CHAPTER # 83

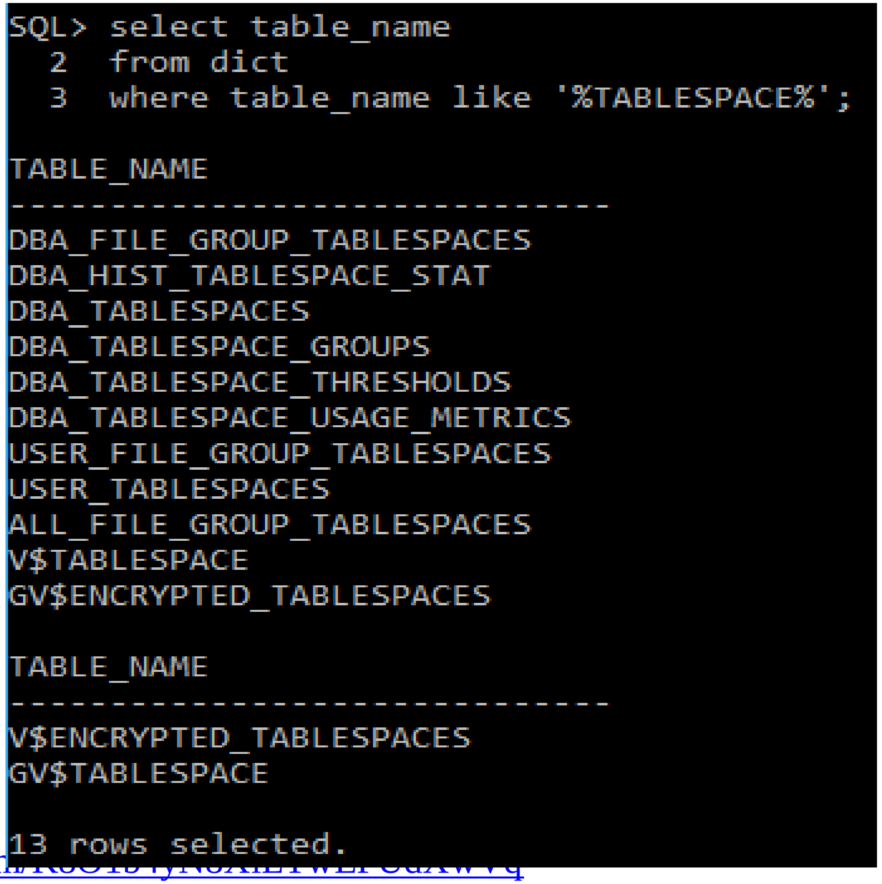
Viewing Tablespace using Data Dictionary Views

Data Dictionary Views of Tablespace

- Dictionary Views of Tablespaces
- V$Tablespaces
- DBA_DATA_FILES
- DBA_TEMP_FILES
- Finding Used Size of Tablespaces
- Finding Free Space of Tablespaces

Dictionary Views of Tablespaces

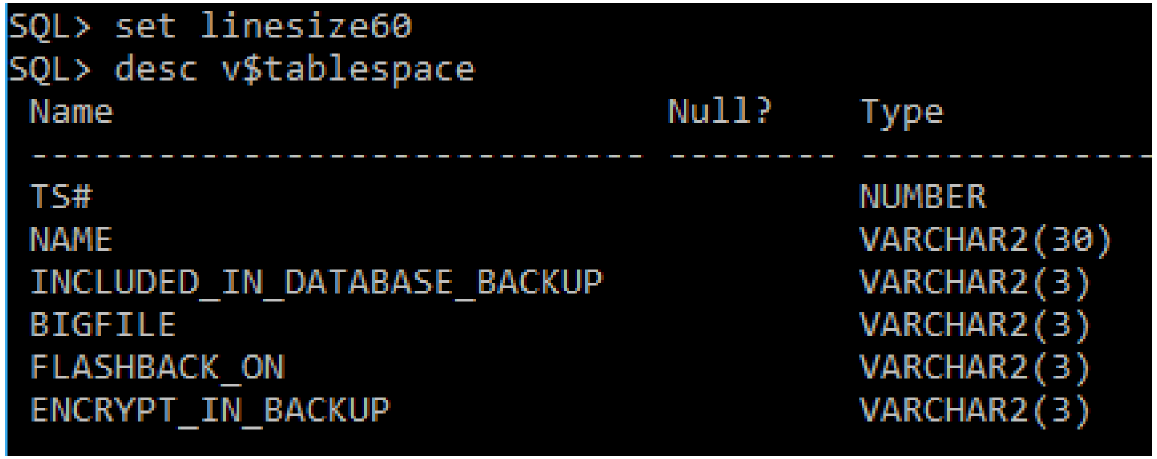
Use Dictionary to display views:



```
SQL> select table_name
  2  from dict
  3  where table_name like '%TABLESPACE%';

TABLE_NAME
------------------------------
DBA_FILE_GROUP_TABLESPACES
DBA_HIST_TABLESPACE_STAT
DBA_TABLESPACES
DBA_TABLESPACE_GROUPS
DBA_TABLESPACE_THRESHOLDS
DBA_TABLESPACE_USAGE_METRICS
USER_FILE_GROUP_TABLESPACES
USER_TABLESPACES
ALL_FILE_GROUP_TABLESPACES
V$TABLESPACE
GV$ENCRYPTED_TABLESPACES

TABLE_NAME
------------------------------
V$ENCRYPTED_TABLESPACES
GV$TABLESPACE

13 rows selected.
```

V$Tablespaces

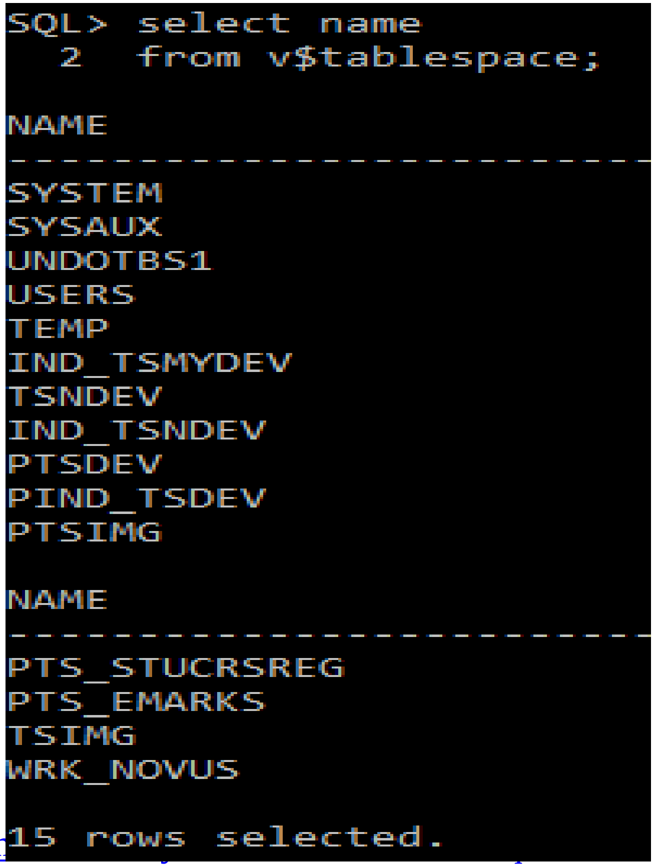
To display columns of V$Tablespaces:



```
SQL> set linesize60
SQL> desc v$tablespace
 Name                          Null?    Type
 ----------------------------- -------- --------------
 TS#                                    NUMBER
 NAME                                   VARCHAR2(30)
 INCLUDED_IN_DATABASE_BACKUP            VARCHAR2(3)
 BIGFILE                                VARCHAR2(3)
 FLASHBACK_ON                           VARCHAR2(3)
 ENCRYPT_IN_BACKUP                      VARCHAR2(3)
```

V$Tablespaces

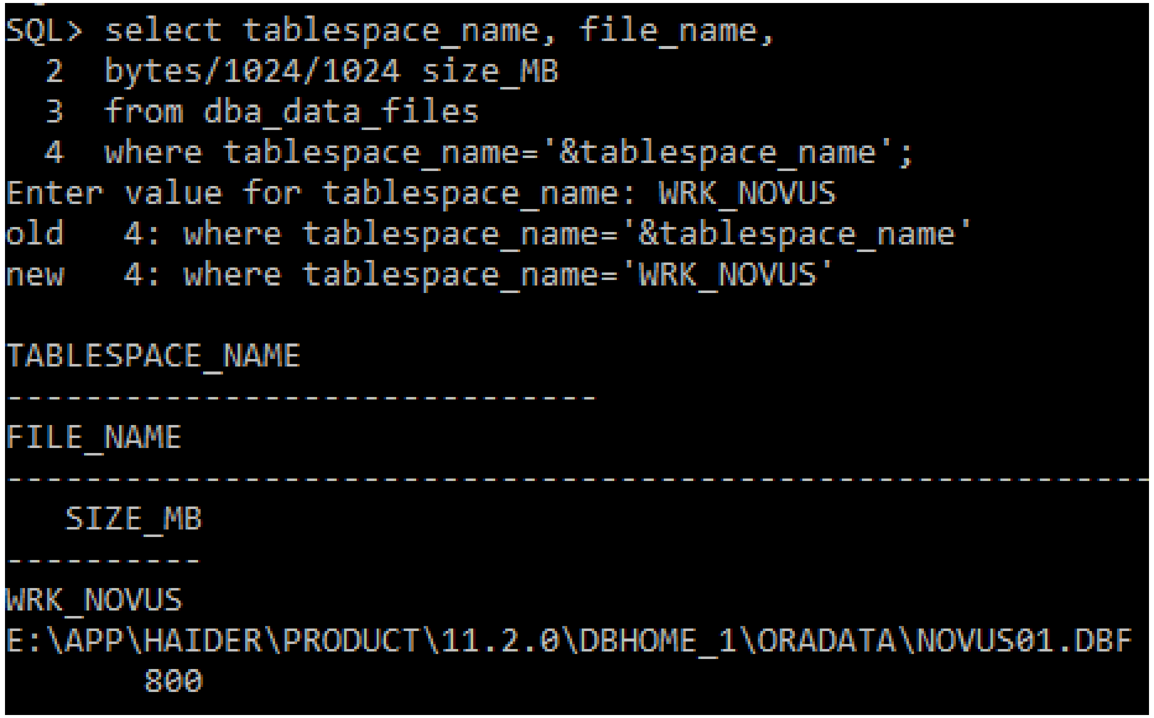
Select column name V$Tablespaces:



```
SQL> select name
  2  from v$tablespace;

NAME
-------------------------
SYSTEM
SYSAUX
UNDOTBS1
USERS
TEMP
IND_TSMYDEV
TSNDEV
IND_TSNDEV
PTSDEV
PIND_TSDEV
PTSIMG

NAME
-------------------------
PTS_STUCRSREG
PTS_EMARKS
TSIMG
WRK_NOVUS

15 rows selected.
```

DBA_DATA_FILES

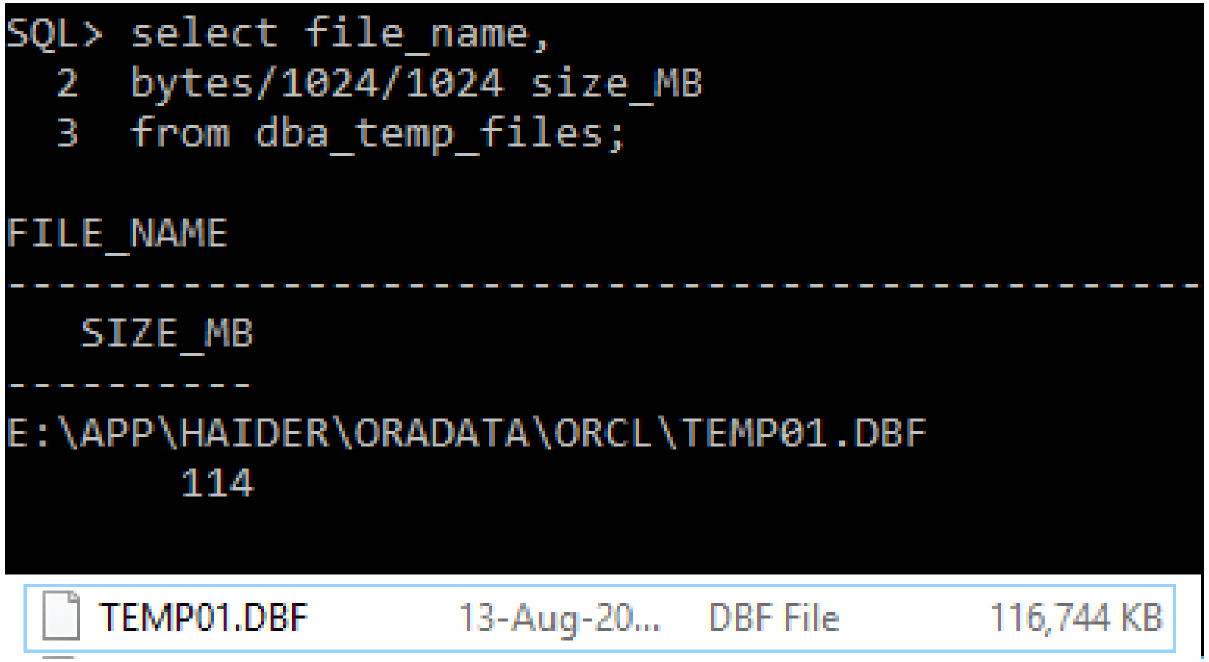
Display name and data file size in MB:



```
SQL> select tablespace_name, file_name,
  2  bytes/1024/1024 size_MB
  3  from dba_data_files
  4  where tablespace_name='&tablespace_name';
Enter value for tablespace_name: WRK_NOVUS
old   4: where tablespace_name='&tablespace_name'
new   4: where tablespace_name='WRK_NOVUS'

TABLESPACE_NAME
------------------------------
FILE_NAME
--------------------------------------------------------------------------------
   SIZE_MB
----------
WRK_NOVUS
E:\APP\HAIDER\PRODUCT\11.2.0\DBHOME_1\ORADATA\NOVUS01.DBF
       800
```

DBA_TEMP_FILES

Display name and data file size in MB:

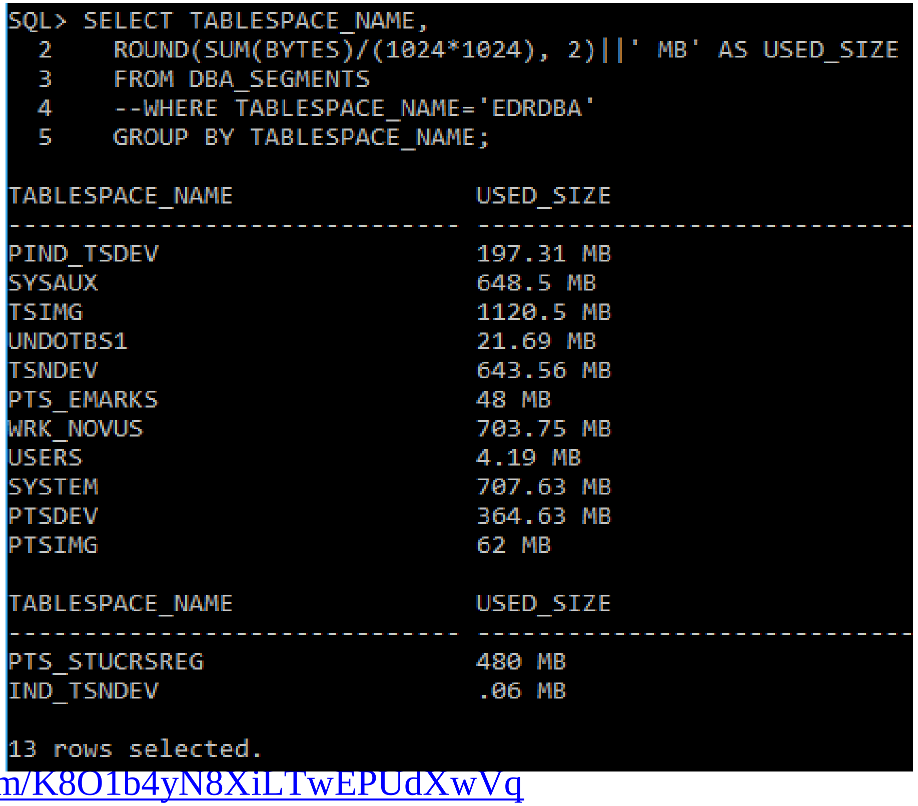


```
SQL> select file_name,
  2  bytes/1024/1024 size_MB
  3  from dba_temp_files;

FILE_NAME
--------------------------------------------------------------------------------
   SIZE_MB
----------
E:\APP\HAIDER\ORADATA\ORCL\TEMP01.DBF
       114
```

TEMP01.DBF   13-Aug-20...   DBF File   116,744 KB

Finding Used Data Size of Tablespaces

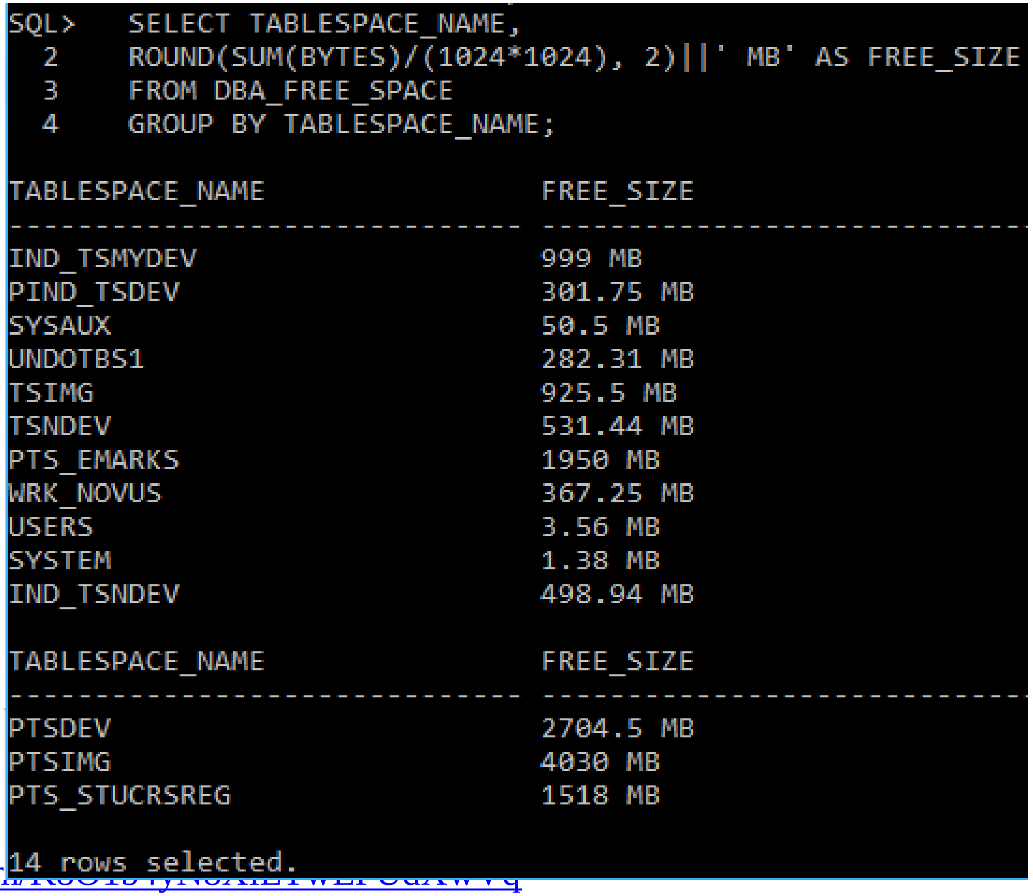
Using data dictionary view DBA_SEGMENTS :



```
SQL> SELECT TABLESPACE_NAME,
       ROUND(SUM(BYTES)/(1024*1024), 2)||' MB' AS USED_SIZE
       FROM DBA_SEGMENTS
       --WHERE TABLESPACE_NAME='EDRDBA'
       GROUP BY TABLESPACE_NAME;

TABLESPACE_NAME                USED_SIZE
------------------------------ ------------------------------
PIND_TSDEV                     197.31 MB
SYSAUX                         648.5 MB
TSIMG                          1120.5 MB
UNDOTBS1                       21.69 MB
TSNDEV                         643.56 MB
PTS_EMARKS                     48 MB
WRK_NOVUS                      703.75 MB
USERS                          4.19 MB
SYSTEM                         707.63 MB
PTSDEV                         364.63 MB
PTSIMG                         62 MB

TABLESPACE_NAME                USED_SIZE
------------------------------ ------------------------------
PTS_STUCRSREG                  480 MB
IND_TSNDEV                     .06 MB

13 rows selected.
```

Finding Free Space of Tablespaces

Using data dictionary view DBA_FREE_SPACE



```
SQL>   SELECT TABLESPACE_NAME,
  2    ROUND(SUM(BYTES)/(1024*1024), 2)||' MB' AS FREE_SIZE
  3    FROM DBA_FREE_SPACE
  4    GROUP BY TABLESPACE_NAME;

TABLESPACE_NAME                FREE_SIZE
------------------------------ -----------------------------
IND_TSMYDEV                    999 MB
PIND_TSDEV                     301.75 MB
SYSAUX                         50.5 MB
UNDOTBS1                       282.31 MB
TSIMG                          925.5 MB
TSNDEV                         531.44 MB
PTS_EMARKS                     1950 MB
WRK_NOVUS                      367.25 MB
USERS                          3.56 MB
SYSTEM                         1.38 MB
IND_TSNDEV                     498.94 MB

TABLESPACE_NAME                FREE_SIZE
------------------------------ -----------------------------
PTSDEV                         2704.5 MB
PTSIMG                         4030 MB
PTS_STUCRSREG                  1518 MB

14 rows selected.
```

