



Data Modeling

Sajid Abbasi

- What is Data Modeling?
 - ◆ Purpose
 - ◆ Types

- The Data Modeling Process
 - ◆ Different Approaches
 - ◆ Conceptual Data Models
 - ◆ Logical Data Models
 - ◆ Physical Data Models

➤ Normalized Relational Models

- ◆ OLTP vs. OLAP
- ◆ What is Normalization?
- ◆ Many-to-many Resolution
- ◆ Generalization Hierarchies

➤ Dimensional Data Modeling

- ◆ Dimensional Characteristics
- ◆ Determining Facts and Dimensions
- ◆ Browsing, Reporting, Slicing and Dicing

- Dimensional schemas
 - ◆ Star schemas
 - ◆ Snowflake schemas
 - ◆ Multi-dimensional schemas

- Dimensional modeling for insurance applications
 - ◆ Characteristics of source systems
 - ◆ Dimensions and facts
 - ◆ Example



Introduction

- What is data modeling?
- Why data model?
- What types of data models do we use?

What is a Data Model?

DATA MODEL

*The specification of **data structures** and **business rules** to represent business requirements.*

STUDENT

student id
student last name
student first name
student dormitory
student major

COURSE

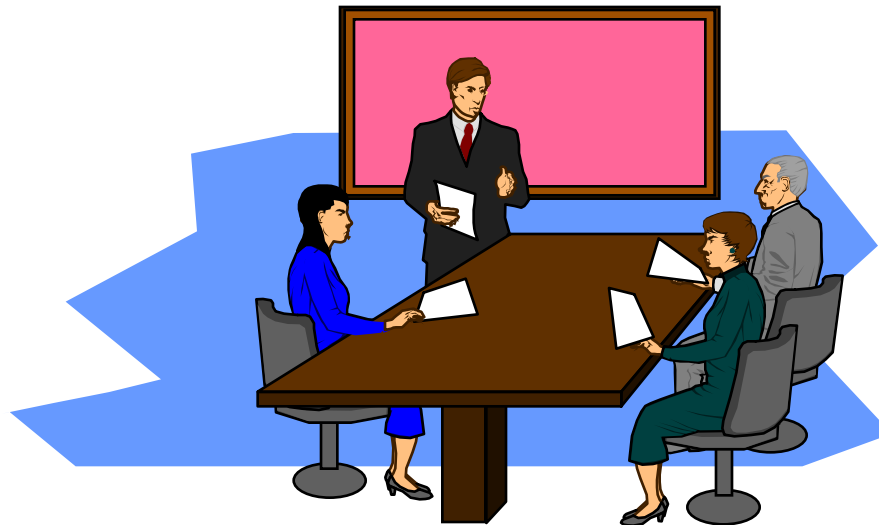
course id
course title
course number of credits
course location
course instructor name

● — attends/
is taught to — ●

What is Data Modeling?

DATA MODELING

A structured approach used to identify major components of an information system's specifications.



- Data Modeling is the process used to:
 - ◆ Analyze the data
 - ◆ Identify the relationships
 - ◆ Create the data model
- We use it to visualize informational needs in the form of an **Entity Relationship Diagram (ERD)**



Types of Modeling

- Business models often coexist with Data Models to enable software development.
- Other types of modeling:
 - ◆ Business Process Model
 - ◆ Functional Model
 - ◆ Object Model



Business Process Modeling

Business Process Modeling is defined as the identification and codification of:

- ◆ the business rules
- ◆ the repetitive activities that surround the data
- ◆ subject areas
- ◆ high level entities

Business Modeling is high level conceptual modeling of business processes (accounting, marketing)



Functional Modeling

Functional Modeling is defined as a more detailed level of business modeling that is closely related to business functions.

- ◆ Assists in application development
- ◆ May include functions like 'Open an Account', 'Close an Account'
- ◆ Used to document needs for front end application.



Object Modeling

- Object Modeling is defined as a type of functional modeling that incorporates data used by the business.
- Object Models represent the roles that entities, attributes and domains play in an organization's structure rather than the relationships between entities.



Introduction

- Purpose: to provide a communications vehicle and a visual representation of the information that is needed, collected and used by the organization.
- Other types of modeling are **business process, functional, and object**.

