will fail. Without MAXSIZE, the entire disk is filled.

- The data file size can also be manually increased:
  
  ```sql
  ALTER DATABASE
  DATAFILE 'D:\User2.ora' RESIZE 100M
  ```

Managing Tablespaces (3)

- Tablespaces can be taken offline (i.e. made not available):
  
  ```sql
  ALTER TABLESPACE USER_DATA OFFLINE;
  ```

- The `SYSTEM` tablespace cannot be taken offline.

- The following command deletes a tablespace with all data in it:
  
  ```sql
  DROP TABLESPACE USER_DATA INCLUDING CONTENTS;
  ```
Managing Tablespaces (4)

- The following command is needed when data files are renamed or moved to another disk:

   \[
   \text{ALTER DATABASE RENAME FILE 'C:\User2.ora' TO 'D:\User2.ora'}
   \]

   The file cannot be currently in use. E.g. the tablespace is offline or the DBMS server is in the MOUNT state, but not OPEN. The command only changes the file name that Oracle uses to access the file (stored in Oracle's control file), one must use OS commands to actually move the data file.

- See the Oracle SQL Reference for more options.

TS in the Data Dictionary (1)

- \text{DBA_TABLESPACES} is list of all tablespaces. Columns are, e.g.

  \[
  \begin{align*}
  &\text{TABLESPACE_NAME: Name of the tablespace.} \\
  &\text{INITIAL_EXTENT, NEXT_EXTENT, MIN_EXTENTS, MAX_EXTENTS, PCT_INCREASE:} \\
  &\text{Default storage parameters for tables created in this tablespace (see next chapter).} \\
  &\text{MIN_EXTLEN: Minimal size for storage pieces allocated in this tablespace.}
  \end{align*}
  \]

TS in the Data Dictionary (2)

- Selected columns of \text{DBA_TABLESPACES}, continued:

  \[
  \begin{align*}
  &\text{STATUS: ONLINE, OFFLINE, READ ONLY.} \\
  &\text{CONTENTS: PERMANENT or TEMPORARY.} \\
  &\text{TEMPORARY: For sorting during query evaluation.} \\
  &\text{LOGGING: LOGGING or NOLOGGING.} \\
  &\text{If changes are not logged, they cannot be recovered.} \\
  &\text{EXTENT_MANAGEMENT: DICTIONARY or LOCAL.} \\
  &\text{Extends are pieces of storage allocated for tables or other database objects. Originally, free space was managed by entries in the data dictionary. Oracle 8i has introduced local extend management (probably a bitmap inside the datafile) that is supposed to be more efficient.}
  \end{align*}
  \]

TS in the Data Dictionary (3)

- \text{USER_TABLESPACES} lists all tablespaces for which the current user has write permission.

- \text{USER_TS_QUOTAS} lists the current file space usage and the allowed maximum for every tablespace writeable by the user.

- \text{USER_FREE_SPACE}: Pieces of free space in tablespaces.

- \text{TABS (= USER_TABLES)} contains the tablespace in which a table is stored:

  \[
  \text{SELECT TABLE_NAME, TABLESPACE_NAME FROM TABS}
  \]
TS in the Data Dictionary (4)

- See also:
  - DBA_DATA_FILES,
  - V$DATAFILE,
  - V$DATAFILE_HEADER,
  - DBA_FREE_SPACE,
  - DBA_FREE_SPACE_COALESCED,
  - DBA_TS_QUOTAS,
  - V$FILESTAT,
  - V$TABLESPACE.

Performance Monitoring (1)

- For performance tuning, the bottlenecks of the system must be found (i.e. the performance problems must be located).
  
  E.g. it is useless to increase the cache if it performs well.

- In Oracle, a lot of statistical information is available in the V$*-tables, e.g. V$SYSSTAT.

  The V$*-tables are called the "Dynamic Performance Views". They give access to data structures inside the Oracle server. They are not stored tables. Of course, the V$*-tables can only be accessed by the DBA.

Performance Monitoring (2)

- V$SYSSTAT contains 226 different performance related numbers (counters, average times, etc.). Its columns are:
  - STATISTIC#: Identifying number of the statistic.
  - NAME: Symbolic name of the statistic.
  - CLASS: Bit pattern to classify the statistic.
  
  E.g. all cache-related statistics have the third bit (8) set.
  - VALUE: The value of the statistic.
Performance Monitoring (3)

- E.g., this query prints the number of data blocks that were physically read since system startup:
  ```sql
  SELECT VALUE 
  FROM V$SYSSTAT 
  WHERE NAME = 'physical reads'
  ```
- The Oracle server maintains a counter that is initialized to 0 when the system is started and incremented each time a block is read from disk.
- There are many different such counters.
  - Some statistics are available only when the initialization parameter TIMED_STATISTICS is set to TRUE (because they cause some overhead).

