

Controversial Issues in Data Modeling

Presented by
Tom Haughey, President
InfoModel LLC
868 Woodfield Road
Franklin Lakes, NJ 07417
201 755 3350(c)
201 337 9094
tom.haughey@InfoModelUSA.com

Debtech Meta-Data and Data Modeling Summit
Feb 28, 2005.
Sheraton Studio City, Orlando, FL



Agenda

- Conceptual-Logical-Physical
- Are Foreign Keys Alien?
- Decoding Codes
- The Domain of Domains
- There Are (Almost) No Natural Keys
- Is ER Dead? And Who Done It?
- There's No Such Thing as Dimensional Modeling!
- No Time for Time
- Data Modeling for Packages?
- There's No Money in Data Modeling!!!

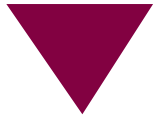


▼ Conceptual-Logical-Physical

- These terms have very different meanings in different approaches

Development Level	Data Model Level Description	Information Engineering (as in this training)	General US Data Architecture Opinion	Merise (Silverrun, Power Designer)
Planning	A rough-cut model, sparsely attributed	High level data model	Conceptual	N/A
Analysis (Requirements)	A detailed representation, which is independent of implementation, technology and organizational structure, of information requirements and business rules. Should be fully detailed and normalized.	Logical	Logical	Conceptual
Design	A specification of the proposed system that is optimized, buildable, maintainable, user-friendly and robust, but still independent of technology. Can be denormalized.	Design	(Physical) Design	Logical
Implementation (Construction)	A representation of specific implementation technologies, structures, techniques, and constraints that will be implemented and how they will be implemented.	Physical	Physical (Implementation)	Physical





Are Foreign Keys Alien to the LDM?

- **Pros:**

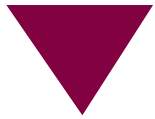
- (1) Foreign keys document the granularity and identify of the entity.
- (2) Foreign keys document the basis for the relationship between two entities, for example, Customer relates to Order because Order has the Customer ID.
- (3) It is easier to normalize data when the foreign keys are evident.
- (4) We like to validate a model (even on paper) before implementing it, which we often do through scenarios and use cases. In providing sample data to illustrate a model, it is meaningless to do so without providing values for the foreign keys.
- (5) Use of foreign keys is familiar to many people, and is in use by most modelers.

- **Cons:**

- (1) The usual argument against foreign keys is that they are a relational concept and have no place in a logical model, which is independent of technology.
- (2) Some people, not accustomed to seeing foreign key, could be confused by them.

- Always remember the principle of **WYLIWYUT!**

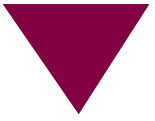




Breaking the Code on Codes

- Code data includes data that is commonly called codes, status and role entities. Part of a larger set of data called reference data.
 - The question is: should such codes and descriptions be an entity or an attribute in the logical model? Should the attribute be the code or the description?
- Reference data is commonly defined as “any kind of data that is used solely to categorize other data found in a database, or solely for relating data in a database to information beyond the boundaries of the enterprise”, according to Malcolm Chisholm.
- There are four kinds of reference data, namely,
 - **Things external to the enterprise**, such as country codes.
 - **Type codes, status codes, and role codes**, such as Customer Type, Order Type.
 - **Classification schemes**, such as market segment classifications and industry classifications.
 - **Constant values**, such as tax rates.

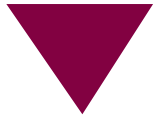




Types of Code Tables

- There are four types of “code” entities or tables:
- **Type 1: A pure and simple code.** No description is used or is needed. No entity is necessary on the logical model. Examples: Gender (M/F); long established Fidelity codes. Also where the code has only two (or very few) values or is an indicator.
- **Type 2: A code and a description**, where these are the translation of a code into a description or vice versa. Example: Account Status; ‘active’ status is one out of 52 statuses.
- **Type 3: Where these are a maintained reference table**, such as Product Hierarchy consisting of Product Type, Product Sub Group, Product Group. Dimension tables in data warehousing are a conspicuous example of this.
- **Type 4: Where the data consists of a code, description and additional attributes.** Example: Product Code, Product Description and Product Price.