End

CHAPTER # 84

Logical Storage Structure & Hierarchy

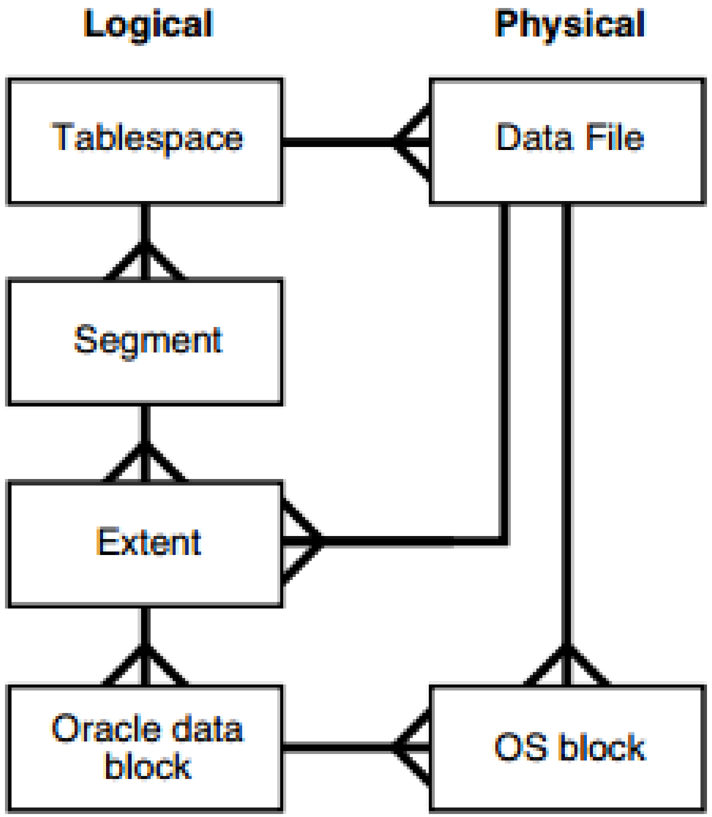
Logical Storage Structure & Hierarchy

- Physical & Logical Database
- Tablespace
- Database Block
- Database Block, Extents and Segments
- Database Block, Extents and Segments within Database

Physical & Logical Database

- Physical Database
  - Control Files
  - Redo Log Files
  - Data Files
- Logical Database
  - Tablespace
    - Segments
    - Extents
    - DB Blocks

Physical & Logical Database



Join VU Group: https://chat.whatsapp.com/K8O1b4yN8XiLTwEPUdXwVq

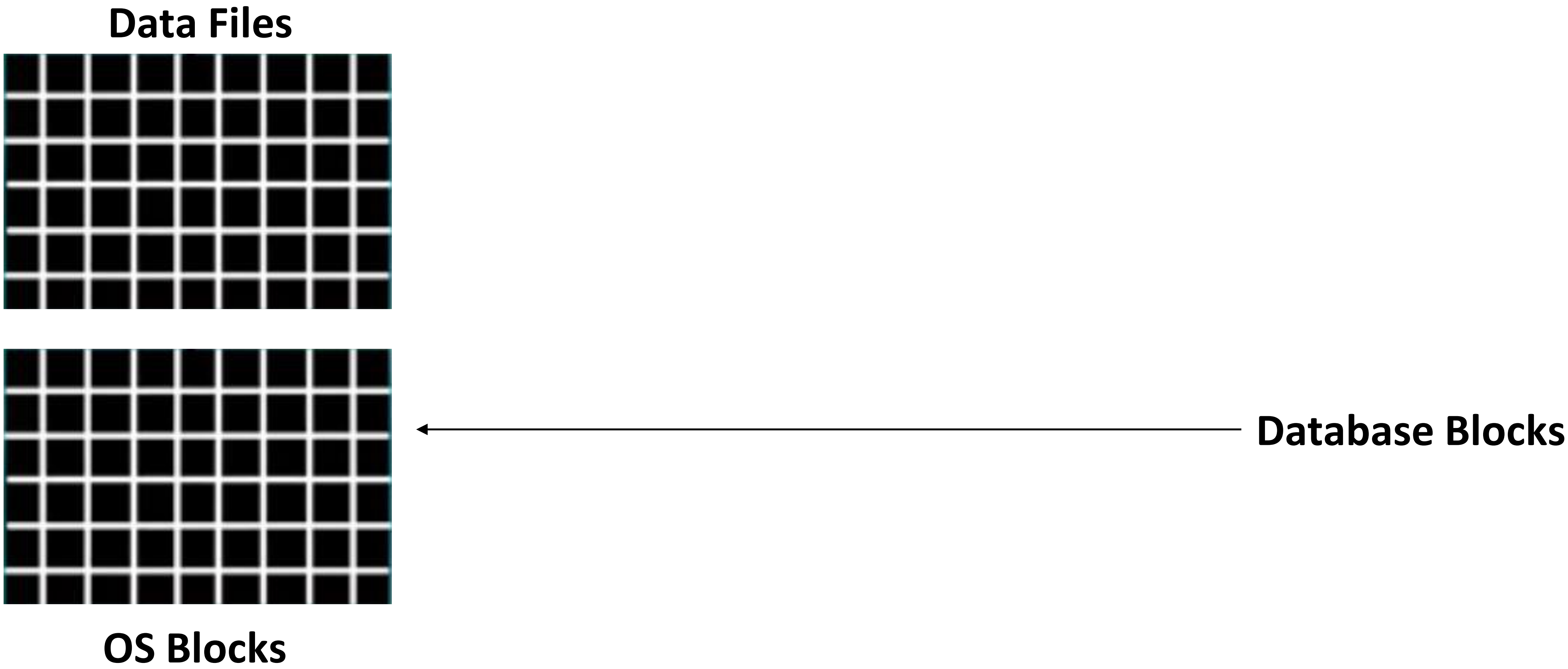
Tablespace

- Tablespace stores space logically to create logical objects such as Tables, Indexes and other objects.
- Tablespace’s physically space stored in associated files.

Database Block

- Logical minimum storage unit is called database block.
- Its size is decided during database creation.
- 5 default sizes: 2K, 4K, 8K, 16K, 32K (K: Kilobytes)

Database Block, Extents and Segments

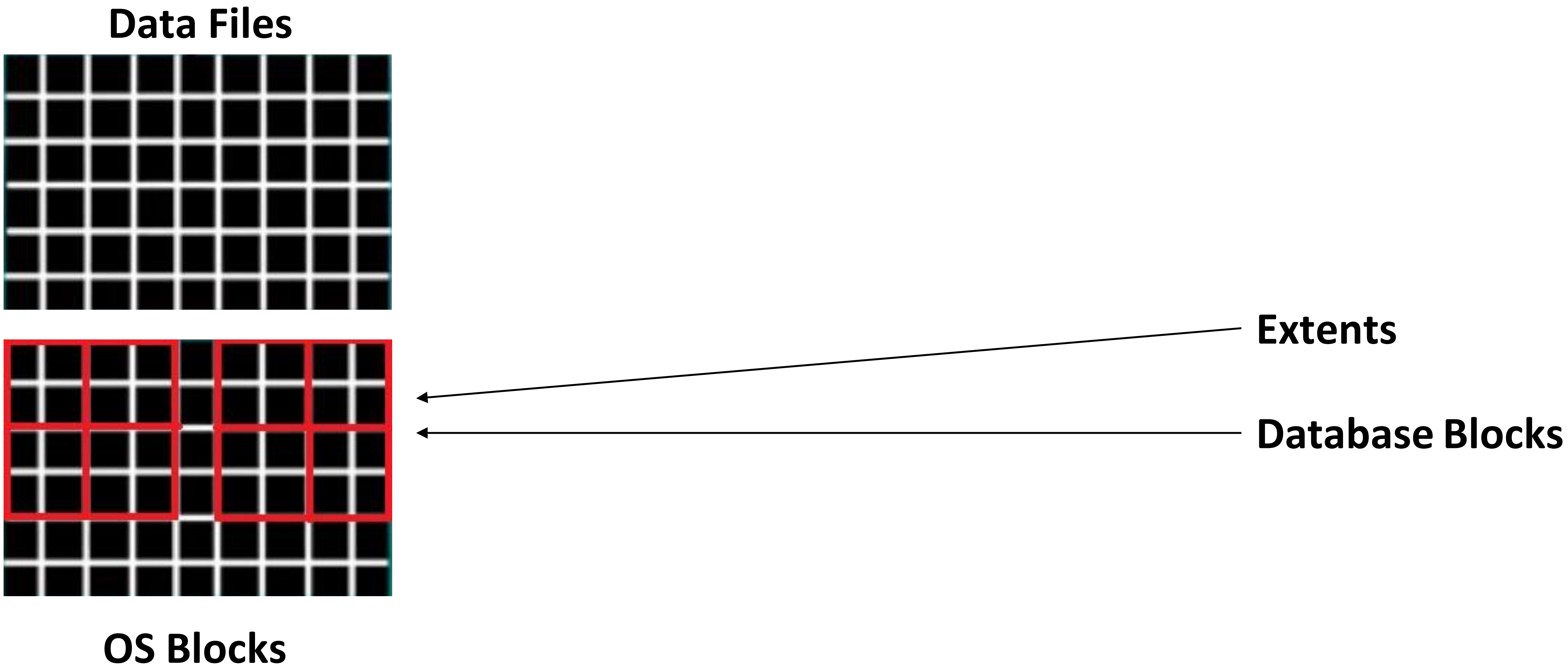


Data Files

Database Blocks

OS Blocks

Database Block, Extents and Segments



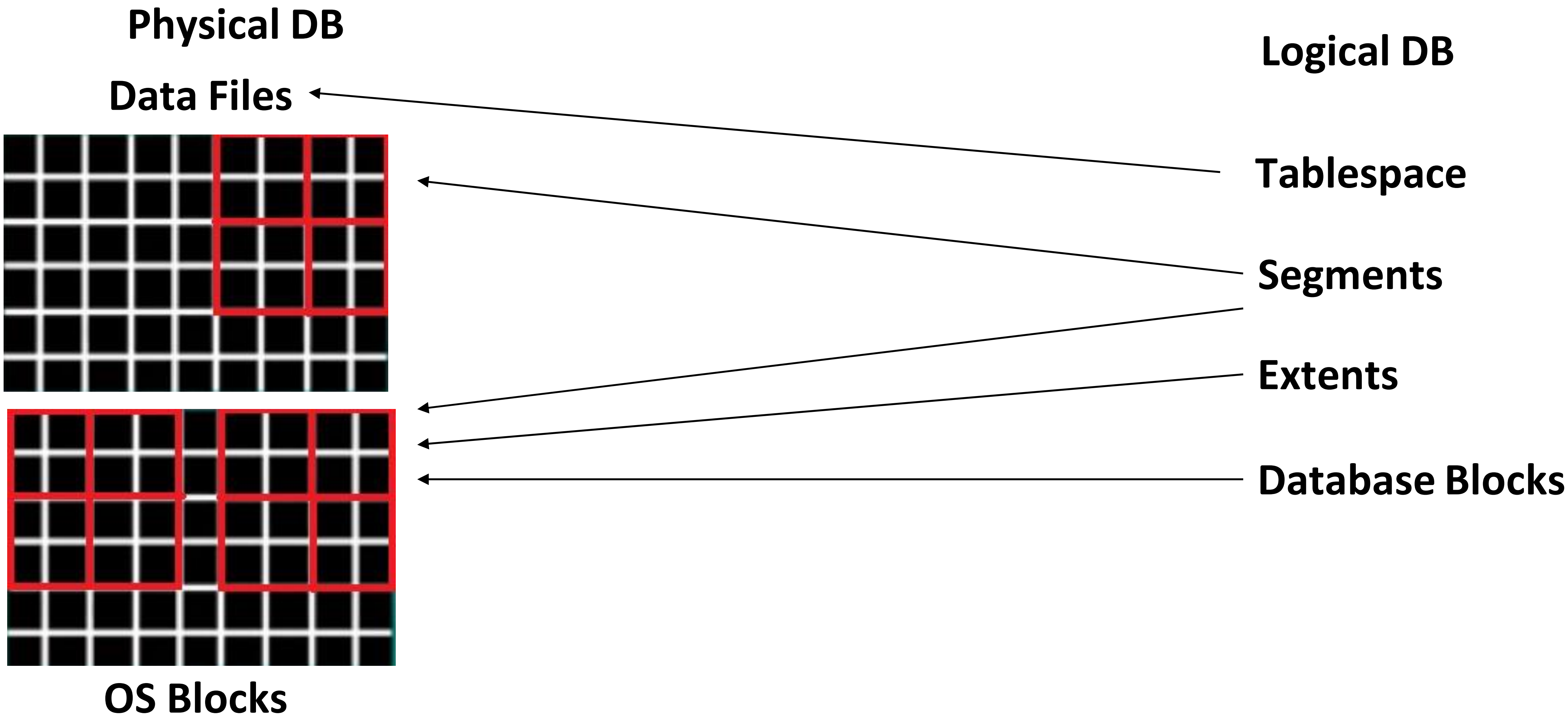
Data Files

Extents

Database Blocks

OS Blocks

Database Block, Extents and Segments



Physical DB

Data Files

OS Blocks

Logical DB

Tablespace

Segments

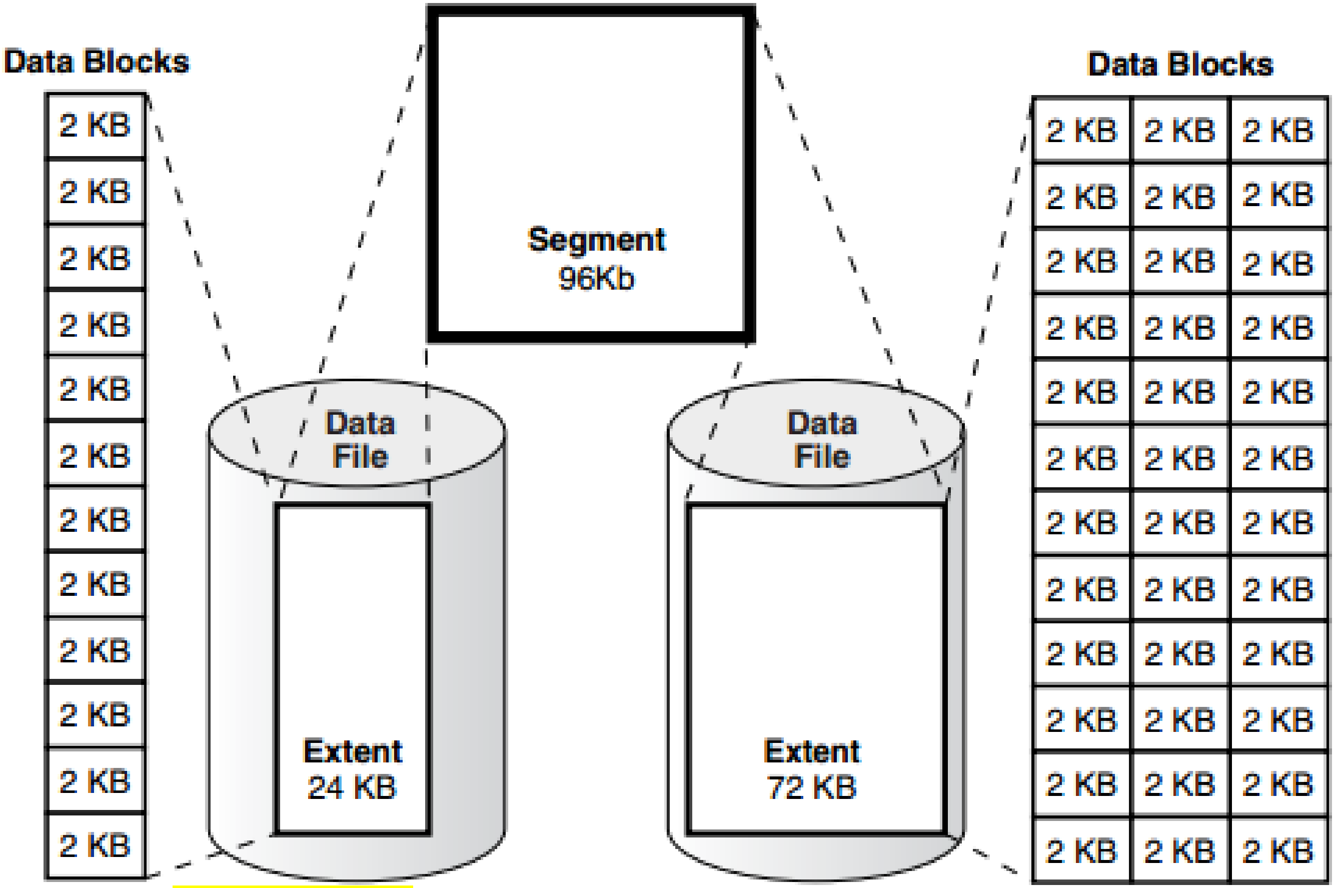
Extents

Database Blocks

Database Block, Extents and Segments within Database

- At the finest level of granularity, Oracle Database stores data in data blocks.
- One logical data block corresponds to a specific number of bytes of physical disk space. Block sizes can be 2 KB, 4KB or more.
- Data block is the smallest unit of storage that Oracle Database can use or allocate.

Database Block, Extents and Segments within Database



CHAPTER # 85

Overview of Data Blocks

Overview of Database Blocks

- What is Data Block?
- Data Blocks & OS Blocks
- Interaction of Data Blocks & OS Blocks
- Database Block Size
- Data Block Format
- Row Format
- RowID Pseudocolumn

What is Data Block?

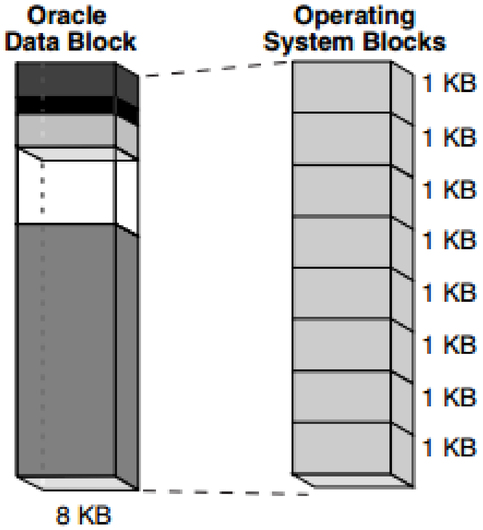
Oracle Database manages the logical storage space in the data files of a database in units called data blocks, also called Oracle blocks or pages. A data block is the minimum unit of database I/O.

Data Blocks & OS Blocks

- At the physical level, database data is stored in disk files made up of operating system blocks.
- An operating system block is the minimum unit of data that the operating system can read or write.
- In contrast, an Oracle block is a logical storage structure whose size and structure are not known to the operating system.

Data Blocks & OS Blocks

Figure shows that operating system blocks may differ in size from data blocks. The database requests data in multiples of data blocks, not operating system blocks.



Interaction of Data Blocks & OS Blocks

When the database requests a data block, the operating system translates this operation into a requests for data in permanent storage. The logical separation of data blocks from operating system blocks has the following implications:

- Applications do not need to determine the physical addresses of data on disk.
- Database data can be striped or mirrored on multiple physical disks.

Database Block Size

- Every database has a database block size. The DB_BLOCK_SIZE initialization parameter sets the data block size for a database when it is created.
- The size is set for the SYSTEM and SYSAUX tablespaces and is the default for all other tablespaces.
- The database block size cannot be changed except by re-creating the database.

Database Block Size

- If DB_BLOCK_SIZE is not set, then the default data block size is operating system-specific. The standard data block size for a database is4 KB or 8 KB.

