Presented by,
MySQL AB® & O'Reilly Media, Inc.

Building Scalable & High Performance Datamarts with MySQL

Tangirala C. Sarma

E-mail: tcsarma2002@yahoo.com

Agenda

- Introduction
- Section 1: Why DW/DM? (30 min)
- Section 2: DW Project Methodology (30 min)
- Break – 10 min
- Section 3: DW Modeling Techniques (60 min)
- Break – 10 min
- Section 4: DW Technologies (30 min)
- Questions

Introduction

- DW/BI Architect
- 15+ years of experience providing consulting / advisory services on DW/BI
- Implemented 10+ multi-terabyte data warehouses at enterprises incl. Wells Fargo, WAMU, REI, Reader's Digest, Marriott and 15+ data marts.
- Started looking at MySQL as a viable database platform for Data Warehousing over the last year

Why Data Warehouse/DM?

Demand for DW/DM

- Business reasons for Data Warehouse
  - Discover and act on “market windows”
  - Competitive pressures/compressed product cycles
  - Reduce business costs: inventory, advertising, production
  - The other guy has one
- IT reasons for Data Warehouse
  - Consolidate data from diverse operational systems
  - Off-load decision support from mainframe systems
  - OLTP systems make poor decision support servers

Data Warehouse Definitions

- Theoretical definition:

  "A data warehouse is a subject-oriented, integrated, time-variant, nonvolatile collection of data in support of management's decision-making process"

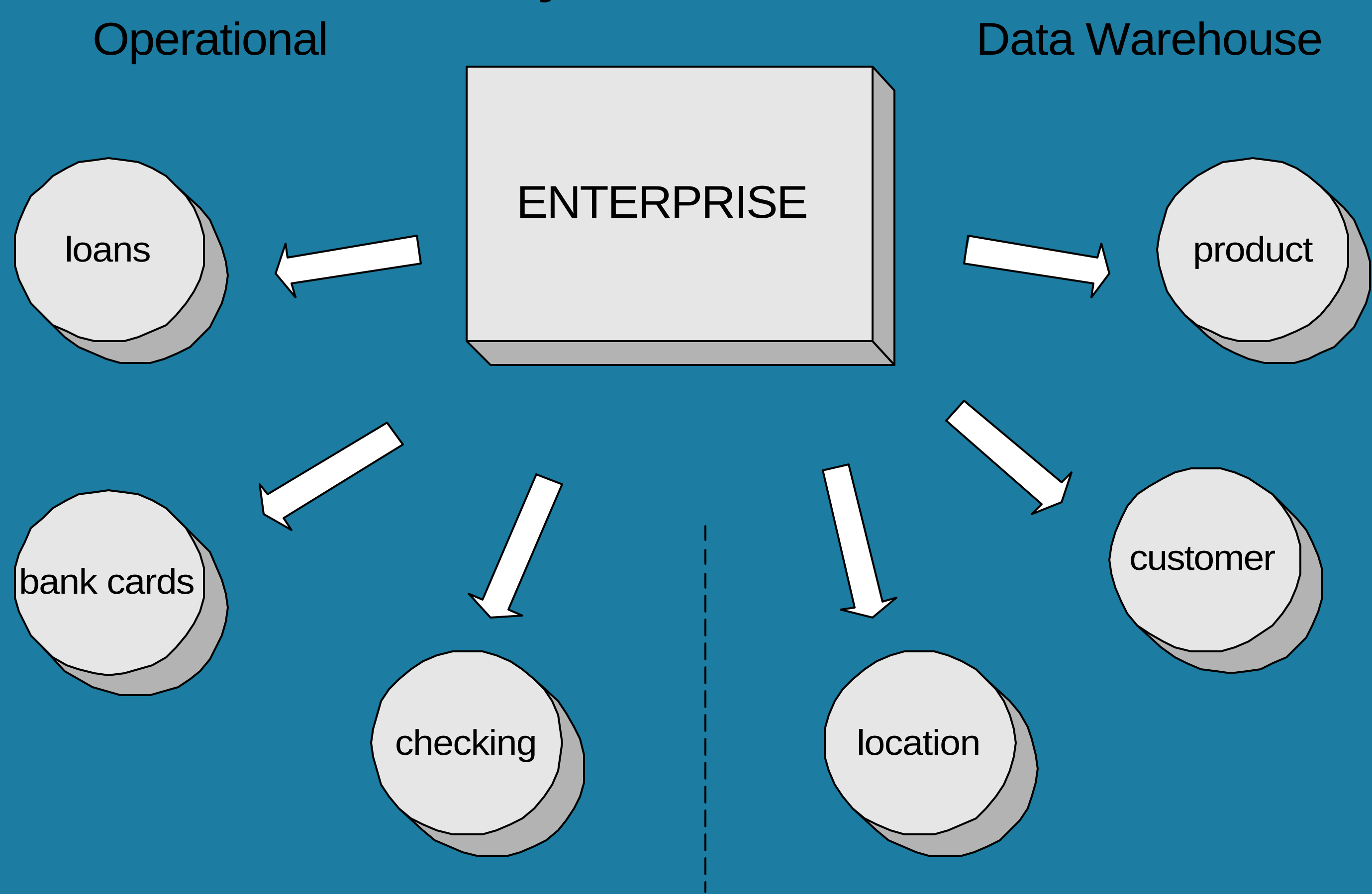
  Using the Data Warehouse - Wiley, W.H. Inmon

Data Warehouse Definitions (cont..)

- Subject Oriented
  - Revolves around business rules
  - “High level entities of the enterprise” (i.e. subject areas)
  - Organized differently than operational/functional environment

Data Warehouse Definitions (cont..)

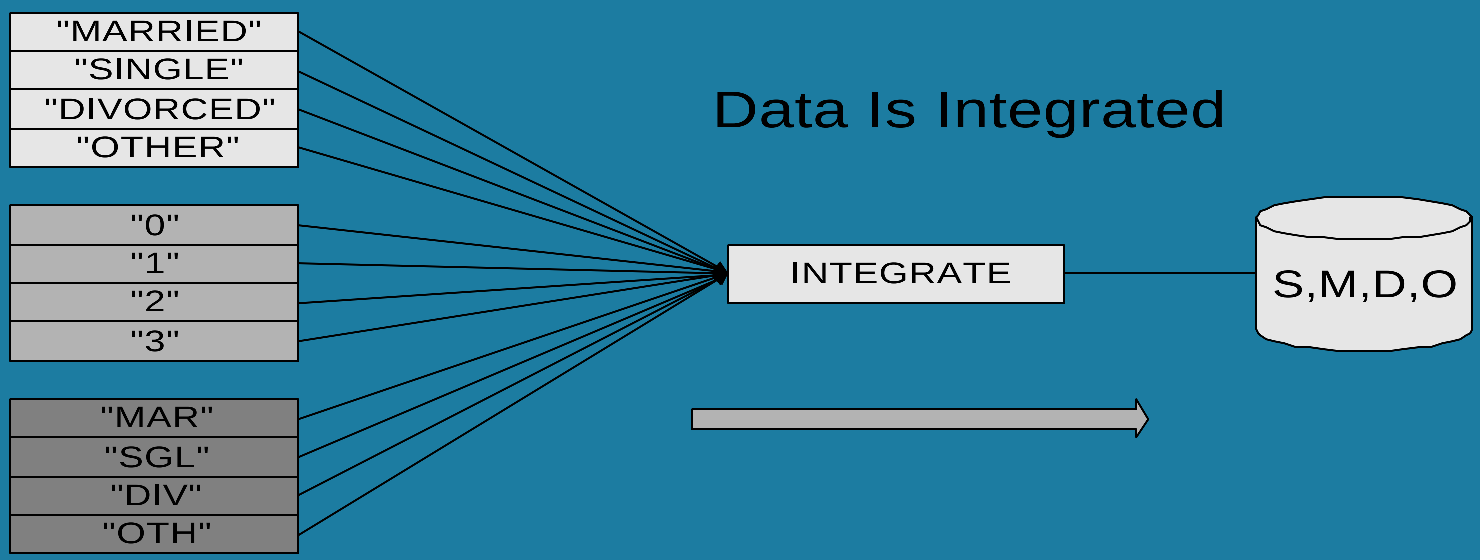
Subject-Oriented



Data Warehouse Definitions (cont..)

Integrated

- Data can come from many different sources
- Each source can look different than the other
- Once it's in the DW, it should look the same



Data Warehouse Definitions (cont..)

Time-Variant

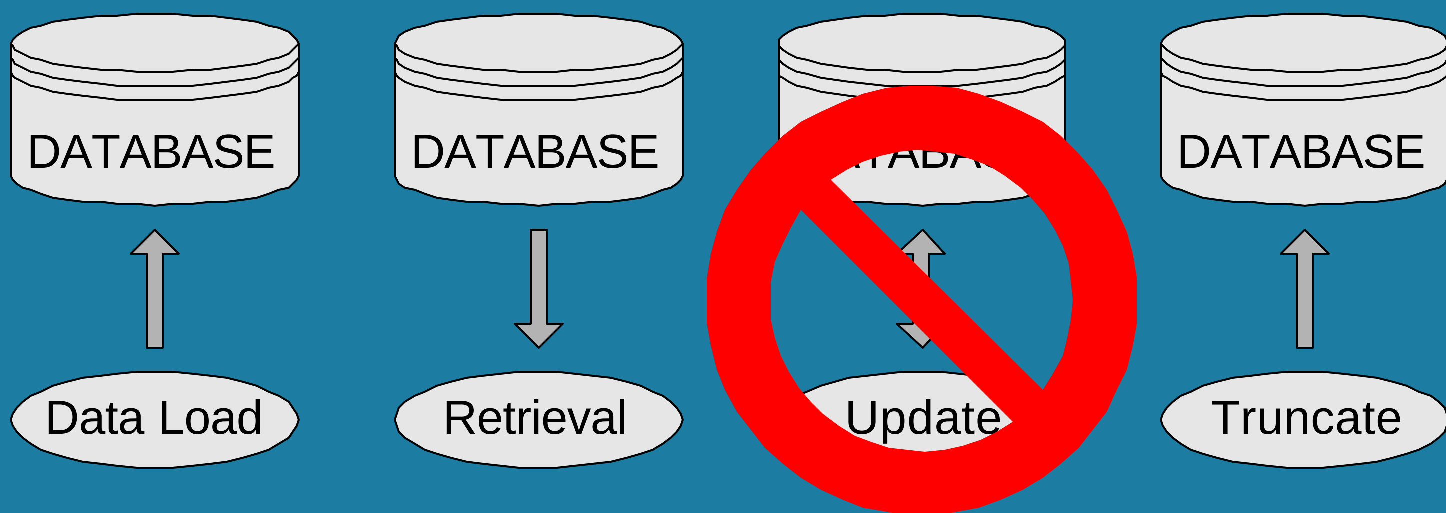
- Key structure is an element of time
- No matter how it's organized, it still represents a series of snapshots
- Snapshots or slices can lose accuracy over time, as opposed to the operational environment, which doesn't lose accuracy.

  Example: "Our product code has changed since last year."

