Data Warehousing

- A Typical Large-Scale Application of Data Engineering
  - Architecture for Decision Support Systems
    - cf. On Line Transaction Processing for operational databases
  - Stock up data of operational transaction processing
    - And use the data for determining strategies of enterprise

Examples of Data Warehouse

- POS (Point Of Sales) data of supermarkets/convenience stores
  - Stock management, Displays of goods, Bargain strategies
  - Combination analysis of purchase (Basket analysis)
- Credit card transaction
  - Dispatching direct mail, analysis of customer reliance
- Cable TV pay-per-view transaction
  - The most popular cable programs for some customer groups
- Telephone call transaction
  - Time and duration analysis for customer packages

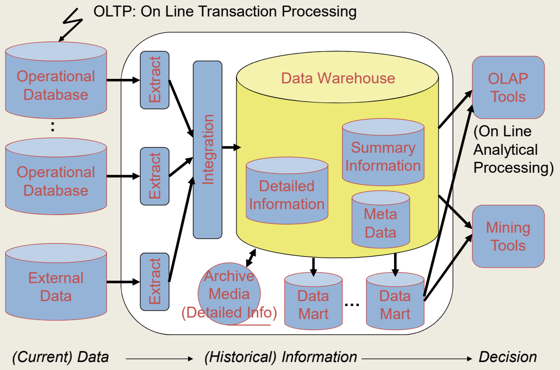
Operational DB vs. DWH

- Current State vs. Transaction History
  - Both size increase, but DHW is faster
- Many transactions including update vs. few, mostly retrieval transactions.
  - Operational DB:
    - Concurrent accesses for small amount of data items
    - Response time intensive
  - DWH:
    - Batch access for a large amount of data set
    - Throughput intensive

Data Warehouse Size

- Huge amount of data should be stored into data warehouse
- Estimation:
  - 1 KB/transaction, 100 transaction/sec (TPS)
  - 100 KB/sec = 360 MB/hour ≈ 10 GB/day ≈ 3 TB/year
- Walmart (A famous supermarket in U.S.A) has 24 TB of DWH (1997)
  - 1999:101TB, 2004:570TB, 2008:2.5PB, 2014:30PB

An Architecture for DWH



Data Loading (1)

- Extract
  - Extract Data from Multiple Foreign Sources
    - Operational Databases
      - RDB, ODB, ORDB, etc.
      - Different vender
    - External Data
      - CSV Files (Excel), etc.
  - Data Cleaning (Cleansing)
    - To guarantee consistency of data
    - e.g., by keeping integrity constraint

Data Loading (2)

- Integration
  - Transform data format of extracting data and merge them
  - Unify synonym and format
    - Personal Computer and PC
    - 15/10/2015 and Oct. 15, 2015
    - ASCII code, JIS code
    - Big/small-endian
- ETL (Extract Transform Load) Tool

Data Loading (3)

- Refresh
  - Recomputation
  - Incremental Loading
    - e.g., Redbrick Table Management Utility
  - Timing for refresh
  - Synchronization among sources

Internal Structure of DWH (1)

- Detailed Information
  - Unified data from each source
  - Some parts of detailed information is stored into archive media
    - Near Line Storage
- Summary Information
  - Summarized by aggregate functions
    - Group-by, SUM, AVG, MAX, MIN, Count
    - Store the results into data warehouse
      - to speed up the performance of common queries
    - Have to maintain the state up-to-date
      - update every time new data is loaded
    - Not have to be backed up

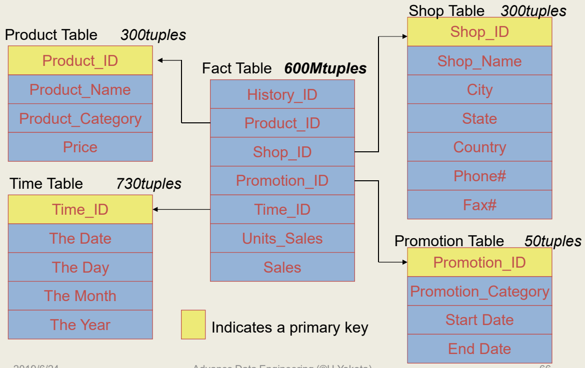
Internal Structure of DWH (2)

- Data Marts
  - Use Decision Support in each section
    - to speed up by reducing amount of data
  - Geographical distribution (e.g., placed in branches)
- Meta Data
  - Data for data location/state
    - Which part is placed in a Data Mart
    - Which part is summarized in Summary Information

Data Structure for DWH (1)

- Storing Detailed Information
  - Mainly in Relational Database Model
  - Star Schema
    - A Fact Table and Dimension Tables
      - Fact Table: History of transaction
      - Dimension Table: Master data
    - Each entry in a Fact Table is a primary key of some Dimension Table
    - A Fact Table has a great number of tuples

An Example of Star Schema



Data Structure for DWH (1)

- Storing Detailed Information
  - Mainly in Relational Database Model
  - Star Schema
    - A Fact Table and Dimension Tables
      - Fact Table: History of transaction
      - Dimension Table: Master data
    - Each entry in a Fact Table is a primary key of some Dimension Table
    - A Fact Table has a great number of tuples
  - Snowflake Schema
    - Hierarchical structure of a Dimension Table
      - Capable of reducing redundant entries