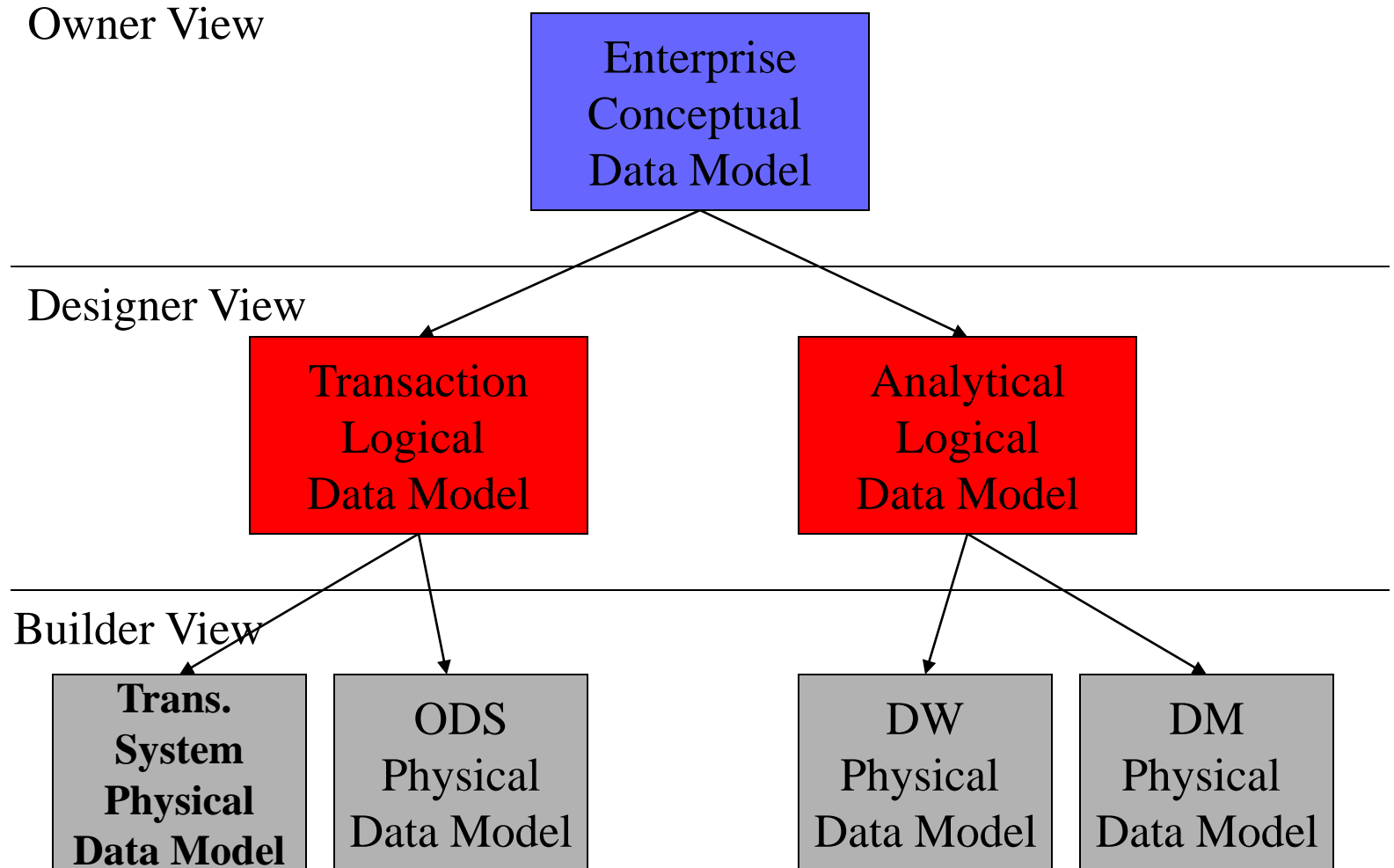


The Data Modeling Process

- How do we define the data modeling process?
- Modeling Tools
- Conceptual Data Models
- Logical Data Models
- Physical Data Models

- What model we need depends on what perspective we need to view the data from.
- Different roles within an organization view the data different ways and use it for different purposes.
- Models will be called something different based on what view they represent.