Recommended General Practice

- In logical models:
 - Code tables should **not** appear in a “conceptual” (high level) model
 - Type 1 codes should **not** appear in the logical model.
 - Type 2 will appear in the logical model **at the discretion** of the modeler.
 - Type 3 codes **should** appear in a logical model. They may appear in separate tables or in a common table.
 - Type 4 codes **must in all cases appear** in the logical model.
- In physical implementation:
 - If a code is updatable it should not be implemented as a check constraint.
 - If the codes Type 1 and 2 have a small number of instances and are not updatable, they can be implemented as check constraints. Type 2 codes with many instances or any that are updatable should be implemented as a separate table.
 - Type 3 and 4 should be implemented as a separate table or included in a generic code table.





What Is the Domain of Domains?

- Domain refers to the valid set of values an attribute can have and its data characteristics, such as data type and length
 - Domains help you characterize the types of data in a model.
 - Domains make it easier to standardize data characteristics across entities.
- Two components to its definition, namely,
 - Conceptual data type: data type, length, etc.
 - Set of values: the valid values an attribute can have
- CASE tool concept of domain:
 - In some tools, mostly enforces domain as conceptual data type
 - Implies a named set of attribute characteristics, such as data type, length, etc.
 - Greatly simplifies definition of attributes and columns
 - In robust CASE tools, domain is a repository object with which you can associate the following information:
 - Data type, length, and precision
 - Check parameters
 - Business rules





Domain Types

- Several types of domains could be recognized.
- The following should be defined for specific attributes:
 - **Enumerated** domain in which codes have a specific description.
 - For example, 1 = Interstate road, 2=U.S. highway, 3=state highway, 4=county road
 - **Range** domain. A from and to-value.
 - For example, a valid range of elevation values is from 1000 to 7000 meters.
 - **Codeset** domain. This refers to a standard list or a discrete set of values which contains the members of an established set.
 - For example, state codes.
- The following can be a generic domain:
 - Serial or **unrepresentable** domain
 - For instance, customer names or street addresses, which are each unique and unrepeating. Any valid text can be used.



▼ There Are (Almost) No Natural Keys

- A natural key is an attribute (or set of attributes) that exist in the real world and that have all the characteristics of any key
 - Mandatory
 - Unique
 - Stable
 - Minimum
- The vast majority of natural keys fail at least one of these characteristics, e.g.,
 - Vendor Name: not unique, not stable
 - Employee Name: *ibid.*
 - Product Name: not stable





Real World Natural Keys

- There are very few real world natural keys, e.g.,
 - State names
 - Animal names
 - Stage names (actors, films)
 - Drug names
 - The periodic table
 - And, of course, date
- Where purely natural keys do not exist, artificial keys are often used. To be considered natural, they have to assume a business meaning (e.g., Customer Number: “What’s your customer number?”)





Sources of Keys

- Keys can come from four sources:
 - **Business supplied**, such as Product Codes, which are useful business attributes.
 - **System generated or artificial keys**, such as Customer ID. Though artificial, a Customer ID becomes a useful business attribute.
 - **Externally supplied**, such a Social Security Number and Federal Tax ID Number, which are business attributes provide by external government agencies (in this case).
 - **Surrogate keys**, which are meaningless identifiers, generated by an algorithm.



▼ Natural vs. Surrogate Keys

- A Natural Key consists of one or more business attributes that serve as the primary identifier of the entity. It has three characteristics:
 - Unique
 - Stable
 - Minimal
- A Surrogate Key is a system generated key that replaces the natural key. A Surrogate Key is a system generated key that replaces the natural key. Not all system generated keys or artificial keys are surrogate keys. It has three additional characteristics:
 - Immutable
 - Simple
 - Non-intelligent
- Surrogate keys are not unique to the dimensional model





Surrogate Keys vs. Natural Keys

■ Pros of Surrogate Keys

- ➔ Reduced space by replacing a large combined primary key, especially when used as a foreign key (large natural keys)
- ➔ Isolation of the data from operational changes (such as reused order numbers)
- ➔ Ability to have multiple instances against a given entity (a changing dimension)
- ➔ Speed of access when the optimizer uses a simple, numeric index (the natural key could have a large composite index)