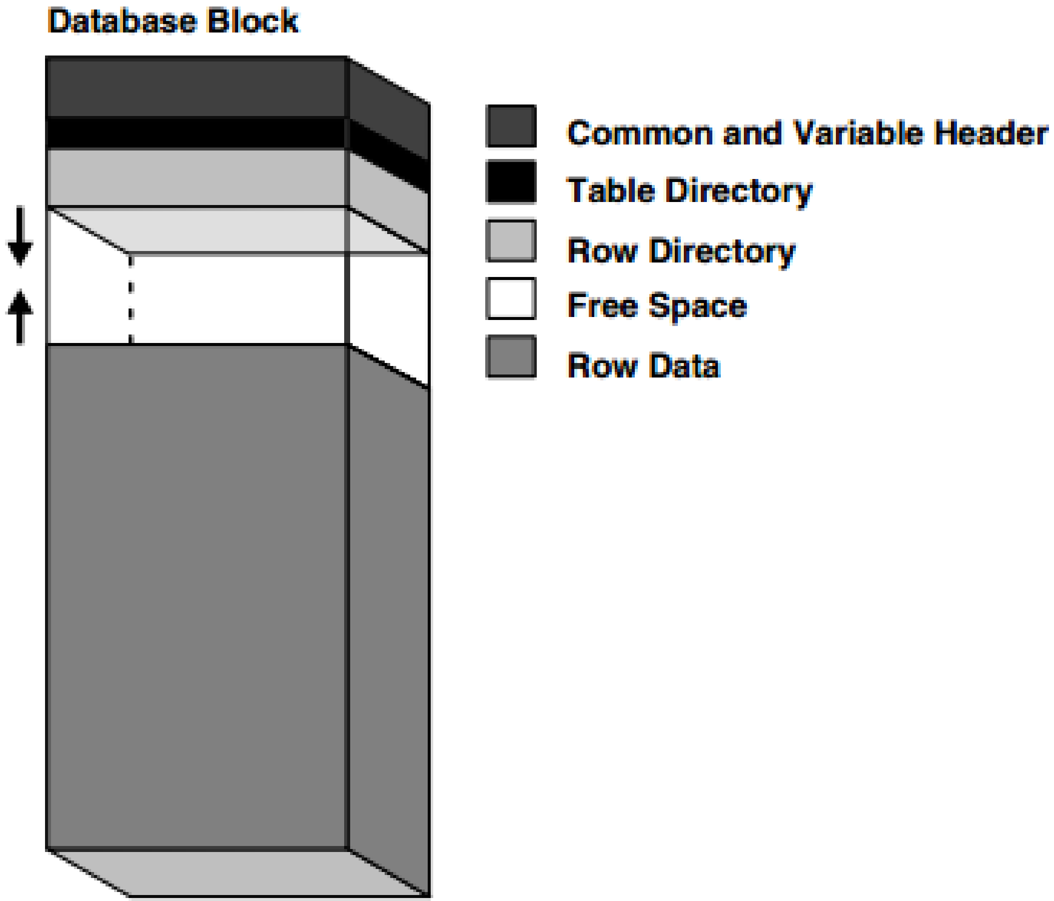
- If the size differs for data blocks and operating system blocks, then the data block size must be a multiple of the operating system block size

Database Block Size

- You can create individual tablespaces whose block size differs from the DB_BLOCK_SIZE setting.
- A nonstandard block size can be useful when moving a transportable tablespace to a different platform.

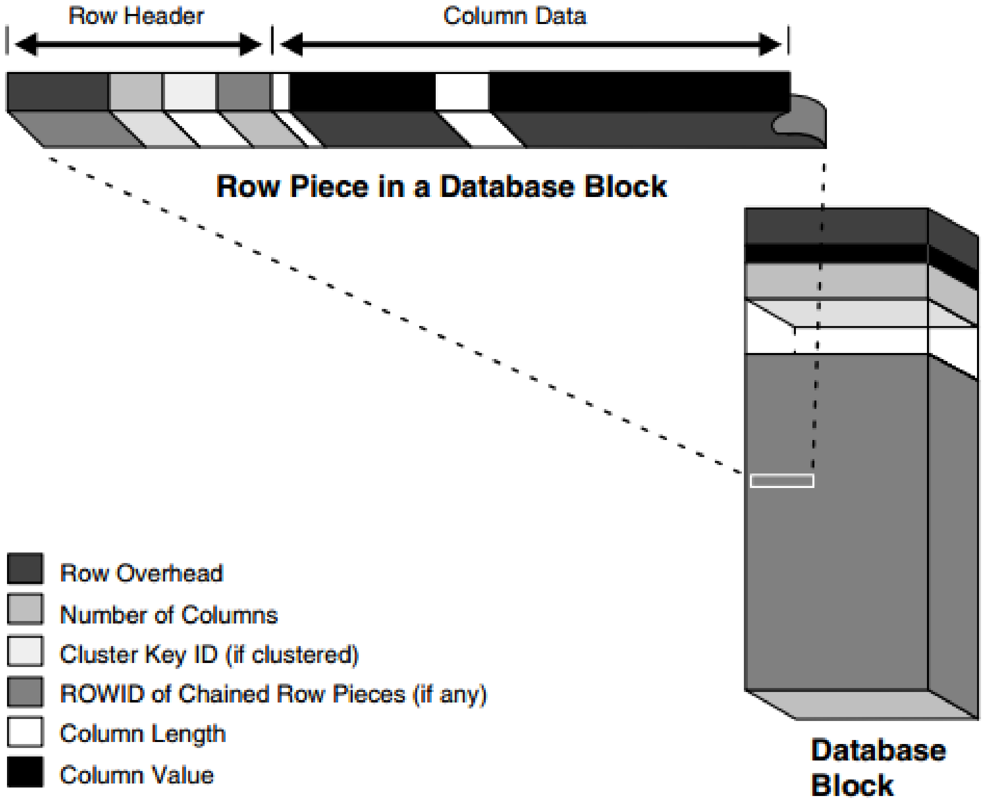
Data Block Format

Every data block has a format or internal structure that enables the database to track the data and free space in the block. This format is similar whether the data block contains table, index, or table cluster data



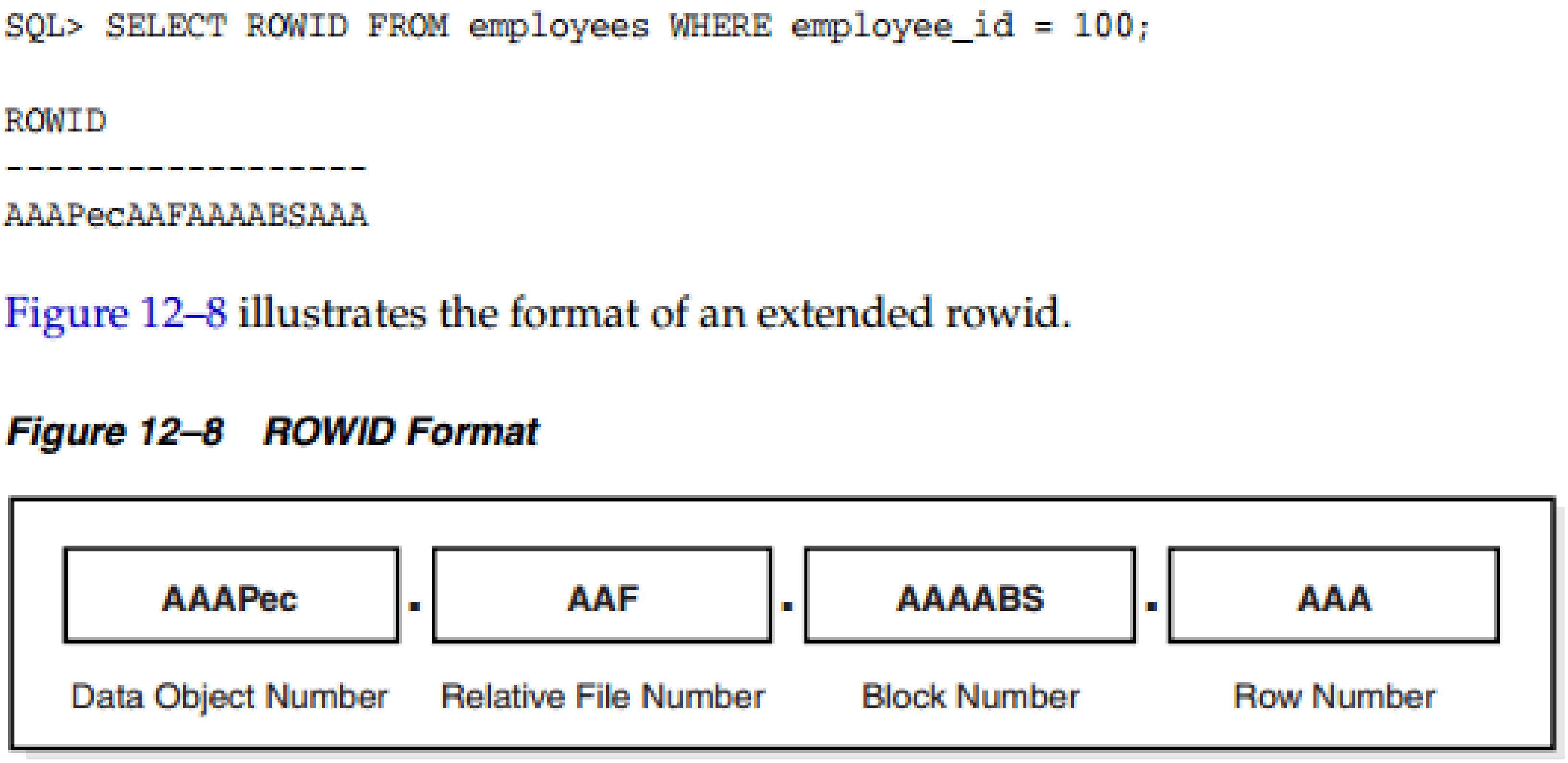
Row Format

The row data part of the block contains the actual data, such as table rows or index key entries. Just as every data block has an internal format, every row has a row format that enables the database to track the data in the row.


Row Header
Column Data
Row Piece in a Database Block
Database Block
Row Overhead
Number of Columns
Cluster Key ID (if clustered)
ROWID of Chained Row Pieces (if any)
Column Length
Column Value

RowID Pseudocolumn

Queries the ROWID pseudocolumn to show the extended rowid of the row in the employee stable for employee 100



```
SQL> SELECT ROWID FROM employees WHERE employee_id = 100;

ROWID
------------------
AAAPecAAFAAAABSAAA
```

Figure 12–8 illustrates the format of an extended rowid.

Figure 12–8 ROWID Format

AAAPec . AAF . AAAABS . AAA

Data Object Number | Relative File Number | Block Number | Row Number

CHAPTER # 86

Percentage of Free Space in Blocks (PCTFREE)

Percentage of Free Space Blocks

- What is Free Space in Data Block?
- Data Blocks & OS Blocks
- Percentage of Free Space in Data Blocks
- PCTFREE Space in Data Block
- Optimizing of Free Space
- Reuse of Index Space
- Row Migration

What is Free Space in Data Block?

- As the database fills a data block from the bottom up, the amount of free space between the row data and the block header decreases. This free space can also shrink during updates, as when changing a trailing null to a non-null value.

- The database manages free space in the data block to optimize performance and avoid wasted space (Tuning Feature).

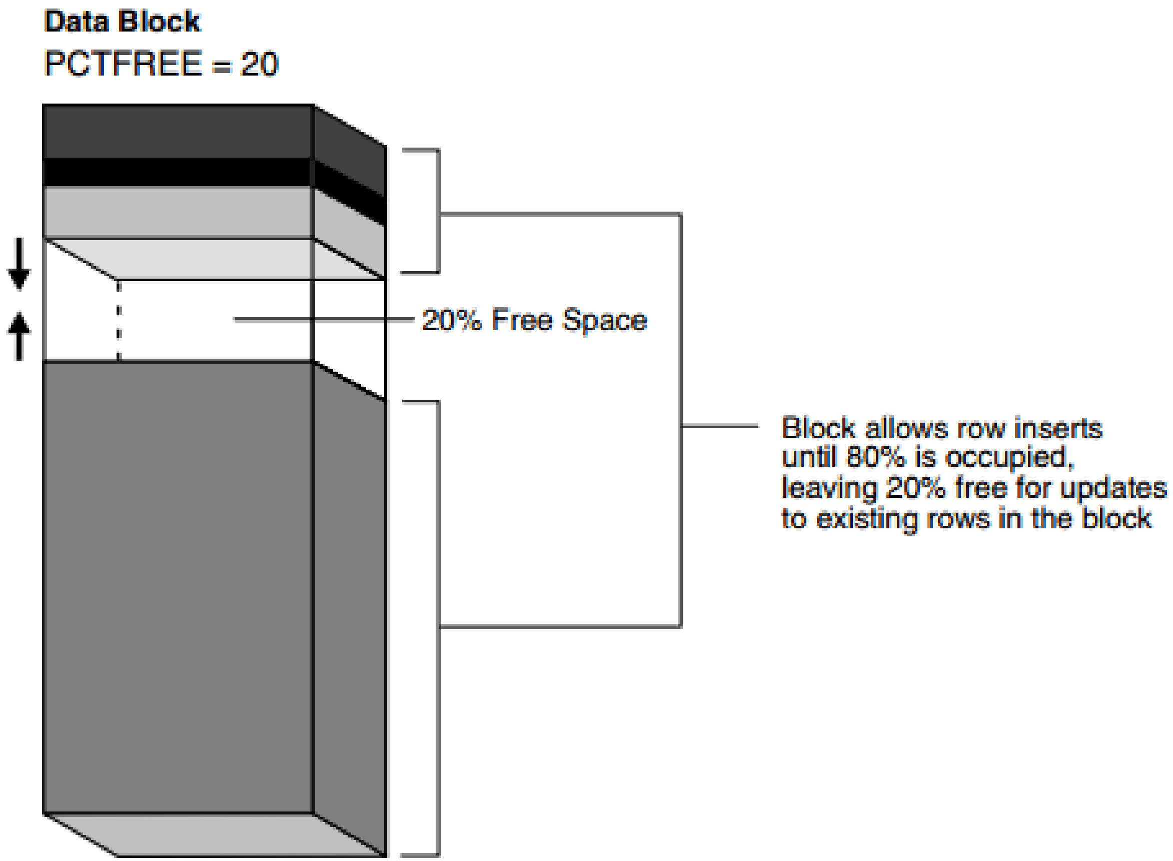
Data Blocks & OS Blocks

- At the physical level, database data is stored in disk files made up of operating system blocks.
- An operating system block is the minimum unit of data that the operating system can read or write.
- In contrast, an Oracle block is a logical storage structure whose size and structure are not known to the operating system.

Percentage of Free Space in Data Blocks

- The PCTFREE storage parameter is essential to how the database manages free space. This SQL parameter sets the minimum percentage of a data block reserved as free space for updates to existing rows.
- PCTFREE is important for preventing row migration and avoiding wasted space.

PCTFREE Space in Data Block


Data Block
PCTFREE = 20
20% Free Space
Block allows row inserts until 80% is occupied, leaving 20% free for updates to existing rows in the block

```
CREATE TABLESPACE userdata
datafile 'C:\Ora11g\ordata\usr01.dbf'
size 500M
EXTENT MANAGEMENT DICTIONARY
DEFAULT STORAGE
(Initial 1M NEXT 1M PCTINCREASE 0);
```

PCTFREE parameter does not applied on LOB Datatypes

Optimizing of Free Space

- While the percentage of free space cannot be less than PCTFREE, the amount of free space can be greater. For example, a PCTFREE setting of 20% prevents the total amount of free space from dropping to 5% of the block, but permits 50% of the block to be free space.

Optimizing of Free Space

The following SQL statements can increase free space:

- DELETE statements
- UPDATE statements that either update existing values to smaller values or increase existing values and force a row to migrate
- INSERT statements on a table that uses OLTP compression. If inserts fill a block with data, then the database invokes block compression, which may result in the block having more free space.

Optimizing of Free Space

The space released is available for INSERT under the following conditions:

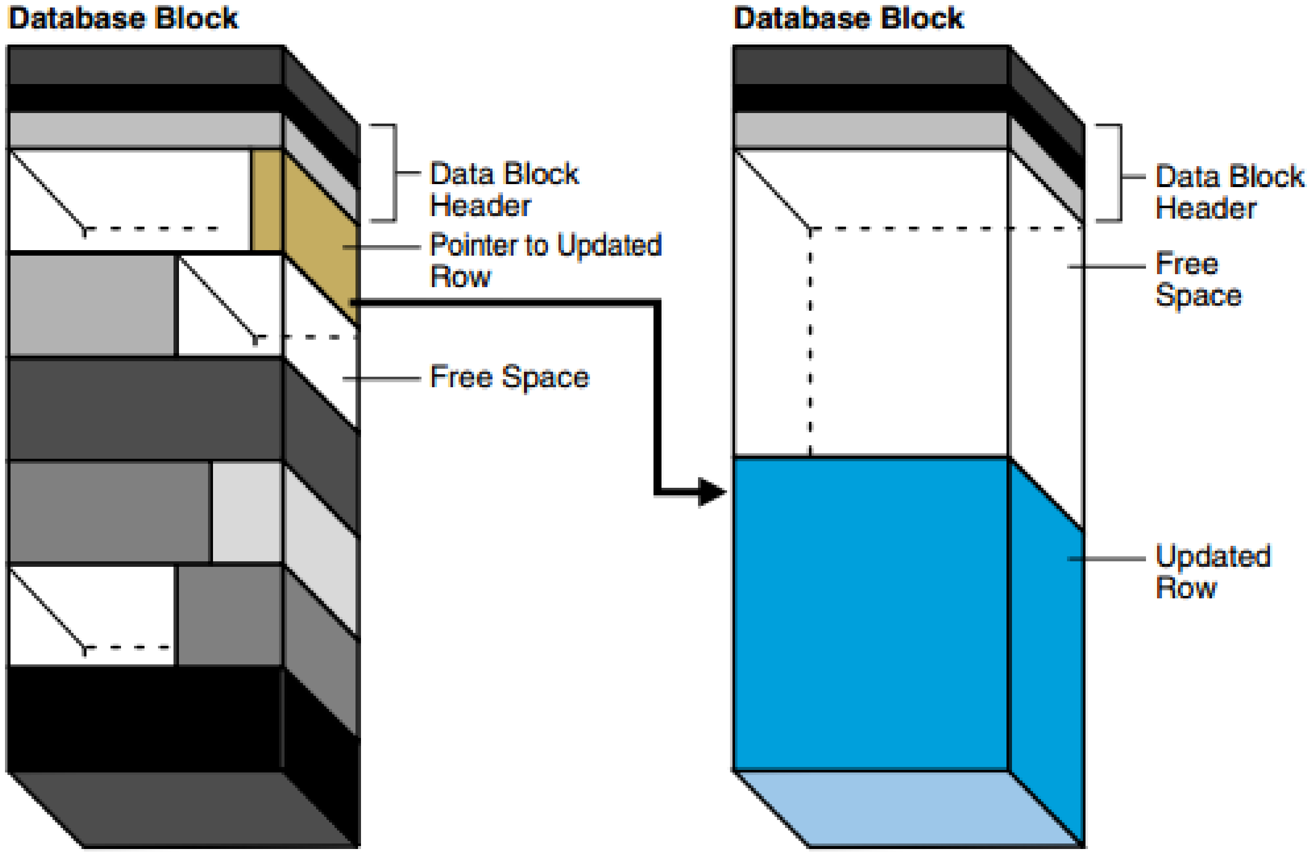
If the INSERT statement is in the same transaction and after the statement that frees space, then the statement can use the space.

If the INSERT statement is in a separate transaction from the statement that frees space (perhaps run by another user), then the statement can use the space made available only after the other transaction commits and only if the space is needed.

Reuse of Index Space

The database can reuse space within an index block.
For example, if you insert a value into a column and delete it, and if an index exists on this column, then the database can reuse the index slot when a row requires it

Row Migration



The left block contains a row that is updated so that the row is now too
large for the block. The database moves the entire row to the right block and leaves a pointer to the migrated row in the left block

CHAPTER # 87

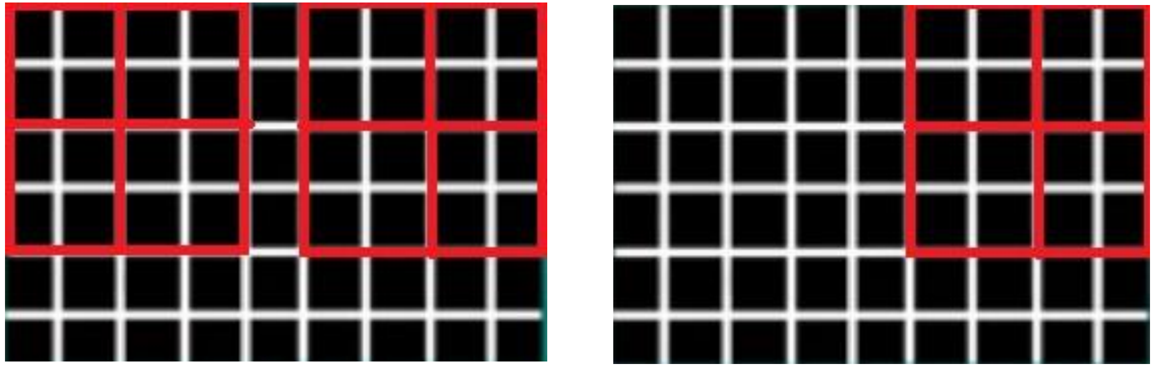
Overview of Extents

Overview of Extents

- What is Extent?
- Incremental Extents of Segment
- Deallocation of Extents
- Manually Deallocate Space
- Sample SQL DDL using Extent

What is Extent?