Different Views





Modeling Tools

- ERwin (Entity Relationship Windows)
- Sterling Software COOL
- System Architect
- PowerDesigner
- DBArtisan
- VISIO

- A **Conceptual Data Model** is a structured business view of the data required to support current business processes, business events, and related performance measures.
- It is a **single integrated data structure** which reflects the structure of the business functions rather than the processing flow or the physical arrangement of data.

Characteristics

- Represents overall logical structure of data
- Independent of software or data storage structure
- Often contains objects not implemented in physical databases
- Represents data needed to run an enterprise or a business activity



Logical Data Model

- A Logical Data Model builds upon the business requirements and includes a further level of detail that supports both the business and system requirements.
- Business rules are incorporated into the LDM and it loses some of the 'generalities' from the Enterprise CDM.

Logical Data Model

Characteristics

- Independent of specific software and data storage structure
- Includes more specific entities and attributes
- Includes business rules and relationships
- Includes foreign keys, alternate keys, and inversion entries

ENTITY

A person, place, thing, event, or concept about which the business keeps data.

- Each ENTITY represents a set/collection of like individual objects called instances.
- Each instance must be uniquely identifiable and distinct from all other instances.

Naming the Entity

- Each ENTITY should be named using:
 - ◆ *Unique* entity names in a model
 - ◆ A non-technical, *business-like* name
 - ◆ A *singular noun* that describes a singular instance (no collective nouns)
 - ◆ All UPPERCASE (Erwin standard)

SERVICE CONTRACT

~~serv_contracts~~

Defining the Entity

- Each entity should be fully defined ***by the business community*** to:
 - ◆ Identify **why** the business needs this information
 - ◆ Improve understanding
 - ◆ Avoid redundancy

A SERVICE CONTRACT is a PRODUCT representing an AGREEMENT between our COMPANY and the PURCHASER that offers service coverage extending beyond the normal WARRANTY period.

Entity Types

- Two types of entities:
 - ◆ Independent: depends on no other entity for its identification

ORDER

- ◆ Dependent: depends on one or more entities for its identification

LINE ITEM

Attributes

ATTRIBUTE

A distinct characteristic of an ENTITY for which data is maintained.

ENTITY Name
(above the box)

→ EMPLOYEE

employee id

employee first name

employee last name

employee address

employee phone number

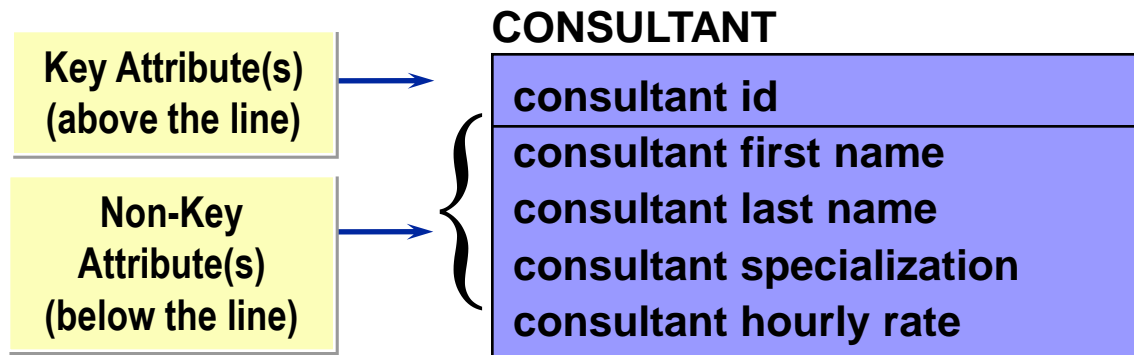
} attributes

Naming Attributes

- Each ATTRIBUTE should be named using:
 - ◆ *Unique* attribute names in a model (or entity)
 - ◆ A non-technical, *business-like* name
 - ◆ A *singular noun* that describes a singular instance (no collective nouns)
 - ◆ All lowercase (Erwin standard), with spaces

Attribute Types

- Two types of attributes:
 - ◆ Key
 - ◆ Non-key



How do we determine keys?