■ Cons of Surrogate Keys

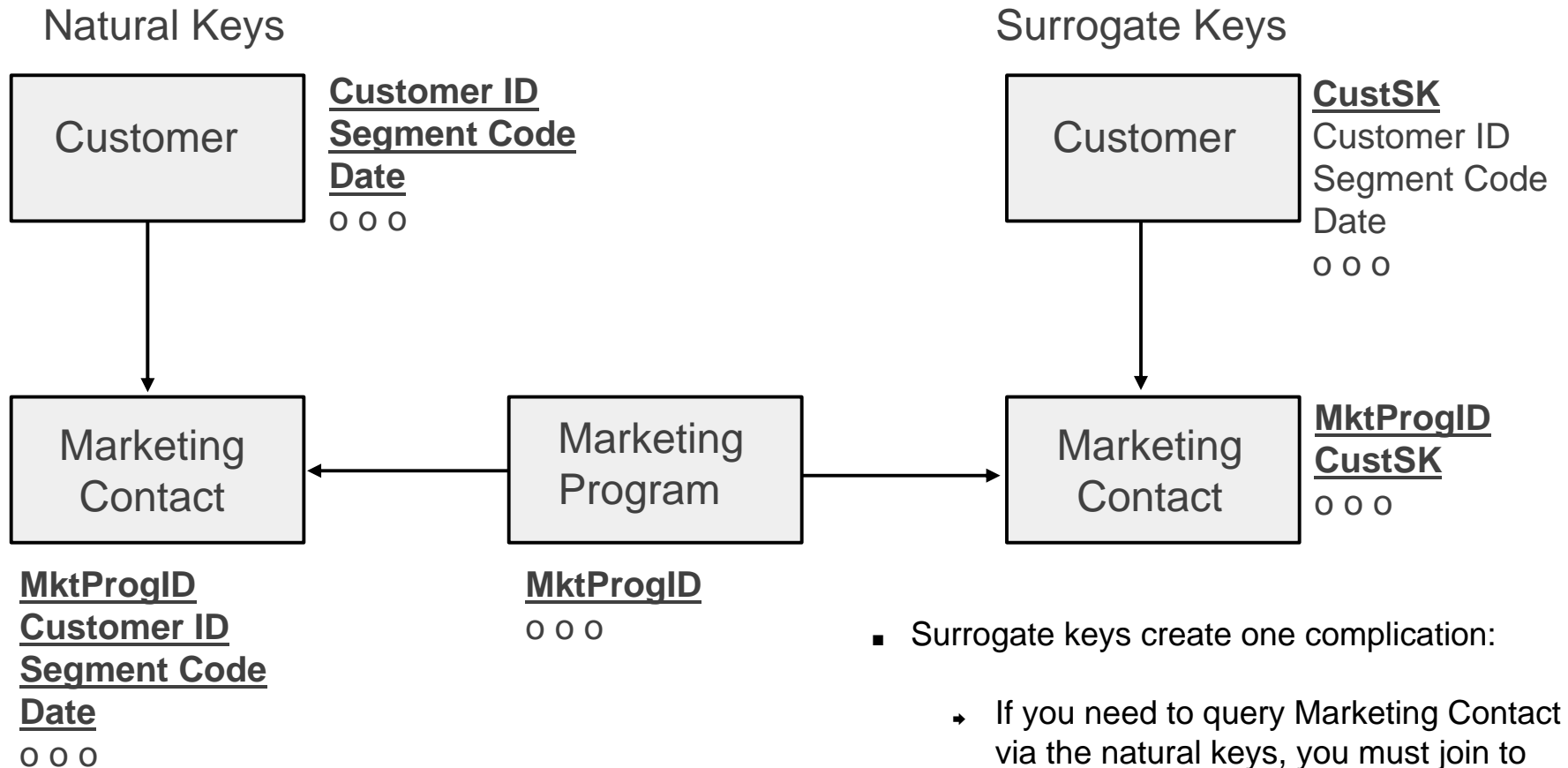
- ➔ (Possible) Extra storage requirement due to an extra column (if both natural and surrogate are retained)
- ➔ Integrity having to be enforced some other way
- ➔ Still need access to the data using natural keys
- ➔ May have to supplement surrogate keys with a mapping table correlating surrogate key and its equivalent natural key.





