Data Warehouse
Logical Design

Munawar, Ir. MMSI. M.Com, PhD

Intro to Logical Design

- Logical design is the most attractive step given that it presents tremendous benefits to system performance.
- It is intended to obtain conceptual schemata based on the data structure that will be applied by a DM or DW, with consideration for a number of constraints, particularly those related to disk space or query retrieval (Trujilo et al, 2000).
- Logical design is relevant in a relational OLAP (ROLAP) environment, and a logical model can be easily derived in a multidimensional OLAP (MOLAP) environment.

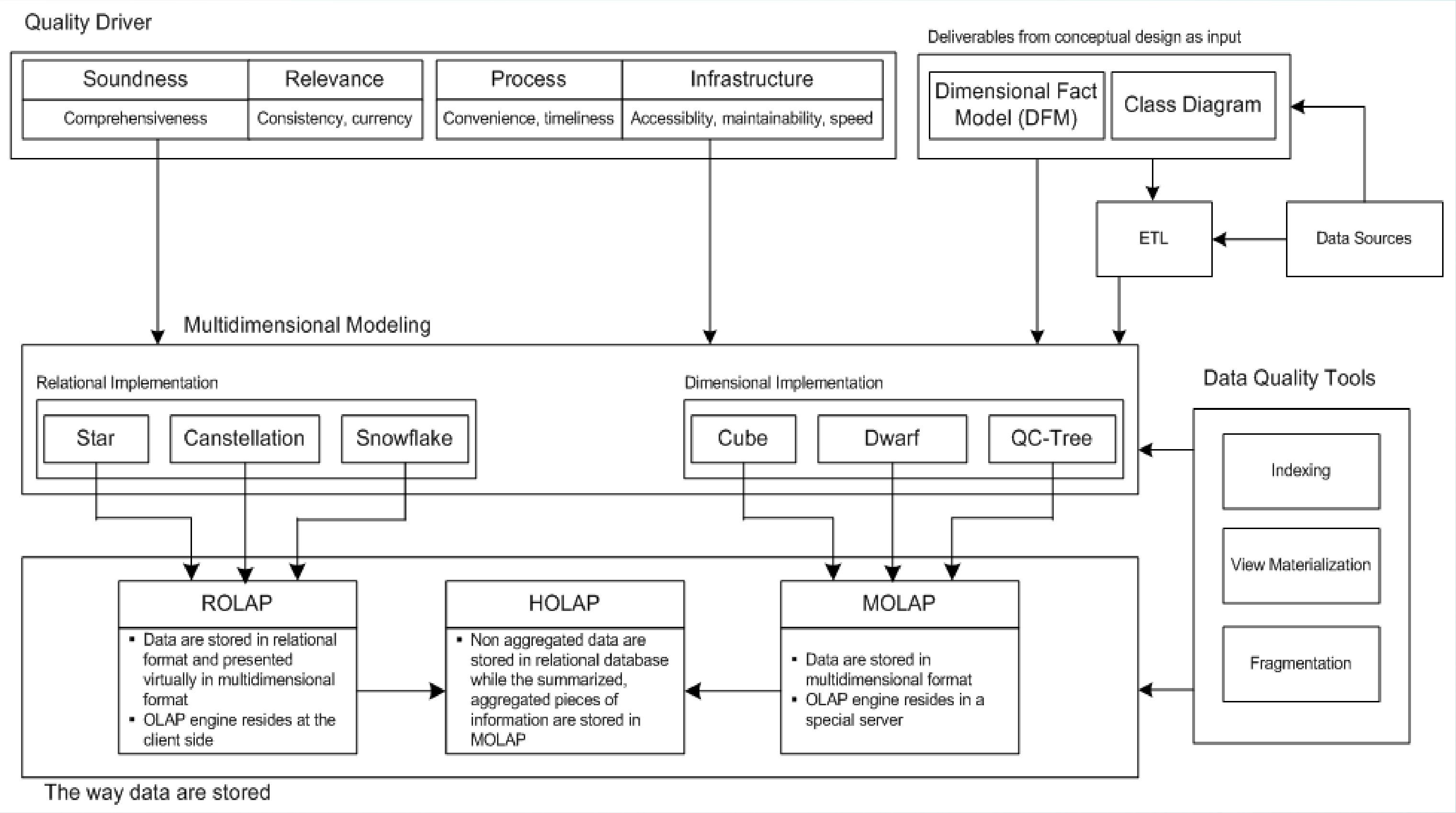
OLAP

- OLAP data are stored in ROLAP, MOLAP and hybrid OLAP (HOLAP) formats (Arun Sen and Sinha, 2007).
- In ROLAP, data in a DW are stored in a relational format and displayed virtually to users in multidimensional form whilst the OLAP engine remains in a client site.
- In MOLAP, both data and presentation are arranged in a multidimensional format through the use of multidimensional databases whilst the OLAP engine remains in a special server.