CANDIDATE KEY

Any attribute or group of attributes which serves to uniquely identify each instance of an ENTITY.

BOOK

author first name
author last name
book title
book edition
book publisher
book year published
book isbn
book lc catalog number

BOOK

author first name
author last name
book title
book edition
book publisher
book year published
book isbn
book lc catalog number

BOOK

author first name
author last name
book title
book edition
book publisher
book year published
book isbn
book lc catalog number

Primary Keys

PRIMARY KEY

An ATTRIBUTE or group of attributes that uniquely identifies an instance of the entity.

- The primary key is always placed above the line in an Entity

BOOK

book isbn

author first name
author last name
book title
book edition
book publisher
book year published
book lc catalog number

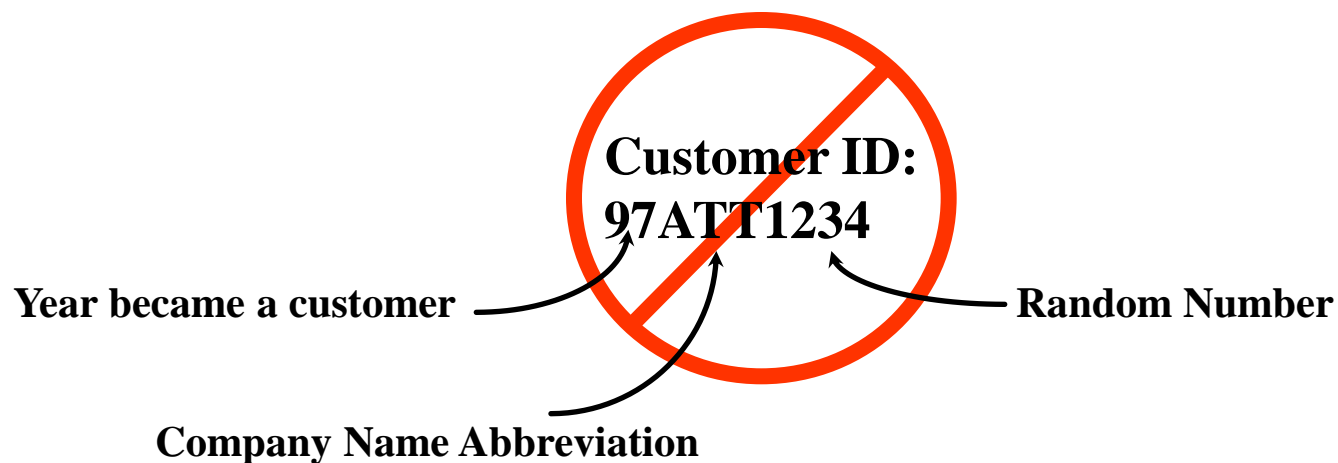


Choosing a Primary Key

- Factors to consider:
 - ◆ Should be efficient
 - ◆ Must not contain any null parts
 - ◆ Values must remain static
 - ◆ Should be a data element in your control

AVOID EMBEDDED INTELLIGENCE

Avoid the use of intelligent keys where a single attribute is granted a structure having hidden meaning.



Surrogate Keys

SURROGATE KEY

A contrived, non-intelligent, single-attribute key used to replace a long composite key.

CASH MACHINE TRANSACTION

account id
customer id
cash machine id
transaction date

Composite Key

CASH MACHINE TRANSACTION

transaction id
account id
customer id
cash machine id
transaction date

Surrogate Key

Alternate Keys

ALTERNATE KEY

An attribute or set of attributes which uniquely identifies each instance, but is not chosen as the PRIMARY KEY.

- ◆ Shown on diagram by (AK x . y) after each attribute in the Alternate Key
 - x represents an integer incremented for each Alternate Key in a given entity
 - y represents the order of the attribute in the key

BOOK

book isbn

author first name (AK1.2)

author last name (AK1.1)

book title (AK1.3)

book edition (AK1.4)

book publisher

book year published

book lc catalog number (AK2.1)

Inversion Entries

INVERSION ENTRY

Use Inversion Entries when one or more attributes are frequently used to access one or more ENTITY instances.