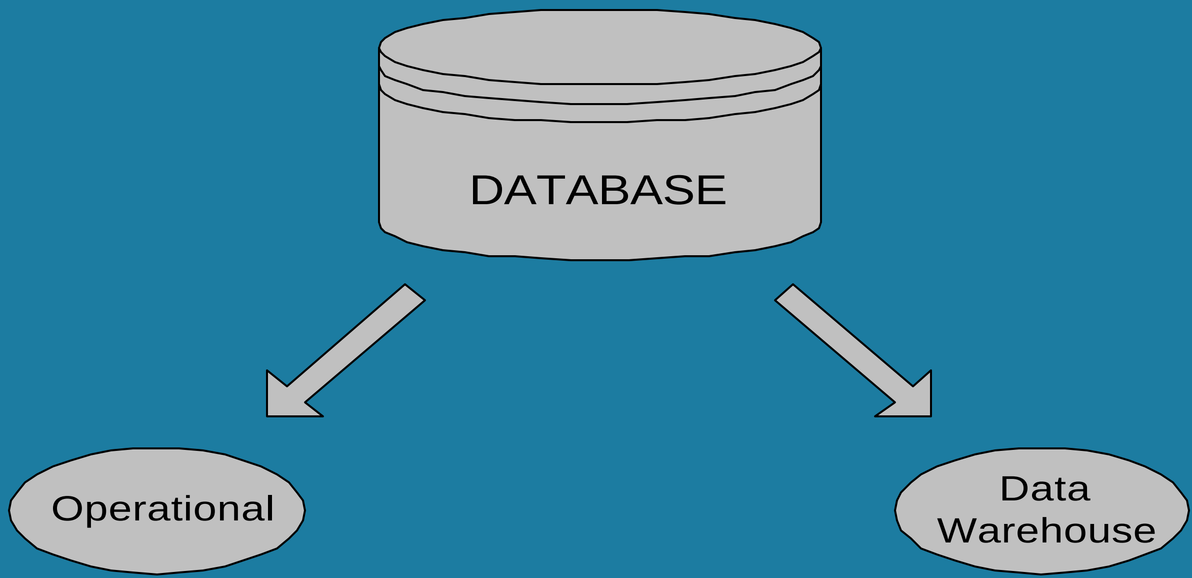
Data Warehouse Definitions (cont..)

Nonvolatile

- Two types of routine operations: Load & Access
- No update
- Removal of data according to business rule



Operational vs. Data Warehouse



Operational	Data Warehouse
- current	- historical
- not time-based	- rolling schedule
- used for “day-to-day”	- business decisions
- updated	- inserted and left alone
- large number of users	- smaller number of users
- well-defined	- ad-hoc climate

Data Warehouse Architecture

Several Flavors

- Operational Data Store
- Enterprise (Data Warehouse)
- Departmental (Data Mart)
- Personal (Mobile)

DW - Enterprise Data Warehouse Level

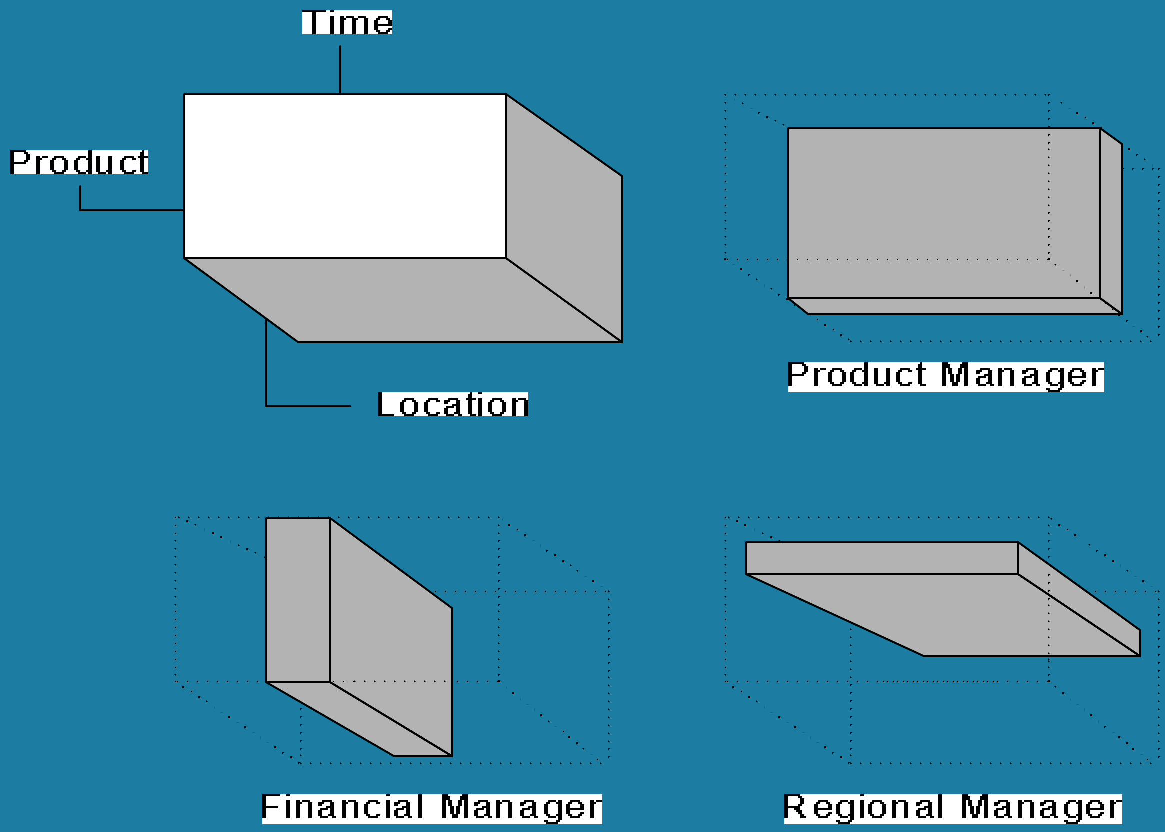
- Time variant
- Integrated
- Subject oriented
- Some summary
- Most granular

Department/Division Level

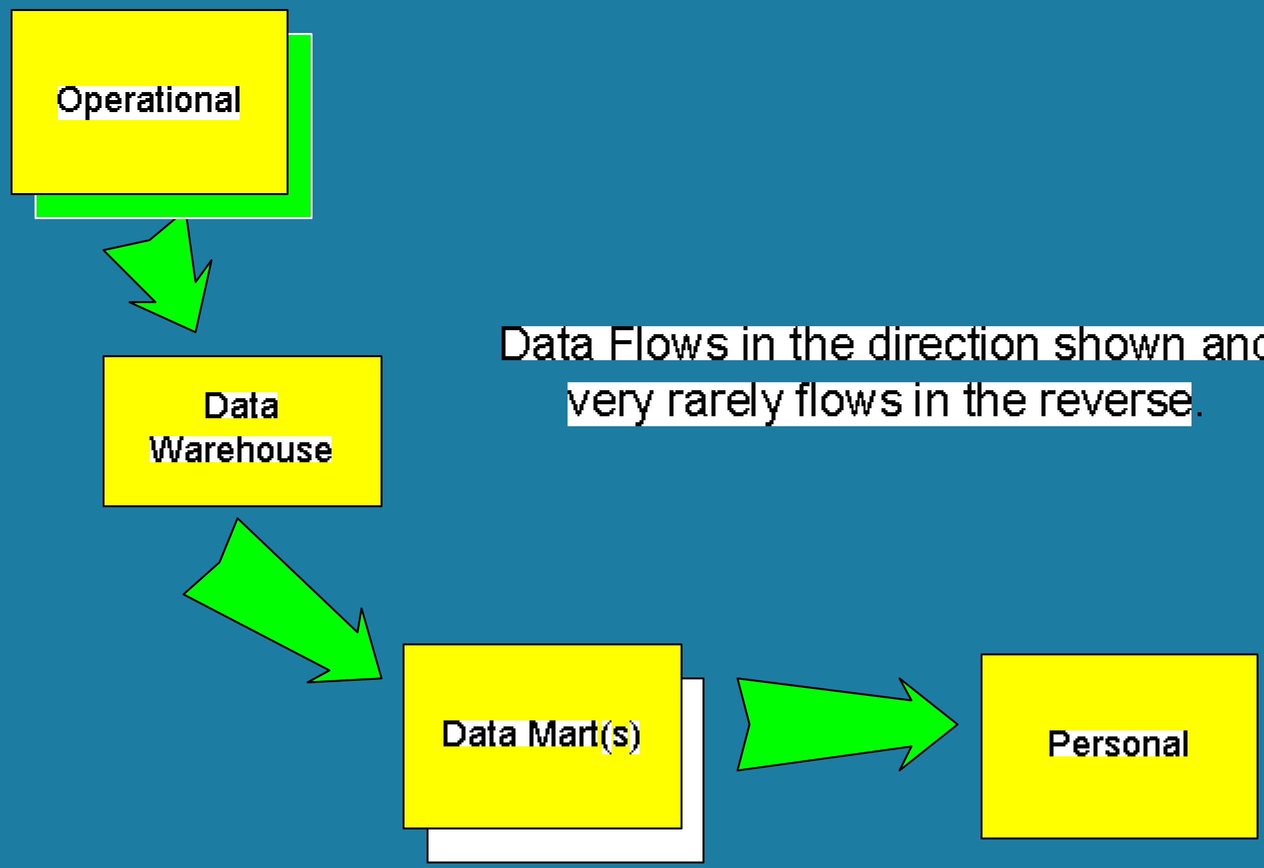
- Sometimes referred to as the Data Mart
- Some derived; some primitive
- Typical Departments

  Accounting, Marketing, Engineering, Sales etc.
- Generally where OLAP resides
- More summary data than Enterprise Level

