Access Structures Laboratory Material

Summary of contents

- Steps to follow in exercises
- Access Structure Sentences
  - Table without clustered index, nor clustered structure, nor hash
  - Table with clustered index
  - B+ tree
  - Table with hash
  - Clustered structure
- Appendix
  - Catalog tables and attributes relevant for the exercises
  - Relevant physical operations in execution plans

Steps to follow in exercises (1/2)

1. Delete all the objects in your database
   - Use the ***Script to delete objects*** given in the course laboratory information
2. Clean the recyclebin (necessary if it is not done in deleting objects).
   - Execute "`PURGE RECYCLEBIN`" from DBeaver
3. Create tables and access structures that you consider as a possible solution. Also fill the tables.
4. Update the statistics of all the objects in your database
   - Use the ***Script to update statistics*** given in the course laboratory information

Steps to follow in exercises (2/2)

5. Check that the access plan of the SELECT to be optimized is the one that you expectes Oracle would use.
   - There is an execution option in DBeaver that allow to do this
     - You can also use EXPLAIN PLAN to get it
   - Look at the cost of using the defined access structures
6. Check the database catalog to see the space and cost of using the access structures
   - Select the ***relevant atributes and tables*** (see appendix of these slides)
   - Look at the space the defined structures use
7. Write down the `username` and `password` of your database in the questionnaire exercise form, and `submit` to check your proposed structures

Access Structure Sentences: Table without access structure

```
CREATE TABLE tableName (attributes)
PCTFREE 0 ENABLE ROW MOVEMENT;
```

PCTFREE 0 causes that there is no free space in the data blocs

After inserting rows in the table we have to execute the following sentence to optimize the distribution of rows in table blocs.

```
ALTER TABLE tableName SHRINK SPACE;
// Compress the table
```

Access Structure Sentences: Table with clustered index

```
CREATE TABLE name (attributes, PRIMARY KEY(...))
ORGANIZATION INDEX PCTFREE 33;
```

After inserting rows in the table we have to execute the following sentence that reconstructs the index, guaranteeing that the index holds the desired load factor (2/3)

```
ALTER TABLE name MOVE;
```

In Oracle, we can only define a clustered index over the primary key of the table.

Access Structure Stentences: B+ Tree

```
CREATE TABLE tableName (attributes)
PCTFREE 0 ENABLE ROW MOVEMENT;
```

After inserting rows in the table the index is created to assure that it holds the desired load factor (2/3) for the index:

```
ALTER TABLE tableName SHRINK SPACE;
// Compress the table

CREATE [UNIQUE] INDEX indexName ON tableName(attributes)
PCTFREE 33;
```

Access Structure Sentences: Table with hash

```
CREATE CLUSTER name (attributes type) SINGLE TABLE
HASHKEYS L PCTFREE 0;

CREATE TABLE name (attributes)
CLUSTER name(attributes);
```

Where L specifies and limits the number of unique hash values that the hash function can generate and has to be the result of the following multiplication: $\lceil 1.25*B \rceil$ (being B the number of blocks that the table occupies without index).

Access Structure Sentences: Clustered structure

```
CREATE CLUSTER cluster_name (a1 type) PCTFREE 33;
CREATE INDEX index_name ON
CLUSTER cluster_name PCTFREE 33;
CREATE TABLE table_name1 (...,a2 type,...)
CLUSTER cluster_name(a2);
CREATE TABLE table_name2 (...,a3 type,...)
CLUSTER cluster_name(a3);
```

After inserting rows in the table we have to execute the following sentence that reconstructs the index, guaranteeing that the index holds the desired load factor (2/3)

```
ALTER INDEX index_name REBUILD;
```

Appendix

- Relevant attributes and tables in the catalog
  - USER_TS_QUOTAS (1/6)
  - USER_TABLES (2/6)
  - USER_TAB_COLS (3/6)
  - USER_INDEXES (4/6)
  - USER_CLUSTERS (5/6)
  - USER_SEGMENTS (6/6)
- Relevant physical operations in execution plans

Relevant attributes and tables in the catalog (1/6)