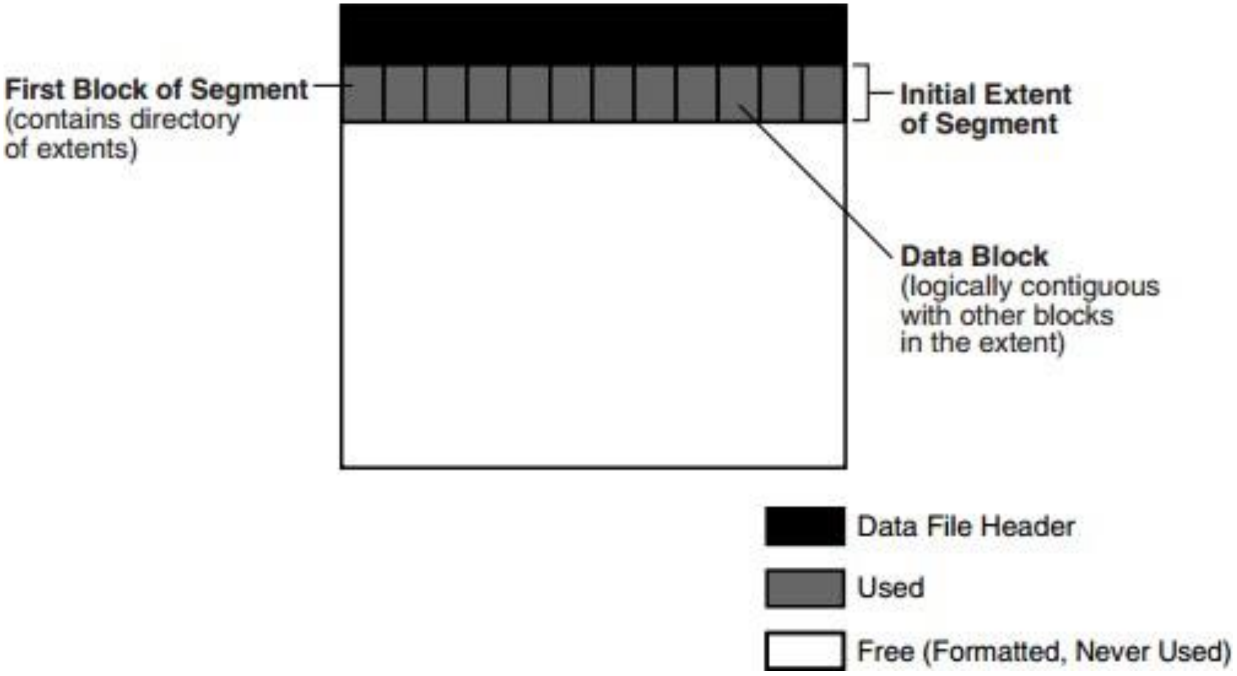
An extent is a logical unit of database storage space allocation made up of contiguous data blocks. Data blocks in an extent are logically contiguous but can be physically spread out on disk because of RAID striping and file system implementations.



What is Extent?

An extent is a logical unit of database storage space allocation made up of contiguous data blocks. Data blocks in an extent are logically contiguous but can be physically spread out on disk because of RAID striping and file system implementations.

What is Extents?


First Block of Segment (contains directory of extents)
Initial Extent of Segment
Data Block (logically contiguous with other blocks in the extent)
Data File Header
Used
Free (Formatted, Never Used)

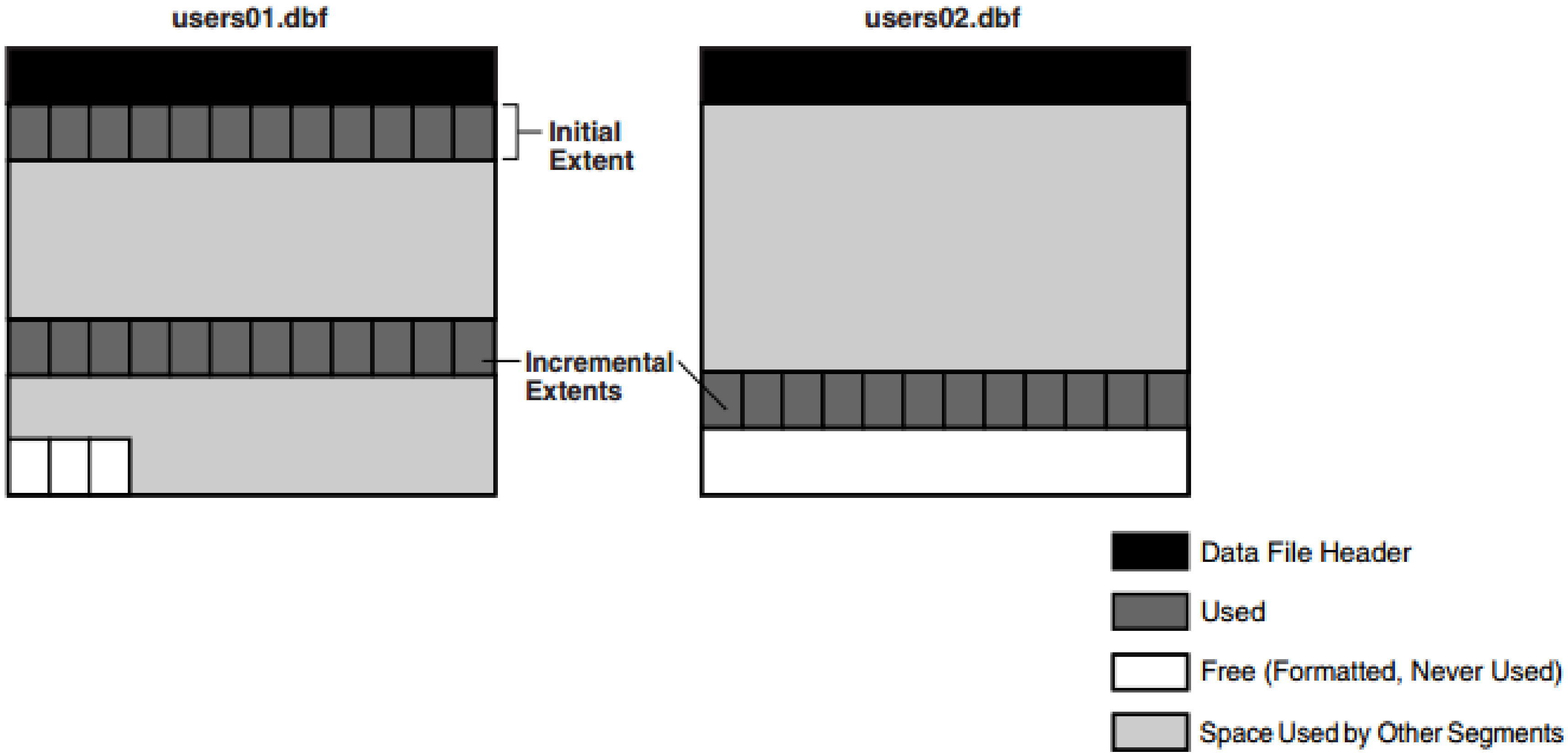
By default, the database allocates an initial extent for a data segment when the segment is created. An extent is always contained in one data file.

Although no data has been added to the segment, the data blocks in the initial extent are reserved for this segment exclusively.

The first data block of every segment contains a directory of the extents in the segment.

Figure shows the initial extent in a segment in a data file that previously contained no data.

Incremental Extents of Segment



Deallocation of Extents

In general, the extents of a user segment do not return to the tablespace unless you drop the object using a DROP command. In Oracle Database 11g Release 2, you can also drop the segment using the DBMS_SPACE_ADMIN package.

For example, if you delete all rows in a table, then the database does not reclaim the data blocks for use by other objects in the tablespace.

Manually Deallocate Space

Following techniques can deallocate extents

- You can use an online segment shrink to reclaim fragmented space in a segment. Segment shrink is an online, in-place operation.
- You can move the data of a non partitioned table or table partition into a new segment, and optionally into a different tablespace for which you have quota.

Manually Deallocate Space

- You can rebuild or coalesce the index
- You can truncate a table or table cluster, which removes all rows. By default, Oracle Database deallocates all space used by the removed rows except that specified by the MINEXTENTS storage parameter.
- You can deallocate unused space, which frees the unused space at the high water mark end of the database segment and makes the space available for other segments in the tablespace

Sample SQL DDL using Extent

```
CREATE TABLESPACE MCare_indx
DATAFILE 'D:\oracle11g\ora11g\oradata\MCare_indx01.dbf'
SIZE 100M
MINIMUM EXTENT 64K
DEFAULT STORAGE (INITIAL 64K NEXT 64K);
```

CHAPTER # 88

Overview of Segments

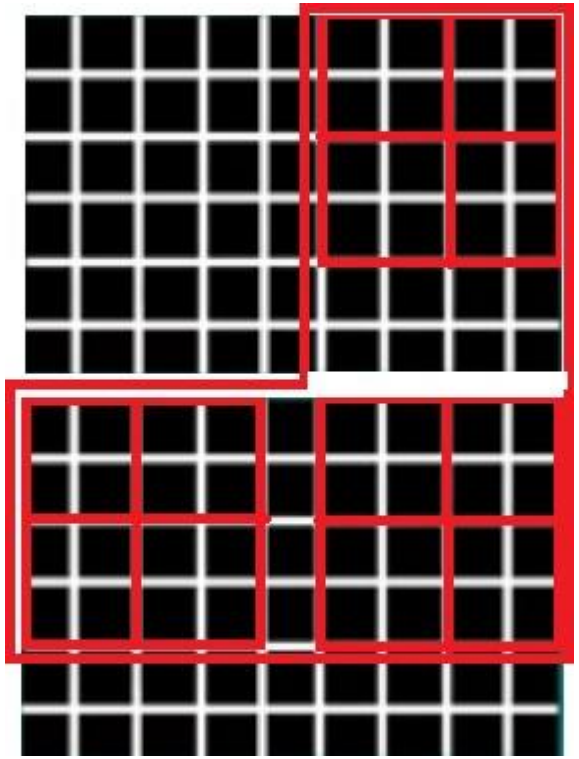
Overview of Segments

- What is Segment?
- Segment & Data Files
- User Segments
- User Segments Creation
- Creation of Multiple Segments
- SQL DDL for Segments under Tablespace
- Temporary Segments
- UNDO Segments
- Tablespace with Segments

What is Segment?

A segment is a set of extents that contains all the data for a logical storage structure within a tablespace. For example, Oracle Database allocates one or more extents to form the data segment for a table. The database also allocates one or more extents to form the index segment for a table implementations.

Segment & Data Files



Data Files

For example, Patient table consists of three segments from two data files

User Segments

A single data segment in a database stores the data for one user object. There are different types of segments. Examples of user segments include:

- Table, table partition, or table cluster
- LOB or LOB partition
- Index or index partition

Each non partitioned object and object partition is stored in its own segment. For example, if an index has five partitions, then five segments contain the index data.

User Segments Creation

By default, the database uses deferred segment creation to update only database metadata when creating tables and indexes.

When a user inserts the first row into a table or partition, the database creates segments for the table or partition, its LOB columns, and its indexes.

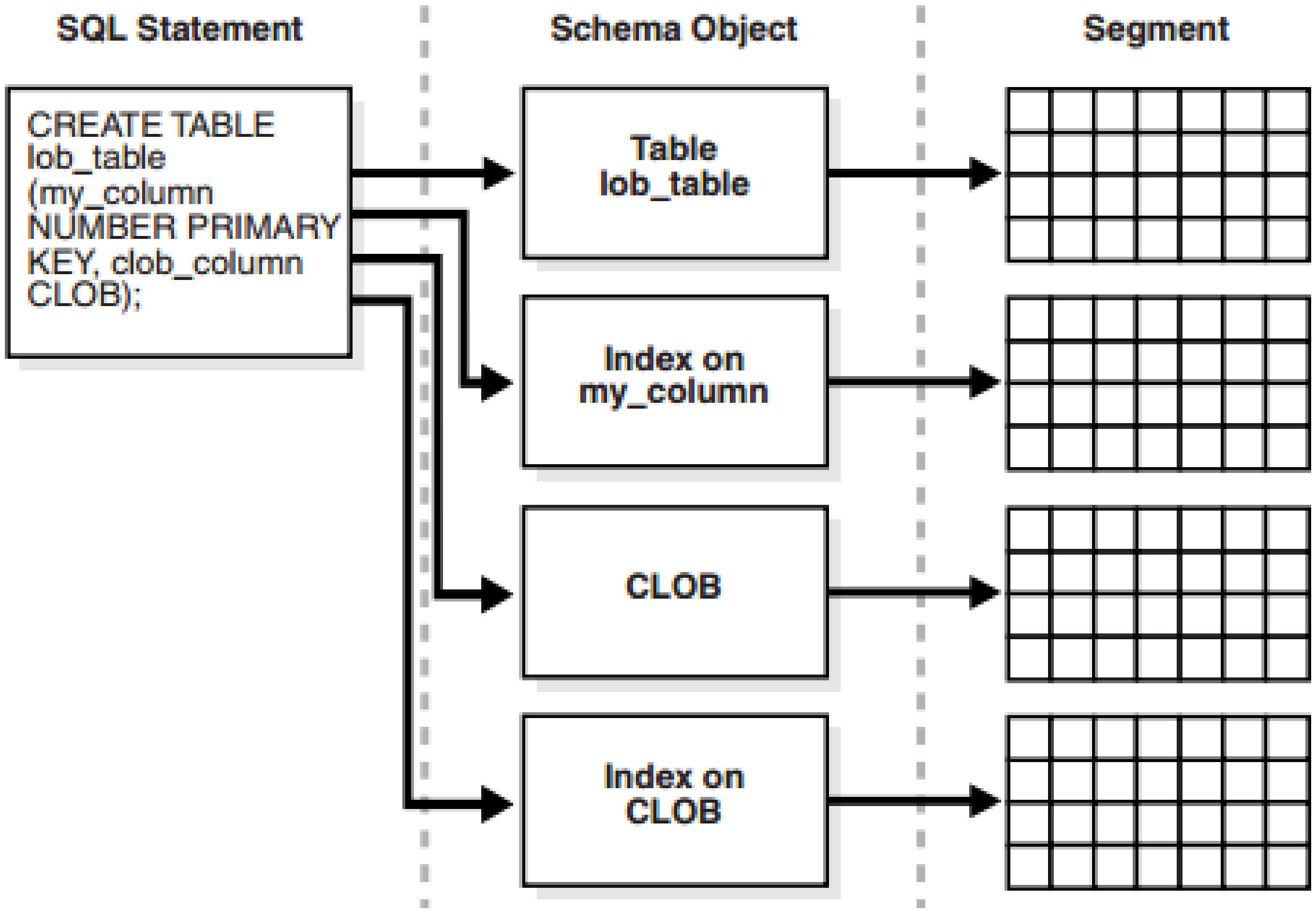
Deferred segment creation enables you to avoid using database resources unnecessarily.

User Segments Creation

For example, installation of an application can create thousands of objects, on consuming significant disk space. Many of these objects may never be used.

You can use the DBMS_SPACE_ADMIN package to manage segments for empty objects.

Creation of Multiple Segments



Join VU Group: https://chat.whatsapp.com/K8O1b4yN8XiLTwEPUdXwVq

SQL DDL for Segments under Tablespace

```
CREATE TABLE PATIENT_HISTORY
(PATIENT_ID CHAR (007),
DEPEND_SNO VARCHAR2 (002),
ALLERGY_DESC VARCHAR2 (250) NOT NULL,
HISTORY_DESC LONG
) tablespace MCare;

ALTER TABLE PATIENT_HISTORY
ADD constraint PK_PATHIST_ID PRIMARY KEY (PATIENT_ID,DEPEND_SNO)
USING INDEX TABLESPACE MCare_indx;
```

Temporary Segments

When processing a query, Oracle Database often requires temporary workspace for intermediate stages of SQL statement execution.

Typical operations that may require a temporary segment include sorting, hashing, and merging bitmaps. While creating an index, Oracle Database also places index segments into temporary segments and then converts them into permanent segments when the index is complete

UNDO Segments

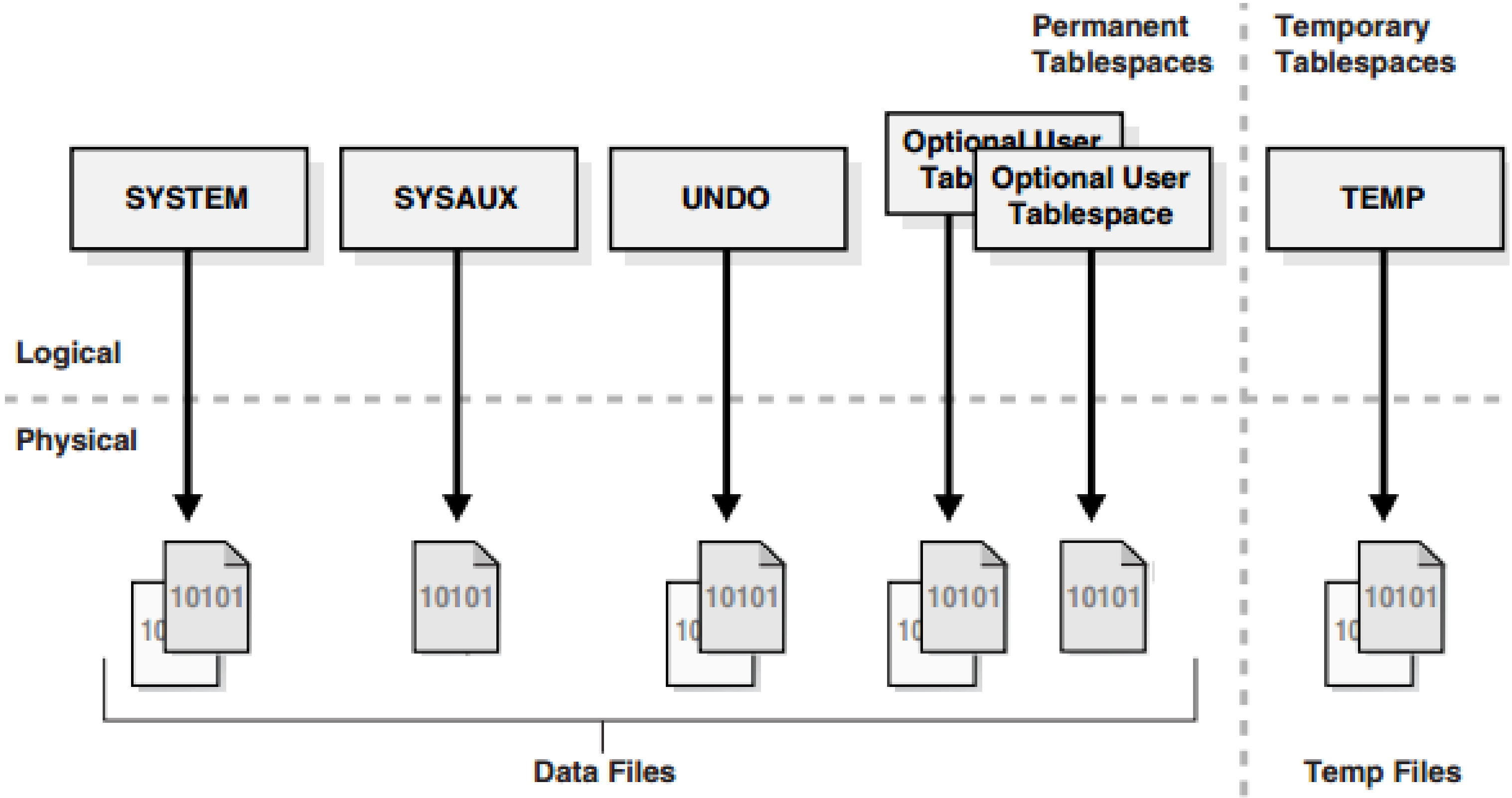
Oracle Database maintains records of the actions of transactions, collectively known as undo data.

Oracle Database uses undo to do the following:

- Roll back an active transaction
- Recover a terminated transaction
- Provide read consistency
- Perform some logical flashback operations

Tablespace with Segments

A tablespace is a logical storage container for segments. Segments are database objects, such as tables and indexes, that consume storage space. At the physical level, a tablespace stores data in one Or more data files or temp files.



CHAPTER # 89

Creating Schema Objects Tables, Views

Overview of Segments

- Introduction to Schema Objects
- User & Schema Objects
- Schema Objects Types
- Overview of Views
- Use of Views
- SQL to Create View
- View of Base Table

Introduction to Schema Objects

- A database schema is a logical container for data structures, called schema objects.

- Examples of schema objects are tables, views, sequences and indexes etc.

- Schema objects are created and manipulated with SQL

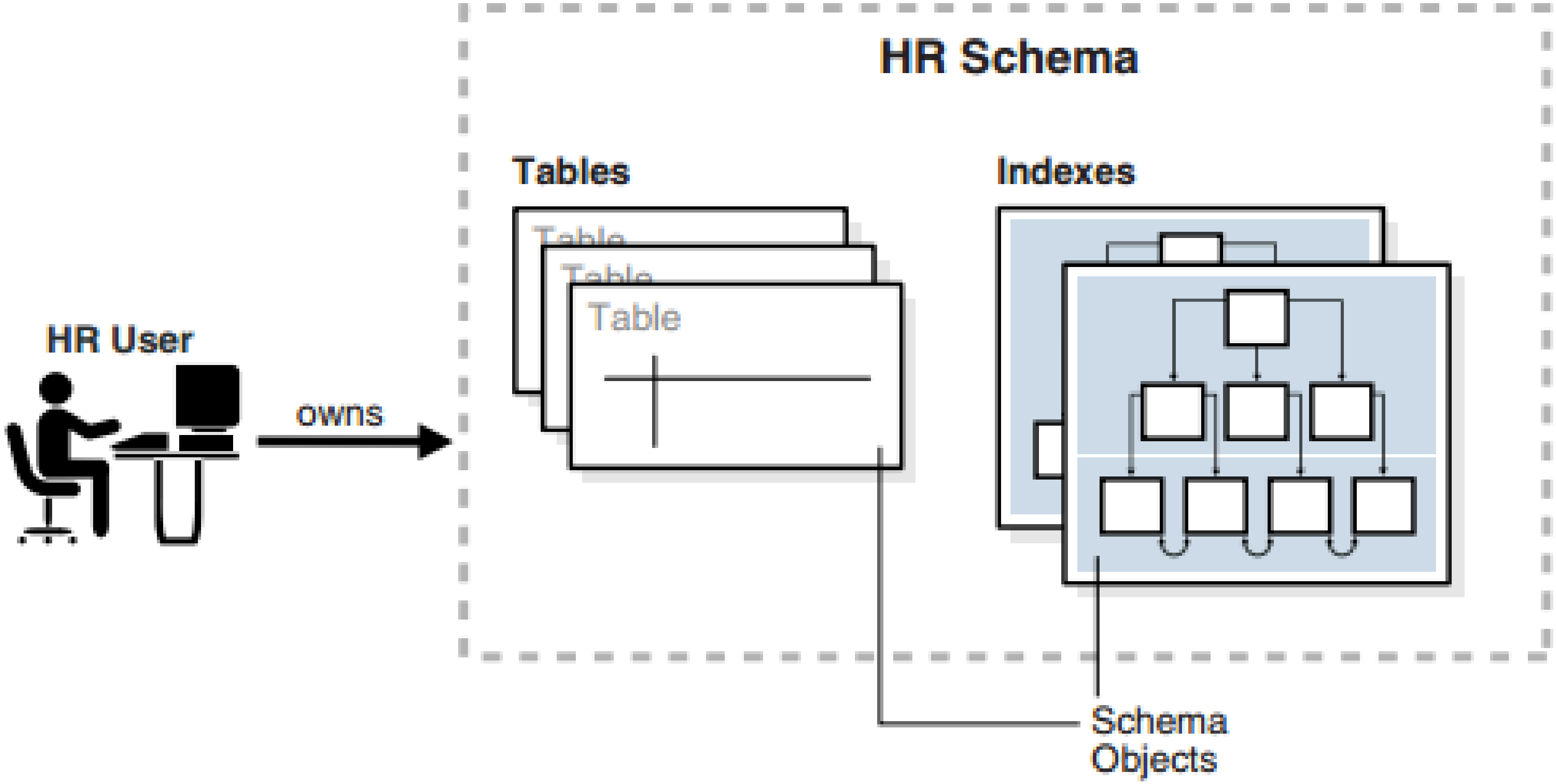
Introduction to Schema Objects

A database user has a password and various database privileges. Each user owns a single schema, which has the same name as the user.

