DMV - DW

26 January 2021 11:48

Conceptual & Logical Design

In the "Top-Down" design approach, a data warehouse is described as a subject-oriented, time-variant, non-volatile and integrated data repository for the entire enterprise data from different sources are validated, reformatted and saved in a normalized (up to 3NF) database as the data warehouse. The data warehouse stores "atomic" information.

Advantages of top-down design

Data Marts are loaded from the data warehouses.

Developing new data mart from the data warehouse is very easy.

Disadvantages of top-down design

This technique is inflexible to changing departmental needs.

The cost of implementing the project is high.

In the "Bottom-Up" approach, a data warehouse is described as "a copy of transaction data specifical architecture for query and analysis," term the star schema. In this approach, a data mart is created first to necessary reporting and analytical capabilities for particular business processes (or subjects). Thus it is needed to be a business-driven approach in contrast to Inmon's data-driven approach.

Advantages of bottom-up design

Documents can be generated quickly.

The data warehouse can be extended to accommodate new business units.

It is just developing new data marts and then integrating with other data marts.

Disadvantages of bottom-up design

the locations of the data warehouse and the data marts are reversed in the bottom-up approach design.

- Facts: Description of relevant events for the company
- Workload: queries expressed in natural language
- Structural requirements: aviable space, system architecture, deployment planning
- Conceptual design: dimensional fact model (facts, dimensions, measures)
    - Measures aare stream (aggregatable), level (not additive over time), unit(not additive)
- Aggregate operatiors: distributive (sum, min, max), non distributive (algebraic (avg), olistic (median))
- Factless fact schema: events not carachterized by measures, used to count occurred events
- Time can be either a snapshot of the current value, a direct relationship with a time dimension, mapped to a time dimension all at the same time
- data volume needs to be estimated dpending on: cardinality and sparsity
- Star vs Snowflake: Start requires more memory but has easier joins, snowflake is good for materialized views
- Multiple edges can be solved with: bridge table (2 id one corresponding to another in a table) or push down (boolean list in the fact)
    - Degenerate dimensions are dimensions with a single attribute, easily implementable in the fact table
    - Junk dimensions can contain several degenerate dimensions (feasible for small cardinality)

Online analytical processing, or **OLAP** is an approach to answer multi-dimensional analytical (MDA) queries swiftly in computing.

Data Warehouse: Data Analysis

- Controlled Query Environment: it encompasses complex queries and ad hoc analysis procedures for predefined reports. Requires ad hoc code developement
- Ad Hoc Environment: arbitrary OLAP queries may be defined, designed on demand by users.AN olap session allows successive refinements of the same query
    - used when predefined queries are not enough
- OLAP analysis:

    - aviable operators: roll up (decrease detail in a dimension by jumping hierarchy such as GROUP BY), drill down (increase the detail by adding more dimensions to group on), slice and dice (select with specific conditions and logical operators), pivot (reorganize the structure without changing the detail level), sort (order by)
- the extension of SQL allows for new aggregate functions based on a computation window and to calculate the rank in a given order
    - window clause: partitioning, row ordering in each partition and aggregate window
    - Sort order is required, an incomplete window will take place on the aviable rows, several different computation windows may be specified
    - the aggregation window may be defined at a logical and physical level(from row to row)
        - Logical intervals are **appropriate for sparse data**

Mateiralized Views

- precompiled summaries for the fact table explicitly stored in the data warehouse, provide an increased performance for aggregate queries

ETL process

- Extraction, Transformation and loading AKA data preprocessing before being loaded into the data WH
- depends on how operational data is collected: historical (all mod stored for a given time), partly historical (only a limited number of states is stored) or transient (only current state)
    - Incremental extraction consists in capturing data modifications by an ad hoc application; rquires specific api and realizes in an increased application load
        - can be trigger based or timestamp based
- data cleaning is performed to solve issues of duplicate, missing, unexpected ec
    - data dictionary, approximate fusion and **prevention**
- transformation goal is to convert operational data into warehouse format (formatting)

Physical Design

- The PD depends on the workload i'm designing the warehouse for
- Structures such as
    - bitmap indexes, join indexes, bitmapped join indexes as well as b+ tree
    - materialized views which require an optimizer to determine the data access strategy
- procedure: select a suited data structure to support the most frequent queries, choose structures that contribute to dealing with more queries at the same time
    - tuning: a posteriori variation of support on physical structures, often required for olap app
    - parallelism: data/query fragmentation for parallel execution
- physical access structures describe how data is stored on the disk for efficient query execution
    - sequential structures, hash structures + indexing to increase efficiency
- Heap file ---> tuples are sequenced in insertion order, all the space in a block is used before starting a new block, but delete/update may cause wasted space
    - a sort key is used to determine the order in which tuples are written
    - typically used with b+tree clustered indices
        - b+tree: provides direct access to data based on the value of a key field, leaves are all at the same distance from the root so that access time is constant regardless of the searched value. Can be either clustered (tuple in leave) or unclustered
            - very efficient for range queries but susceptible to updates and delete
- bitmap index: matrix in which the relationship between a value and its field is stored as a boolean variable
    - very efficient for boolean predicaments but inappropriate for continuous attributes
- join index: precomputes the join between two tables is precomputed in a table
- star index: precomputes the join between two or more tables
    - stores the rid n-uples of the tuples that satisfy the join predicament
        - efficient computation of column based joins but useful only for specific joins, the storage space might become big

- bitmapped join index: a column for each dimension and a row for each fact to precompile the join between dimension and the fact table
    - a boolean value describes wheter the join is there or not
- hash structure: guarantees direct and efficient access to data based on the value of a key field
    - the hash function is applied to the key field to point to the position in memory
    - blocks should never be filled to allow insertion
    - can be either clustered or unclustered (blocks contain the pointer to the data instead of the data itself)
    - very efficient for queries with equality predicates with no disk sorting required, but inefficient for range queries