OLAP Consideration

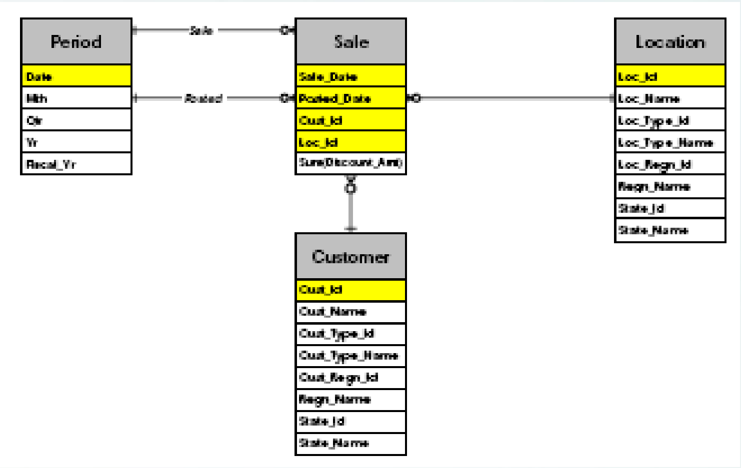
- The ROLAP or MOLAP option depends on the complexity of queries and performance.
- MOLAP should be used in multidimensional databases with extensive OLAP capabilities to carry out complex queries and achieve excellent response times.
- ROLAP can be used in relational tables to perform operations when query complexity is low and response time requests are minimal because of multidimensional views that are generated on the fly.

Logical Design for DW Development

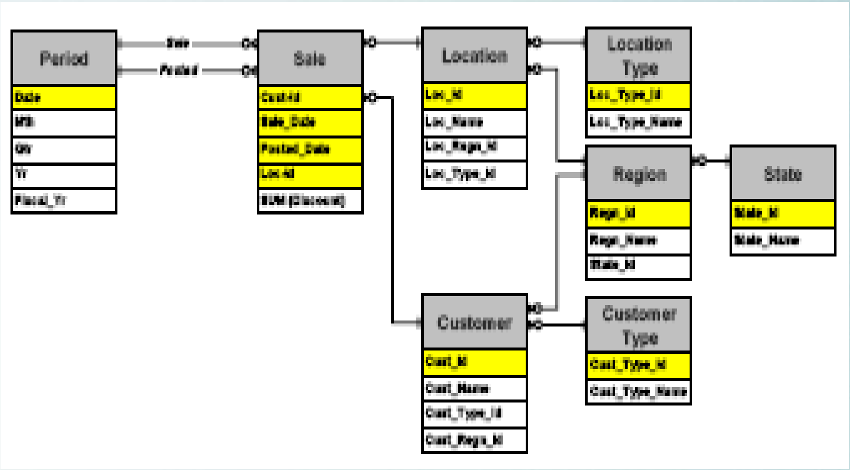


Source : Munawar, 2016

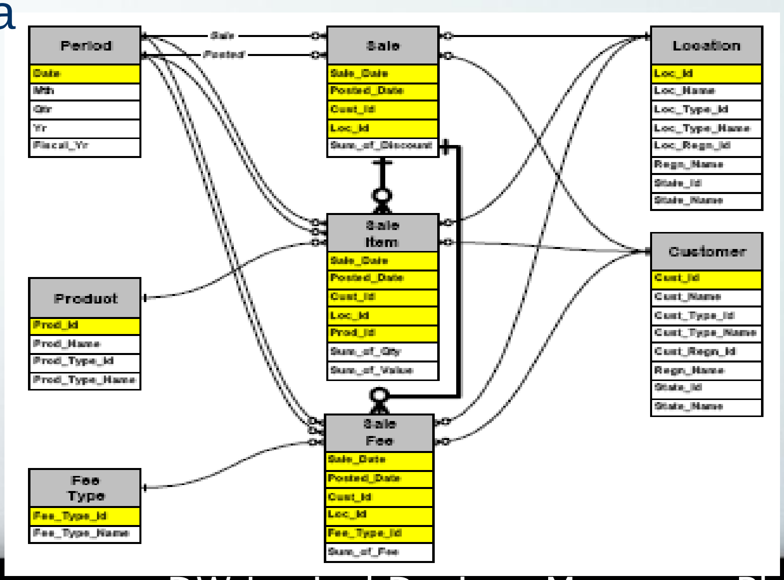
Relational Implementation



Star Schema



Snowflake Schema



Constellation Schema

Comparison Between Relational Implementation

	Star Schema	Fact Constellation Schema	Snowflake Schema
Efficiency	High	High	Moderate
Usability	High	Moderate	Moderate
Reusability	Low	Low	High
Flexibility	High	High	Moderate
Redundancy	High	High	Low
Complexity	Low	Moderate	Moderate