The schema contains the data for the user owning the schema. For example, the hr user owns the hr schema, which contains schema objects such as the employees table.

In a production database, the schema owner usually represents a database application rather than a person.

User & Schema Objects



Schema Objects Types

- Tables

  Storing data in rows of tables

- Indexes

  Indexes are schema objects that contains an entry for each indexed row of the table

  or table cluster and provide direct, fast access to rows.

- Partitions

  Partitions are pieces of large tables and indexes.

Schema Objects Types

- Partitions

  Partitions are pieces of large tables and indexes.
- Views

  Views are customized presentations of data in one or more tables or other views.
- Sequences

  A sequence is a user-created object that can be shared by multiple users to generate integers.

Schema Objects Types

- Dimensions

  A dimension defines a parent-child relationship between pairs of column sets,
  where all the columns of a column set must come from the same table.
- Synonyms

  A synonym is an alias for another schema object.
- PL/SQL Programs

  Procedural server programming

Overview of Views

- A view is a logical representation of one or more tables.
- In essence, a view is a stored query.
- A view derives its data from the tables on which it is based, called base tables.
- Base tables can be tables or other views.
- All operations performed on a view actually affect the base tables.
- You can use views in most places where tables are used.

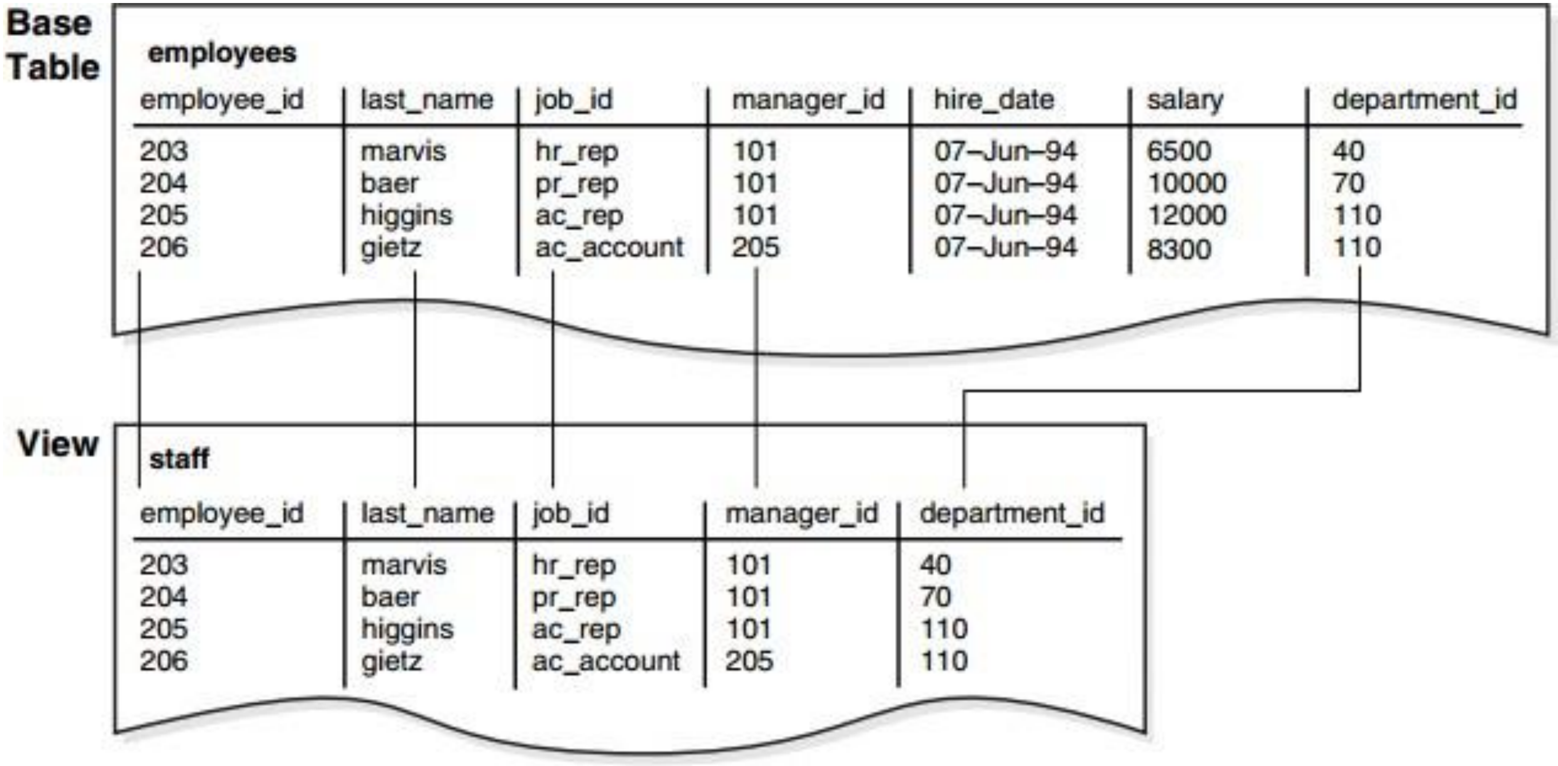
Use of Views

- Provide an additional level of table security by restricting access to a predetermined set of rows or columns of a table
- Hide data complexity
- Present the data in a different perspective from that of the base table For example, the columns of a view can be renamed without affecting the tables on which the view is based.
- Isolate applications from changes in definitions of base tables used.

SQL to Create View

```
CREATE VIEW staff AS
SELECT employee_id, last_name, job_id,
manager_id, department_id
FROM employees;
```

View of Base Table



Base Table

employees

employee_id	last_name	job_id	manager_id	hire_date	salary	department_id
203	marvis	hr_rep	101	07–Jun–94	6500	40
204	baer	pr_rep	101	07–Jun–94	10000	70
205	higgins	ac_rep	101	07–Jun–94	12000	110
206	gietz	ac_account	205	07–Jun–94	8300	110

View

staff

employee_id	last_name	job_id	manager_id	department_id
203	marvis	hr_rep	101	40
204	baer	pr_rep	101	70
205	higgins	ac_rep	101	110
206	gietz	ac_account	205	110

CHAPTER # 90

Analyzing Tables, Indexes & Clusters

Analyzing Tables, Indexes & Clusters

- Overview of Table Clusters
- Cluster Key
- Overview of Indexed Clusters
- DDL for Indexed Clusters
- Overview of Indexes
- Composite Indexes

Overview of Table Clusters

- A table cluster is a group of tables that share common columns and store related data in the same blocks. When tables are clustered, a single data block can contain rows from multiple tables.

- For example, a block can store rows from both the employees and Departments tables rather than from only a single table.

Cluster Key

- The cluster key is the column or columns that the clustered tables have in common.
- For example, the employees and departments tables share the department_id column.
- You specify the cluster key when creating the table cluster and when creating every table added to the table cluster.

Cluster Key Value

- The cluster key value is the value of the cluster key columns for a particular set of rows. All data that contains the same cluster key value, such as department_id=20, is physically stored together. Each cluster key value is stored only once in the cluster and the cluster index, no matter how many rows of different tables contain the value

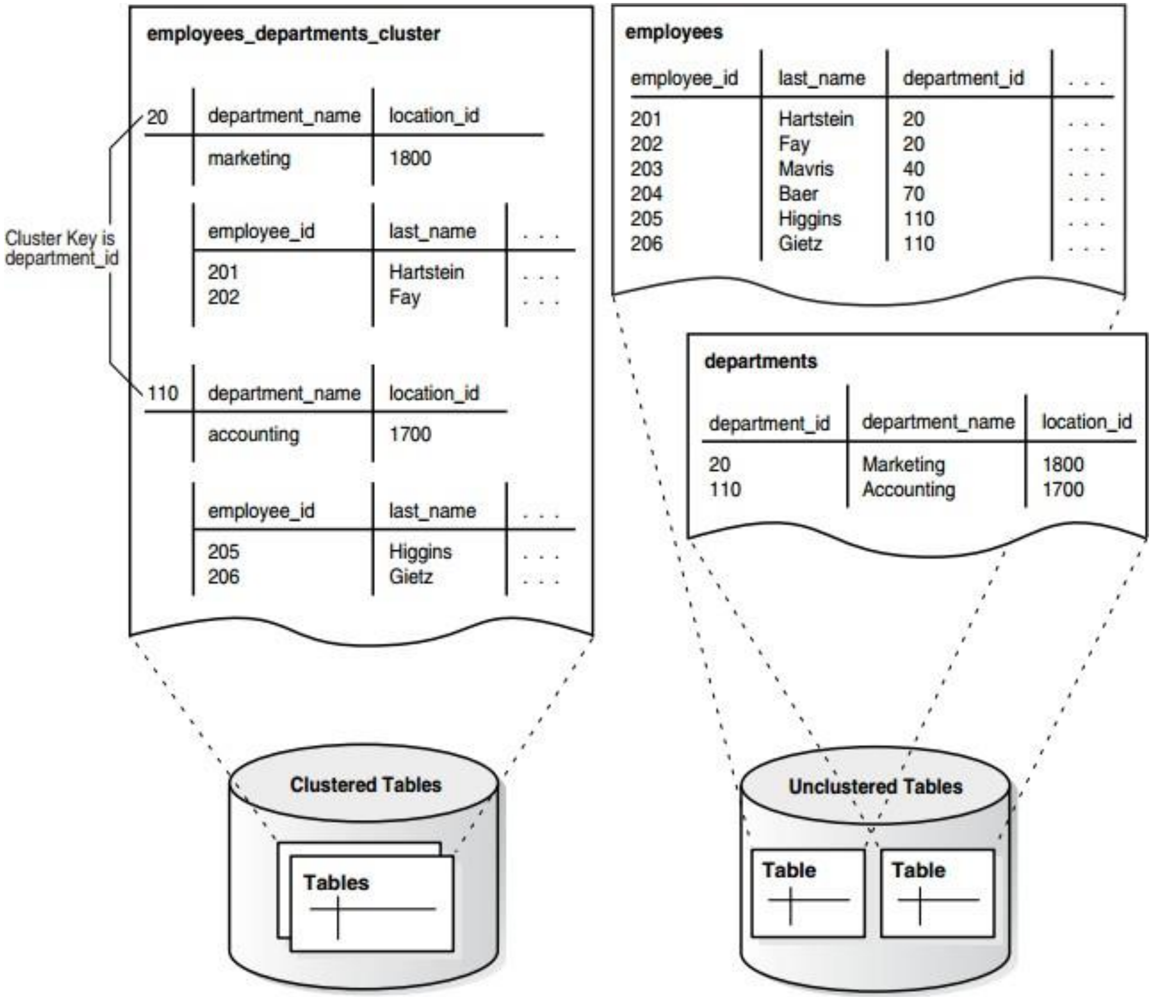
Overview of Indexed Clusters

An indexed cluster is a table cluster that uses an index to locate data. The cluster index is a B-tree index on the cluster key. A cluster index must be created before any rows can be inserted into clustered tables

DDL for Indexed Clusters

```
CREATE CLUSTER emp_dept_cluster
(department_id NUMBER(4))
SIZE 512;

CREATE INDEX idx_emp_dept_cluster ON
CLUSTER emp_dept_cluster;
```

DDL for Indexed Clusters



employees_departments_cluster

20	department_name	location_id
	marketing	1800

employee_id	last_name	. . .
201	Hartstein	. . .
202	Fay	. . .

110	department_name	location_id
	accounting	1700

employee_id	last_name	. . .
205	Higgins	. . .
206	Gietz	. . .

employees

employee_id	last_name	department_id	. . .
201	Hartstein	20	. . .
202	Fay	20	. . .
203	Mavris	40	. . .
204	Baer	70	. . .
205	Higgins	110	. . .
206	Gietz	110	. . .

departments

department_id	department_name	location_id
20	Marketing	1800
110	Accounting	1700

Overview of Indexes

An index is an optional structure, associated with a table or table cluster, that can sometimes speed data access. By creating an index on one or more columns of a table, you gain the ability in some cases to retrieve a small set of randomly distributed rows from the table.

Indexes are one of many means of reducing disk I/O.

Composite Indexes

An index is an optional structure, associated with a table or table cluster, that can sometimes speed data access. By creating an index on one or more columns of a table, you gain the ability in some cases to retrieve a small set of randomly distributed rows from the table.

Indexes are one of many means of reducing disk I/O.

Composite Indexes

A composite index, also called a concatenated index, is an index on multiple columns in a table. Columns in a composite index should appear in the order that makes the most sense for the queries that will retrieve data and need not be adjacent in the table.

```
CREATE INDEX employees_ix
ON employees (last_name, job_id, salary);
```

End

CHAPTER # 91 Managing Integrity Constraints-1

Integrity Constraints

- Data Integrity
- Techniques for Data Integrity
- Advantages of Integrity Constraints
- Types of Integrity Constraints

Data Integrity

- Business rules specify conditions and relationships that must always be true or must always be false. For example, each company defines its own policies about salaries, employee numbers, inventory tracking, and so on.

- It is important that data maintain data integrity, which is adherence to these rules, as determined by the database administrator or application developer.

Techniques for Data Integrity

Following are possible options:

- Enforcing business rules with triggered stored database procedures
- Using stored procedures to completely control access to data
- Enforcing business rules in the code of a database application
- Using Oracle Database integrity constraints, which are rules defined at the column or object level that restrict values in the database

Advantages of Integrity Constraints

An integrity constraint is a schema object that is created and dropped using SQL.

Advantages of integrity constraints over alternatives for enforcing data integrity include:

- Declarative ease
- Centralized rules
- Flexibility when loading data

Advantages of Integrity Constraints

- Declarative ease

  Because you define integrity constraints using SQL DDL statements.
- Centralized rules

  Integrity constraints are defined for tables and are stored in the data dictionary
- Flexibility when loading data

  You can disable integrity constraints temporarily to avoid performance overhead when loading large amounts of data.

Types of Integrity Constraints

Oracle Database enables you to apply constraints both at the table and column level. A constraint specified as part of the definition of a column or attribute is called an inline specification.

NOT NULL
UNIQUE
CHECK
DEFAULT
PK/ FK: Standard Integrity Constraints

End

CHAPTER # 92

Managing Integrity Constraints-2

Integrity Constraints

- NOT NULL Constraint
- UNIQUE Constraint
- CHECK Constraint
- DEFAULT Constraint
- FK CASCADE Constraints
- FK CASCADE Constraints – Example
- Trigger’s Constraint
- ENABLE/ DISABLE or Drop Constraints

NOT NULL Constraint

- A NOT NULL constraint requires that a column of a table contain no null values. A null is the absence of a value. By default, all columns in a table allow nulls.
- By default, FK is NULL.

```
Create table Student(
(RollNo          CHAR(10),
FullName         VARCHAR2(50) NOT NULL,
….. );
```

UNIQUE Constraint

- A unique key (Null able) constraint requires that every value in a column or set of columns be unique. No rows of a table may have duplicate values in a column (the unique key) or set of columns (the composite unique key) with a unique key constraint.

```
Create table Medicine(
(ItemNo          CHAR(10),
Name   VARCHAR2(50) NOT NULL UNIQUE,
….. );
```

CHECK Constraint

- A check constraint on a column or set of columns requires that a specified condition be true or unknown for every row. If DML results in the condition of the constraint evaluating to false, then the SQL statement is rolled back.

```
Alter table Medicine
Add constraint chk_medType
CHECK (MedType IN ('S','P'));
```

DEFAULT Constraint

- A default constraint on a column requires that a specified value be true when null value is entered.

```
Create table Medicine(
(ItemNo          CHAR(10),
Name             VARCHAR2(50) NOT NULL,
Status           CHAR(1) default 'Y‘,
….. );
```

FK CASCADE Constraints

- CASCADE is applied with INSERT, DELETE or UPDATE.
- For example, a foreign key with cascade delete means that if a record in the parent table is deleted, then the corresponding records in the child table will automatically be deleted. This is called a cascade delete in Oracle.

FK CASCADE Constraints - Example

```
CREATE TABLE supplier
( supplier_id numeric(10) not null,
  supplier_name varchar2(50) not null,
  contact_name varchar2(50),
  CONSTRAINT supplier_pk PRIMARY KEY (supplier_id));

CREATE TABLE products
( product_id numeric(10) not null,
  supplier_id numeric(10) not null,
  CONSTRAINT fk_supplier FOREIGN KEY (supplier_id)
    REFERENCES supplier(supplier_id)  ON DELETE CASCADE);
```

Trigger’s Constraints

```
CREATE TRIGGER cascade_del_stu_record
BEFORE DELETE ON STUDENT
FOR EACH ROW
BEGIN
   DELETE FROM RECORD
   WHERE RECORD.SID = :OLD.SID;
END;
```

Trigger’s Constraints

```
CREATE TRIGGER cascade_del_stu_record
BEFORE DELETE ON STUDENT
FOR EACH ROW
BEGIN
   DELETE FROM RECORD
   WHERE RECORD.SID = :OLD.SID;
END;
```

ENABLE/ DISABLE or Drop Constraints

Existing constraints on table can be

- Enabled
- Disabled
- Dropped

For example,
Alter table Invoice
DROP constraint fk_custid;

End

CHAPTER # 93

Schema Objects, Queries & Views

Schema Objects

- How a Schema is Created?
- How a Schema is Implemented?
- Simple & Complex Views
- Creating Tables
- Creating Sequence or AutoNumber
- Data Dictionary for Tables & Views

How a Schema is Created

- We have studied that DBA has to create default Tablespace and assign temporary Tablespace with storage in terms of data files.
- Create first user (owner of schema) with grants such as rights and privileges.
- We will cover schema objects Tables and Views.
- Ref. DBA-73

How a Schema is Implemented?

- A view is a logical representation of a table or combination of tables.
- In essence, a view is a stored query.
- A view derives its data from the tables on which it is based.
- These tables are called base tables. Base tables might in turn be actual tables or might be views themselves.
- All operations performed on a view actually affect the base table of the view.

How a Schema is Implemented?

- You can use views in almost the same way as tables. You can query, update, insert into, and delete from views, just as you can standard tables.
- Views can provide a different representation (such as subsets or supersets) of the data that resides within other tables and views.
- Views can be Simple or Complex
- Views can be used for security purposes
- View does not contain its own data

Simple & Complex Views

```
CREATE VIEW sales_staff AS
SELECT empno, ename, deptno
FROM emp
WHERE deptno = 10;
```

```
CREATE VIEW sales_staff AS
SELECT job, count(*)
FROM emp
WHERE deptno = 10
Group by job;
```

Simple & Complex Views

```
CREATE OR REPLACE VIEW division1_staff
AS
SELECT ename, empno, job, dname
FROM emp, dept
WHERE emp.deptno IN (10, 30)
AND emp.deptno = dept.deptno;
```

Insert, delete and update operations for base tables are applied on simple views.

