



Datawarehousing for OLTP Data Modelers

Leslie M. Tierstein

newScale, Inc.

- Transactional systems vs Data Marts/Warehouses
 - OLTP vs OLAP – transactional vs. analytical processing
 - Technology and Methodology
 - What skills must be are common to both
 - What skills must be “unlearned”?
 - What skills are new?

● Topics:

– Database Design

- Logical Design
- Physical Design

– Updating and Reporting on Database Contents

– Methodology

- Project Team
- Development Life Cycle
- Tools: build or buy?

Database Design



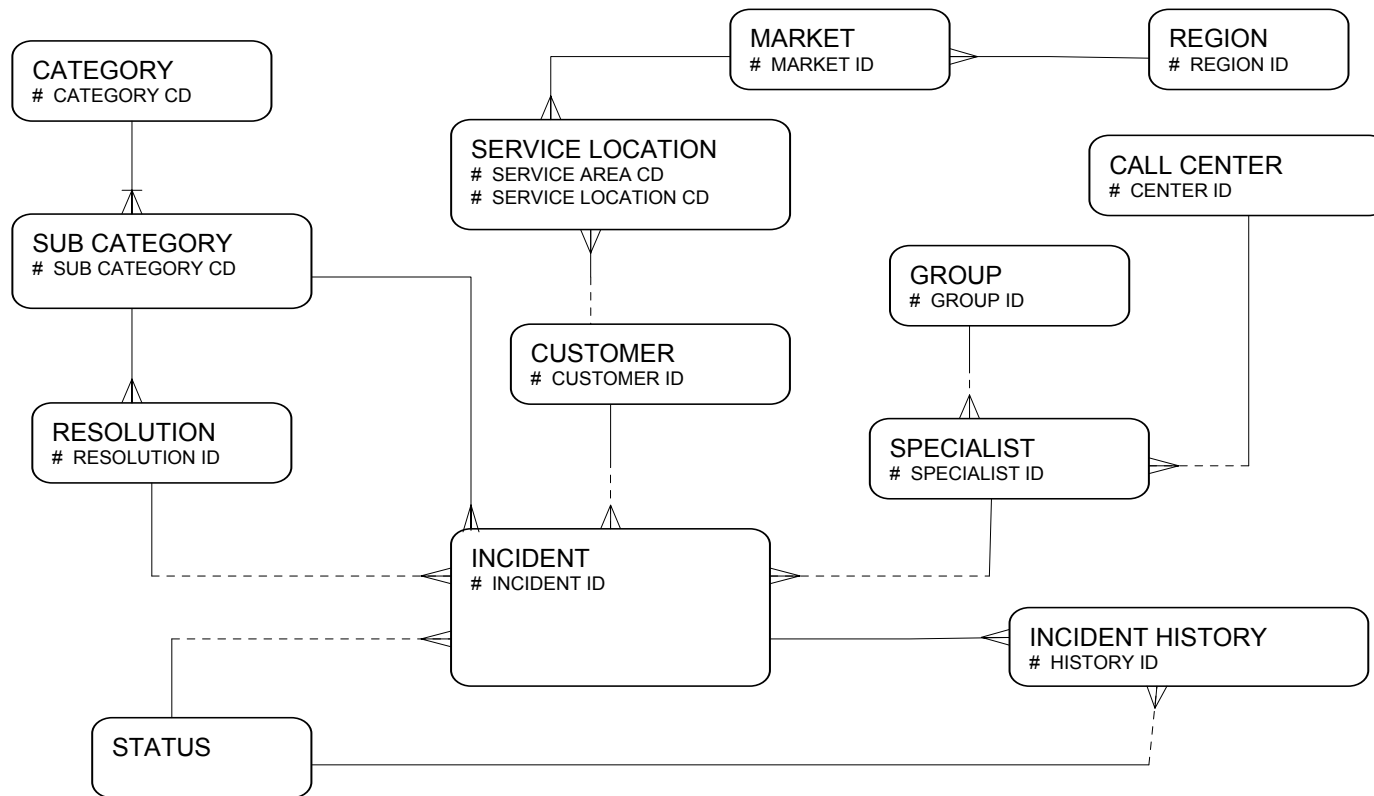
- Sample OLTP design
- Sample Data Mart design for the same data
- Differences in methodology