Comparison …

DW schema	Advantages	Drawbacks
Star schema	• The simplest structure (Moody and Kortink, 2008) • Reduces number of tables and therefore enables optimisation (Basaran, 2005) • The number of relationships between the tables can be reduced (Basaran, 2005). • The number of joins needed in user queries can be reduced (Basaran, 2005). • Query performance can be accelerated.	• Very inflexible (Teklitz, 2000) • For every gigabyte of row data, a schema will require at least an additional gigabyte for aggregations (Teklitz, 2000) • Requires considerable development maintenance effort to manage schema-oriented DW (Teklitz, 2000)
Fact constellation schema	• Storage space can be saved through reusable dimension tables (Levene and Loizou, 2003).	• It may not be helpful for small organisations because of its complexity (Feng et al, 2004)
Snowflake schema	• Hierarchical structures of each dimension can be shown explicitly (Teklitz, 2000) • Intuitive and easy to understand (Arfaoui and Akaichi, 2012) • Data aggregation can be accommodated (Arfaoui and Akaichi, 2012) • Easy to extend through additional new attributes without inference with existing database programmes (Arfaoui and Akaichi, 2012).	• Increases unnecessary complexity (Teklitz, 2000) • Diminishes query performance (Teklitz, 2000)

Comparison between multidimensional Implementation

	Condensed cube (Feng et al, 2004; Wang et al, 2002)	Dwarf (Sismanis et al, 2003; Sismanis and Roussopoulos, 2004)	QC-Tree (Lakshmanan, 2003)
Size	Much smaller size	Highly compressed and clustered data cubes	Very compact data structure
Compression	• Fully pre-computed cube without compression • Neither decompression nor further aggregation is required when answering queries.	• Complete architecture that support queries, updates and roll-up data • A tuneable granularity parameter that controls the degree of materialisation performed	Elegant and thin

Optimization Technique can be adopted

- Index
- Materialized View
- Data Fragmentation (Aouiche, 2005)
- Without optimisation techniques, queries may take hours or days to execute because of the high complexity of queries that are related to a large number of joins with dimension tables

Comparison of Indexing Technique

Indexing techniques	Advantages	Drawbacks
Bitmap indexing	• Widely used in DW environment • Response time can be minimised. • Storage needs can be minimised compared with other indexing techniques • Dramatic performance for a small amount of memory or CPU • Efficient maintenance	• Slow performance for high-cardinality column data. • More work is required if index is modified. • Concurrency occurs if any modification on bitmap indexes is inequitable.
Cluster indexing	• Performance can be optimised. • Good for range-based queries but needs sorted data	• Increasing sorting costs for unsorted data • Costly operation because the re-ordering of data is needed for data insertions (Davidson, 2008; Aizawa, 2002)
Hash-based indexing	• Large amounts of data can be minimised (Delmarco, 2006). • Average look-up cost can be minimised through hash function, bucket table size and internal data structures. • No key to be sorted • Best option for equality selections	• Leads to collusion • Range queries is unsupported • Leads to long chains in static hashing • Impossible for hash reversing

Fragmentation Technique

- Vertical → splits tables by column; one table is divided into two or more tables →
- Horizontal → splits tables by row; the tables are the same as those in the original, except that the rows are split → to minimise irrelevant data access → is designed for partitioning a relation into a set of smaller relations so that only one fragment is executed by many applications
- Hybrid → horizontal fragmentation followed by vertical fragmentation or vice versa

Aggregation

- Aggregates are needed in case of high load predictable queries exist. In such a case, faster response can be obtained by aggregates, as well be having results already stored in aggregates. Summary data must be applied only in critical condition.
- Based on practical experience, in case of small data in the fact table, no aggregation is needed in such case
- Commonly, selective aggregation is used, depending upon the requirements of organisation and often asked questions
- The possible total number of aggregations can be defined by simply multiplying the number of levels in each dimension hierarchy.

Aggregation can be stored

- As new field in existing fact table.
- As a new field in existing fact table, aggregations face the following issues:
  - ✓ Issue of double counting.
  - ✓ Aggregation can be seen by the user.
- As new fact table
- As new fact table, the following benefits can be obtained compared with previous method:
  - ✓ Double counting issue resolved.
  - ✓ Aggregation unseen by the user.
  - ✓ Easily updated in future without any problem to tables.
  - ✓ Field size of aggregation does not affect the size of the field for the basic data.