- The used space value that is taken into account if there is a space constraint in the DBD exercises is the one in the USER_TS_QUOTAS

- Table: USER_TS_QUOTAS
- Relevant attributes
  - TABLESPACE_NAME
  - BYTES
  - BLOCKS

```
-- Select blocks in the users_ts_quotas
SELECT TABLESPACE_NAME, BYTES, BLOCKS
FROM USER_TS_QUOTAS;
```

Relevant attributes and tables in the catalog (2/6)

- Table: USER_TABLES
- Relevant attributes
  - TABLE_NAME
  - CLUSTER_NAME
  - IOT_TYPE
  - IOT_NAME
  - PCT_FREE
  - BLOCKS
  - NUM_ROWS
  - LAST_ANALYZED

```
-- Select attributes
SELECT TABLE_NAME, BLOCKS, NUM_ROWS, PCT_FREE,
       CLUSTER_NAME, IOT_TYPE, IOT_NAME, LAST_ANALYZED
FROM USER_TABLES;
```

Relevant attributes and tables in the catalog (3/6)

- Table: USER_TAB_COLS
- Relevant attributes
  - TABLE_NAME
  - COLUMN_NAME
  - DATA_TYPE
  - DATA_LENGTH
  - AVG_COL_LEN
  - NULLABLE
  - LAST_ANALYZED

```
-- Select attributes
SELECT TABLE_NAME, COLUMN_NAME, DATA_TYPE,
       DATA_LENGTH, AVG_COL_LEN, NULLABLE, LAST_ANALYZED
FROM USER_TAB_COLS;
```

Relevant attributes and tables in the catalog (4/6)

- Table: USER_INDEXES
- Relevant attributes
  - INDEX_NAME
  - TABLE_NAME
  - INDEX_TYPE,
  - UNIQUENESS
  - PCT_FREE
  - BLEVEL
  - LEAF_BLOCKS,
  - DISTINCT_KEYS
  - LAST_ANALYZED
  - JOIN_INDEX

```
-- Select attributes
SELECT INDEX_NAME, TABLE_NAME, INDEX_TYPE, UNIQUENESS,
       PCT_FREE, BLEVEL, LEAF_BLOCKS,
       DISTINCT_KEYS, LAST_ANALYZED, JOIN_INDEX
FROM USER_INDEXES;
```

Relevant attributes and tables in the catalog (5/6)

- Table: USER_CLUSTERS
- Relevant attributes
  - CLUSTER_NAME
  - PCT_FREE
  - CLUSTER_TYPE
  - HASHKEYS
  - SINGLE_TABLE

```
-- Select attributes
SELECT CLUSTER_NAME, PCT_FREE,
       CLUSTER_TYPE, HASHKEYS, SINGLE_TABLE
FROM USER_CLUSTERS;
```

Relevant attributes and tables in the catalog (6/6)

- Table: USER_SEGMENTS
- Relevant attributes
  - SEGMENT_NAME
  - SEGMENT_TYPE
  - BYTES
  - BLOCKS

```
-- Select attributes
SELECT SEGMENT_NAME, SEGMENT_TYPE, BYTES, BLOCKS
FROM USER_SEGMENTS;
```

Relevant physical operations in execution plans (*)

Operation (option)	Description
Table access (full)	Retrieval of all rows from a table.
Table access (cluster)	Retrieval of rows from a table based on a value of an indexed cluster key
Table access (hash)	Retrieval of rows from table based on hash cluster key value
Table access (by rowid range)	Retrieval of rows from a table based on a rowid range
Table access (by index rowid)	If rows are located using index(es).
Index unique scan	Retrieval of a single rowid from an index
Index range scan	Retrieval of one or more rowids from an index. Indexed values are scanned in ascending order.
Index range scan descending	Retrieval of one or more rowids from an index. Indexed values are scanned in descending order
Index full scan	Retrieval of all rowids from an index when there is no start or stop key. Indexed values are scanned in ascending order.
Index full scan descending	Retrieval of all rowids from an index when there is no start or stop key. Indexed values are scanned in descending order

(*) the complete list of operations can be found in the session materials