- ◆ Shown on diagram by (IEx.y) after each attribute in the Inversion Entry.
 - x represents an integer incremented for each inversion entry in a given entity
 - y represents the order of the attribute in the Inversion Entry

BOOK

book isbn

author first name (AK1.2, IE1.2)

author last name (AK1.1, IE1.1)

book title (AK1.3)

book edition (AK1.4)

book publisher (IE2.1)

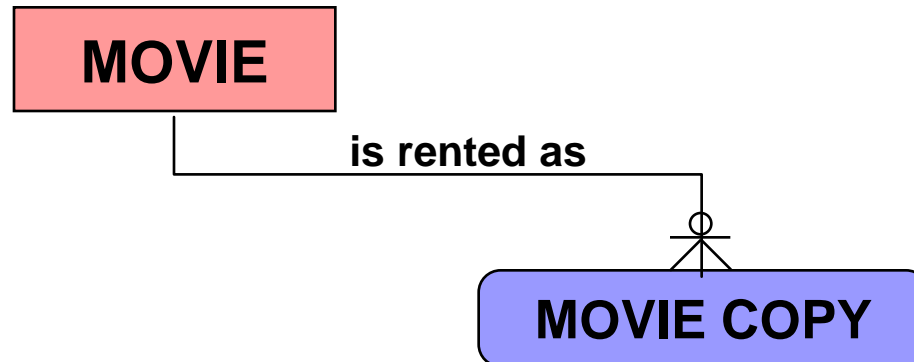
book year published (IE2.2)

book lc catalog number (AK2.1)

Relationships

RELATIONSHIP

A logical link between two entities that represents a business rule or constraint.

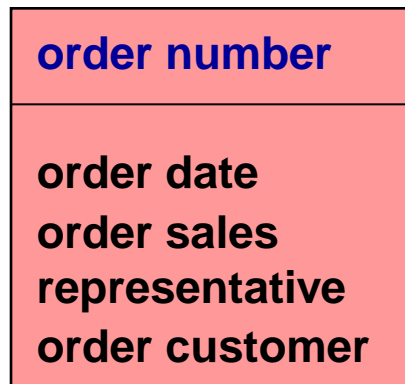


Foreign Keys

FOREIGN KEY (FK)

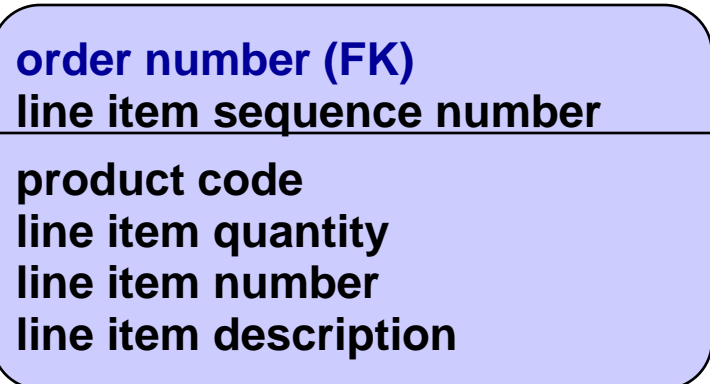
A primary key of a parent entity that is contributed to a child entity across a relationship.