Multidimensional Cube



Data Flow in the Warehouse

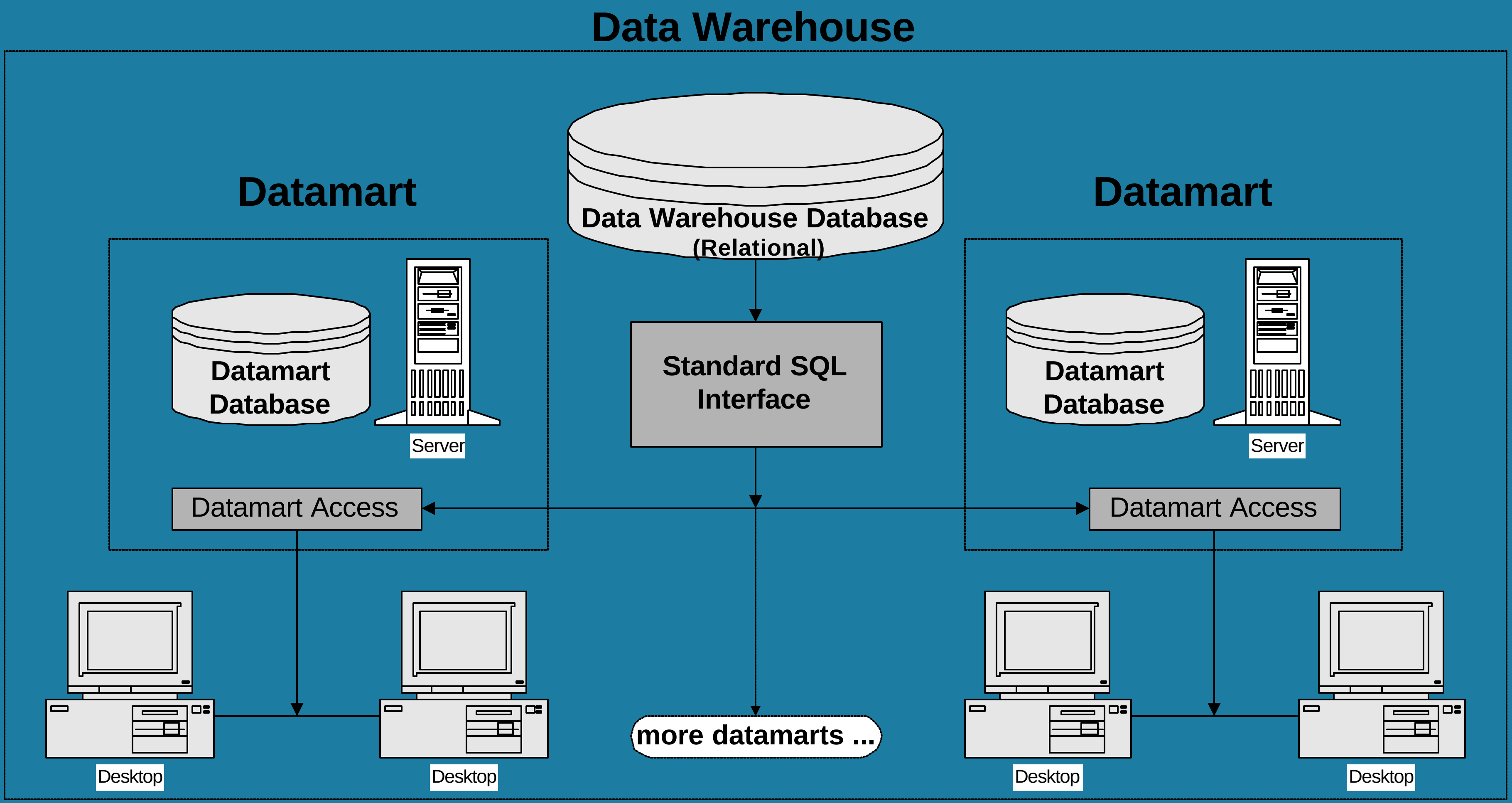


Data Flows in the direction shown and very rarely flows in the reverse.

Data Flow

- Operational to Data Warehouse
  - Significant amount of transformation takes place
- Data is lightly summarized in the Data Warehouse
- Data generally becomes more summarized as it moves to the lower levels

Physical Architecture



DW/DM Methodology

Why Methodology?

- 60% - 70% of major data warehouse initiatives fail
- Mismatch between user expectations and the reality of the end result leads to perception of failure
- Structured approach that can engage users and measure progress at every milestone is key to success

Project Approach

Goals, Planning & Preparation	Discovery/ Requirements	Design	Developmen	Testing and Deployment	Training & Roll-Out
• Project Definition • Scoping • ROI Metrics and Success Criteria • SOW • Contracts • Deliverables & Milestones • Project Plan	• Business • Functional • Technical (B&R) • Data • Logical Model • Process • Security Definition	• Databases/Physical Model • Interfaces/ETL • Applications • Application Installation • Hardware setup • Error handling and exception reporting	• System • Data Feeds • Applications Configuration	• Unit Testing • System Testing • Pre-Training • UAT	• Application Training • Documentation • Go-Live • Support

Project Management, Status Reporting, Issue and Risk Management, Facilitated Workshops, Process Controls, Joint Project Teams, Scope Management, Consensus Building, and Phase-Based Delivery

Project Approach

Goals, Planning, Preparation

- Establish Goals & Objectives with project stakeholders
- Develop Key success Metrics

Deliverables

Deliverable	Description
Objective & Goals document	High level objective of the overall initiative and key success metrics should be established and documented here.

Project Approach

Discovery, Business & Data Requirements

- Develop the vision for presentation of information with the business users.
- Determine the appropriate level of granularity
- Define usability requirements, such as uptime and average query length, of the system

Deliverables

Deliverable	**Description**
Business requirements document finalized	Contains business requirements for the DW/DM. This is the synthesized information resulting from the interviews and workshops.

Project Approach

Conceptual Model

- Conceptual model is blueprint for detailed design
- Identifies subject areas and high level data entities
- Helps with project scoping and overall project plan

Deliverables

Deliverable	Description
Conceptual Data Model	High level data model that identifies data entities, relationships and key metrics

Project Approach

Data Discovery & Data Quality Analysis

- Spend time to understand the source data & assess data quality
- Use data profiling tools to identify data availability, anomalies
- This information will help with ETL design in later phases

Project Approach

Data Discovery & Data Quality Analysis - Deliverables

Deliverable	Description
Data Source Gap Matrix	The Data Source Gap Matrix defines specific data element gaps in the available data required to support the solution and the Business Data Model as defined by the representative users. It details the gaps, and their characteristics (that is, uniqueness, availability, granularity, quality, etc.) and defines possible options or alternate solutions to support the business requirements (where feasible).
High-Level Data Quality Assessment	The High-Level Data Quality Assessment documents findings related to source data, its integrity, availability, ability to meet business requirements, and overall quality. It includes issues and possible approaches to resolution.

Project Approach

Technical Architecture

- Develop Architecture blueprint to support Development, Test and Production environments
- Develop capacity plan to support concurrency and usage needs
- Architecture should identify all the components of the solution including interfaces to the external systems
- This blueprint should help with procuring the right HW/SW to support the DW/DM initiative

Project Approach

Technical Architecture - Deliverables

Deliverable	Description
Architecture Blueprint	• Defines technical design based on business requirements • Clarifies technical components of overall solution • Where there are integration requirements, this document should summarize the configuration in a series of diagrams.
Capacity Plan	This plan defines the requirements for storage capacity for current and future needs for database storage, staging area and archival needs. Other items it should cover include Processor Performance, Memory Capacity, Storage Capacity, Backup bandwidth, Network bandwidth, extensibility of the platform etc.

Project Approach

Data Modeling & Database Design

- Design dimensions and facts to support business requirements
- Uses conceptual model (from earlier step) as starting point
- Includes creation of Logical and Physical model
- Use modeling tools to support this task

Project Approach

Data Modeling & Database Design - Deliverables

Deliverable Name	Description
Logical Database Design	The Logical Database Design documents the following design elements: ? Table, column and view definitions ? Primary, unique and foreign key definitions, column and row level validation rules (check constraints) ? Rules for populating specific columns (sequences, derivations).
Physical Database Design	The initial Physical Database Design consists of a narrative description of the database design decisions and a number of listings of the physical storage aspects of the database and its associated objects. ? Database, rollback segment, table space definitions ? File and storage definitions ? Index types ? Database object definitions (physical storage including partitioning) ? Specialized indexes ? Fact table partitioning schemes

Project Approach

Data Acquisition (ETL) Design

- Design the processes necessary to extract, transform and load the required data into the DW/DM
- Should cover design for first-time load and on-going loads
- If there is a requirement for data consolidation (e.g., identifying unique customers, house holding etc.), these rules need to be clearly documented here
- Design should include Data archival process

Project Approach

Data Acquisition (ETL) Design - Deliverables

Deliverable Name	Description
High-Level Data Source Model	The High-Level Data Source Model identifies the selected list of operational and external data sources required to meet the information requirements described in the Detailed Business Descriptions. It creates a system of record for those original data sources. This deliverable defines the flow of data between business functions and source systems, and identifies any operational constraints, which could impact the DW/DM solution. Data Volumes will also be collected.
Data Acquisition Approach	The Data Acquisition Approach defines the scope and objectives as well as the critical success factors and risks associated with the data extraction, transportation, transformation and data loads to support the solution. It describes the business' conversion and interface requirements. Topics include: ? Data acquisition techniques for extract, transport (move), transform (map) ? Data conversion ? Refresh strategies ? Load frequencies ? Data availability ? Functional requirements for extracting and loading the data ? Requirements for validating and cleaning up both extract and load ? Error condition handling

Project Approach

Data Acquisition (ETL) Design - Deliverables

Deliverable Name	Description
Customer Standardization & Matching rules	This document captures the standardization and matching rules to identify unique sites and build the customer hierarchy.
Source Data To Target Logical Database Matrix	The Source Data to Target Logical Database Matrix defines the key assumptions, mapping between source and target, and mapping rules and logic that is needed to create the conversions and interfaces necessary to support the solution. It is intended to provide the developer with the necessary information for writing accurate transformation and load logic.

Project Approach

Data Acquisition (ETL) Development

- ETL components and processes are built during this phase to cover
  - First Time Load
  - On-going Load
  - Customer and House holding rules
  - Process automation
  - Error and reject record handling
- Use of ETL tools / Data Quality tools are recommended; Writing custom SQL may introduce maintenance issues later

Project Approach

Data Acquisition (ETL) Development - Deliverables

Deliverable Name	Description
Data Cleansing and standardization components for on-going load	• These components for the on-going load are the modules that standardize the name, address and e-mail address etc. This standardized information is used by other components to de-dup and consolidate the customer records. This consolidated information is fed through the mappings to be loaded into the corresponding database objects.
Data Acquisition (ETL) components for on-going load	• The Data Acquisition (ETL) Components for on-going load are the modules that move data from sources to the DW/DM. Developers are also responsible for Unit Testing the respective modules before making them available in the test environment.
Data Acquisition (ETL) for History Load	• The Data Acquisition (ETL) Components for historical load are the modules that move data from sources to the DW/DM. Sometimes, the first time sources are different from on-going sources.

Project Approach

Data Acquisition (ETL) Development - Deliverables

Deliverable Name	Description
Data Acquisition (ETL) Automation for standard, on-going loads	The Data Acquisition (ETL) Automation for on-going loads includes the scripts and control information to manage the automated receipt and loading of data that is scheduled to be loaded into the database on a regularly scheduled basis. It will log activity and send out appropriate notices of success and failure of the data loads.
Data Acquisition (ETL) for extracts	These ETL components extract data from the DW/DM in a pre-defined format to provide to the downstream systems.
Data Acquisition (ETL) components for Data Archival	The Data Acquisition (ETL) Components for rolling off the data from DW/DM and promote the same to higher levels of aggregation tables as the data reaches history requirement ceiling.

Project Approach

System Integration Testing (SIT)

- SIT is critical to deliver high quality data and early delivery of dependable solution
- SIT team should be different from development team
- Load full sets of data (for at least 2 periods) through ETL process flow
- Monitor the system for load issues. Any load performance issues should be handled here.
- Generate reports from DW/DM and compare to known results to determine data accuracy

Project Approach

System Integration Testing (SIT) - Deliverables

Deliverable	Description
Integration And Testing Plan	The Integration and Testing Plan defines the scope and approach to handle the system, system integration and component testing activities. It includes information about: Test roles and responsibilities Test Cases and expected Test Results Test data Test estimates and scheduling
System Integration Test Results	The Component, System, and System Integration Test Results are a compilation of the sets of results that were produced from component, system and system integration testing.