● Sample - Customer Care System

- Many accounts, each in a marketing hierarchy (region, market, service area)
- Each account may generate numerous trouble calls (incidents)
 - Each incident is assigned to a specialist at a call center
 - Each incident may take many calls to resolve
 - Each incident is categorized as to its type and resolution

Logical DB Design

● Customer Care - ERD



- Customer Care - ERD
 - Lots of one-to-many relationships (hierarchies; master-detail)
 - Lots of many-to-one relationships (descriptions; codes)
 - Normalized (3NF) design

- Requirements Analysis

- Determine the data required in the system and the relationships between entities
- Determine process/functional requirements with user sign-off - a formal Functional Requirements List

- Agile/Extreme Methods

- How much ahead-of-time analysis is appropriate?

● Normalization

- “A column in a table is a fact about the key, the whole key, and nothing but the key, so help me Codd.”
- Normalization eliminates “update anomalies”
- The trade-off is that many tables must be joined to retrieve all relevant information

● Requirements Analysis - OLAP

- How much analysis is required/desirable, given that the system's goal is "adhoc" inquiries and/or to support data mining?
- Tierstein Analysis: What are the top 10 questions you need to be able to answer?
- Data Mining: What are the "groupings" that you will be interested in?

- Performance/Functional Requirements
 - Data is static, so updates are not required
 - Retrieval speed is paramount
 - Capacity planning/scalability is critical
 - Database refresh must fit in maintenance window

● Star Schema

- Find the central “fact” that the user is interested in:
 - OLTP Hint: Follow the master-detail relationships down to the appropriate level of detail; that’s probably your fact
 - OLTP Hint: Think “transaction” -- sale, history, scheduling

