# Project on Database Optimization

The database project is focused on database optimization, and more specifically, on physical optimization. The objective of the project is to learn how to *fine-tune* a real database.

For this project, we will use Oracle 11g. Oracle is one of the most powerful database management systems and we chose it due to its wide range of algorithm implementation and data structures available.

*Fine-tuning* or *tuning-up* a database is the set of actions, typically conducted by the database administrator, to improve the performance of the database. At high level, it typically consists of two main kind of actions:

- Tuning the system parameters (e.g., the size of a block in disk, the size of memory pages, the degree of the trees, etc.). Even whether from the cost models we discussed during the lectures you can infer the impact of a larger or smaller block / page size or a higher / lower degree of a tree, we will consider the default values for the system parameters during the project.
- Creating additional data structures (e.g., B+, hash, index cluster, bitmaps, multi-attribute indexes) in order to improve the current *workload* (i.e., the current known set of queries the database is answering). We will focus on this aspect for the project.

Careful!!

The project has two parts, (i) the optimization of a database given a single query and (ii) the optimization of a database given a workload.

(i) Optimitzation of the database given a single query

In this part, you will solve different problems, each of them consisting of a query and a current database. The objective of this part is to build additional data structures so that you maximize the database performance.

You can look at the problem as an optimization problem: the search space would be all the combinations of potential data structures with a certain system parametrization (we call each of these combinations a potential **configuration**). For example, one configuration of the search space could be a B+ on a certain attribute A of a table T, and given a block and a page memory size and a certain tree order). As we mentioned before, in these exercises we will assume the pre-defined values for the system parameters so **you do not have to play with them**. Thus, these exercises boil down to choose the best data structures.

(ii) Optimization of a workload

Essentially, this problem is a generalization of the previous one. Now, instead of a single query we may have several of them. Thus, a configuration might be beneficial for a query and counterproductive for another one. Accordingly, the way to decide what is the best option when this happens is by considering the frequency or relevance of each query. In the end, we want to optimize the overall database performance (i.e., knowing the frequency of each query, the solution that minimizes the overall execution time of all the queries for a certain period). Nevertheless, from your point of view, this exercise requires creating the data structures improving the database performance the most, for a given workload and a period of time (note that the queries frequency embed, implicitly, a period of time).

For the workload optimization exercises, you need to consider all the structures seen in the lectures. Additionally, **note that bitmaps and multi-attributes are also candidates for these exercises**.

## Evaluation

The project consists of different exercises: some of type (i) and some of type (ii). You can solve them all, however, out of the exercises of kind (i), the three with the higher mark will prevail. The top-3 exercises will account, proportionally, for 2p **each** (if x, ranging [0,10], is the mark given by Learn-SQL for a top-3 exercise, the mark obtained is 0,2*x). The other exercises will not be considered for the mark).

The other 4p will come from the exercises of kind (ii). The same rationale apply: the exercise with the top mark will be considered as, at most, 4p.

*Finally, for your top-3 (i) exercises and your top (ii) exercise upload to the Explanation Doc event a document explaining, for the current database configuration you submitted, the access plan the database would execute for each query.*

The explanation document weights (by means of a geometric mean) the marks obtained for (i) and (ii). Thus, please, be sure you understand the plan access the database is choosing given the data structures you created.

## Connecting to Oracle

Use DBeaver to connect to Oracle. Click on “add a connection” and choose Oracle. Add then the following information:

*Host: oraclefib.fib.upc.edu*

*SID: ORABD*

*Username: same as for PostgreSQL (i.e., the one for the LCFIB system)*

*Password: same as for PostgreSQL (DBddmmaa; where ddmmaa is your birth date)*

When connecting for the first time, DBeaver will ask you for the Oracle driver. Find it in the LearnSQL (*ojdbc6.jar*).

## Objectives

Thus, the objectives of the project summarise as:

Given a simple situation (one table and one query, or two tables and a join) or a workload (i.e., several queries) optionally constrained with respect to space:

- Propose suitable configurations (structure type and involved attributes),
- Estimate space and cost corresponding to several configurations, and
- Compare estimations and real values taken from Oracle and explain observed differences

**Constraints**

For each exercise, start experimenting in your Oracle account. Each exercise has an attachment, which contains:

- A set of CREATE tables (i.e., the database schema),
- An script to insert data into the given tables (in the form of PL/SQL),
- A script to update the database statistics

**When creating your data structures, follow these instructions:**

- Table without clustered index, nor clustered structure, nor hash:

  CREATE TABLE name (attributes) PCTFREE 0 ENABLE ROW MOVEMENT;

  Insertions

  ALTER TABLE name SHRINK SPACE; // Compress the table

- Table with clustered index:

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

  Insertions

  ALTER TABLE name MOVE; // Compress the index cluster

  - This way, we reconstruct the index and we assure that it holds the desired load factor (2/3).
  - **We can only define a clustered index over the primary key of the table.**

- B+ tree (after the insertions):

  CREATE TABLE name (attributes) PCTFREE 0 ENABLE ROW MOVEMENT;

  Insertions

  ALTER TABLE name SHRINK SPACE; // Compress the table
  CREATE {UNIQUE} INDEX name ON table (attributes) PCTFREE 33;

  - We create the index after the insertions to assure that it holds the desired load factor (2/3).