Project Approach

User Acceptance Testing (UAT)

- UAT gives a first glimpse of the DW/DM to selected users
- Users should access the system using Reporting / OLAP tools
- UAT should cover standard reports / ad-hoc queries
- UAT is performed after the history data is loaded and data becomes current (on full data volumes)
- Have periodic meetings to capture feedback
- Any Query related performance issues should be fixed during this phase

Project Approach

User Acceptance Testing (UAT) - Deliverables

Deliverable Name	Description
UAT Test Plan	The User Acceptance Test Plan defines the scope and approach to handle the UAT activities. This plan is prepared by the user team working with the representative users. It includes information about: ?Test roles and responsibilities ?Test Cases and expected Test Results ?Test data ?Test estimates and scheduling
Pre-Production Validation Results	The Pre-Production Validation Results is a user statement confirming that the system is ready to go into production. Users will be responsible for conducting the UAT and documenting the test results.

Project Approach

Transition

- Transition phase moves the DW/DM from testing to production
- All production support folks need to be adequately trained in data model, ETL process flows
- Develop a Run Book to help troubleshoot potential problems with data loads

Dimension Modeling Design Techniques

Presented by
MySQL Conference & Expo
O'REILLY

Four Step Design Process

- Select Business Process
- Declare the Grain
- Choose Dimensions
- Identify Facts

Select Business Process

- Talk to users to gather what they would like to understand using the data

In this Case Study, we will model a Sales Datamart.

Users from Finance dept would like to understand sales numbers to help them with monthly reporting and forecasting.

Declare The Grain

- Grain defines the level of detail that should be made available in the dimensional model
- Ideally we should capture the lowest level of detail possible; This would provide for more flexible queries from users

In our Case Study, we will select the daily transaction summary by product by customers by Zip Code as the grain.

Choose Dimensions

- Once Granularity of the data is identified, that will determine the primary dimensionality of the fact table
- You can add new dimensions to the Fact without reloading the data as long as it does not change the grain

In our example, Customer, Product, Date and Zip Code are the primary dimensions of the fact

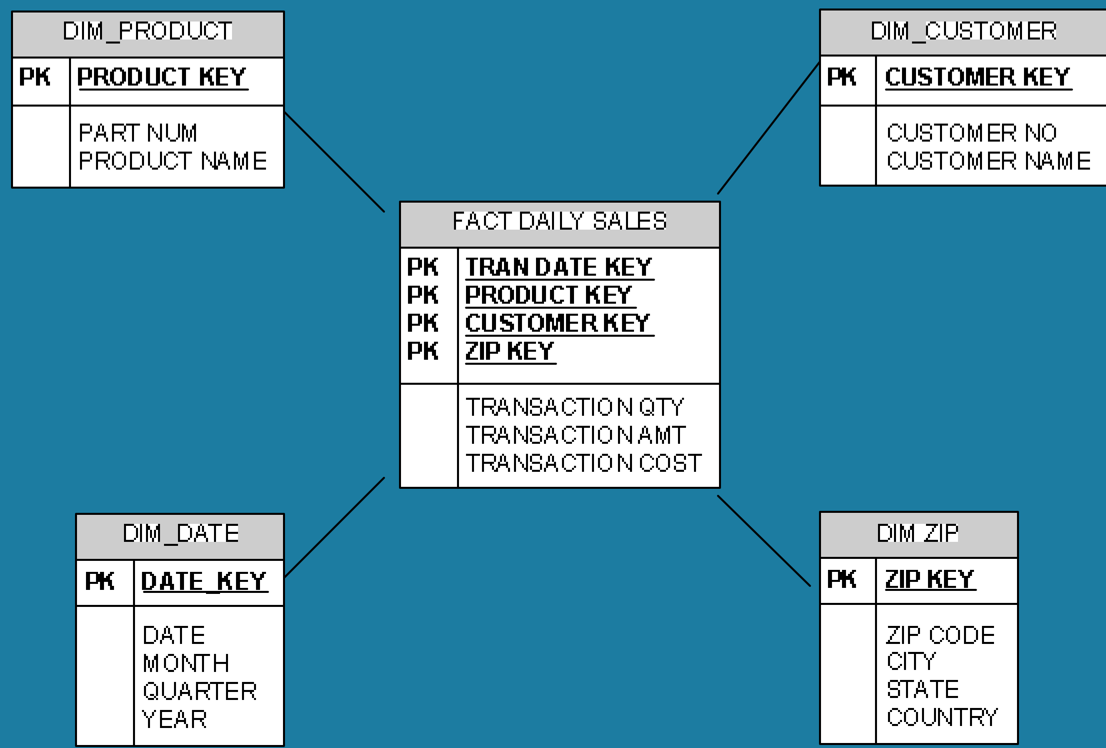
Identify The Measures

- Once Granularity of the data is identified, identify the measures in which users are interested in
- If there is a requirement for non-additive measures such as percentages and ratios, store the underlying measures in the FACT

In our example, users are interested in analyzing sales quantity and sales amount.

We can add additional measures to the fact without compromising the granularity of the fact. In our example, say, our business users are interested in analyzing the gross margin, we should add Cost to the FACT table not the calculated gross margin.

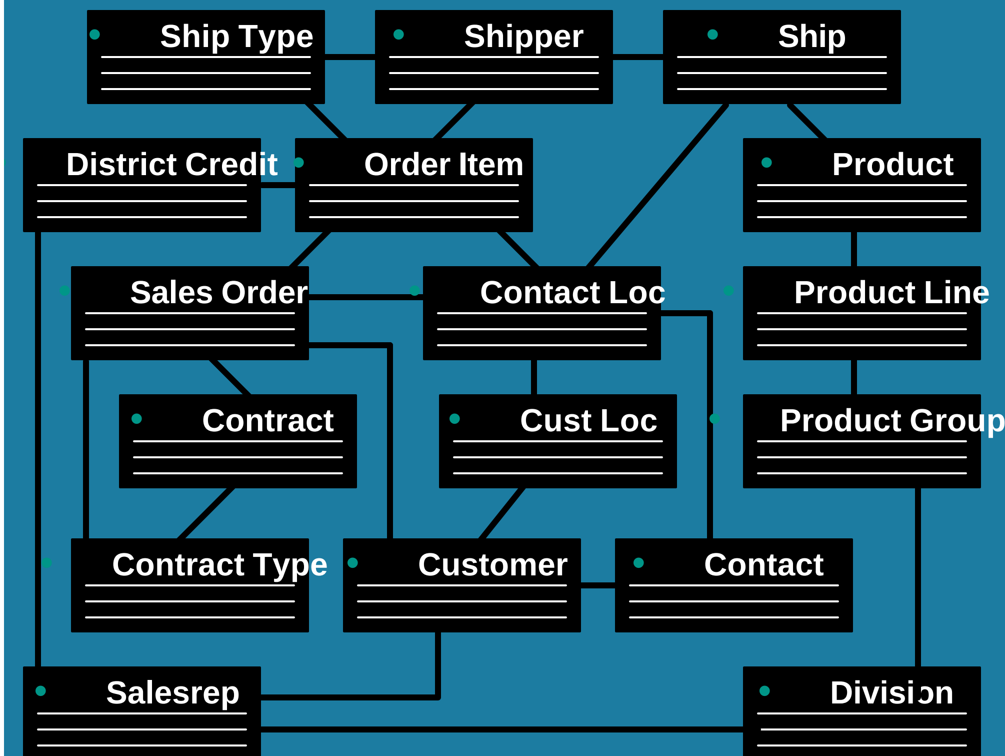
The Sales Fact



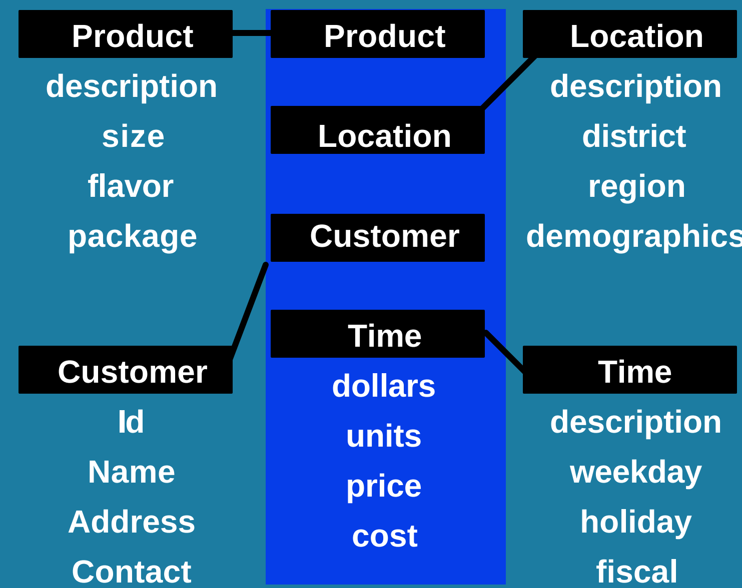
- Reference table linked to Fact table by key value
- Queries require joining reference tables with fact table

OLTP vs. OLAP Schema

Transaction-Oriented Entity-Relation Model



Dimensional Model for Star Queries



Dimension Tables

- Dimensions provide context to the data
- Define business in terms already familiar to users
- Wide rows with lots of descriptive text
- Generally Small Tables (there are exceptions)
- Joined to Fact table by a foreign key (not necessarily enforced)
- Typical Dimensions
  - Time, Geography, Product, Customer etc.

Fact Table

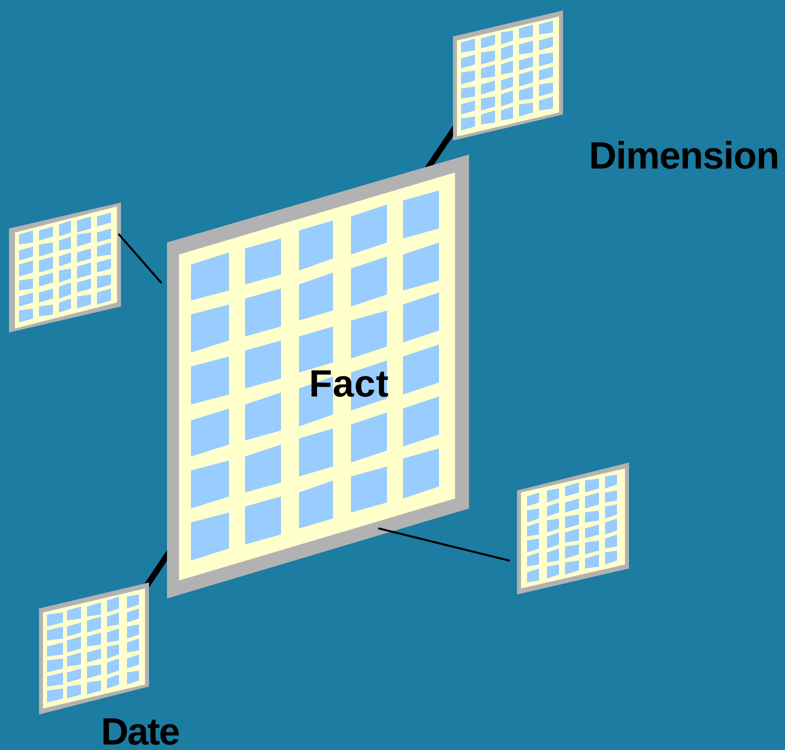
- Central table in the star schema
- Typical example: Sales Information (Quantity sold, Amount Sold)
- Contains mostly raw numeric items
- Narrow rows, a few columns at most
- Large number of rows (billions in some cases)
- Access via dimensions

STAR Dimension table

- Generally has a sequence key (also called warehouse key) that is the PK on the table
- Has a source key which will be used to match rows while loading
- Hierarchy levels are stored as columns in the same table (e.g., DATE, MONTH, QUARTER, YEAR)
- Also has attributes to enable rich data analysis (e.g., HOLIDAY IND, WEEKDAY IND etc.)