Performance Monitoring (4)

- In addition, there is a table V$SESSTAT that contains statistics for each session. Columns are:
  - SID: Session identifier (more info in V$SESSION).
  - STATISTIC#: Identifying number of the statistic.
  - VALUE: The value of the statistic.
- Here, a join with V$STATNAME is necessary in order to decode the statistic numbers.
  ```sql
  V$STATNAME lists all available statistics, it has the columns STATISTIC#, NAME, CLASS. Some of the statistics are only meaningful in V$SYSSTAT, others only in V$SESSTAT.
  ```

Performance Monitoring (5)

- Two scripts in "$ORACLE_HOME/rdbms/admin" can be used to print a report containing many statistics:
  - utlbstat.sql (begin statistics) records the current values of the statistics counters.
    - The scripts are executed with SQL*Plus. They log in as INTERNAL. It might be necessary to belong to the OS user group "dba".
  - Then there should be normal production usage of the DBMS for some time.
  - utlestat.sql (end statistics) computes the differences of the then current values with the stored ones and generates a report (in report.txt).

Performance Monitoring (6)

- SQL*Plus shows a few statistics for each executed query after
  ```sql
  SET AUTOTRACE ON
  ```
- In addition, the query execution plan is shown.
  ```sql
  SET AUTOTRACE ON STATISTICS shows only the statistics, SET AUTOTRACE ON EXPLAIN only the execution plan. Try also SET TIMING ON. If a user has rights on a view, but not the base tables, the execution plan is not shown. Before one can see the execution plan, a table for storing information about that plan must be created by executing the script $ORACLE_HOME/rdbms/admin/utlxplan.sql. Before a user can see the statistics, the DBA must grant the role PLUSTRACE to that user. The role is created with the script $ORACLE_HOME/sqlplus/admin/plustrce.sql.
Buffer Performance (1)

- In Oracle, the number of cache misses is the value of the counter "physical reads".
- The total number of requests are the sum of two statistics values (this sum is called "logical reads"):
  - **consistent gets**: Requests for block versions that contain only changes that were committed before the query started.
  - **db block gets**: Requests for the current version of a block.

Buffer Performance (2)

- In Oracle, the hit ratio is computed as:
  \[
  \frac{\text{consistent gets} + \text{db block gets}}{\text{consistent gets} + \text{db block gets}} - \frac{\text{physical reads}}{\text{consistent gets} + \text{db block gets}}
  \]
- Exercise: Write an SQL query for this.
- The hit ratio should be above 90% or 95%.
  - At least for OLTP (online transaction processing) applications.
- If the hit ratio is below 60%, 70% or 80%, the buffering is not working well and something should be done.

Buffer Performance (3)

- E.g., in order to improve the hit ratio, it might be possible to increase the initialization parameter `DB_BLOCK_BUFFERS` (number of buffer frames).
  - Total buffer memory: `DB_BLOCK_BUFFERS * DB_BLOCK_SIZE`.
- It is important that the entire SGA (system global area, includes the cache) remains in real memory.
  - If the increase of the number of buffer frames leads to paging on the operating system level (i.e. "virtual memory" is used), the situation is worse than before ("double paging").
- If necessary, more memory must be bought.

Buffer Performance (4)

- However, before one tunes the buffer cache, there are many other things to check and improve.
- E.g., indexes might reduce the number of accessed disk blocks. Then the hit ratio will improve without adding more buffer frames.
  - Oracle therefore recommends a specific sequence for tuning: Business rules, data design, application design, logical DB structure, DB operations, access paths, memory allocation, I/O and physical structure, resource contention, OS/hardware.
Buffer Performance (5)

- The hit ratio can also be improved by caching only blocks from certain tables.
  
  E.g., if blocks from a very large table are accessed at random, they do not profit from the valuable buffer space, but push other blocks out of the cache.

- Besides the DEFAULT buffer pool, Oracle can manage two other buffer pools: KEEP and RECYCLE.

- One can distribute the available buffer frames between these three buffer pools and assign database objects to a specific buffer pool:

  ```sql
  CREATE TABLE ...(...) STORAGE(BUFFER_POOL KEEP)
  ```

Buffer Performance (6)

- Oracle normally places blocks read during a full table scan at the front of the LRU queue, so that the buffer frames are immediately reused.

- For small lookup tables one should request that they are even if read in a full table scan:

  ```sql
  CREATE TABLE EXERCISES (CAT CHAR(1), ...) TABLESPACE USER_DATA CACHE
  ```

  Small tables are nearly always read in full table scans.

  See also: `V$BUFFER_POOL`, `V$BUFFER_POOL_STATISTICS`, `V$BH`.

Disk Performance (1)

- `V$FILESTAT` contains performance statistics for each file. It has the following columns (continued below):
  
  - `FILENAME#`: File number.
  
  `V$DATAFILE` relates `FILENAME#` and `NAME`.
  
  - `PHYRDS`: Number of read operations for this file.
  
  Physical reads, i.e. real reads (not from buffer).
  
  - `PHYWRTS`: Number of write operations for this file.
  
  - `PHYBLKRD`: Number of blocks read.
  
  `PHYBLKRD` can be larger than `PHYRDS` since sometimes a chunk of several consecutive blocks is read in one call.
  
  - `PHYBLKWRT`: Number of blocks written.
3. Physical Storage of Relations

### Segments (1)

- If tablespaces are the “logical disks” of Oracle, segments are the “logical files”.
- Segments are sequences of data blocks within a tablespace.
  The sequence does not have to be the physical sequence. The blocks are not necessarily stored in contiguous places.
- Segments can grow (blocks can be appended at the end) and shrink (blocks are removed at the end).
  In Oracle, segments shrink only when explicitly requested.

### Segments (2)

- The used storage in a tablespace is partitioned into segments.
  Every data block can belong to at most one segment.
- A tablespace can contain many segments.
- For every table, Oracle creates a segment inside the tablespace that is mentioned in the `CREATE TABLE`.
- In the same way, each index is stored in a segment.
  The four basic kinds of segments are: Data segments (for tables), index segments, rollback segments (for storing old versions of blocks), temporary segments (for sorting during query evaluation).

### Segments (3)

- Normally, the relationship between data segments and tables is 1:1. But in general, it can be n:m:
  - Partitioned tables have more than one segment (usually in different tablespaces on several disks).
    A partitioned table is stored in several pieces, where each piece is basically a table with the same structure: The complete table is then the union of the pieces. When rows are inserted, conditions on the data determine in which piece the row is stored.
  - Clusters can contain rows from several tables having one or more attributes in common.
    Clusters are an Oracle-specific data structure that permits very efficient joins because the rows to be joined together are already stored together (ideally, in the same block).

### Segments (4)

- ![Diagram of segment relationship](image)
Segments (5)

- The data dictionary view `DBA_SEGMENTS` contains one row for each segment. It has the following columns:
  - `OWNER`: User who created the table etc.
  - `SEGMENT_NAME`: Table name, index name, etc.
  - `PARTITION_NAME`: For partitioned tables (else null).
  - `SEGMENT_TYPE`: Type of the segment, e.g. `TABLE`, `INDEX`, `CLUSTER`, `TABLE PARTITION`, `INDEX PARTITION`, `ROLLBACK`, `DEFERRED ROLLBACK`, `TEMPORARY`, `CACHE`, `LOBINDEX`, `LOBSEGMENT`.
  - `TABLESPACE_NAME`: Tablespace in which the segment is stored.

Extents (1)

- Oracle allocates storage in units called "extents".
- An extent is a sequence of contiguous disk blocks. Thus, an extent can be especially fast read from the disk.
- An extent belongs to a single segment and thus to a single table (or index etc.).
- A segment can consist of many extents. But too many extents give bad performance.

Extents (2)

- Extent sizes are specified in the table declaration:

```
CREATE TABLE STUDENTS(SID NUMERIC(3), ...)
TABLESPACE USER_DATA
STORAGE(INITIAL 200K NEXT 50K
         PCTINCREASE 100)
```

- When the table is created, the initial extent is allocated.

Although it does not yet contain any rows, it needs disk space for the initial extent (200 KB in the example). The extent size should be a multiple of `DB_BLOCK_SIZE * DB_FILE_MULTIBLOCK_READ_COUNT` (the size that Oracle reads during a full table scan in one disk access).

Extents (3)

- Whenever the disk space allocated for a table is full, another extent will be allocated.
- In the example, the second extent will be 50 KByte (`NEXT`). Normally, all following extents have this size.
- However, with the parameter `PCTINCREASE` one can request that each following extent will be larger than the previous one (reduces number of extents).

`PCTINCREASE 100` means that the extent size is doubled. Third extent: 100 KB, fourth: 200 KB, etc. If the extent size grows so fast, there will certainly not be very many extents. However, since one soon gets very large extents, space may be wasted.
3. Physical Storage of Relations

Extents (4)

Example:

File 1:

1 2 3 4 5 6 7 8 9 10 11
Table R, Extent 1 | Table R, Extent 2 | Free

File 2:

1 2 3 4 5 6 7 8 9 10 11
Table S, Extent 1 | Free | Table R, Extent 3

Tables R and S are stored in a tablespace which consists of two data files. Table R has three extents: Block 1 to 4 in File 1, Block 5 to 8 in File 1, and Block 7 to 11 in File 2. Oracle does not merge contiguous extents of a table. Table S consists of a single extent (Block 1 to 4 in File 2).

Stefan Brass: Datenbanken II, Universität Halle, 2005

Extents (5)

Segment (1, 1) in (0, *) Tablespace (1, *)

consists of (1, 1)

Extent (1, 1) in (0, *) Data File (1, *)

consists of (0, 1)

Block (1, 1)

Extents (7)

- DBA_EXTENTS contains one row for each extent.

  - OWNER, SEGMENT_NAME, PARTITION_NAME: Identification of the segment to which this extent belongs.
  - SEGMENT_TYPE, TABLESPACE_NAME: See DBA_SEGMENTS.
  - EXTENT_ID: Extent number within segment. Counted from 0, i.e. 0, 1, 2, ...
  - FILE_ID: File containing the extent.
  - BLOCK_ID: Start of the extent within the file.
  - BYTES, BLOCKS: Size of the extent.
  - RELATIVE_FNO: Relative file number of first block.

    I am not sure what relative file number means. Please help.

Extents (8)

- DBA_FREE_SPACE contains one row for each contiguous sequence of blocks that is currently not allocated to a segment (“free extents”).
  - TABLESPACE_NAME, FILE_ID: Tablespace, data file.
  - BLOCK_ID: First block of free piece.
  - BYTES, BLOCKS: Size of free piece.
  - RELATIVE_FNO: Relative file no of first extent block.

DBA_FREE_SPACE might contain two adjacent pieces. Oracle checks only from time to time (or when necessary) whether adjacent pieces can be merged (“coalesced”).

- See also: USER_EXTENTS, USER_FREE_SPACE.
3. Physical Storage of Relations

### TS Declaration: Extents (1)

- In the `CREATE TABLESPACE` command, default values for the extent parameters can be specified:

  ```
  CREATE TABLESPACE USER_DATA
  DATAFILE 'D:\User1.ora' SIZE 20M
  MINIMUM EXTENT 32K
  DEFAULT STORAGE (INITIAL 100K NEXT 50K
  PCTINCREASE 5
  MINEXTENTS 1 MAXEXTENTS 50
  BUFFER_POOL KEEP)
  ```

- `DBA_TABLESPACES` lists these values (used for all segments in the tablespace unless otherwise specified).

### TS Declaration: Extents (2)

- The `DEFAULT STORAGE` parameters have no meaning for the tablespace itself, they only apply to tables created within it.

- E.g. if one does not specify `PCTINCREASE` for a table, it will not be 0, but the value defined in the tablespace declaration.

  If one does not define it there, defaults set by Oracle are used: `PCTINCREASE=50`, 5 blocks for `INITIAL` and `NEXT`. The small default values for `INITIAL` and `NEXT` show that at least for large tables, it is important to set these parameters.

### Extent Allocation (1)

- The following is basically the explanation from the Oracle manual.
  - Experiments show that extents are sometimes slightly larger than expected.
  - First, Oracle searches through the list of all “free extents” of the requested tablespace for an exactly fitting piece of disk space.

  Of course, the requested extent size is rounded up to the next multiple of `DB_BLOCK_SIZE` (or to the minimal extent size declared for the tablespace). The first extent must consist of at least two blocks, because the first block of each segment is the segment header and cannot be used for table data.

### Extent Allocation (2)

- If an extent is found, the data dictionary and the segment header are updated to reflect the allocation of the disk space.

- If no free space is found that has a size equal to the requested amount, Oracle searches the list again for a piece that is larger than the requested one.
  - If the first piece found is larger by 5 blocks or more, a piece of the requested size is cut off.
  - If the piece found is larger by less than 5 blocks, it is completely allocated as the new extent.
### Extent Allocation (3)

- If all existing pieces of free space are smaller than requested, Oracle merges adjacent pieces. Then both steps are repeated.
- If still no piece is found, and **AUTOEXTEND** is on for at least one data file, the data file is extended (i.e. more disk space is requested from the OS).
- Else the operation fails and an error message is returned (“tablespace full”).

### Local Extent Management (1)

- Since Oracle 8i, free space can alternatively be managed by an array of bits showing which “extents” are allocated.
- For such tablespaces, one can
  - either define a uniform extent size (then one bit is used for each piece of that size)
  - or let Oracle determine the extent size (the algorithm is not disclosed in the documentation).
- The bitmaps are stored in each data file, and not in the data dictionary, thus the name.

### Local Extent Management (2)

- When using bitmaps for free space management, there is no need to search for adjacent pieces of free space in order to merge them.
  - It also avoids recursive calls: The data dictionary entry might itself need space. Also requires to change a rollback segment.
- The parameters **NEXT** and **PCTINCREASE** are not possible for such tablespaces.
  - But **INITIAL** is of course possible.