Creating Tables

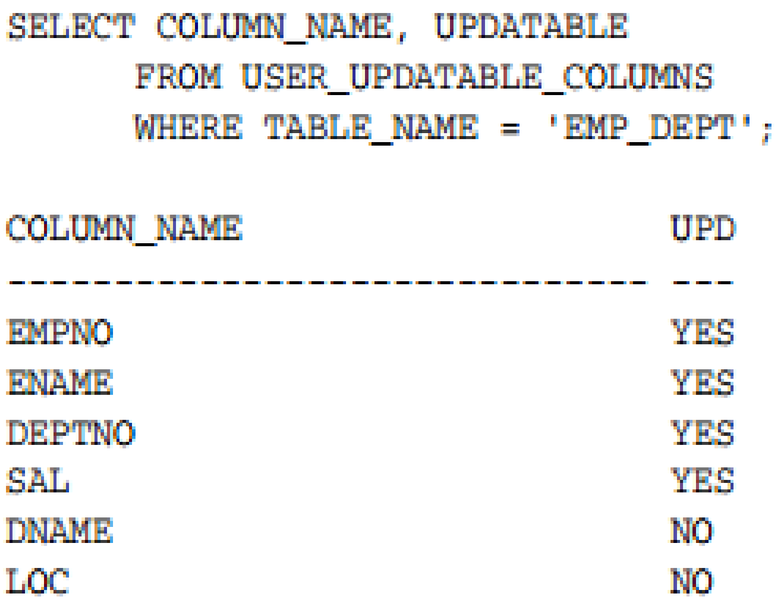
```
CREATE TABLE dept (
deptno NUMBER(4) PRIMARY KEY,
dname VARCHAR2(14),
loc VARCHAR2(13));

CREATE TABLE emp (
empno NUMBER(4) PRIMARY KEY,
ename VARCHAR2(10),
…
deptno NUMBER(2);
…
```

Creating Sequence or AutoNumber

```
CREATE SEQUENCE emp_sequence
INCREMENT BY 1
START WITH 1
NOMAXVALUE
NOCYCLE
CACHE 10;
```

Data Dictionary for Tables & Views

SELECT COLUMN_NAME, UPDATABLE
FROM USER_UPDATABLE_COLUMNS
WHERE TABLE_NAME = 'EMP_DEPT';



```
SELECT COLUMN_NAME, UPDATABLE
      FROM USER_UPDATABLE_COLUMNS
      WHERE TABLE_NAME = 'EMP_DEPT';

COLUMN_NAME                    UPD
------------------------------ ---
EMPNO                          YES
ENAME                          YES
DEPTNO                         YES
SAL                            YES
DNAME                          NO
LOC                            NO
```

End

CHAPTER # 94

Creating Tables & Alter Tables

Creating & Alter Tables

- What is a Table?
- Design Table Guidelines
- Heap-Organized Table
- Clustered Table
- Indexed-Organized Table
- Partitioned Table
- Storage Location of a Table
- Alter Table
- Tables with Segment Creation

What is a Table?

- Tables are the basic unit of data storage in an Oracle Database. Data is stored in rows and columns. You define a table with a table name, such as employees, and a set of columns.
- You give each column a column name, such as emp_id, last_name, and job_id; a data type, such as VARCHAR2, DATE, or NUMBER; and a width. The width can be predetermined by the data type, as in DATE.

What is a Table?

- If columns are of the NUMBER data type, define precision and scale instead of width. A row is a collection of column information corresponding to a single record.
- You can specify rules for each column of a table. These rules are called integrity constraints. One example is a NOT NULL integrity constraint. This constraint forces the column to contain a value in every row.
- Some column types, such as LOBs, varrays, and nested tables, are stored in their own segments. LOBs and varrays are stored in LOB segments, while nested tables are stored in storage tables.

Design Table Guidelines

- Use descriptive names for tables, columns, indexes, and clusters.
- Be consistent in abbreviations and in the use of singular and plural forms of table names and columns.
- Document the meaning of each table and its columns with the COMMENT command.
- Normalize each table.

Design Table Guidelines

- Select the appropriate datatype for each column. Consider whether your applications would benefit from adding one or more virtual columns to some tables.
- Define columns that allow nulls last, to conserve storage space.
- Cluster tables whenever appropriate, to conserve storage space and optimize performance of SQL statements.

Heap-Organized Table

- This is the basic, general purpose type of table which is the primary subject of this chapter. Its data (heap).

Clustered Table

- A clustered table is a table that is part of a cluster. A cluster is a group of tables that share the same data blocks because they share common columns and are often used together

Indexed-Organized Table

- Unlike an ordinary (heap-organized) table, data for an index-organized table is stored in a B-tree index structure in a primary key sorted manner.
- Besides storing the primary key column values of an index-organized table row, each index entry in the B-tree stores the non key column values as well

Partitioned Table

- Partitioned tables enable your data to be broken down into smaller, more manageable pieces called partitions, or even subpartitions. Each partition can have separate physical attributes, such as compression enabled or disabled, type of compression, physical storage settings, and tablespace, thus providing a structure that can be better tuned for availability and performance.

Partitioned Table

- In addition, each partition can be managed individually, which can simplify and reduce the time required for backup and administration.

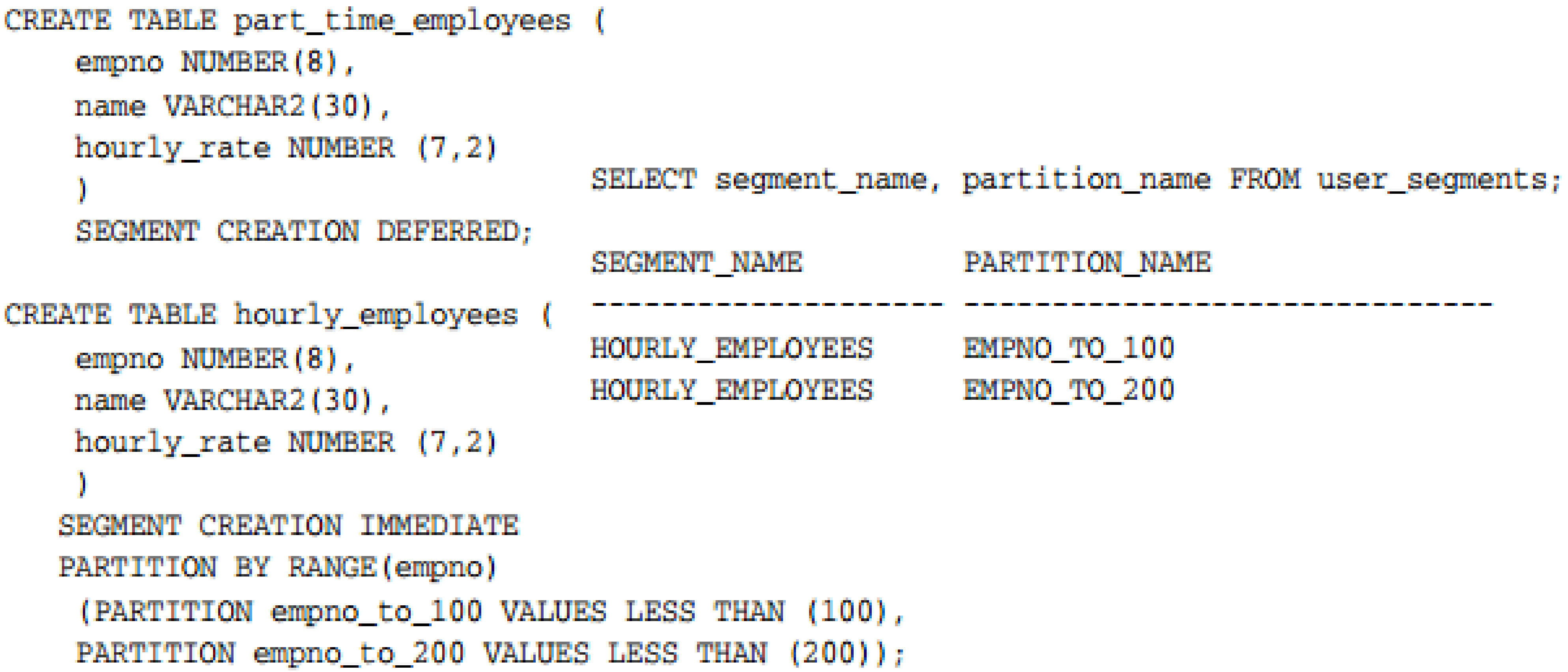
Storage Location of a Table

- It is advisable to specify the TABLESPACE clause in a CREATE TABLE statement to identify the tablespace that is to store the new table.
- For partitioned tables, you can optionally identify the tablespace that is to store each partition. Ensure that you have the appropriate privileges and quota on any tablespaces that you use.
- If you do not specify a tablespace in a CREATE TABLE statement, the table is created in your default tablespace.

Alter Table

- Alter table command applies when table is already exist in a schema.
- We can add constraints, columns and modifications operations on table using Alter table DDL statements.

Tables with Segment Creation



```
CREATE TABLE part_time_employees (
    empno NUMBER(8),
    name VARCHAR2(30),
    hourly_rate NUMBER (7,2)
    )
    SEGMENT CREATION DEFERRED;

CREATE TABLE hourly_employees (
    empno NUMBER(8),
    name VARCHAR2(30),
    hourly_rate NUMBER (7,2)
    )
   SEGMENT CREATION IMMEDIATE
   PARTITION BY RANGE(empno)
    (PARTITION empno_to_100 VALUES LESS THAN (100),
    PARTITION empno_to_200 VALUES LESS THAN (200));
```

```
SELECT segment_name, partition_name FROM user_segments;

SEGMENT_NAME         PARTITION_NAME
-------------------- ------------------------------
HOURLY_EMPLOYEES     EMPNO_TO_100
HOURLY_EMPLOYEES     EMPNO_TO_200
```

CHAPTER # 95

DDL Script Writing-1

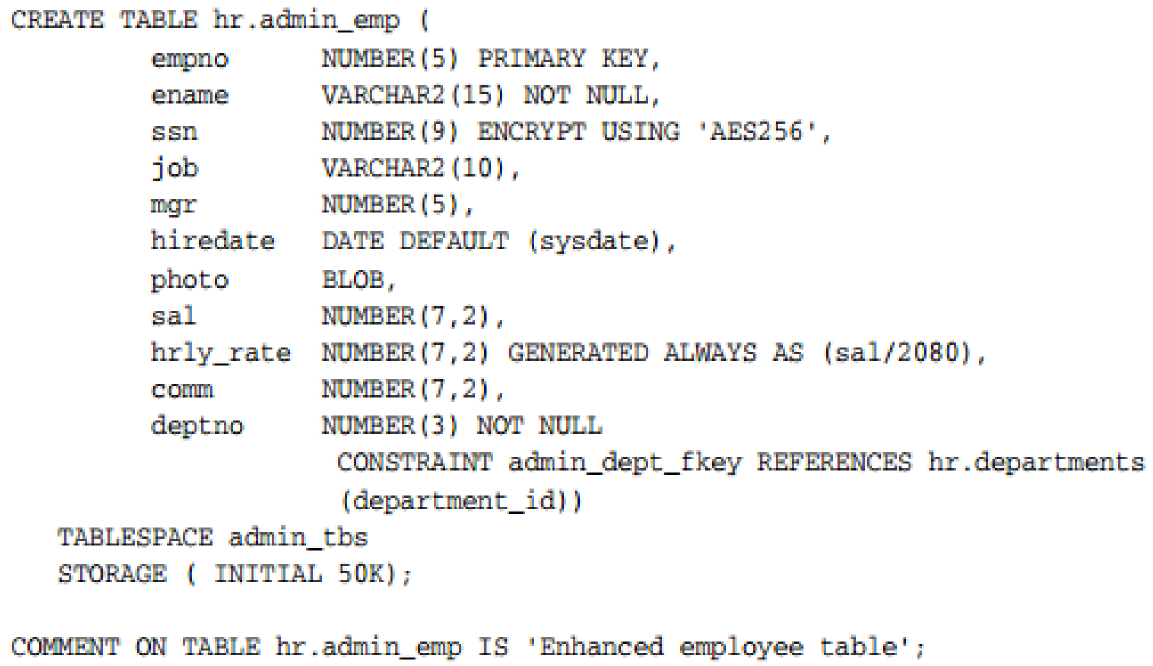
Script Writing

- What is a Script?
- A Sample DDL in a Script?
- Creating Temporary Tables
- Table with TEMP Tablespace
- Bulk Data Load in a Table
- Alter Table
- Index with Tablespace
- DDL Partition Table

What is a Script?

- Normally writing or generating DDL commands to create/ alter tables, views, constraints and storages etc., in an order in plain text using Notepad or text files
- Mostly scripts contain structure of tables. Sometimes, DML commands are also put in scripts as per management requirement.
- Scripts can be with Create, Alter and Drop Tables

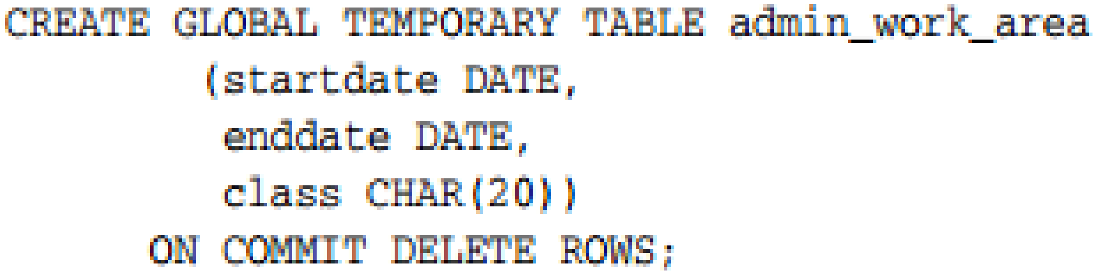
A Sample DDL in a Script



```
CREATE TABLE hr.admin_emp (
         empno      NUMBER(5) PRIMARY KEY,
         ename      VARCHAR2(15) NOT NULL,
         ssn        NUMBER(9) ENCRYPT USING 'AES256',
         job        VARCHAR2(10),
         mgr        NUMBER(5),
         hiredate   DATE DEFAULT (sysdate),
         photo      BLOB,
         sal        NUMBER(7,2),
         hrly_rate  NUMBER(7,2) GENERATED ALWAYS AS (sal/2080),
         comm       NUMBER(7,2),
         deptno     NUMBER(3) NOT NULL
                     CONSTRAINT admin_dept_fkey REFERENCES hr.departments
                     (department_id))
   TABLESPACE admin_tbs
   STORAGE ( INITIAL 50K);

COMMENT ON TABLE hr.admin_emp IS 'Enhanced employee table';
```

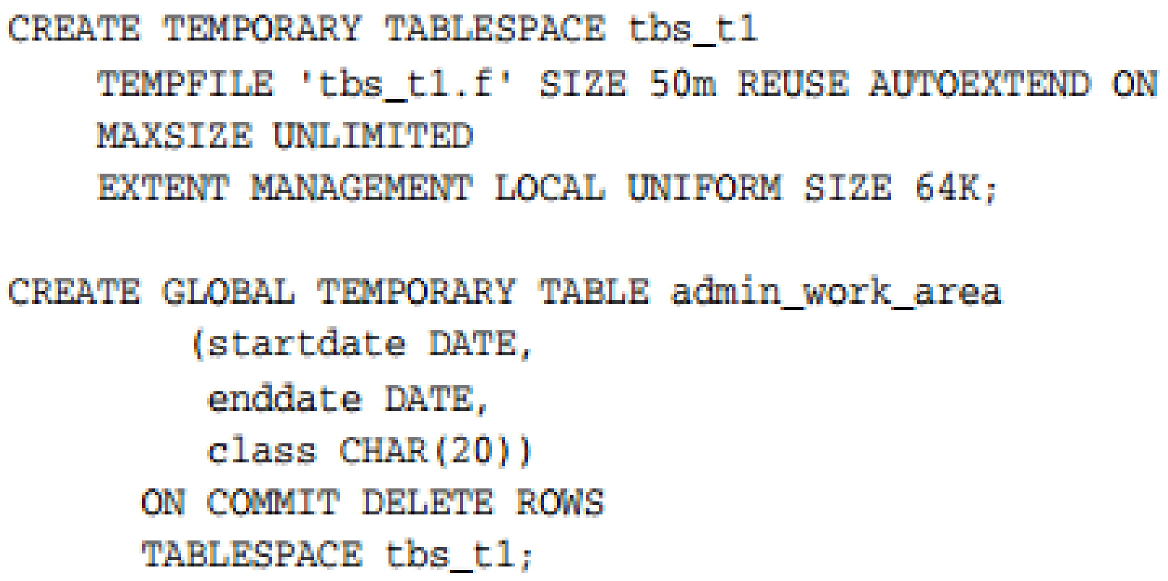
Creating Temporary Tables

Temporary tables are useful in applications where a result set is to be buffered (temporarily persisted), perhaps because it is constructed by running multiple DML operations.
For example, consider the following:



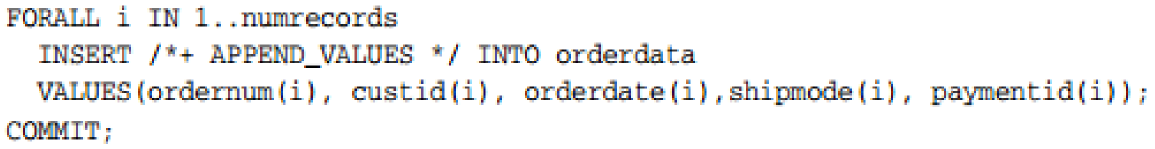
```
CREATE GLOBAL TEMPORARY TABLE admin_work_area
        (startdate DATE,
         enddate DATE,
         class CHAR(20))
      ON COMMIT DELETE ROWS;
```

Table with TEMP Tablespace



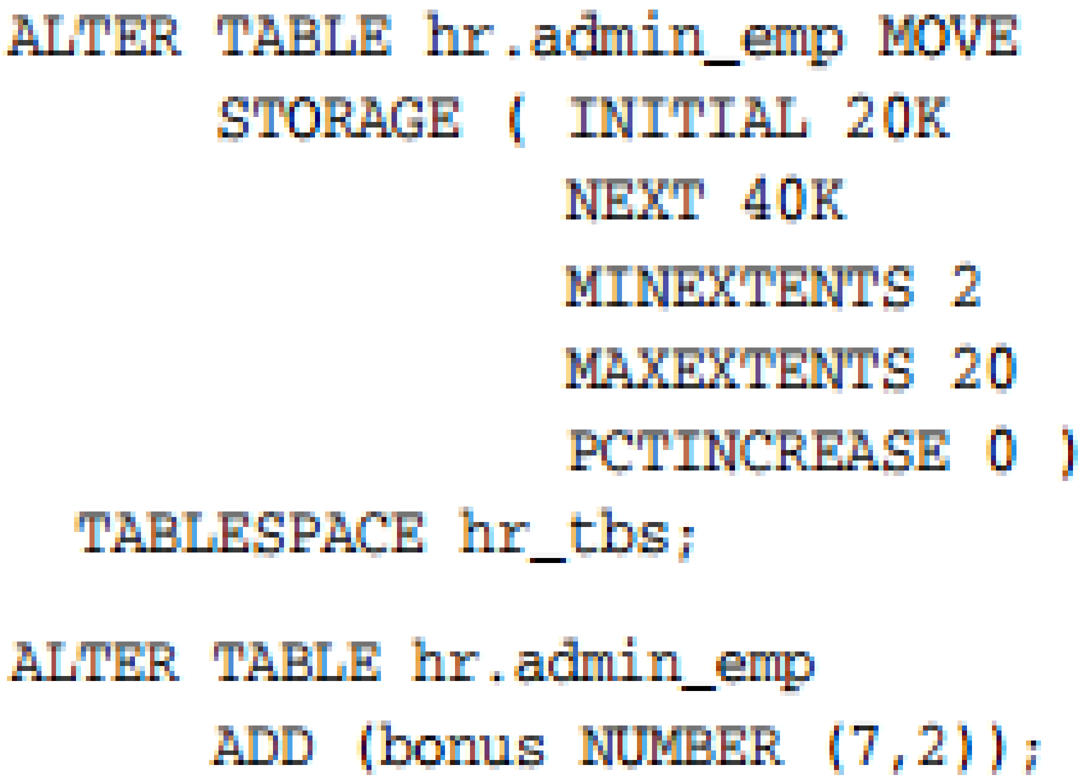
```
CREATE TEMPORARY TABLESPACE tbs_t1
    TEMPFILE 'tbs_t1.f' SIZE 50m REUSE AUTOEXTEND ON
    MAXSIZE UNLIMITED
    EXTENT MANAGEMENT LOCAL UNIFORM SIZE 64K;

CREATE GLOBAL TEMPORARY TABLE admin_work_area
        (startdate DATE,
         enddate DATE,
         class CHAR(20))
      ON COMMIT DELETE ROWS
      TABLESPACE tbs_t1;
```

Bulk Data Load in a Table



```
FORALL i IN 1..numrecords
  INSERT /*+ APPEND_VALUES */ INTO orderdata
  VALUES(ordernum(i), custid(i), orderdate(i),shipmode(i), paymentid(i));
COMMIT;
```

Alter Table

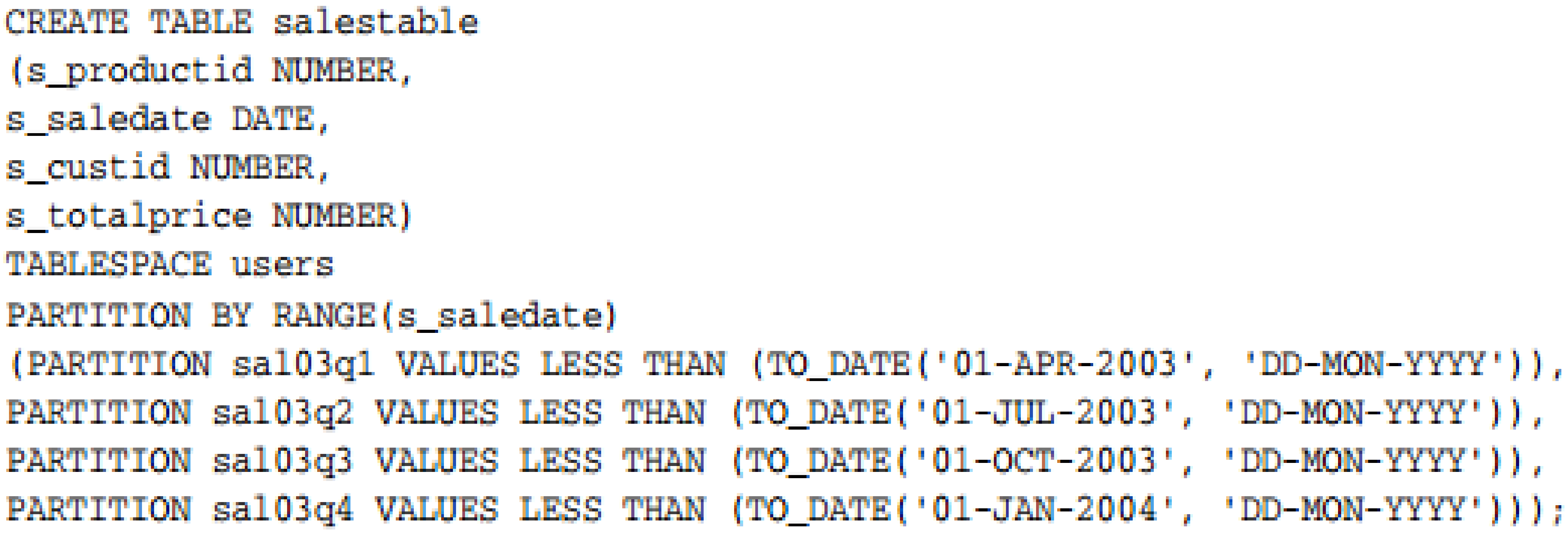


```
ALTER TABLE hr.admin_emp MOVE
      STORAGE ( INITIAL 20K
                NEXT 40K
                MINEXTENTS 2
                MAXEXTENTS 20
                PCTINCREASE 0 )
  TABLESPACE hr_tbs;

ALTER TABLE hr.admin_emp
      ADD (bonus NUMBER (7,2));
```