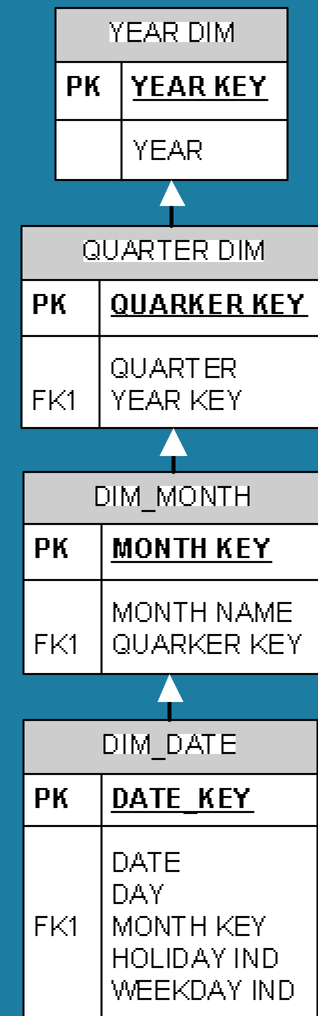
DIM_DATE	
PK	**DATE_KEY**
	DATE DAY MONTH QUARTER YEAR HOLIDAY IND WEEKDAY IND

Star Schema Example

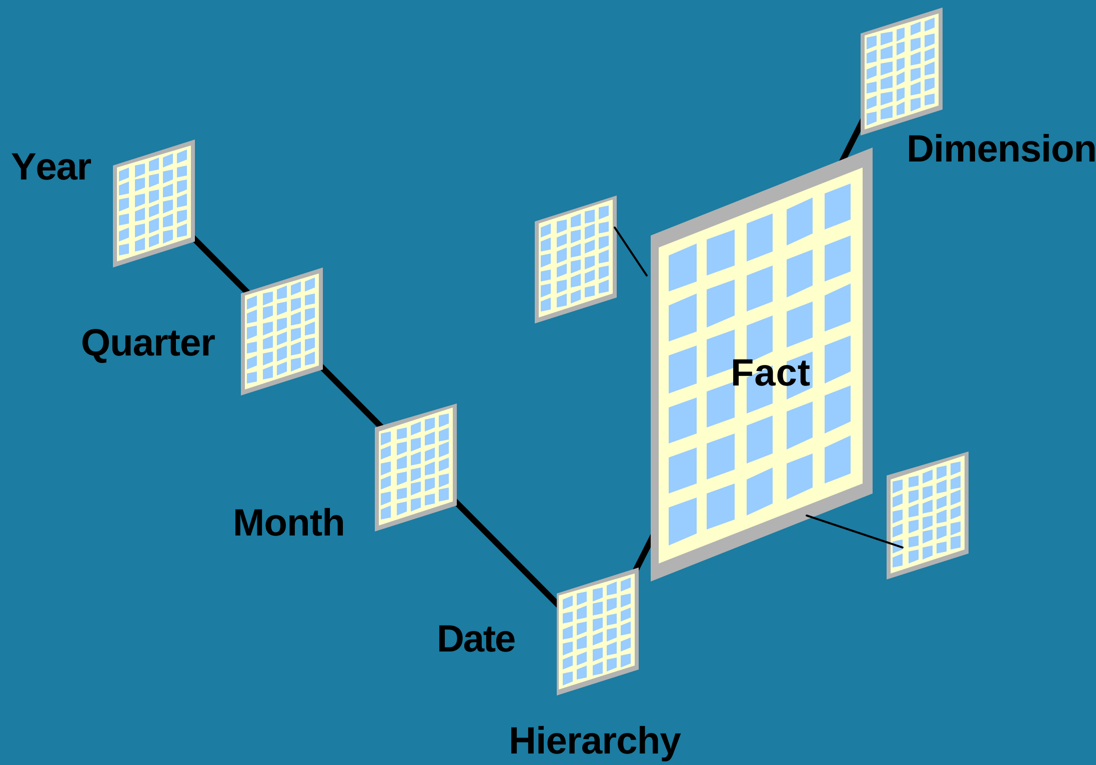


Dimension Normalization (SnowFlaking)

- Each Level in the hierarchy is represented as its own table
- Attributes for each level reside in the corresponding DIM table
- Each level is mapped to the Parent level table using FK relationship
- Has a source key which will be used to match rows while loading
- Use this technique only for very large dimensions



SnowFlake Schema Example



The DATE Dimension

- DATE dimension is one dimension that you will find in every DM/DW
- You can build the DATE dimension in advance with 5 to 10 years of data depending on the history requirements
- You may have to model for Calendar and Fiscal hierarchy in the DATE dimension
- To allow for non standard calendar calculations (such as Holiday sales, Weekend sales, seasonal sales etc.), use indicators to flag the days

DIM_DATE	
PK	**DATE_KEY**
	DATE DAY HOLIDAY IND WEEKDAY IND DAY OF WEEK CALENDAR WEEK NO CALENDAR MONTH CALENDAR QUARTER CALENDAR YEAR FISCAL MONTH FISCAL QUARTER FISCAL YEAR

Example of a simple DATE dimension

Other Dimensions

- Other dimensions are generally sourced from operational sources
- Capture as many descriptive attributes as possible to allow for robust and complete analysis
- Model the dimension as a STAR or SNOWFLAKE depending on the size
- Really large dimensions, should use SNOWFLAKE model to avoid storing the attribute values redundantly at every level.
- Always add a dummy row to represent ‘unknown’ dimension value to avoid storing NULL keys in the FACT table.

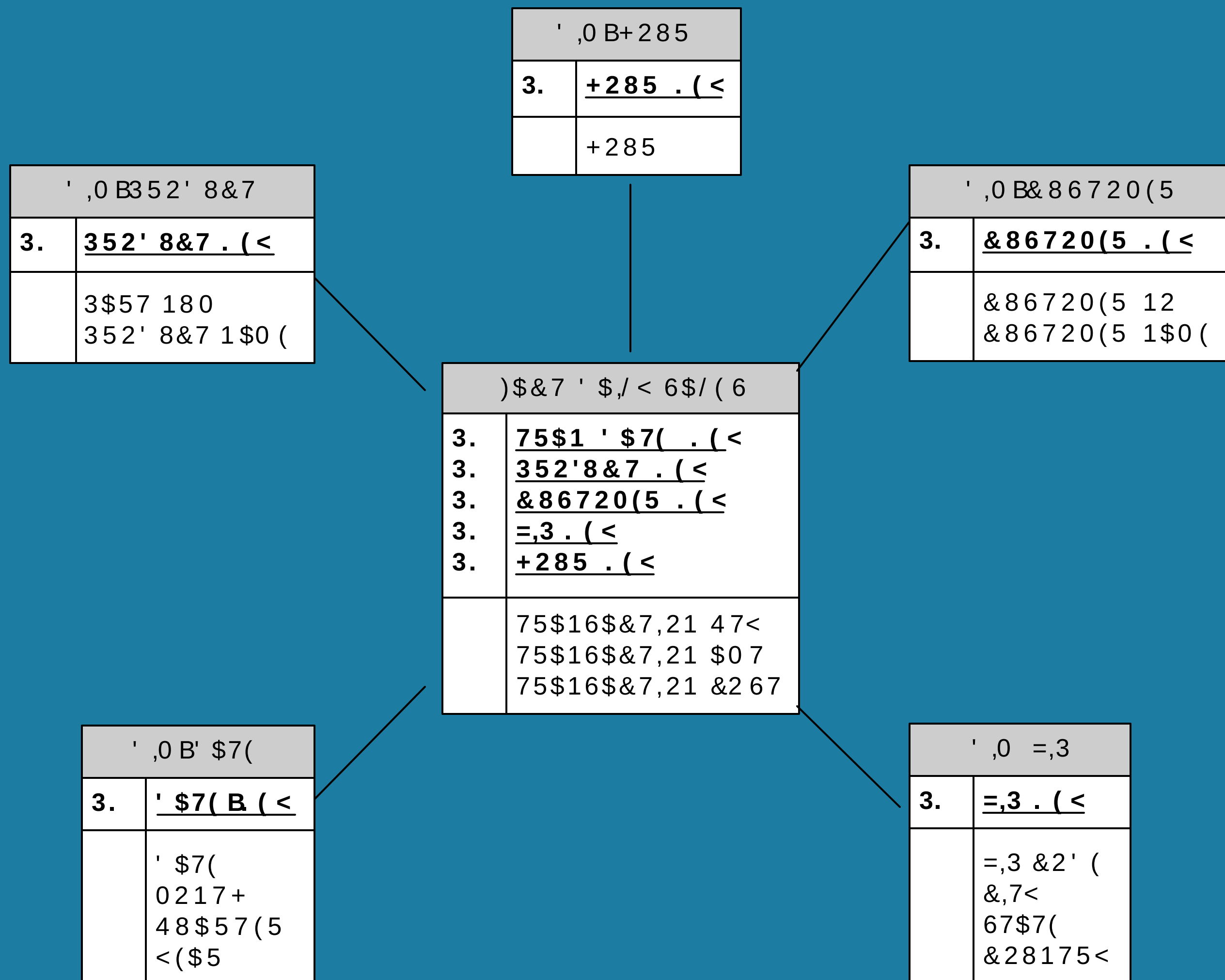
Changing The Grain

- Let's say after we designed the DW/DM and loaded the data, users would like access to more detailed level data, say HOURLY sales not DAILY sales

  (Another example would be if the users want to see the daily summary at the store level within Zip code)

- HOURLY will change the Grain of the Fact
- Changing the Grain, will require reloading all the historical data which will be painful
- Make sure you capture the requirements early on, to avoid these painful scenarios

The New SALES FACT



Degenerate Dimensions

- In our example, say, users now would like to see Transaction level info not hourly or daily summary
- This requires, getting Transaction Number from the source
- Transaction number does not have any additional attributes except that it identifies the grain of the fact
- Empty dimensions such as Order Number, Invoice Number, Transaction number are generally referred to as Degenerate dimensions
- Degenerate dimensions do not require a separate dimension table

Schema Extensibility

- New Dimension attributes – results in new columns added to the dimension table
- New Dimensions – Results in new FK to Fact table as long as it does not change granularity
- New Measured facts – New columns to the Fact table as long as the dimensionality is same
- Dimension becomes more granular – Requires rebuilding Fact table
- Adding new data source which may result in new dimensions – May require creating new fact table if granularity or dimensionality changes

Determining FACT tables in DW/DM

An Example

Dimension	Date	Hour	Month	Customer	Product	Geography	SalesRep
Measure							
Sales Quantity	X	X		X	X	X	X
Sales Amount	X	X	X	X	X	X	X
Cost Amt	X		X	X	X	X	
Gross Profit	X		X	X	X	X	

How Many Fact tables in DW/DM?