ORDER



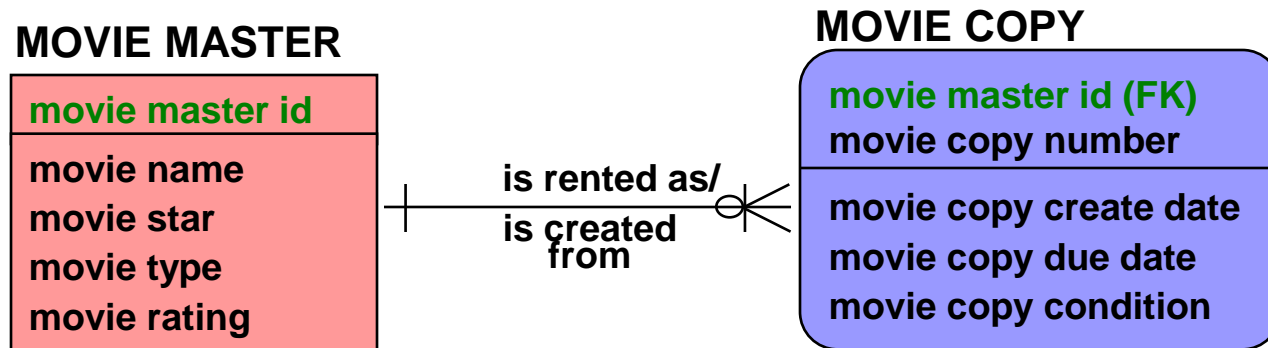
contains

LINE ITEM



Identifying Relationship

- The parent Primary Key migrates through the relationship to become part of the Primary Key (*IDENTITY*) of the child
- The child is *DEPENDENT* upon the parent for its identification and cannot exist without the parent



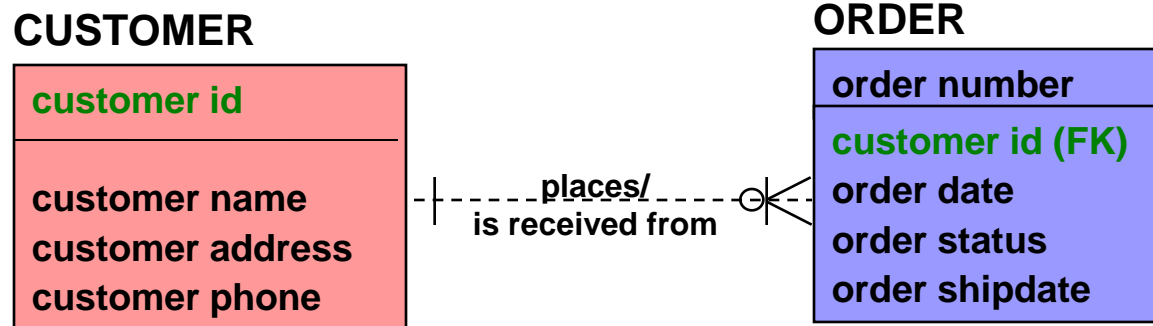
Assertion 1: *Each MOVIE MASTER <is rented as> 0, 1, or more MOVIE COPYs.*

Assertion 2: *Each MOVIE COPY <is created from> one and only one MOVIE MASTER.*

Assertion 3: *A MOVIE COPY cannot exist without its MOVIE MASTER.*

Non-identifying Mandatory Relationship

- The parent Primary Key migrates as a non-key attribute to the child and does *NOT IDENTIFY* the child
- The child is *INDEPENDENT* of the parent for its identification, but still cannot exist without a parent (*mandatory*)



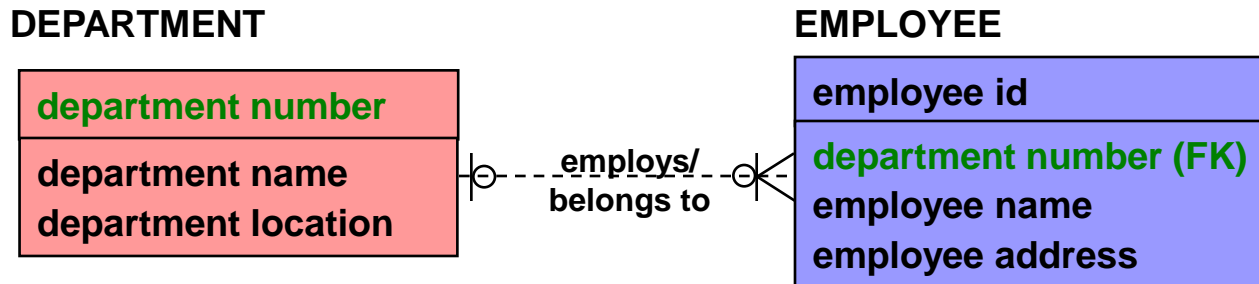
Assertion 1: *Each CUSTOMER* <places> 0, 1, or more *ORDERs*.

Assertion 2: *Each ORDER* <is received from> one and only one *CUSTOMER*.

Assertion 3: An *ORDER* can be identified without information about a *CUSTOMER*, but requires a value for the customer id (MUST HAVE A PARENT).