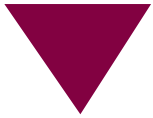
Example of Surrogate Keys

- Surrogate key adds one attribute at the one-end
- It saves space at the many-end



- Surrogate keys create one complication:
 - If you need to query Marketing Contact via the natural keys, you must join to Customer.
 - With natural keys, the foreign key is present in Marketing Contact.





ER Is Dead – But Who Done It?

- “Reports of my death are greatly exaggerated!”
 - » Mark Twain
- Challengers to ER:
 - Object Role Modeling: too cluttered; good for business rules
 - UML (Unified Modeling Language): too weak in modeling principles
 - Dimensional Modeling: oversimplified, stylized, DW only
- While each of these has its strengths, none can hold a candle to ER for strength and rigor of its data structures
- ER is like Aspirin
 - Aspirin, once disparaged, is now considered a miracle drug
 - “Our drug is compatible with Aspirin therapy!!!”





There's No Such Thing as Dimensional Modeling

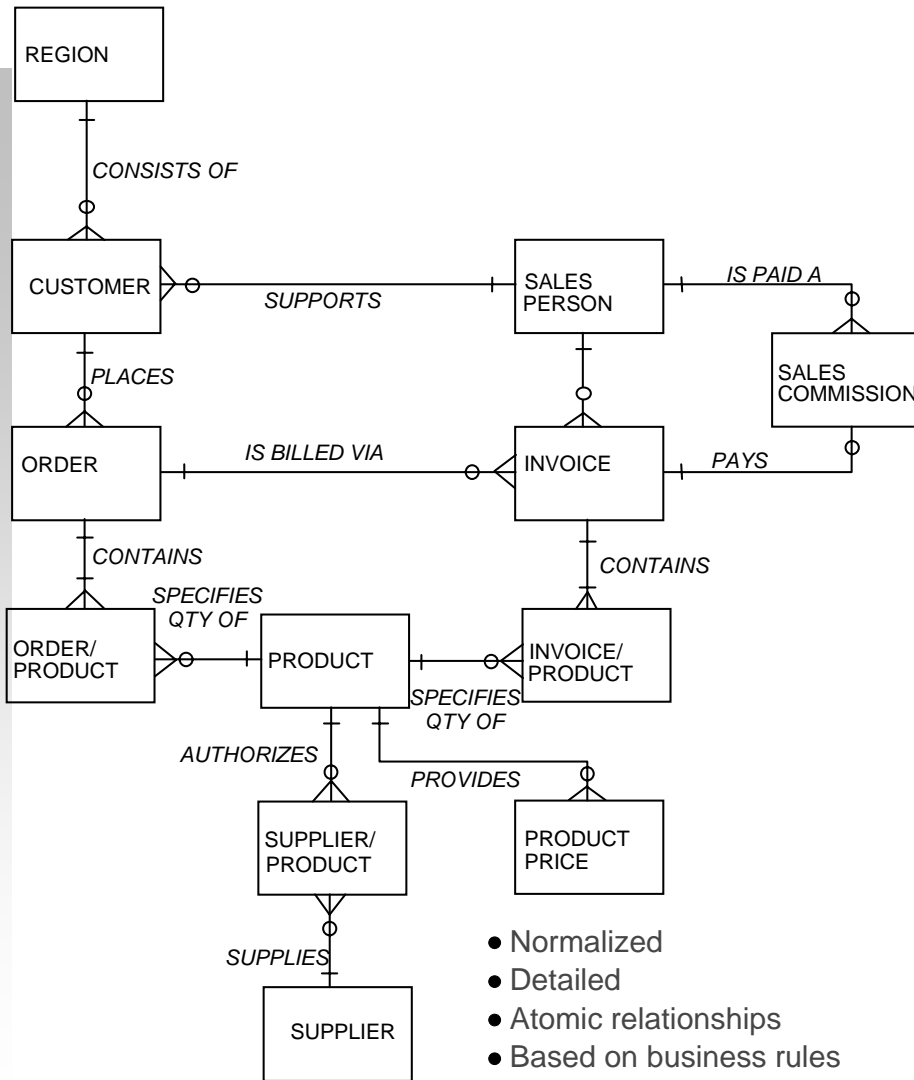
- This is not as demented as it sounds.
- Why? Because DM is merely the application of data modeling to analytical data.
- Take analytical data and model it; it comes out that way
- Plus aggregates are naturally dimensional
 - “Give me Customer by Product by Week”
- The only real difference in the creation of the dimensional model is reduction of the structure to a star (which is a physical optimization)