- Granularity and Dimensionality of measures determine the number of Fact tables required
- Identify all the measures that the users would like to analyze on
- Identify the granularity at which these measures should be stored
- Determine the dimensionality of each measure
- Tabulate these results in a spreadsheet
- This information should guide in determining the number of FACT tables in DW/DM

Surrogate Keys

- Surrogate keys are integers that are assigned sequentially while populating dimension tables
- Avoid embedding intelligence in generating these keys
- If the source id has intelligence built in (say part num has 1st 3 letters identifying manufacturer), these codes should appear as separate columns in the dimension table
- Surrogate keys shield the DW/DM from operational changes
- Always add a dummy row with '-1' as sequence id, to identify 'unknown' codes in the source data

Surrogate Key

DIM_PRODUCT	
PK	PRODUCT KEY
	PART NUM PRODUCT NAME PROD SUB CATEGORY PROD CATEGORY

Natural Key

Slowly Changing Dimensions

- Dimension attributes change over time
- Users may want to track these attribute changes, to help them with analysis of the data
- Early on, work with the users to identify which attributes they would like to track history
- You should educate the users to resist the urge to track 'everything' since it will have performance implications
- Do proper analysis of the source to determine the frequency of change to help identifying the right strategy in handling the changes

Handling Slowly Changing Dimensions

- Type 1 – Overwrite the attribute value
- Type 2 – Track every change by adding a new row
- Type 3 - Track only the last change by adding a new column
- Combination of these techniques can be used in the same dimension table as required

Type 1 – Overwrite the value

- This technique will be used if users don't want to track history of an attribute
- Match on the source id and replace the value of the attribute with the new value, if there is a change

		Product Key	Part Num	Prod Name	Category
	Original Row	1001	ABC110	iPod 1GB	Entertainment
	Changed Row	1001	ABC110	iPod 1GB	Music

Type 2 – Add a new Row

- This technique should be used if users want to track history on all the changes to the attribute(s)
- Create a new row with a new sequence id
- Use Effective Date and Expiry date to identify point in time attribute value (Eff Date <= ‘date value’ and Exp Date > ‘date value’)

	Product Key	Part Num	Prod Name	Category	Eff Date	Exp Date
Original Row	1001	ABC110	iPod 1GB	Entertainment	1/31/2008	
Changed Row	1001	ABC110	iPod 1GB	Entertainment	1/31/2088	2/28/2008
Changed Row	1002	ABC110	iPod 1GB	Music	2/28/2008	12/31/9999

Type 2 – Add a new Row (Cont..)

- Match dimension row on source id; if exists and attribute changes, create a new row with latest values; Update the old row to set the expiry date
- Set Expiry date to '12/31/9999' to identify current row
- If attribute frequently changes, it may result in a new row on every load which will increase the size of the dimension resulting in performance issues

Type 3 – Track only last change

- Create a new column for the corresponding attribute to track the ‘Old Value’
- Allows the users to see the current or historical fact data by new or prior attribute values

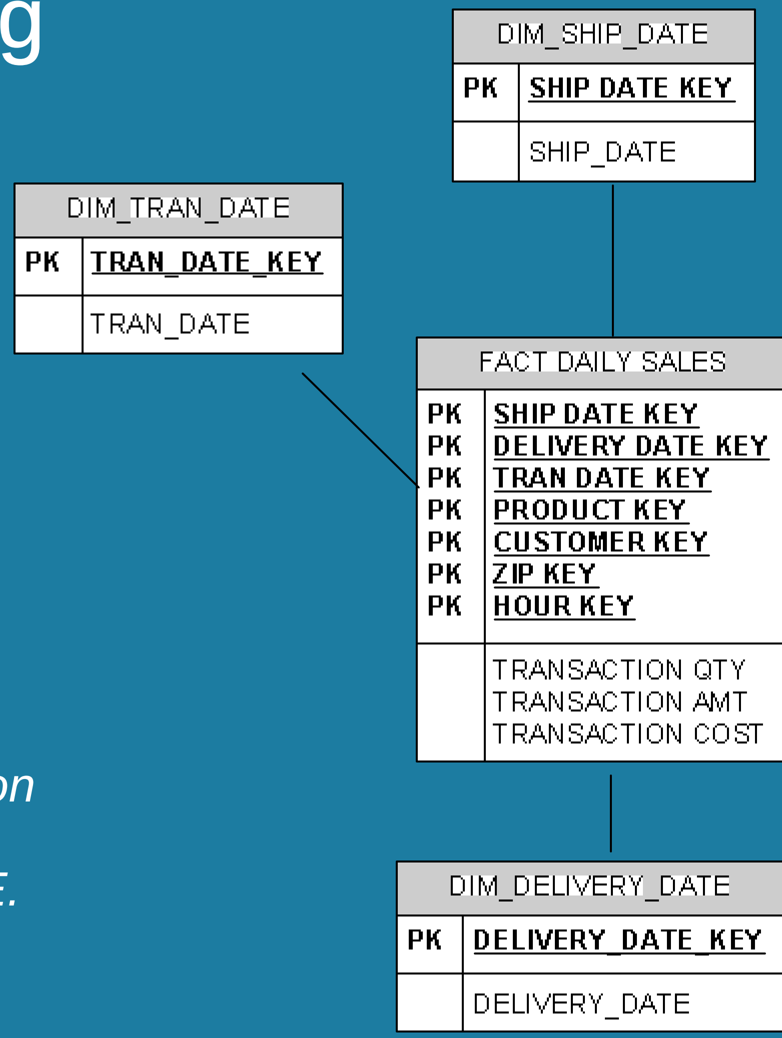
Example: To track last change to category, create last_category as a new column in the product dim table

	Product Key	Part Num	Prod Name	Category	Last Category
Original Row	1001	ABC110	iPod 1GB	Entertainment	
Changed Row	1001	ABC110	iPod 1GB	Music	Entertainment

Dimension Role-Playing

- Sometimes a single dimension may appear several times in the same fact table
- The underlying dimension is a single physical table but this should be presented to the data access tools as separately labeled view

Example: Say, in our fact table we have 'shipped date' and 'delivered date' in addition to the 'transaction date'. All these dates point to the same physical table DIM_DATE. We will need to create 'Ship Date Dim' and 'Delivery Date Dim' as views on the DATE_DIM table.



Handling Flags and Indicators

- Sometimes there are variety of flags and indicators that are part of source fact data (e.g., payment type, channel code, order type etc.)
- Options to handle the flags & indicators
  - Create a dimension table for each flag/indicator – may end having too many meaningless dimensions
  - Leave them as part of the FACT row – Not a good design. Will leave textual attributes on the FACT row
  - Strip out all flags / indicators in DM/DW – Users don't like it

Junk Dimension

- Junk dimension is a grouping of low cardinality flags and indicators
- By creating these groupings, we can eliminate having these flags as part of FACT table

In our example, create a junk dimension to combine payment type, order type, channel code. Create a sequence key for each combination and FK to Fact table.

Populating Junk Dimension

- Pre populate the table with all possible combinations of the codes

  In our example, if there are 4 payment types, 2 order types, 4 channel codes, we will end up with 4 x 2 x 4 = 32 rows in the table.

- If the combinations are too large, populate only what is observed in the source at load time

Rapidly Changing Dimensions

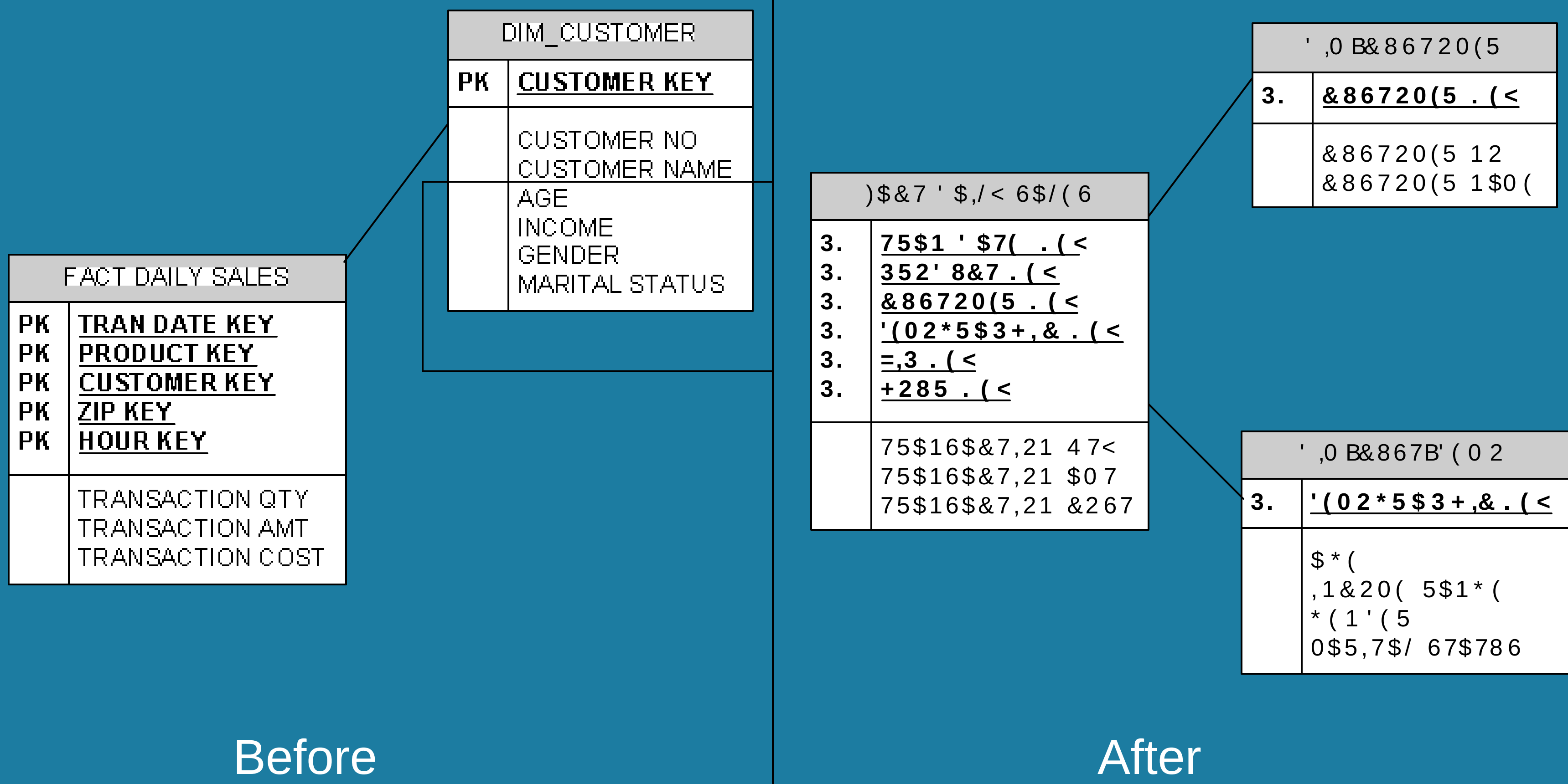
- If there are attributes of a dimension that change more frequently, these may trigger creation of new rows for every load
- To avoid dimension explosion, follow any of these approaches
  - Leave as-is
  - Create a mini-dimension by combining rapidly changing attributes with a FK to the fact table
  - Add rapidly changing attributes directly to the fact table