- Bitmap[1]:

  CREATE TABLE name (attributes) PCTFREE 0 ENABLE ROW MOVEMENT;

  Insertions

  ALTER TABLE name SHRINK SPACE; // Compress the table

  ALTER TABLE table MINIMIZE RECORDS_PER_BLOCK; // Compress the bitmap

  CREATE BITMAP INDEX name ON table(attributes) PCTFREE 0;

  - **Bitmaps should only be considered for attributes with values repeated – i.e., the *k*- over 100 times**

- Table with hash:

  CREATE CLUSTER name (attributes type) SINGLE TABLE HASHKEYS 1.25*B PCTFREE 0;
  CREATE TABLE name (attributes) CLUSTER name(attributes);

In order to facilitate deleting and creating different configurations, find in Learn-SQL a script (*DropPurge.sql*) that deletes all the structures from your database.

Remember than when submitting your answer through Learn-SQL you must guarantee:

- Your Oracle database only contains those objects corresponding to your answer.
- Purge the recycling bin ("PURGE RECYCLEBIN").
- Update the statistics of all the objects (use the corresponding code in the attachment).

In addition, to guarantee the automatic correction done by Learn-SQL is sound, follow these general rules:

- General rules: ????
  - "ROW MOVEMENT" must be enabled for any table not in an index cluster. It will allow you to "SHRINK SPACE" just after the insertions.

[1] Due their compression techniques, bitmaps generate little unpredictable variations in query costs (usually one or two blocks). This potential deviation will never result in more than one point in the exercise, and will be manually (off-line) corrected by the lecturer.

- "MINIMIZE RECORDS_PER_BLOCK" must be executed after the insertions of the table and before the creation of any bitmap index on it (you only need to execute it once, even if you create many bitmap indexes over the same table).

Trigger: Cause a device to function  Prune = reduce  Empirically = in a manner based on observation or experience

**Hints**

Trying all potential configurations, even for these toy exercises, is simply not feasible. So that, a database administrator needs to prune the search space and try empirically those configurations of potential interest. Follow these steps to do so:

- If you are not taking Data Warehousing, go through the advanced indexing techniques slides to get familiar with bitmaps, multi-attributes and index-only query answering. These structures are allowed in the optimization of a workload (type ii) exercises.
- Check the slides "*differences between the theory and Oracle*". There, you will find the small implementation changes Oracles does to the most common data structures. Accordingly, you need to adapt the formulas to these changes.
- Use the formulas to select the best candidate data structures.
- Go to Oracle, delete all the objects (use the script we provide) and create each potential configuration of interest one by one. For each configuration, trigger the involved queries and use the "*explain access plan*" button of DBeaver.
    - Understand the access plan and thus, if Oracle is using the structures as you expected.
    - Choose the configuration with the smallest cost according to Oracle.
- Once you are satisfied with your option, prepare your database for the submission. Delete everything and implement from scratch your configuration following the instructions above. Then, go to Learn-SQL and submit your proposal by introducing your login and password there.
    - Learn-SQL will tell you if your approach is the best or still there could be other options by using the data structures we saw during the lectures.

When needing to check how much space a configuration does occupy (remember some exercises fix a maximum amount of disk you may use) use Oracle's catalog. The catalog is where Oracle stores all metadata extracted from the database. Specifically, there is a table called USER_TS_QUOTAS (which you can query with SQL) that show the quota (i.e., disk space) used by your database. The same information but more detailed can be found in USER_SEGMENTS. Other relevant catalog tables to check for relevant statistics (such as the number of blocks used by a table, etc.) are the following ones:

-- Info about all the tables you have created

USER_TABLES (TABLE_NAME, CLUSTER_NAME, IOT_TYPE, IOT_NAME, PCT_FREE, BLOCKS, NUM_ROWS, AVG_ROW_LEN, LAST_ANALYZED)

-- Info about all the columns of all the tables you have created

USER_TAB_COLS (TABLE_NAME, COLUMN_NAME, DATA_TYPE, DATA_LENGTH, AVG_COL_LEN, NULLABLE, LAST_ANALYZED)

-- Info about all the indexes you have created

USER_INDEXES (INDEX_NAME, TABLE_NAME, INDEX_TYPE, UNIQUENESS, PCT_FREE, BLEVEL, LEAF_BLOCKS, DISTINCT_KEYS, LAST_ANALYZED, JOIN_INDEX)

-- Info about all the clusters created

USER_CLUSTERS (CLUSTER_NAME, PCT_FREE, CLUSTER_TYPE, HASHKEYS, SINGLE_TABLE)

-- Info about the space (disk) used in your current configuration

USER_SEGMENTS (SEGMENT_NAME, SEGMENT_TYPE, BYTES, BLOCKS)

USER_TS_QUOTAS (TABLESPACE_NAME, BYTES, BLOCKS)

RECYCLEBIN (ORIGINAL_NAME)

Finally, some information about the default system parameters:

- Blocks are 8 Kbytes (either table blocks and B+ nodes)
- Addresses use 32 bits
- Variable *u* is the number of entries in a B+ leaf
- Variable *d* is half the number of entries that fit in a B+ leaf with a load factor of 1
- Variable *h* is the B+ height after insertions (BLEVEL attribute in the catalog)
- Variable B is the number of blocks of a non-indexed table. In these exercises, compute it as (NUM_ROWS * AVG_ROW_LEN) / (8 * 1024).