Index with Tablespace

```
CREATE INDEX int_sales_index ON
int_salestable
(s_saledate, s_productid, s_custid)
TABLESPACE tbs_low_freq;
```

DDL Partition Table



```
CREATE TABLE salestable
(s_productid NUMBER,
s_saledate DATE,
s_custid NUMBER,
s_totalprice NUMBER)
TABLESPACE users
PARTITION BY RANGE(s_saledate)
(PARTITION sal03q1 VALUES LESS THAN (TO_DATE('01-APR-2003', 'DD-MON-YYYY')),
PARTITION sal03q2 VALUES LESS THAN (TO_DATE('01-JUL-2003', 'DD-MON-YYYY')),
PARTITION sal03q3 VALUES LESS THAN (TO_DATE('01-OCT-2003', 'DD-MON-YYYY')),
PARTITION sal03q4 VALUES LESS THAN (TO_DATE('01-JAN-2004', 'DD-MON-YYYY')));
```

CHAPTER # 96

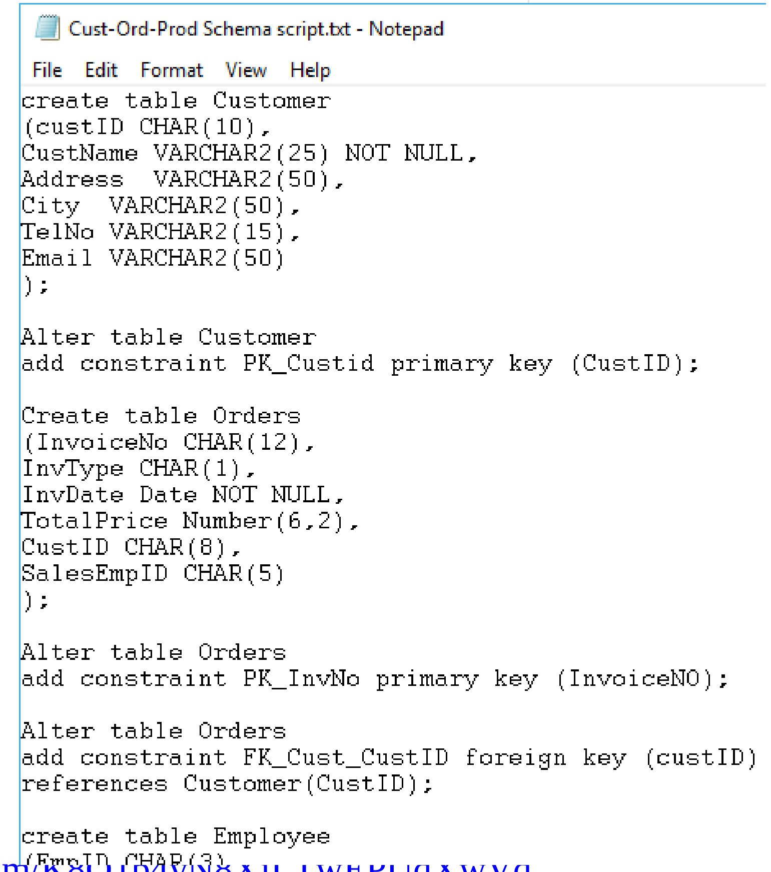
DDL Script Writing-2

Script Writing

- Writing Script using Notepad
- Tables in an Order
- Drop Tables in an Order
- Other Schema Objects

Writing Script using Notepad

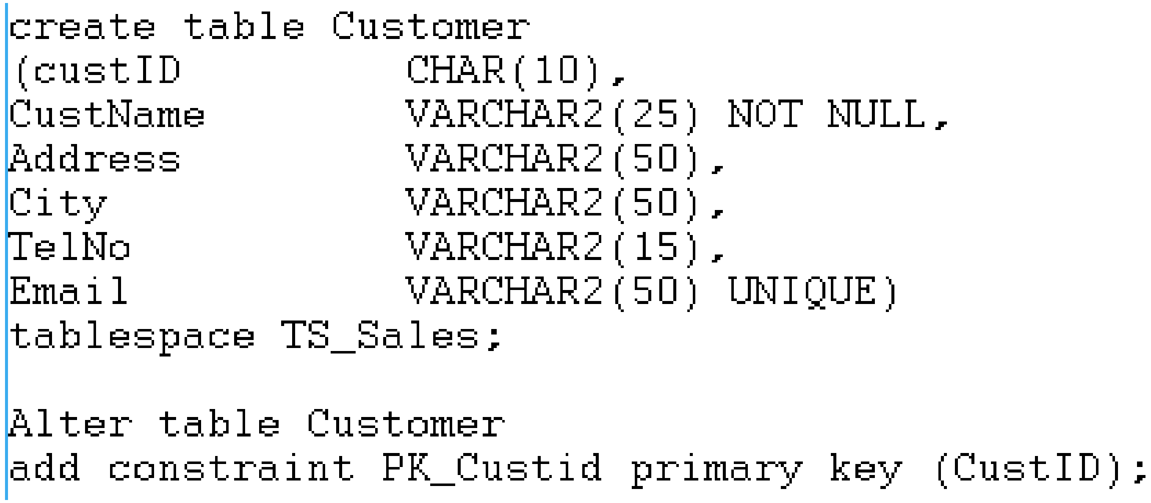
- Script is a plain text structure of schema



Cust-Ord-Prod Schema script.txt - Notepad
File Edit Format View Help

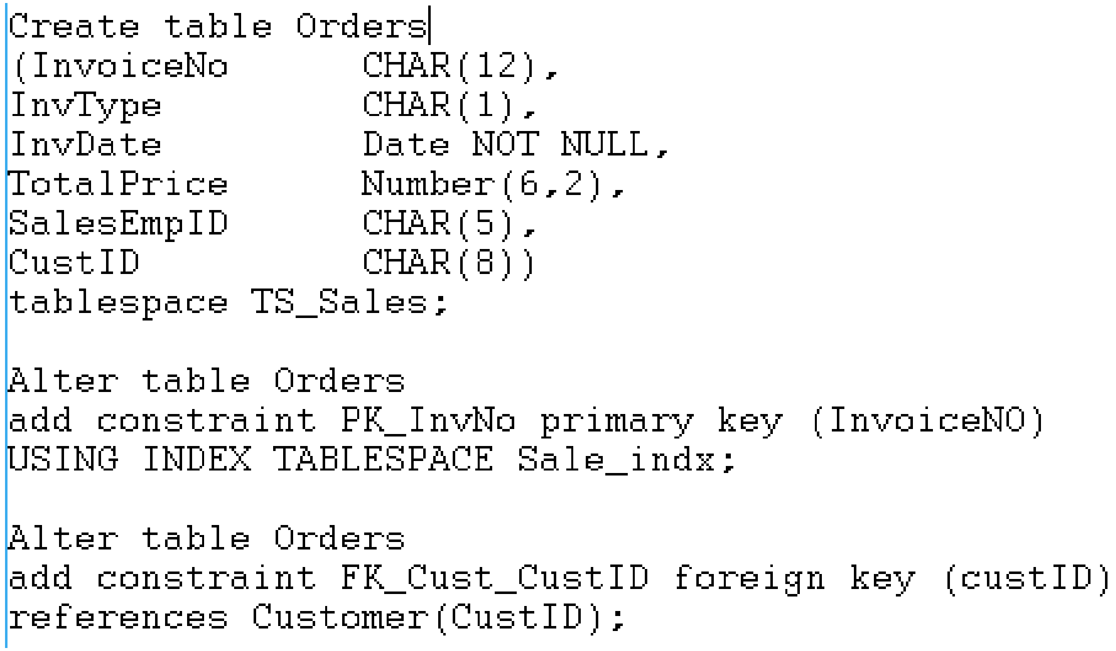
```
create table Customer
(custID CHAR(10),
CustName VARCHAR2(25) NOT NULL,
Address  VARCHAR2(50),
City  VARCHAR2(50),
TelNo VARCHAR2(15),
Email VARCHAR2(50)
);

Alter table Customer
add constraint PK_Custid primary key (CustID);

Create table Orders
(InvoiceNo CHAR(12),
InvType CHAR(1),
InvDate Date NOT NULL,
TotalPrice Number(6,2),
CustID CHAR(8),
SalesEmpID CHAR(5)
);

Alter table Orders
add constraint PK_InvNo primary key (InvoiceNO);

Alter table Orders
add constraint FK_Cust_CustID foreign key (custID)
references Customer(CustID);

create table Employee
(EmpID CHAR(3)
```

Tables in an Order



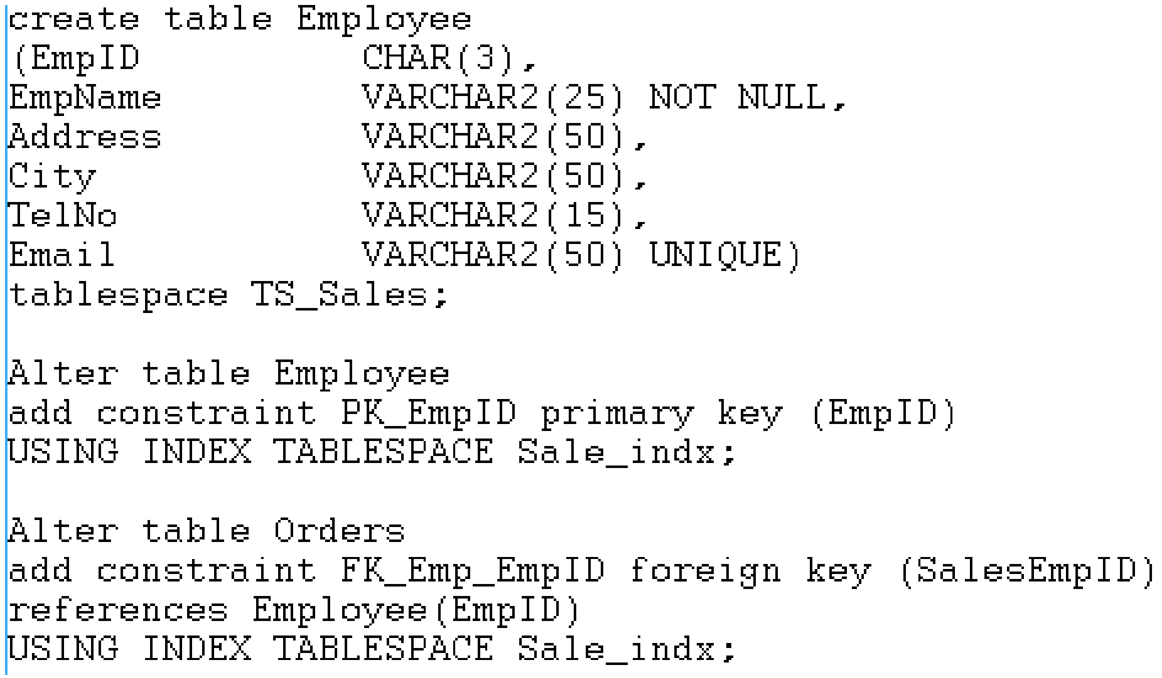
```
create table Customer
(custID         CHAR(10),
CustName        VARCHAR2(25) NOT NULL,
Address         VARCHAR2(50),
City            VARCHAR2(50),
TelNo           VARCHAR2(15),
Email           VARCHAR2(50) UNIQUE)
tablespace TS_Sales;

Alter table Customer
add constraint PK_Custid primary key (CustID);
```

Tables in an Order



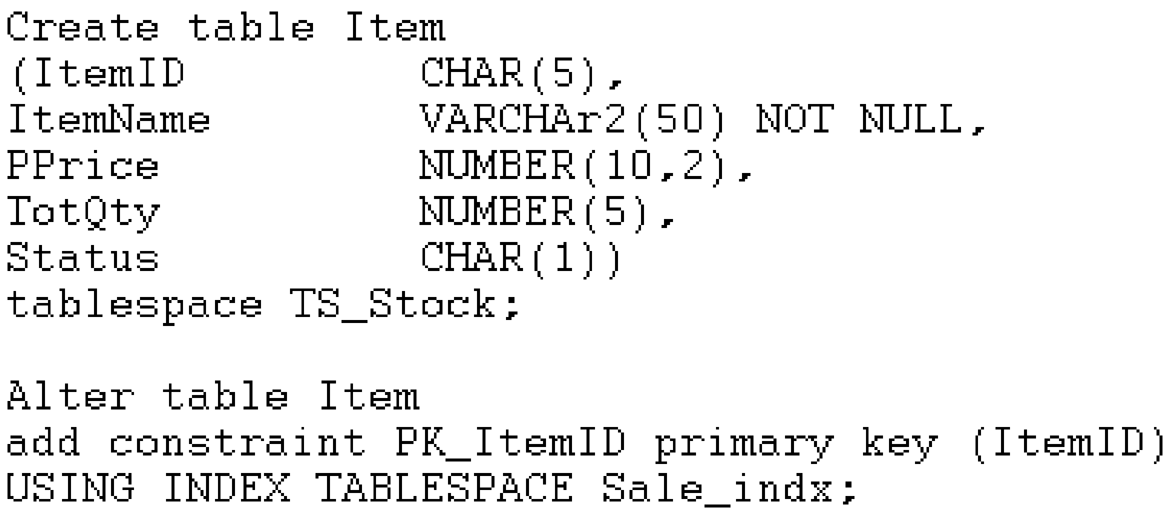
```
Create table Orders
(InvoiceNo      CHAR(12),
InvType         CHAR(1),
InvDate         Date NOT NULL,
TotalPrice      Number(6,2),
SalesEmpID      CHAR(5),
CustID          CHAR(8))
tablespace TS_Sales;

Alter table Orders
add constraint PK_InvNo primary key (InvoiceNO)
USING INDEX TABLESPACE Sale_indx;

Alter table Orders
add constraint FK_Cust_CustID foreign key (custID)
references Customer(CustID);
```

Tables in an Order



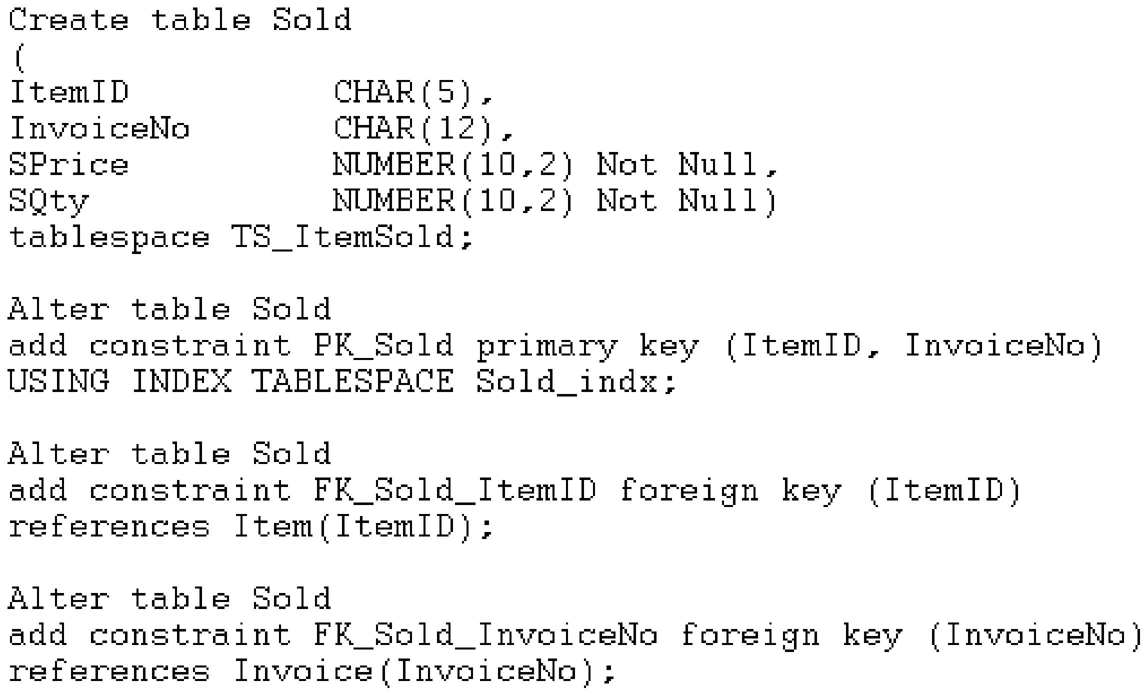
```
create table Employee
(EmpID           CHAR(3),
EmpName          VARCHAR2(25) NOT NULL,
Address          VARCHAR2(50),
City             VARCHAR2(50),
TelNo            VARCHAR2(15),
Email            VARCHAR2(50) UNIQUE)
tablespace TS_Sales;

Alter table Employee
add constraint PK_EmpID primary key (EmpID)
USING INDEX TABLESPACE Sale_indx;

Alter table Orders
add constraint FK_Emp_EmpID foreign key (SalesEmpID)
references Employee(EmpID)
USING INDEX TABLESPACE Sale_indx;
```

Tables in an Order



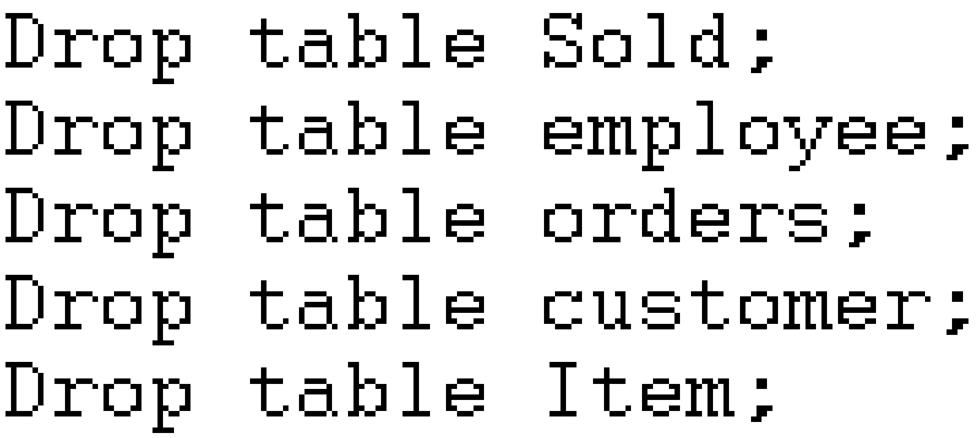
```
Create table Item
(ItemID         CHAR(5),
ItemName        VARCHAr2(50) NOT NULL,
PPrice          NUMBER(10,2),
TotQty          NUMBER(5),
Status          CHAR(1))
tablespace TS_Stock;

Alter table Item
add constraint PK_ItemID primary key (ItemID)
USING INDEX TABLESPACE Sale_indx;
```

Tables in an Order



```
Create table Sold
(
ItemID          CHAR(5),
InvoiceNo       CHAR(12),
SPrice          NUMBER(10,2) Not Null,
SQty            NUMBER(10,2) Not Null)
tablespace TS_ItemSold;

Alter table Sold
add constraint PK_Sold primary key (ItemID, InvoiceNo)
USING INDEX TABLESPACE Sold_indx;

Alter table Sold
add constraint FK_Sold_ItemID foreign key (ItemID)
references Item(ItemID);

Alter table Sold
add constraint FK_Sold_InvoiceNo foreign key (InvoiceNo)
references Invoice(InvoiceNo);
```

Drop Tables in an Order



```
Drop table Sold;
Drop table employee;
Drop table orders;
Drop table customer;
Drop table Item;
```

Other Schema Objects

Script may contains other objects such as

Views
Sequences
Synonym
PL/SQL Triggers & Code
DML for sample data

End

CHAPTER # 97

Drop Tables & Recycle Bin

Overview of Segments

- Drop Tables
- Drop Tables Precautions
- Drop Tables Commands
- What is the Recycle Bin?
- Object Naming in Recycle Bin
- Enable & Disable the Recycle Bin
- Purging is the Recycle Bin
- Data Dictionary Views for Recycle Bin

Drop Tables

- To drop a table that you no longer need, use the DROP TABLE statement. The table must be contained in your schema or you must have the DROP ANY TABLE system privilege.