Mini Dimension

- Separate rapidly changing attributes of a dimension into its own dimension (Mini Dimension)
- If there are continuously changing attributes (e.g. income, age, total purchases etc.), use banded ranges
- Join mini dimension directly to the Fact table using FK relationship

  Generally customer demographic attributes fall into this category. Create a separate mini dimension with demographic attributes and join to the fact table.
- Populate the mini dimension as a one time load (if the possible values are low) or based on what is observed during the load time
- During the Fact table load, resolve to the right mini dim key based on the dimension attributes

Mini Dimension - Example



Presented by

MySQL Conference & Expo

O'REILLY

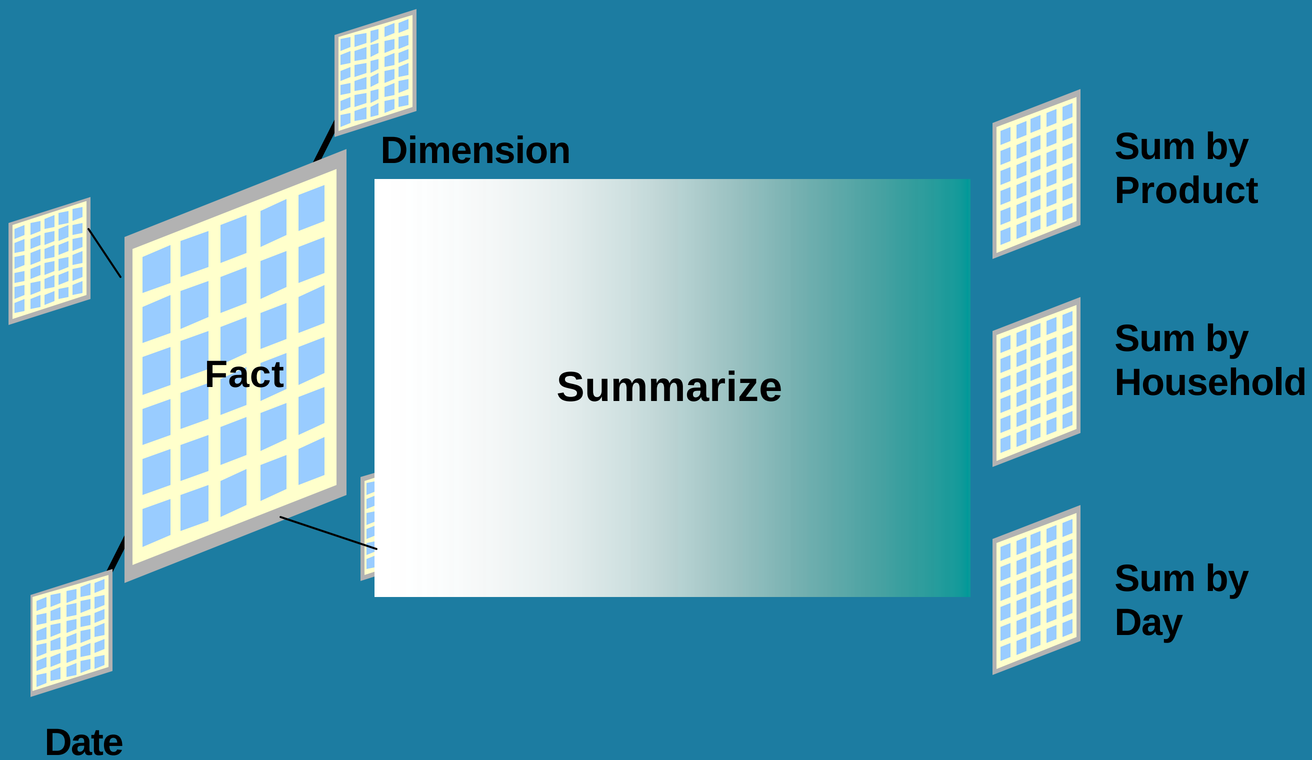
Multi Sourced Dimension tables

- In some instances, there will be a situation where one dimension table is populated from more than one source
- Identify rules for matching dimension records between source systems
- Determine survival rules for dimension attributes

Data Structures - Summary Tables

- Should have a mix of summary data
- Usually most accessed area of Data Warehouse (at least for Ad-hoc/Open-ended area).
- OLAP environments usually revolve around summary design
- Requires frequent monitoring to avoid unnecessary summary build-up

Data Structures - Summary Tables



- **Performance gain if queries go against pre-summarized tables**

Data Structures - Summary Tables

daily transactions
daily transactions
daily transactions
daily transactions
daily transactions

More Detail

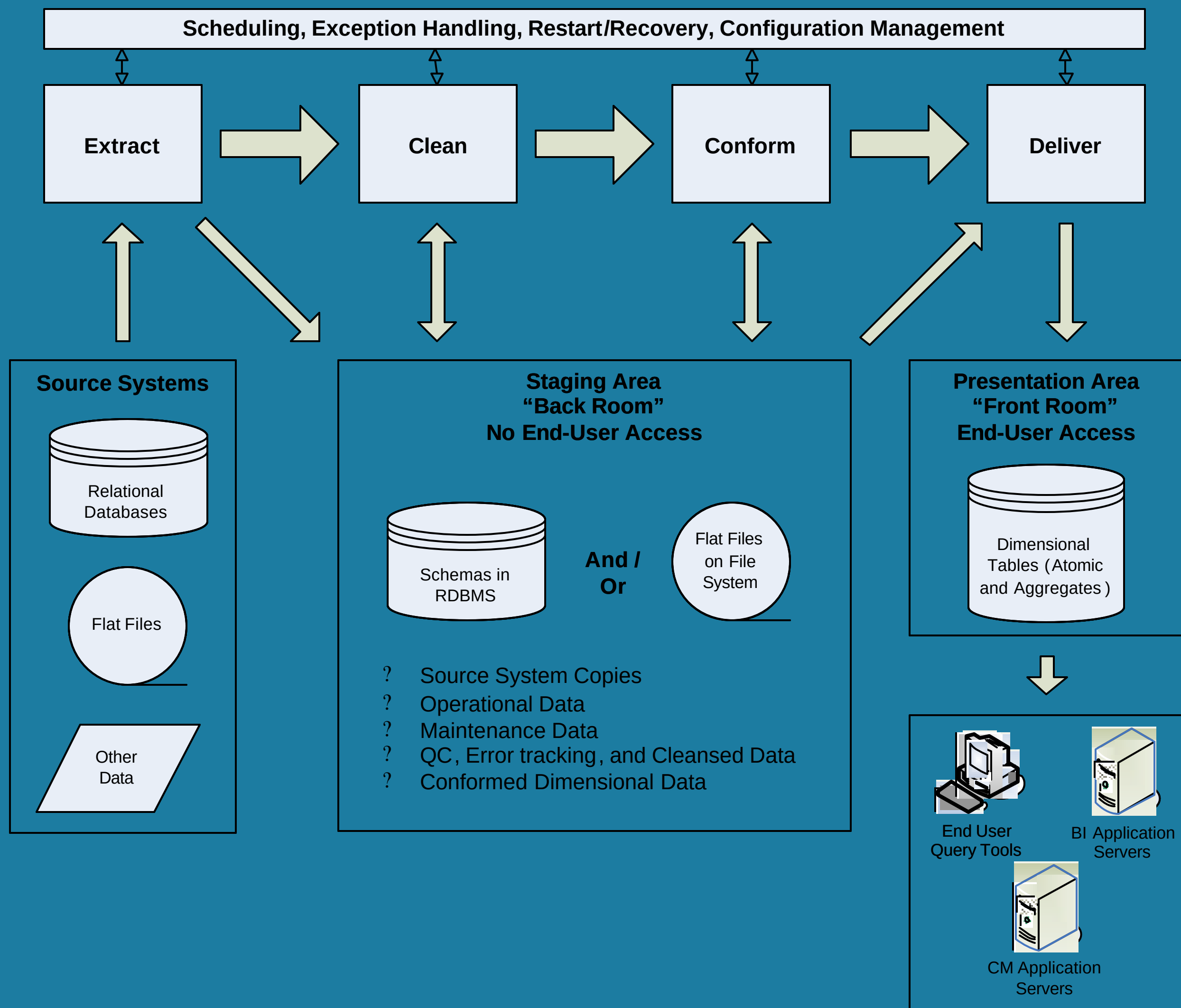
weekly transactions
weekly transactions
weekly transactions