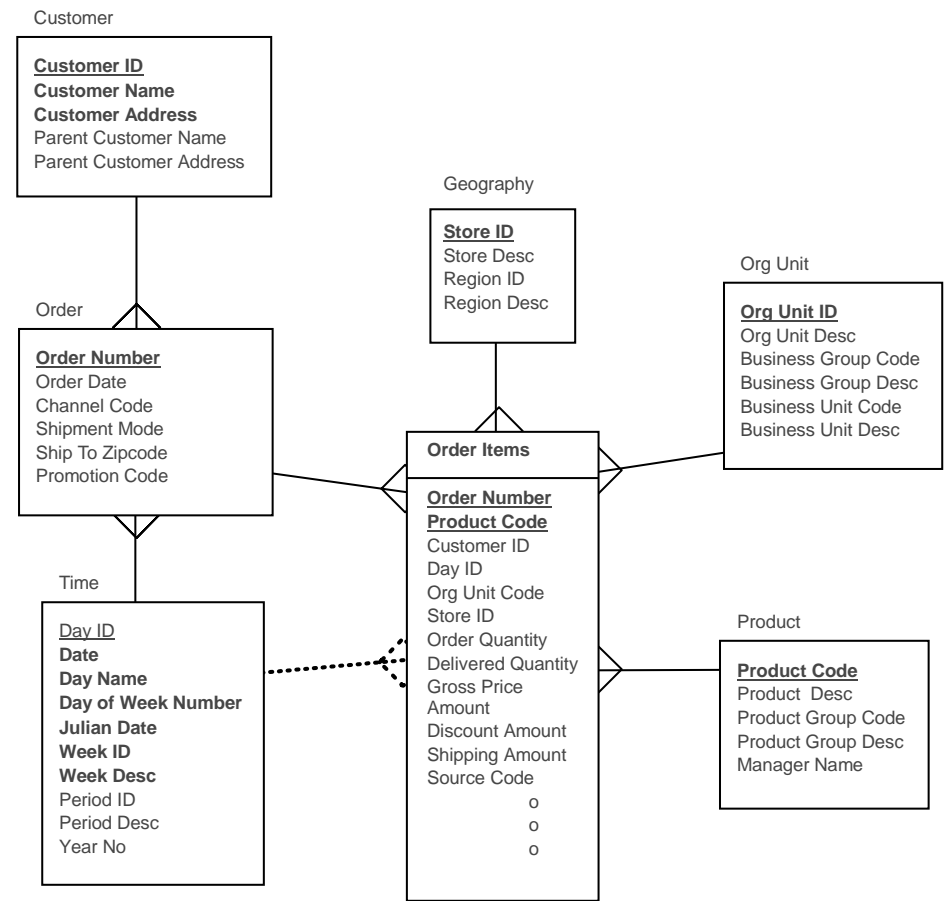
Model Comparison

Operational



- Normalized
- Detailed
- Atomic relationships
- Based on business rules

Analytical



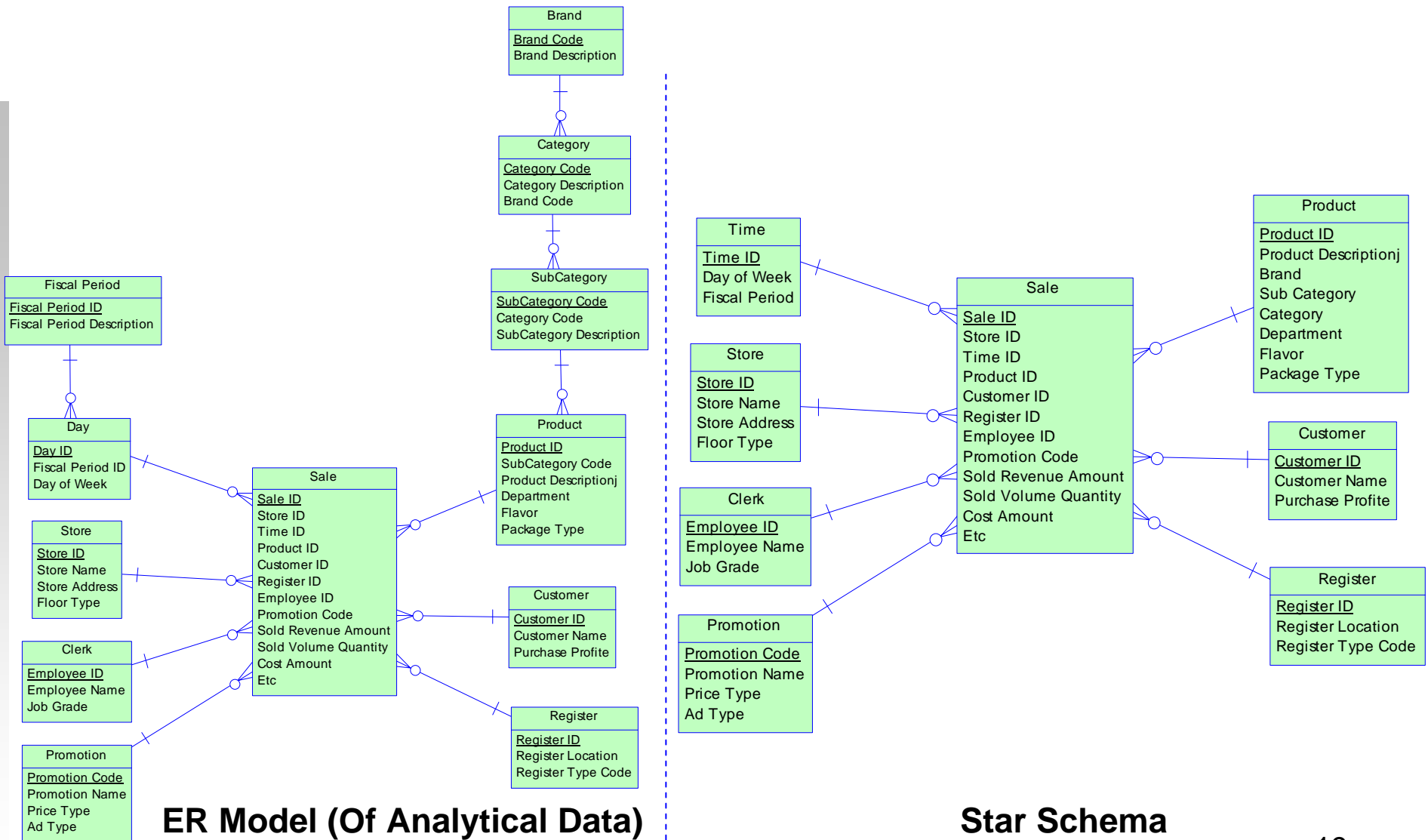
..... Optional, only as possible





Models of Analytical Data

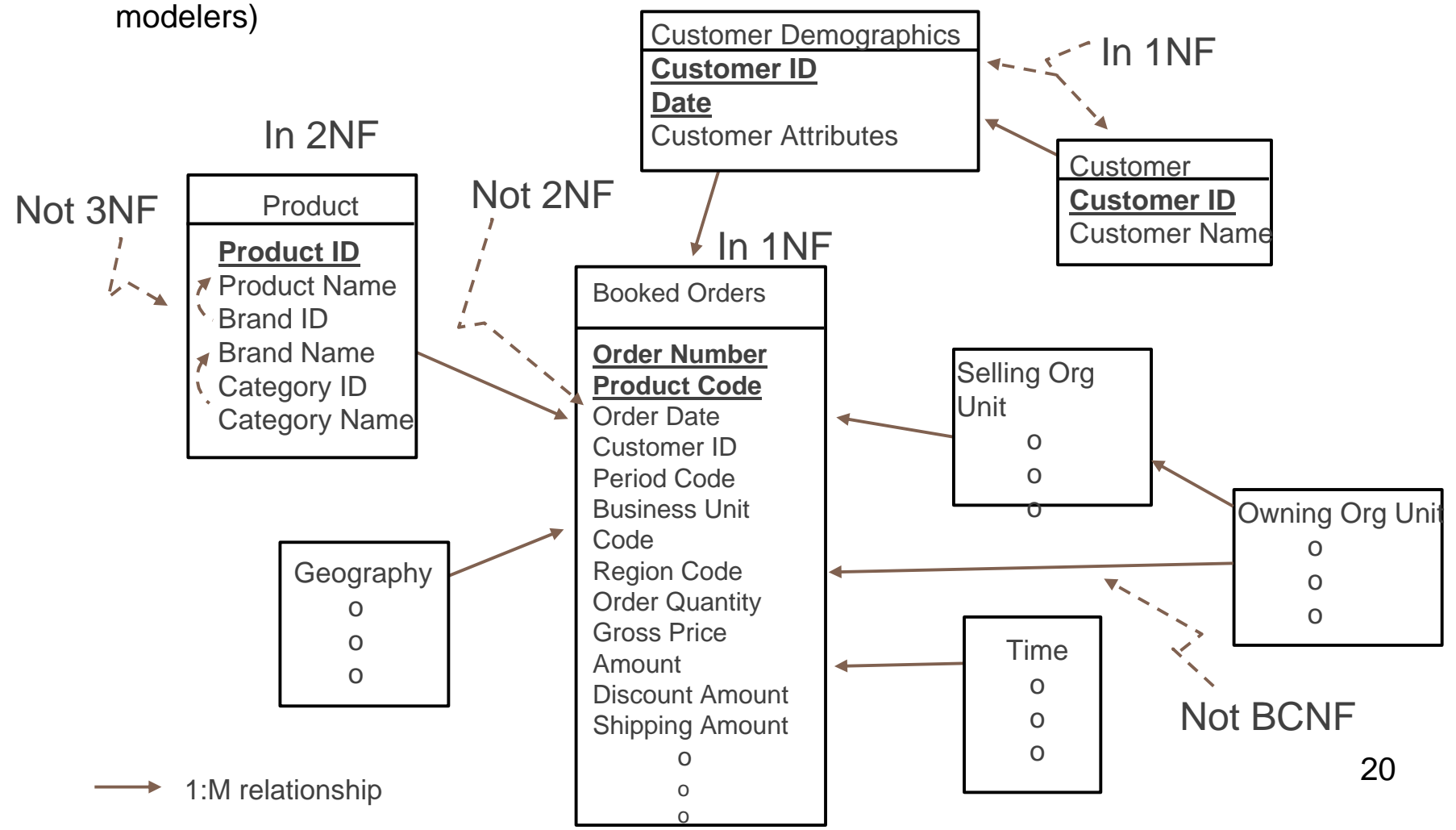
- The main difference between a 3NF model of the same analytical data and a star schema is that the 3NF model will be snowflaked.

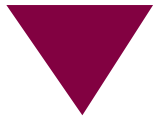




Denormalization in the Dimensional Model

- A typical dimensional model uses denormalization in the following way:
 - Violates 3NF in dimensions by collapsing dimension hierarchies
 - Violates 2NF in facts by collapsing common data into the transaction
 - But, should preserve 1NF in changing dimensions (not always done by dimensional modelers)





No Time for Time in an LDM

- For a long time, we were told not to include time in the logical data model
- The data model captured what the data looked like at a given point in time
- However, this thinking was oversimplified and is outdated
- Whatever concept of time is part of the business requirements (not archiving), must be included in the model without exception