Non-identifying Optional Relationship

- The parent Primary Key migrates as a non-key attribute to the child and does *NOT IDENTIFY* the child
- The child is *INDEPENDENT* of the parent for its identification
- The diamond (optionality indicator) indicates that a child may exist without parent information



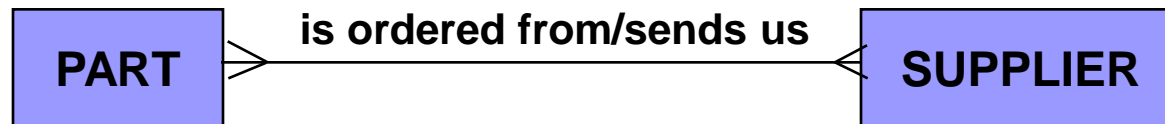
Assertion 1: *Each DEPARTMENT <employs> 0, 1, or more EMPLOYEEs.*

Assertion 2: *Each EMPLOYEE OPTIONALLY <belongs to> a DEPARTMENT.*

Assertion 3: *An EMPLOYEE can be identified without information about the DEPARTMENT, and does not have to be associated with a DEPARTMENT (CAN EXIST WITHOUT A PARENT).*

Many-to-Many Relationship

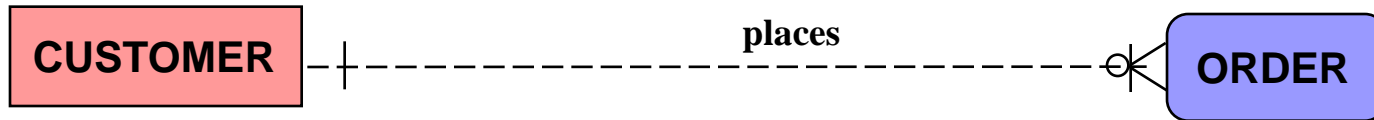
- A non-specific relationship in which primary keys do not migrate as foreign keys
- Must have two verb phrases
- Each phrase states the rule from the perspective of:
 - ◆ parent to child
 - ◆ child to parent



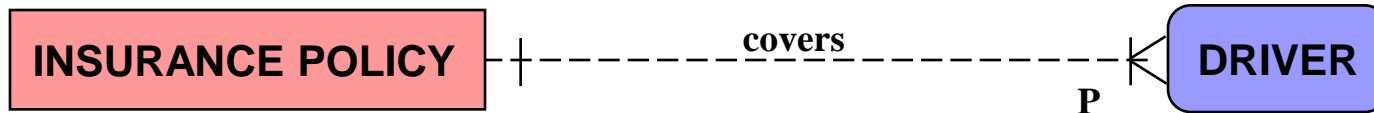
Assertion 1: *Each PART <is ordered from> 0, 1, or more SUPPLIERS.*

Assertion 2: *Each SUPPLIER <sends us> 0, 1, or more PARTs.*

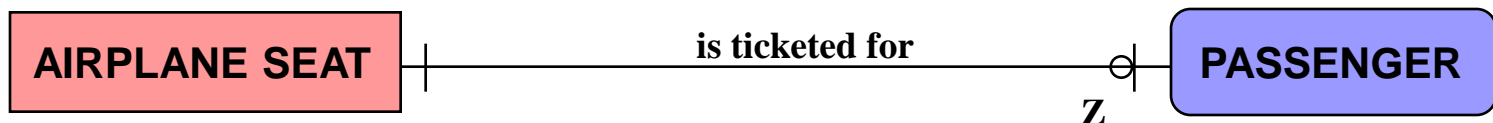
Relationship Cardinality



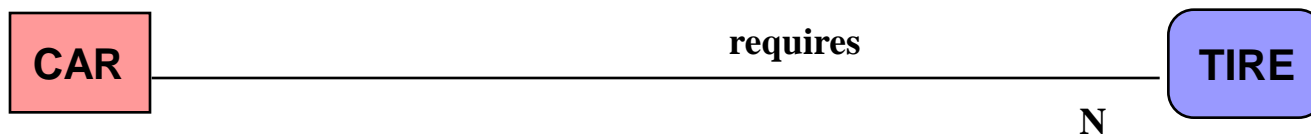
Each parent instance is related to zero, one, or more child instances