Drop Tables Precautions

- Dropping a table removes the table definition from the data dictionary. All rows of the table are no longer accessible.
- All indexes and triggers associated with a table are dropped.
- All views and PL/SQL program units dependent on a dropped table remain, yet become invalid (not usable).
- All synonyms for a dropped table remain, but return an error when used.

Drop Tables Precautions

- All extents allocated for a table that is dropped are returned to the free space of the tablespace and can be used by any other object requiring new extents or new objects. All rows corresponding to a clustered table are deleted from the blocks of the cluster

Drop Tables Commands

DROP TABLE hr.int_admin_emp;

- If the table to be dropped contains any primary or unique keys referenced by foreign keys of other tables and you intend to drop the FOREIGN KEY constraints of the child tables, then include the CASCADE clause in the DROP TABLE statement

DROP TABLE hr.admin_emp CASCADE CONSTRAINTS;

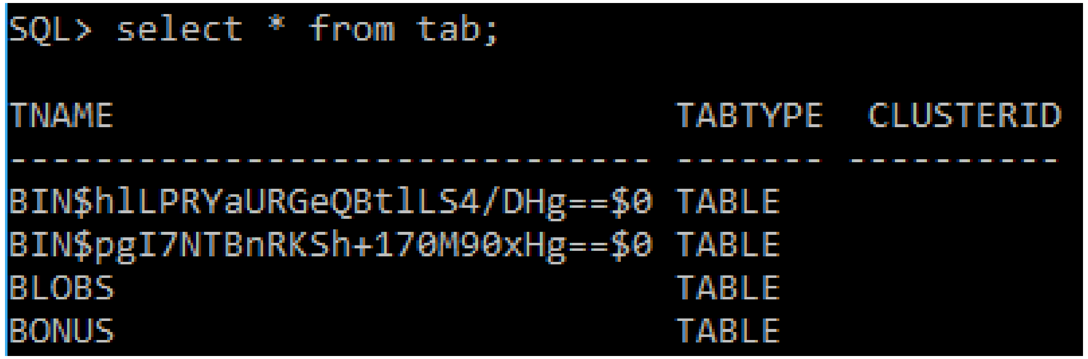
Drop Tables Commands

If you should want to immediately release the space associated with the table at the time you issue the DROP TABLE statement, include the PURGE clause as shown in the following statement:

```
DROP TABLE hr.admin_emp PURGE;
```

What is the Recycle Bin?

- The recycle bin is actually a data dictionary table containing information about dropped objects.
- Dropped tables and any associated objects such as indexes, constraints, nested tables, and the likes are not removed and still occupy space.
- They continue to count against user space quotas, until specifically purged from the recycle bin.
- SELECT * FROM RECYCLEBIN;



```
SQL> select * from tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
BIN$hlLPRYaURGeQBtlLS4/DHg==$0 TABLE
BIN$pgI7NTBnRKSh+170M90xHg==$0 TABLE
BLOBS                          TABLE
BONUS                          TABLE
```

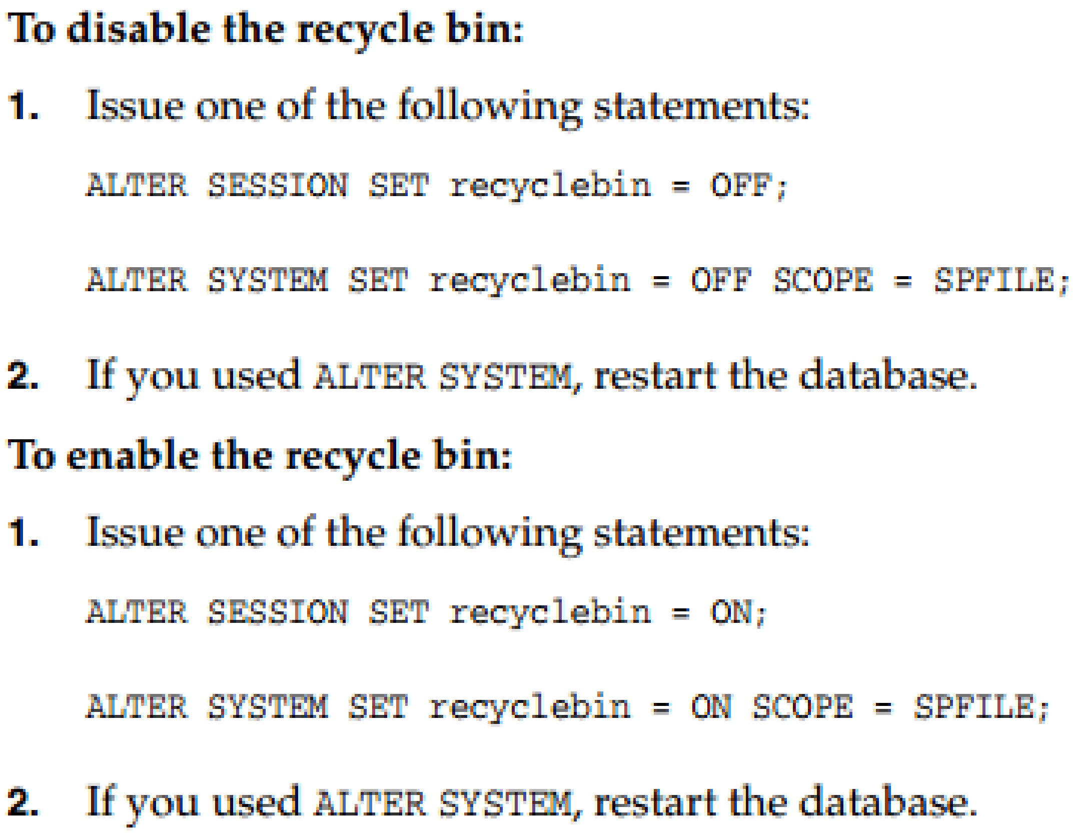
Object Naming in Recycle Bin

- The renaming convention is as follows:

  BIN$unique_id$version

- unique_id is a 26-character globally unique identifier for this object, which makes the recycle bin name unique across all databases
- Version is a version number assigned by the database

Enable & Disable the Recycle Bin



To disable the recycle bin:

1. Issue one of the following statements:

```
ALTER SESSION SET recyclebin = OFF;

ALTER SYSTEM SET recyclebin = OFF SCOPE = SPFILE;
```

2. If you used `ALTER SYSTEM`, restart the database.

To enable the recycle bin:

1. Issue one of the following statements:

```
ALTER SESSION SET recyclebin = ON;

ALTER SYSTEM SET recyclebin = ON SCOPE = SPFILE;
```

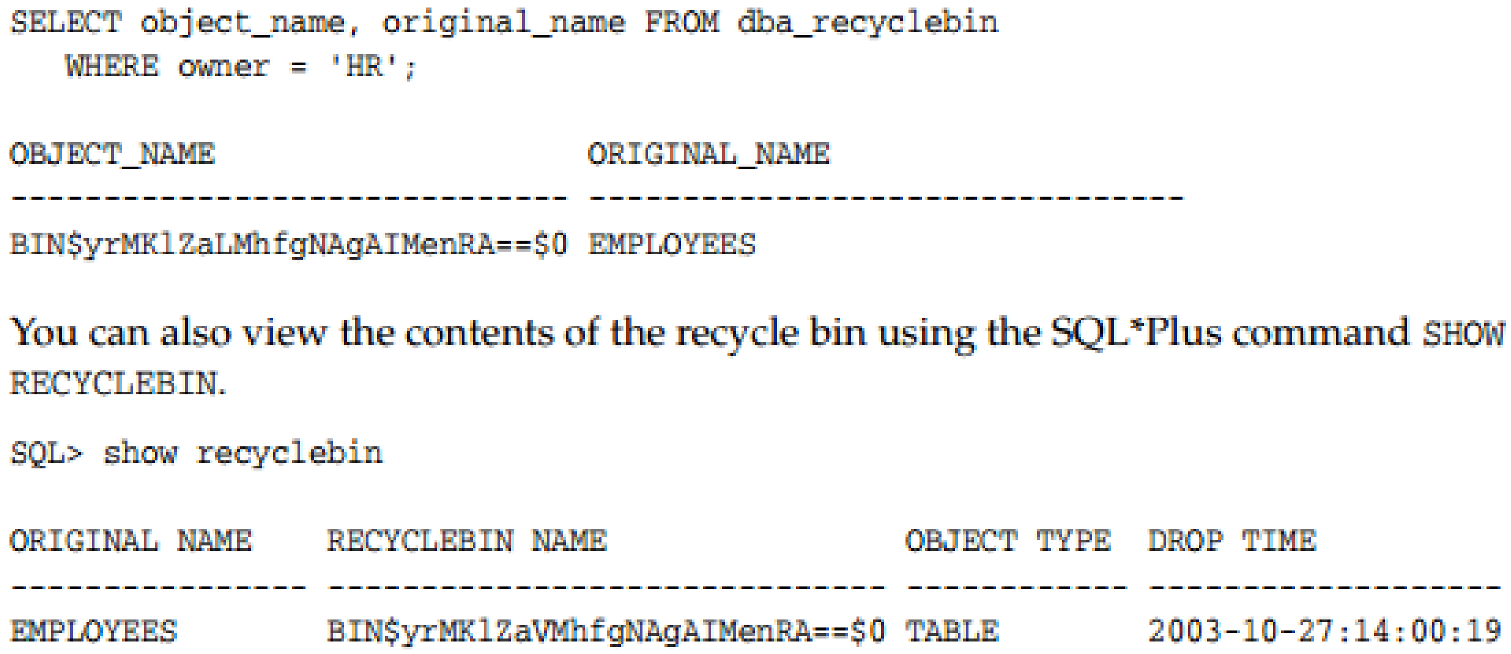
2. If you used `ALTER SYSTEM`, restart the database.

Purging is the Recycle Bin

- If you decide that you are never going to restore an item from the recycle bin, you can use the PURGE statement to remove the items and their associated objects from the recycle bin and release their storage space.

```
PURGE TABLE "BIN$jsleilx392mk2=293$0";
PURGE TABLESPACE example;
PURGE RECYCLEBIN;
```

Data Dictionary Views for Recycle Bin



```
SELECT object_name, original_name FROM dba_recyclebin
   WHERE owner = 'HR';

OBJECT_NAME                    ORIGINAL_NAME
------------------------------ --------------------------------
BIN$yrMKlZaLMhfgNAgAIMenRA==$0 EMPLOYEES
```

You can also view the contents of the recycle bin using the SQL*Plus command `SHOW RECYCLEBIN`.

```
SQL> show recyclebin

ORIGINAL NAME    RECYCLEBIN NAME                OBJECT TYPE  DROP TIME
---------------- ------------------------------ ------------ -------------------
EMPLOYEES        BIN$yrMKlZaVMhfgNAgAIMenRA==$0 TABLE        2003-10-27:14:00:19
```

Join VU Group: https://chat.whatsapp.com/K8O1b4yN8XiLTwEPUdXwVq

CHAPTER # 98

Indexed Organized Tables

Overview of Segments

- Drop Tables

Drop Tables

- To drop a table that you no longer need, use the DROP TABLE statement. The table must be contained in your schema or you must have the DROP ANY TABLE system privilege.

CHAPTER # 99

Data Dictionary Views for Tables

Integrity Constraints

- Three Types of Views
- Displaying Views for Tables
- Size of a Table
- Displaying Table’s Constraints
- Displaying Table’s Columns Details

Three Types of Views

- There are three types of Views to manage Tables related information:

  DBA_
  ALL_
  USER_

Displaying Views for Tables

- SQL statement to display Tables related Views:

```
Select Table_name
From Dict
Where Table_name like '%TAB%‘;
```

You can use following choices for more views:

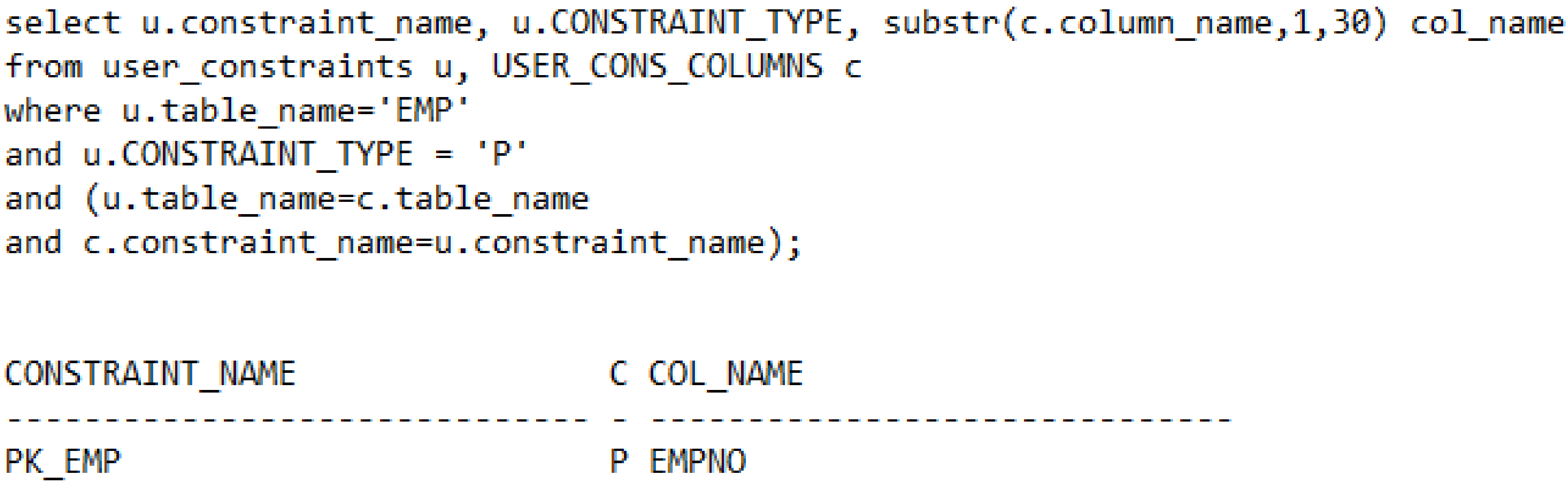
```
Table_name like '%COL%'
Table_name like '%EXTERNAL%‘
Table_name like '%CONSTRAINT%'
```

Size of a Table

- To find size of table in MB

```
select sum(bytes)/1024/1024 SizeMB
from dba_segments
where segment_name ='EMP';
```

Displaying Table’s Constraints

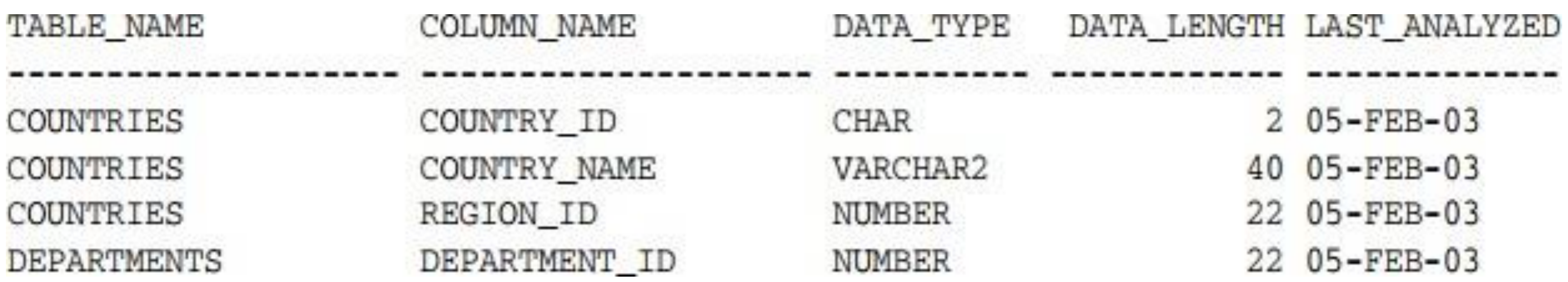


```
select u.constraint_name, u.CONSTRAINT_TYPE, substr(c.column_name,1,30) col_name
from user_constraints u, USER_CONS_COLUMNS c
where u.table_name='EMP'
and u.CONSTRAINT_TYPE = 'P'
and (u.table_name=c.table_name
and c.constraint_name=u.constraint_name);


CONSTRAINT_NAME                C COL_NAME
------------------------------ - ------------------------------
PK_EMP                         P EMPNO
```

Displaying Table’s Columns Details

Column information, such as name, data type, length, precision, scale, and default data values can be listed using one of the views ending with the _COLUMNS suffix. For example, the following query lists all of the default column values for the emp and dept tables:

```
SELECT TABLE_NAME, COLUMN_NAME, DATA_TYPE, DATA_LENGTH, LAST_ANALYZED
FROM DBA_TAB_COLUMNS
WHERE OWNER = 'HR'
ORDER BY TABLE_NAME;
```



```
TABLE_NAME           COLUMN_NAME          DATA_TYPE  DATA_LENGTH LAST_ANALYZED
-------------------- -------------------- ---------- ----------- -------------
COUNTRIES            COUNTRY_ID           CHAR                 2 05-FEB-03
COUNTRIES            COUNTRY_NAME         VARCHAR2            40 05-FEB-03
COUNTRIES            REGION_ID            NUMBER              22 05-FEB-03
DEPARTMENTS          DEPARTMENT_ID        NUMBER              22 05-FEB-03
```

CHAPTER # 100

Creating & Altering an Index

Creating & Altering Index

- Altering Indexes
- Creating Indexes
- Creating & Alter Indexes
- Rebuild Index
- Making Index Visible/ Invisible
- Making Index Useable

Altering Indexes

To alter an index, your schema must contain the index or you must have the ALTER ANY INDEX system privilege. With the ALTER INDEX statement, you can:

- Rebuild or coalesce an existing index
- Deallocate unused space or allocate a new extent Specify parallel execution (or not) and alter the degree of parallelism
- Alter storage parameters or physical attributes Specify LOGGING or NOLOGGING

Creating Indexes

- Enable or disable key compression
- Mark the index unusable
- Make the index invisible
- Rename the index
- Start or stop the monitoring of index usage

Creating Indexes

```
CREATE INDEX emp_ename ON
emp(ename)
TABLESPACE users
STORAGE (INITIAL 20K NEXT 20k);

CREATE UNIQUE INDEX
dept_unique_index ON dept (dname)
TABLESPACE indx;
```

Creating & Alter Indexes

```
CREATE TABLE emp (
empno NUMBER(5) PRIMARY KEY,
age       INTEGER)
ENABLE PRIMARY KEY
USING INDEX TABLESPACE Work_Indx;

ALTER INDEX emp_ename
STORAGE (NEXT 40);
```

Rebuild Index

You have the option of rebuilding the index online. Rebuilding online enables you to update base tables at the same time that you are rebuilding.
The following statement rebuilds the emp_name index online:

```
ALTER INDEX emp_name REBUILD ONLINE;
```

Making Index Visible/ Invisible

An invisible index is ignored by the optimizer unless you explicitly set the OPTIMIZER_USE_INVISIBLE_INDEXES initialization parameter to TRUE at the session or system level. Making an index invisible is an alternative to making it unusable or dropping it.

ALTER INDEX *index* INVISIBLE;

ALTER INDEX *index* VISIBLE;

Making Index Visible/ Invisible

An invisible index is ignored by the optimizer unless you explicitly set the OPTIMIZER_USE_INVISIBLE_INDEXES initialization parameter to TRUE at the session or system level. Making an index invisible is an alternative to making it unusable or dropping it.

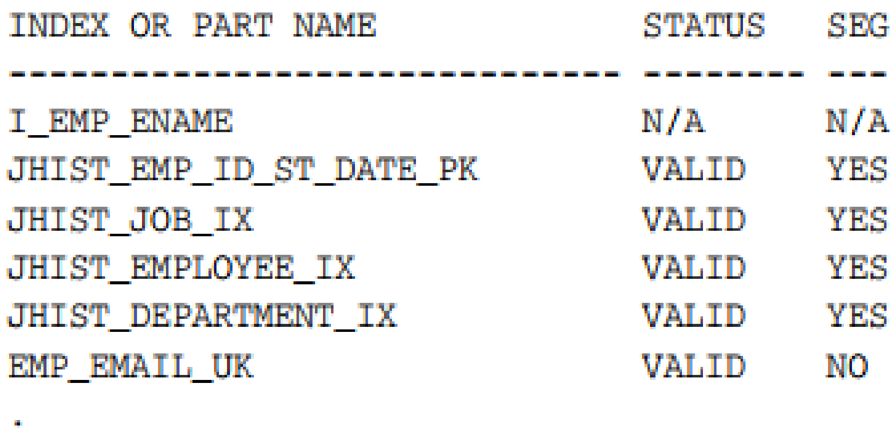
ALTER INDEX *index* INVISIBLE;

ALTER INDEX *index* VISIBLE;

Making Index Useable

Query the data dictionary to determine whether an existing index or index partition is usable or unusable.

```
SELECT INDEX_NAME AS "INDEX OR PART NAME",
STATUS, SEGMENT_CREATED
FROM USER_INDEXES
UNION ALL
SELECT PARTITION_NAME AS "INDEX OR PART NAME",
STATUS, SEGMENT_CREATED
FROM USER_IND_PARTITIONS;
```



```
INDEX OR PART NAME              STATUS   SEG
------------------------------ -------- ---
I_EMP_ENAME                     N/A      N/A
JHIST_EMP_ID_ST_DATE_PK         VALID    YES
JHIST_JOB_IX                    VALID    YES
JHIST_EMPLOYEE_IX               VALID    YES
JHIST_DEPARTMENT_IX             VALID    YES
EMP_EMAIL_UK                    VALID    NO
.
```