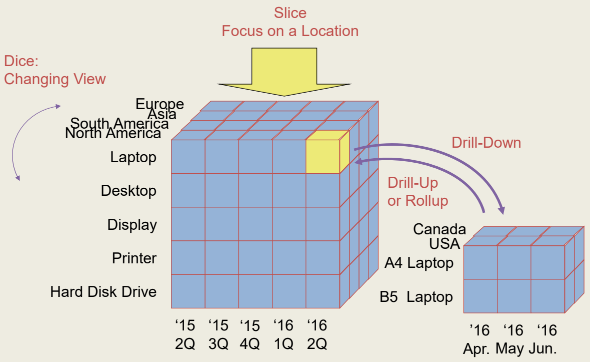
Data Structure for DWH (2)

- Storing Summary Information
  - Multidimensional Data
    - also called as a **Data Cube**
  - Applying Group-By and other aggregate functions for the Fact Table by some attributes of Dimension tables, beforehand.
    - to speed up the performance of common queries
    - some operations are available for Data Cubes

Operations on a Data Cube

- Dice:
  - Changing the view
- Slice:
  - Focusing on some dimensions.
- Drill-Down:
  - See more detailed veiws
- Drill-up (Rollup)
  - See more global view

An Example of a Data Cube



2019/6/24 Advance Data Engineering (©H.Yokota) 70

OLAP Architecture

- On Line Analytical Processing
- ROLAP
  - Relational OLAP
  - Based on Relational Operations
- MOLAP
  - Multi-dimensional OLAP
  - Based on Multi-dimensional Data
- Hybrid OLAP
  - Combine ROLAP + MOLAP

2019/6/24 Advance Data Engineering (©H.Yokota) 71

An Example of ROLAP Queries

- Query:
  'Derive the total sales of A4-type laptop personal computers that were sold in the U.S.A. as the Summer Campaign during August, 2015.'
- Many Join operations between the Fact Table and the Dimension Tables are required.

```
SELECT  SUM(Sales)
FROM    Fact_Table
WHERE   Product_ID IN
 (SELECT Product_ID
  FROM   Product_Table
  WHERE  Product_Category = 'A4 Laptop')
AND     Shop_ID IN
 (SELECT Shop_ID
  FROM   Shop_Table
  WHERE  Country = 'U.S.A.')
AND     Promotion_ID IN
 (SELECT Promotion_ID
  FROM   Promotion_Table
  WHERE  Promotion_Category = 'Summer Campaign')
AND     Time_ID IN
 (SELECT Time_ID
  FROM   Time_Table
  WHERE  The_Year = 2015 AND The_Month = 'Aug.')
```

2019/6/24 Advance Data Engineering (©H.Yokota) 72

View Materialization

- Keeping the intermediate state of query results as a relation (Materialized View).
  - It reduces time for retrieval
- A Problem
  - When the contents of original database are update, the change has also to be applied to the view.
    - It takes costs (especially for aggregate results)
- Trade-off between retrieval speed and update cost

Multiple Aggregation

- It is usual to derive multiple aggregation in Data Warehouse (for a Data Cube)
- A method of applying multiple aggregate functions to a tuple that is read from the local disk in parallel is also proposed.

↓

- Multidimensional Aggregate Function

Partial Order Relation for Multidimensional Aggregate Function

- Query Example 1

```
SELECT Product_ID, Shop_ID, SUM(Sales)
FROM   Fact_Table
GROUP BY Product_ID, Shop_ID
```

- Query Example 2

```
SELECT Product_ID, SUM(Sales)
FROM   Fact_Table
GROUP BY Product_ID
```

- The result of Query Example 1 can be used for calculating Query Example 2.
  (Product_ID, Shop_ID) ≥ Product_ID

Result Examples

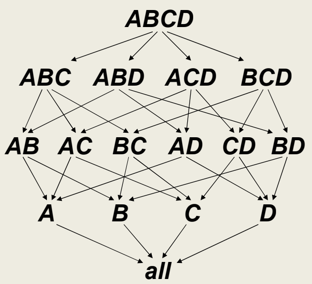
- Q1

Product ID	Shop ID	SUM(Sales)
P001	Shop-A	10
P002	Shop-A	35
P001	Shop-B	20
P002	Shop-B	50

- Q2

Product ID	SUM(Sales)
P001	30
P002	85

A Lattice of Multidimensional Aggregation



- Form a Lattice by the Partial Order of Aggregate Function
  - *A*, *B*, *C*, and *D* are the objective attributes of the aggregation
  - *all*: aggregate function of entire data cube

Optimization of Calculating A Data Cube

- In the Lattice of Multidimensional Aggregate Functions
- Smallest Parent
  - It is better to calculate *A* from *AB* or *AC* than to derive from *ABC*
  - Select smaller one between *AB* and *AC*
- Cache Effect
  - Use a result of the previous aggregate function as much as possible
- Optimization of Disk Scan
  - Consider the location of disk head, for example *ABC*, *ACD*, *ABD*, *BCD* for the attribute of *ABCD*

Smallest Parent Example

Product ID	Shop ID	SUM(Sales)
P001	Shop-A	10
P002	Shop-A	35
P001	Shop-B	20
P002	Shop-B	50

Product ID	Time ID	SUM(Sales)
P001	'15 1Q	5
P002	'15 1Q	20
P001	'15 2Q	15
P002	'15 2Q	40
P001	'15 3Q	10
P002	'15 3Q	25

Product ID	Promotion ID	SUM(Sales)
:		

Product ID	SUM(Sales)
P001	30
P002	85

Cache Effect Example

NG

1. {Product ID, Shop ID, Time ID} → {Product ID, Shop ID}
2. {Product ID, Time ID, Promotion ID}→{Time ID, Promotion ID}
3. {Product ID, Shop ID}→ {Product ID}

OK

1. {Product ID, Time ID, Promotion ID}→{Time ID, Promotion ID}
2. {Product ID, Shop ID, Time ID} → {Product ID, Shop ID}
3. {Product ID, Shop ID}→ {Product ID}