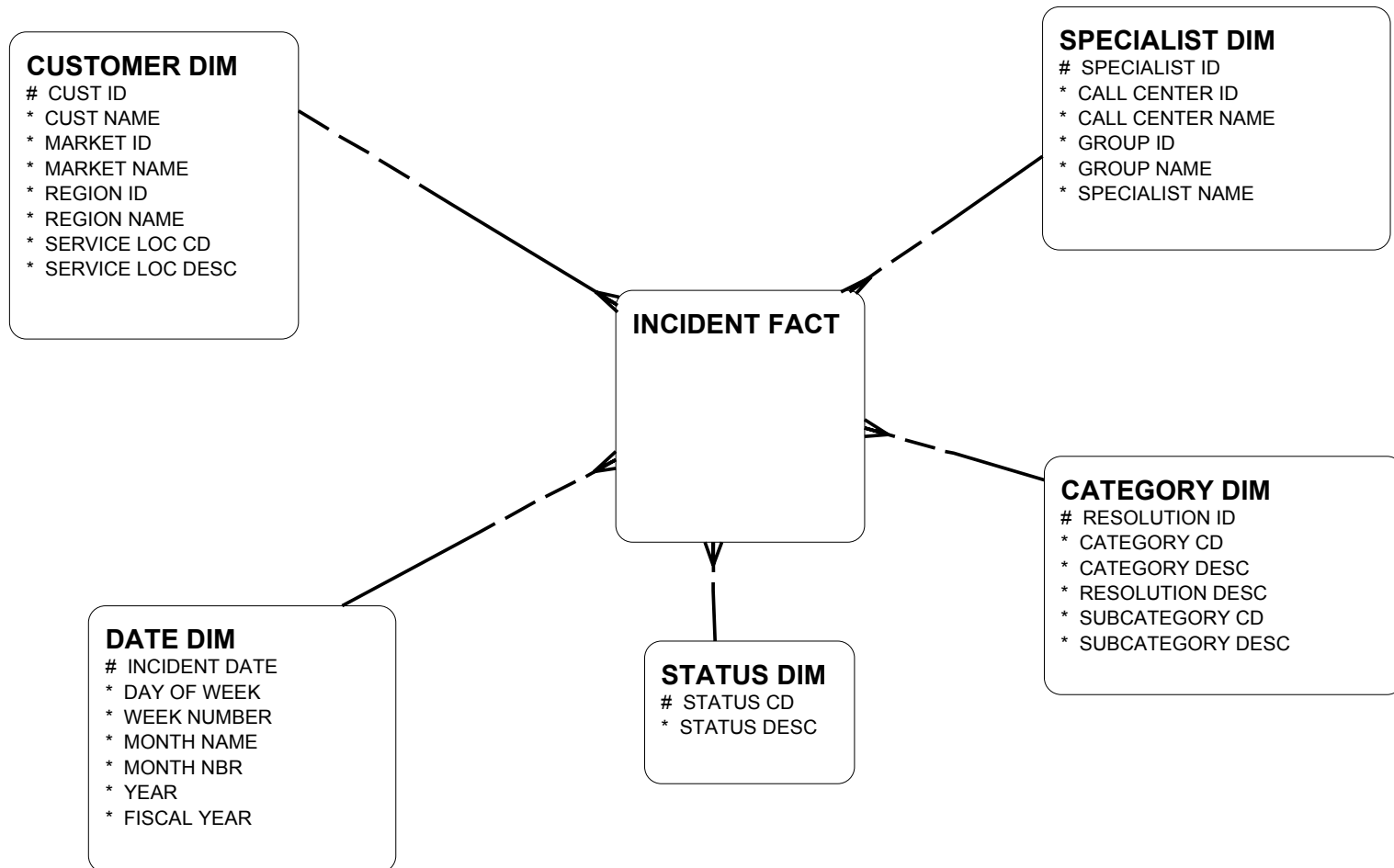
● Star Schema

- The descriptive codes describing the fact are “dimensions”
 - OLTP Hint: The date is almost always a dimension.
 - OLTP Hint: The OLTP reference (code) tables of the fact are also one dimension, with different levels of detail “denormalized” into one table
- What is a “small” dimension?

● Star Schema

- An ERD ends up with a central fact, and dimensions radiating out from it - a star
- A data mart (or data warehouse) can consist of one or more stars
- The stars can (should) share dimensions

Data Mart Star Schema



● Snowflake

- Sometimes, a dimension in a star schema will, itself, have dimensions
- This results in a snowflake configuration
- Snowflakes may have performance issues
- Some BI tools perform better with different DB designs: Normalized, star, snowflake

● Fact and Dimension Tables

– Attributes: Alphanumeric descriptive data

- Derived attribute: age range, salary range, call length

– Metrics: Numeric data about the fact

- “Factless fact” – fact table with no metrics
- OLTP hint: An intersection table with no additional attributes

– Sparsity vs. denseness of data

Physical Design Issues



- Natural vs. Artificial Primary Keys
- Denormalization
- Summarization
- Server-Side Referential Integrity Constraints
- Database Partitioning
- Application Tuning, including Indexes

Assumption: Relational implementation, not a multi-dimensional cube

Natural vs. Artificial Keys



- Natural Primary Key - Value is intelligible to the user, and occurs naturally in the application
- Artificial Primary Key - Value is artificially derived, eg, from an Oracle sequence

- Can be a religious argument
- Use artificial keys if:
 - The natural key value is subject to change
 - The key structure is too complex (> 5 columns, 64 characters)
 - Part of the natural key may be null
 - To reduce code lookups
 - Your project standards say to

- Always use artificial keys:
 - The natural key value might not be unique (for example, when collecting data from multiple systems)
 - Indicated for use with bitmap indexes
 - Supports “slowly changing dimensions”, when the natural key value is the same but its semantics change

- What if a dimension changes:
 - Example: Market A used to be in the Western region, now it's in a new, Mountain region
- How can we compare summaries by region from before and after the change?
- Approaches:
 - 1: Lose the history and realign (or not) data
 - 2: Add new dimension records for new data
- Several different implementations

