SESSION ON PHYSICAL DW DESIGN

OBJECTIVES

Tune a provided physical DW schema for a given workload (set of KPIs and given data load). Specifically:

- Decide the database configuration (i.e., B+, Clustered index, and Bitmaps) that optimizes the execution of a given workload (composed of a subset of 4 queries answering required KPIs and the frequency of their execution).

ACTIVITIES

Given a DW logical schema for the ACME use case and a workload (set of 4 KPIs), optionally constrained with respect to space:

- Translate given KPIs into the SQL queries over the DW schema.
- Propose suitable configurations (structure type and involved attributes) to optimize DW design for the given workload – i.e., SQL queries,
- Estimate space and cost corresponding to these configurations,
- Create structures in Oracle, use the catalog and see the execution plan to empirically contrast the space and cost needed in Oracle for such configurations.

CONSTRAINTS

- <u>The query cost taken into account is that provided by Oracle</u> in the execution (measured in terms of gets from the cache) for each query (weighted by the query frequency).
  - Thus, only use the theoretical formulas as a guide.
  - Note that applying a greedy algorithm <u>might</u> lead to a suboptimum solution, which, in turn, would impact on your mark.
- <u>The used space value that is taken into account, if there is a space constraint, is the one corresponding to USER_TS_QUOTAS</u>. More details can be found in USER_SEGMENTS (see the additional information section).
- Materialized views and partitions should **not** be considered.
- You can enable (or not) Oracle's **star-join transformation** (default is disabled) as explained in the lectures. However, for this to be taken into account during the assessment, besides setting the corresponding session attribute, you must also add the following comment to any of the tables in the schema (e.g., the fact table):

```
COMMENT ON TABLE name IS 'star_transformation_enabled';
```

- Data structures can only be created in the following way:
  - 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 IOT

  - This way, we reconstruct the index and we assure that it holds the desired load factor (2/3).
  - As it can be deduced, **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 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

  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 in the average**.

[1] Due their compression techniques, bitmaps generate little unforeseen 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 proctor.

    - Primary keys (after the insertions):

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

      Insertions

      ALTER TABLE name SHRINK SPACE; // Compress the table

      ALTER TABLE name ADD PRIMARY KEY (attributes) USING INDEX PCTFREE 33;

        - We create the primary key after the insertions to assure that the associated B+ tree holds the desired load factor (2/3).
    - Consider as available join algorithms any of the algorithms considered by Oracle.
- General rules:
    - “ROW MOVEMENT” must be enabled for any table not in a cluster or IOT. It will allow you to “SHRINK SPACE” just after the insertions.

REQUIRED KNOWLEDGE

- Taught material: physical design (see "Physical optimization: Basics", "Physical optimization: Sorting and join algorithms", "Physical optimization: Selectivity factor and star-join", "Physical optimization: Bitmap and join index" in the slides section in Moodle).

ADDITIONAL INFORMATION

- The following catalog tables let you know what is happening and give the actual values for space and cost (inside parenthesis you have some attributes that may be interesting):
    - USER_TABLES (TABLE_NAME, CLUSTER_NAME, IOT_TYPE, IOT_NAME, PCT_FREE, BLOCKS, NUM_ROWS, AVG_ROW_LEN, LAST_ANALYZED)
    - USER_TAB_COLS (TABLE_NAME, COLUMN_NAME, DATA_TYPE, DATA_LENGTH, AVG_COL_LEN, NULLABLE, LAST_ANALYZED)
    - USER_INDEXES (INDEX_NAME, TABLE_NAME, INDEX_TYPE, UNIQUENESS, PCT_FREE, BLEVEL, LEAF_BLOCKS, DISTINCT_KEYS, LAST_ANALYZED, JOIN_INDEX)
    - USER_SEGMENTS (SEGMENT_NAME, SEGMENT_TYPE, BYTES, BLOCKS)
    - USER_TS_QUOTAS (TABLESPACE_NAME, BYTES, BLOCKS)
    - RECYCLEBIN (ORIGINAL_NAME)

- Some interesting hints to know or remember:
    - 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)