Typical Queries on DW

(1) Roll-up: aggregate fact attributes to view data at a higher level of abstraction.

(2) Drill-down: disaggregate fact attributes in order to introduce further details.

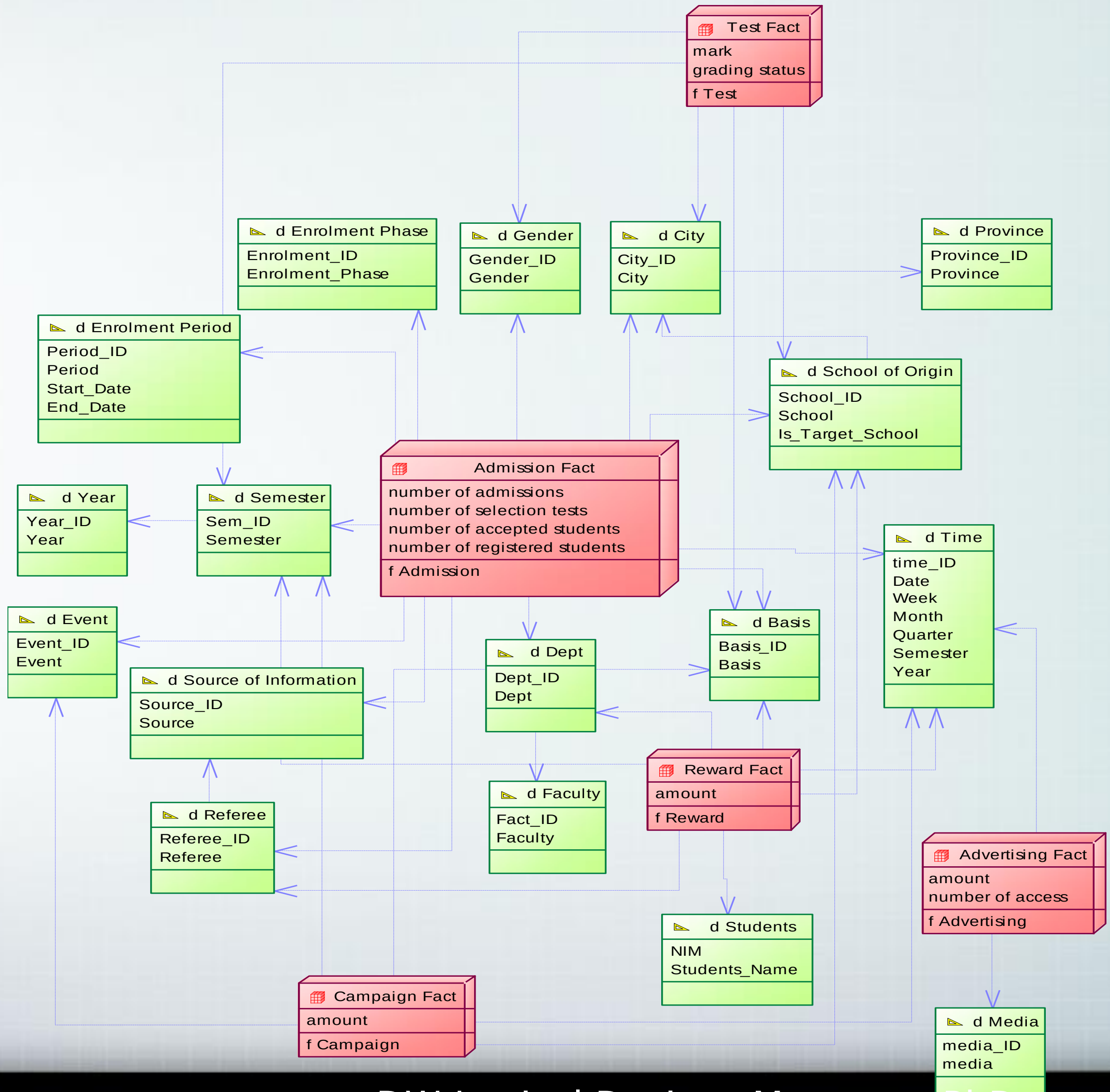
(3) Drill-cross: relate and compare distinct facts.

(4) Slice-and-dice: select and project facts so as to reduce their dimensionality

Practical Evidence

To illustrate the proposed model, a case study on the student admission process that is specifically related to marketing activities was conducted. A private university in Jakarta intends to build a monitoring system for student admissions. A series of related marketing activities have been carried out to increase student enrolment. Improvements in decision making related to the admission system is the expected benefit from the implementation of a DM.

Translation from conceptual to logical design



Thank You !

Munawar, PhD