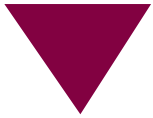




History vs. Audit

- History is a business requirement
 - History
 - Time series
- Auditing and archiving are system requirements
 - Legal requirement
 - Separate, mirror tables
 - Querying and business users not permitted to touch audit data
 - System requirement, added in design





General Advice on Time

- Define temporal requirements early on
- Embed time into design at an early stage
- Better to have more history than not enough
 - Too much, you have a performance problem
 - Too little, you have a costly redesign and even re-platform problem

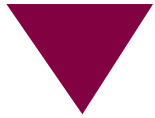




Are Use Cases Useless for Data Modeling?

- First, use cases are not object-oriented (I'm just saying)
 - They are simply process modeling
 - They are a simplified form of process modeling and an enhanced form of process definition
- Second, there are three forms of use cases:
 - Formal: all the bells and whistles
 - Informal: 1-5 narrative paragraphs
 - Basic: only the major parts of the use case
- Scenarios are a main component of use cases:
 - Concrete scenarios: a particular path with sample values
 - Abstract scenarios: a particular path without sample values
- You can't do data modeling only by doing data modeling
 - Use cases provide the context and
 - The means of manual validation and data usage analysis





The Logical Modeler Should Do the Physical

- The logical modeler should do at least the first-cut physical
- The logical modeler best knows the business requirements
- The transition to the first-cut physical is generally quick and simple





First-Cut Physical

- **1. Ensure physical columns characteristics**
 - Use domains or assign data type, length, nullability, optionality, etc.
- **2. Ensure proper physical database object names are used**
 - Follow standard (enforced by CASE tool) for physical names
- **3. Add necessary constraints and business rules**
 - For example, an Order Item must belong to Order and Order must have Order Item
 - The first is enforced by RI, the second requires a trigger or stored procedure
- **4. Resolve implementation of subtypes**
 - Decide how the entire hierarchy will be implemented
- **5. Perform any obvious denormalization or other safe compromises**
 - Safe compromises do not compromise the integrity and non-redundancy.
 - For example, splitting or partitioning a table based on usage.
- **6. Adjust keys**
 - Ensure keys truly provide uniqueness and stability (consider time). Consider surrogate keys.
- **7. Add indexes on primary keys, major foreign keys only**
 - Introduce indexing sparingly at first to determine raw capabilities of the design
- **8. Add any production objects**
 - Examples: transient tables, processing tables/columns (e.g. batch control tables).

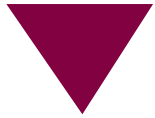




Data Modelers Should Write SQL

- Every data modeler should be made to write SQL for one year
- Why? So that:
 - They don't model things that cannot be built
 - They understand better what it takes to use the data
 - They can better create the first-cut physical

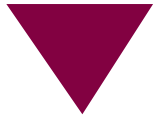




Don't Data Model for Packages(?)

- Data modeling plays three roles with regard to packages:
 - Package selection: you don't need to do a data model – unless of course you are concerned about picking the right package
 - Package customization: no packages come matching your needs 100%. They all need tailoring. Tailoring the data is a major part of that.
 - Data mapping: transforming existing data stores to the package.
 - The data model for most packages is (deceptively) complex. Good data modeling skills are needed to understand it.





There's No Money in Data Modeling!!!

- You're trying to justify data modeling
- How can you do that?
- Data modeling provides monetary value in three main areas:
 - **Reduced maintenance:** say you have a maintenance budget of \$150MM. What would a reduction of 1% save you? 5%? 10%?
 - **Reduced interfaces:** what % of jobs in your organization just **move** data? How much of the project budget is directed toward interfaces?
 - **Reduced development:** how much do you save by reusing databases? Models? Documentation?





Summary

- Conceptual-Logical-Physical
- Are Foreign Keys Alien?
- Breaking the Code on Codes
- What's the Domain of Domains?
- There are (Almost) No Natural Keys
- ER Is Dead -- But Who Done It?
- There's No Such Thing as Dimensional Modeling!
- Time for Time
- Data Modeling for Packages?
- There's No Money in Data Modeling!!!