- OLTP Referential Integrity
 - Server-side declarative constraints
 - Server-side procedural code
 - Client-side GUI controls

● Data Mart Referential Integrity

- Are server-side RI constraints needed?
 - All updates are done via one load program
 - Load program should reject dirty data -- and report on it
- RI constraints should be disabled when loading data
 - Ability to use direct path load
- RI constraints may be required by BI tools

- Methodology whereby a normalized design is “broken”, typically to enhance performance
 - Store summary of detailed data in the master table (to decrease accesses)
 - Store derivable data in the table (to enable indexed searches)

- Determine the level of detail of data to be stored
- Redundantly store derivable (summary) data, typically to enhance performance
- Use materialized views if
 - You can predict your most frequent queries
 - You have sufficient disk space (for views and view logs)
 - You have time to refresh the views

- Summarization/Aggregation - Approaches
 - Store individual “transactions”
 - Summarize transactions on load
 - Summarize transactions after a period of time

● Summarization

- Maintain summary table(s) (materialized views) which summarize the facts by the most frequently combined dimensions
 - Example: Incident by Resolution by Region
- Summary tables must be refreshed whenever the database is refreshed
 - Refresh “on demand” as part of the load process

- Dividing tables and indexes into partitions
 - Read performance – “partition pruning”
 - Write performance – ability to drop a partition, rather than delete rows
 - Admin performance – ability to assign different partitions to different tablespaces (for backup)
- Must be designed into the ETL process

- These disciplines differ greatly between OLTP and Data Mart database
- Examples:
 - Estimating table size (extents; volatility)
 - Indexes (bit maps vs. b-tree searches)
 - In OLAP, a Full Table Scan (FTS) may be good

Refreshing Database Contents



- OLTP

- Convert data from legacy system(s)
- One-time task

- Data Mart

- Initially load the data mart from source system(s)
- Refresh database contents at regular intervals

OLTP Data Conversion



- Methodology and Technology
 - Too often, “seat of the pants”
 - Tools are expensive for one-time use
 - Legacy system experts may be hard to find
- “One-time” use
 - But may have to reload data
 - Phased cutovers

- Methodology and Technology

- E(T)TL tool is required:

- Extract source data
 - (Transport data to new platform)
 - Transform data to new format
 - Load data into new database

- Tools

- Oracle Warehouse Builder
 - Informatica
 - Tools with domain-specific “adapters”

- ETL tool

- Maintain metadata about source system(s) - still in use and being maintained
- Maintain metadata about data mart - should be user accessible
- Maintain history of refresh cycles

- ETL Tool/Code

- Periodically add new data to the data mart

- Modes of operation

- Batch/File-based: Must be run in the “maintenance window” for the source and target systems
 - Near real-time: Message queues

- Operational Data Store (ODS)
 - Normalized database which acts as the feeder system to the data mart
 - Extract data from source system(s) into ODS
 - Load from ODS using refresh routines

● Design Issues

- Change Data Propagation - How do you know which source records are new and need to be loaded?
- Are records ever purged from the data mart?
Summarized?
- Are records ever updated in the data mart?

- Logical Design

- Replace 3NF databases with stars and snowflakes
- A normalized database may be used for an ODS
- “Requirements” may not be as formal

● Physical Design

- Know how to denormalize and summarize (ie, “enhance” the underlying model for performance)
- Pay more attention to tuning (the data mart is born large)

● Data Mart Refresh

- Formalized methodology and technology required
- Metadata is an issue
- Performance (load time)
 - Change data propagation
 - Materialized views and view logs
 - Partitioning
 - Parallel object creation
 - Direct path loads and inserts

● Reporting

- BI tool required for end-user adhoc report creation
- Data mining
- Performance (reporting)
 - Materialized views for aggregates/summaries
 - Partitions
 - Bitmap indexes

About the Author



- Leslie Tierstein is principal technical architect for newScale, Inc, a Silicon Valley company which specializes in ITIL (IT Infrastructure Library) implementations
- She can be reached at:
ltierstein@earthlink.net