SUMMARY ROLLUPS

Less Detail

monthly transactions

Typical ETL Architecture



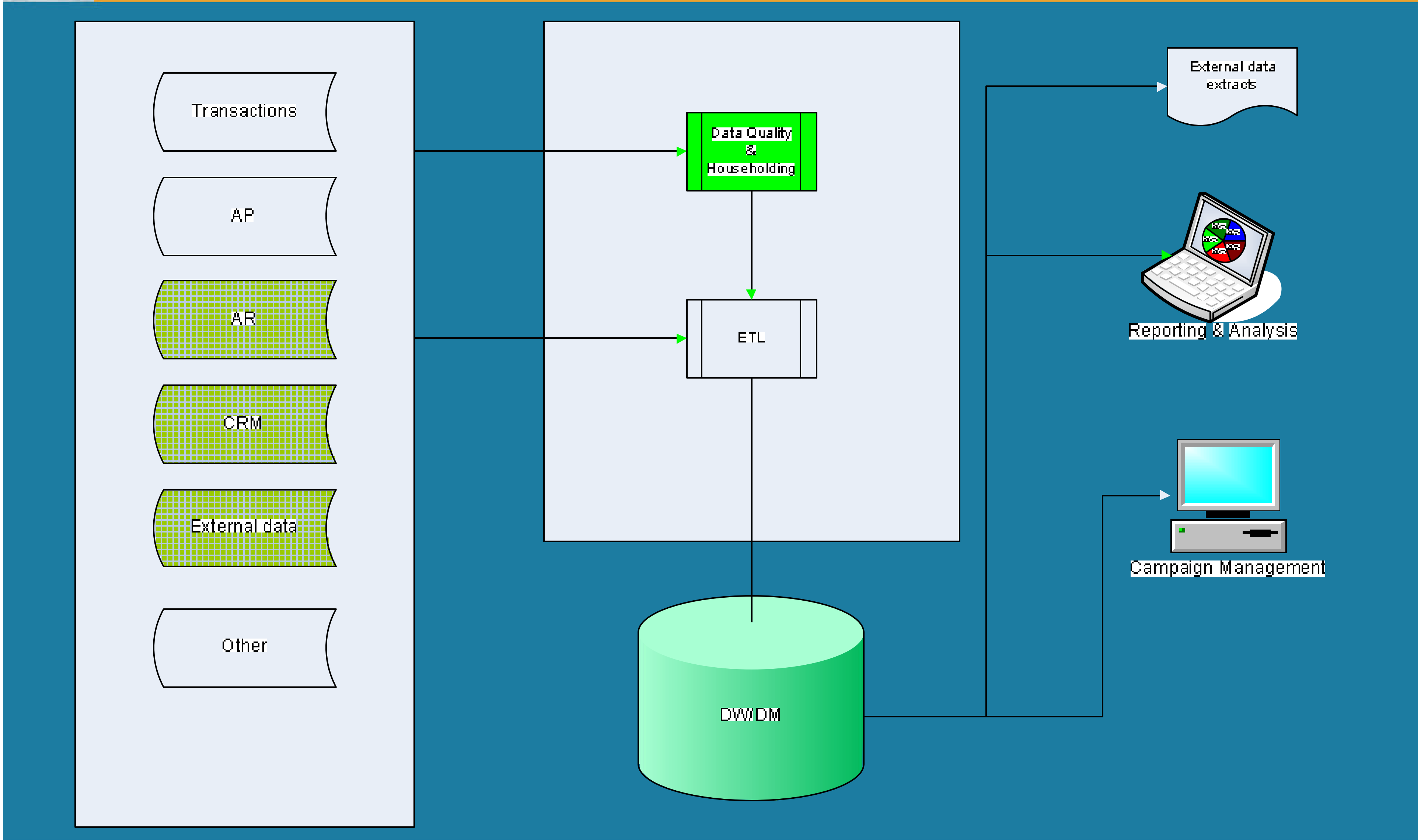
DW/DM Technology

Presented by

MySQL Conference & Expo

O'REILLY

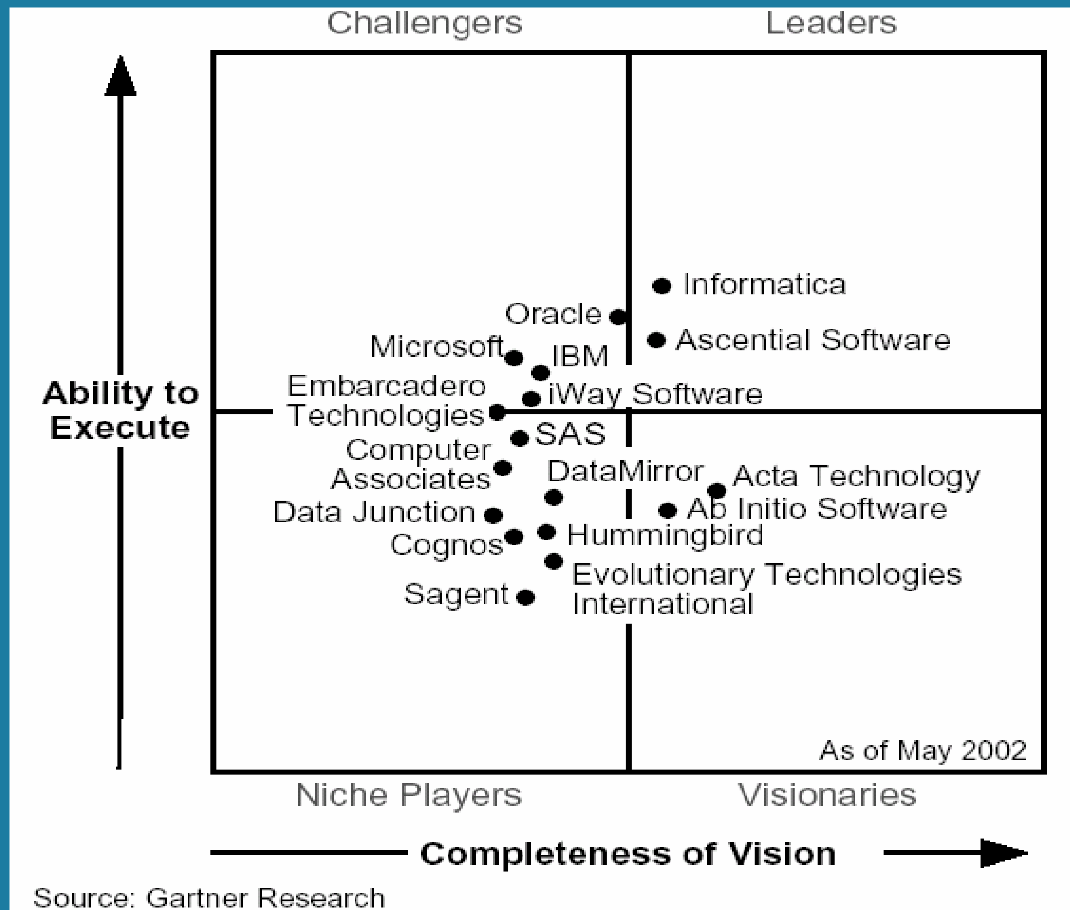
DW/DM Technology Architecture



ETL tool Requirements

- Connectivity to Heterogeneous databases and Mainframes
- Comprehensive data integration
- Extensibility with comprehensive development tools and API
- Performance & scalability
- Real-time connectivity
- Metadata-centric design

ETL Players (Commercial Vendors)



Presented by MySQL Conference & Expo O'REILLY

ETL tools (Open Source Options)

- Clover ETL
- Pentaho – KETL
- Talend

Integrated Business Intelligence

BI tools (Commercial Vendors)

- There are numerous vendors in this space
- Some of the major players
  - Business Objects
  - Hyperion
  - Oracle EEBI
  - Cognos
  - SAS

BI tools (OpenSource Options)

- Jasper and Pentaho are major players
- Jasper Reports, Jasper Decisions
- Pentaho BI Suite (Mondrian for OLAP, Dashboard, Weka for Data Mining, Reporting)
- JFreeChart for graphs
- Actuate BIRT (for reporting)

Required Database Functionality

- Data Partitioning
- Columnar Storage
- Data Compression
- Parallel Query
- Specialized data loaders
- Materialized Views (in the MySQL roadmap, but not currently available)
- Specialized analytical functions

Commercial Vendors like Oracle have evolved over the last 10 years to support Data Warehousing features.
Some built-in features and support of 3rd party storage engines is making MySQL a viable database platform for DW/DM

Partitioning

- ‘Not if to partition, but how to partition’
- Partitioning Benefits
  - Series of separate tables under a “view”.
  - Partition Schemes: RANGE, HASH, LIST, KEY, COMPOSITE
  - Alternative to managing one extremely large table.
  - Targets fact/detail tables most of the time.
  - Partition Pruning helps examine only required data
  - Easy to Manage
- MySQL 5.1 and above supports Partitioning

Data Loading

- One of the most forgotten and neglected issues
- Perhaps highest in critical path for daily operation of the warehouse
- Database should support fast / incremental loaders optimized for bulk data loads
- Some 3rd party MySQL storage engines have specialized loaders that provide screaming load performance

Data Compression

- Data Compression provides enormous storage savings
- Data compression may impact query performance if server has to uncompress to analyze the data
- There are storage engines like KickFire coming up that support data queries without uncompressing data

Columnar Storage

- Traditional databases write data to the disk as rows ; Columnar storage writes data to the disk as columns
- Columnar storage requires less I/O if a subset of columns are selected in the Query thus improving Query performance
- 3rd party storage engines such as KickFire and InfoBright support Columnar Storage

MySQL Storage Engines supporting DW/DM

- Internal
  - MyISAM
  - Archive
  - Memory
- 3rd Party
  - KickFire
  - BrightHouse
  - NitroEDB

Feature Comparison

Feature	MyISAM	NitroEDB	BrightHouse	KickFire
Data Partitioning	X			
Parallel Query				X
Columnar Storage			X	X
Data Compression			X	X
Specialized Data Loaders			X	X
H/W based Query acceleration				X

Why MySQL for DW/DM?

- Focus around native support for Data Warehousing features in MySQL roadmap
- 3rd party storage engines and specialized technologies are coming up to support DW (Ex. KickFire)
- Availability of Open Source ETL and Reporting tools
- Low Total Cost Of Ownership

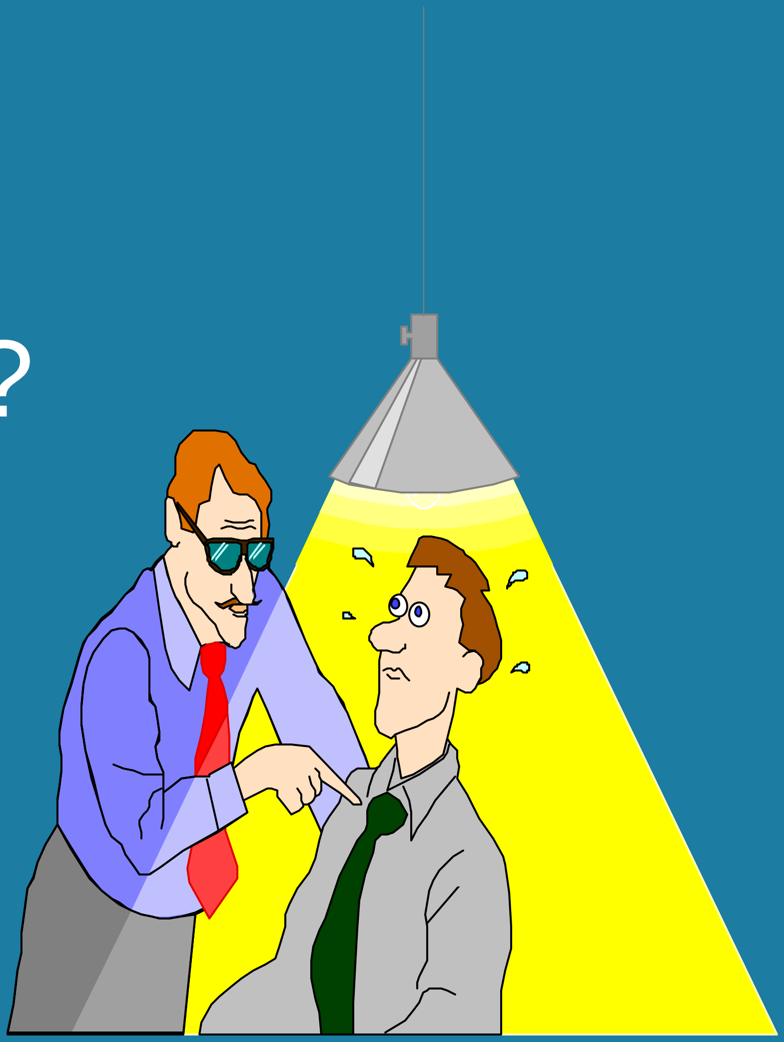
Things to Watch Out For

- Focus on capturing detailed business requirements early in the project
- Involve business users through out the life cycle to keep them engaged.
- Data Quality is Key. Spend time understanding the data anomalies to avoid surprises during development
- Select the right tools and technologies. Once selected, it will be difficult to change.
- Have a phased approach to realizing the DW/DM vision instead of the big bang approach

Reference Material

- The Data Warehouse ToolKit – Ralph Kimball
- Enterprise Data Warehousing with MySQL – MySQL AB
- MySQL Roadmap 2008 – 2009 – MySQL AB

Any questions?



MySQL Conference & Expo

Presented by,
MySQL AB® & O'Reilly Media, Inc.