Each parent instance is related to one or more child instances



Each parent instance is related to zero or one child instances



Each parent instance is related to exactly "N" child instances

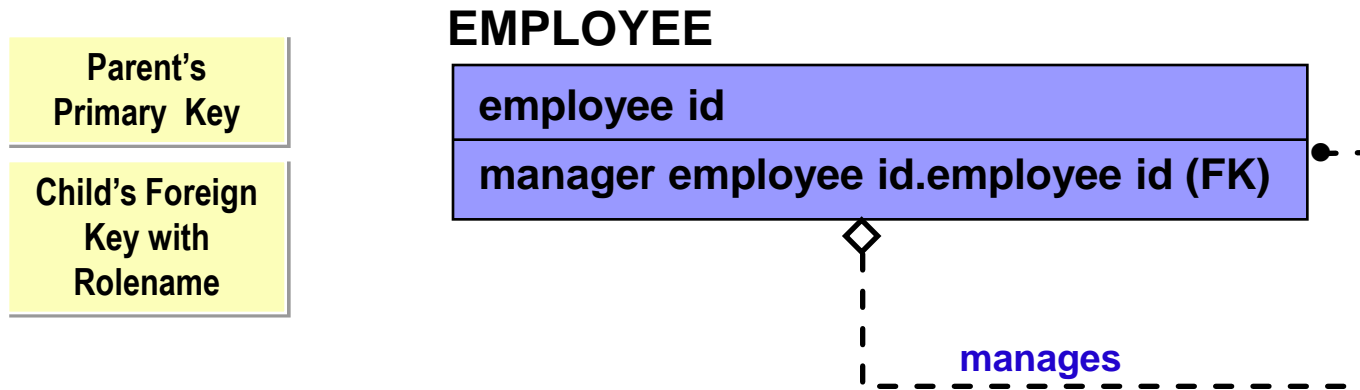
Recursive Relationships

RECURSIVE RELATIONSHIP

A non-identifying non-mandatory relationship in which the same entity is both the parent and the child.

- A recursive relationship:
 - ◆ is always a non-identifying relationship
 - ◆ allows null values for foreign key

Recursive Relationships



- The parent entity *instance*'s primary key is migrated to the non-key area of the child entity *instance*.
- Each migrating primary key attribute *must* be given a rolename to clarify the attribute's foreign key role.

Generalization Hierarchy

GENERALIZATION HIERARCHY

A hierarchical grouping of entities that share common attributes.

ACCOUNT

account number
account balance
account holder
account open date
account review date
account minimum balance

**Generic Parent
(Supertype) contains
generalized
attributes and Key**

SAVINGS ACCOUNT

account number (FK)
savings interest rate

CHECKING ACCOUNT

account number (FK)
available balance
per check charge

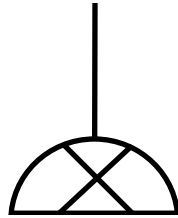
**Category (Subtype) Entity
contains migrated Foreign
Key and distinct Category
Attributes**



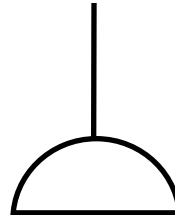
Generalization Hierarchy

- Two reasons to create a generalization hierarchy:
 - ◆ An entity can be partitioned into ‘type’ ranges and the ‘subtypes’ have distinct attributes.
 - ◆ A subtype requires relationships that cannot be generalized at the ‘super type’ level.

Generalization Hierarchy Relationship Symbols



Exclusive
(one)



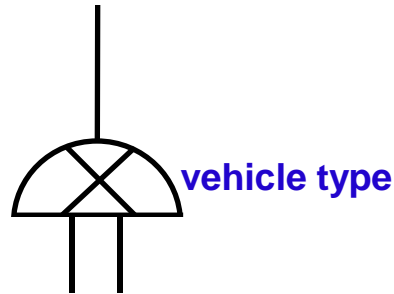
Inclusive
(one or more)

- Use the Exclusive symbol where each instance of the supertype has an entry in exactly one of the subtypes
- Use the Inclusive symbol where each instance of the supertype may have an entry in one or more subtypes

Category Discriminator

CATEGORY DISCRIMINATOR

An attribute in the supertype entity that is chosen to depict the category to which a given instance belongs.

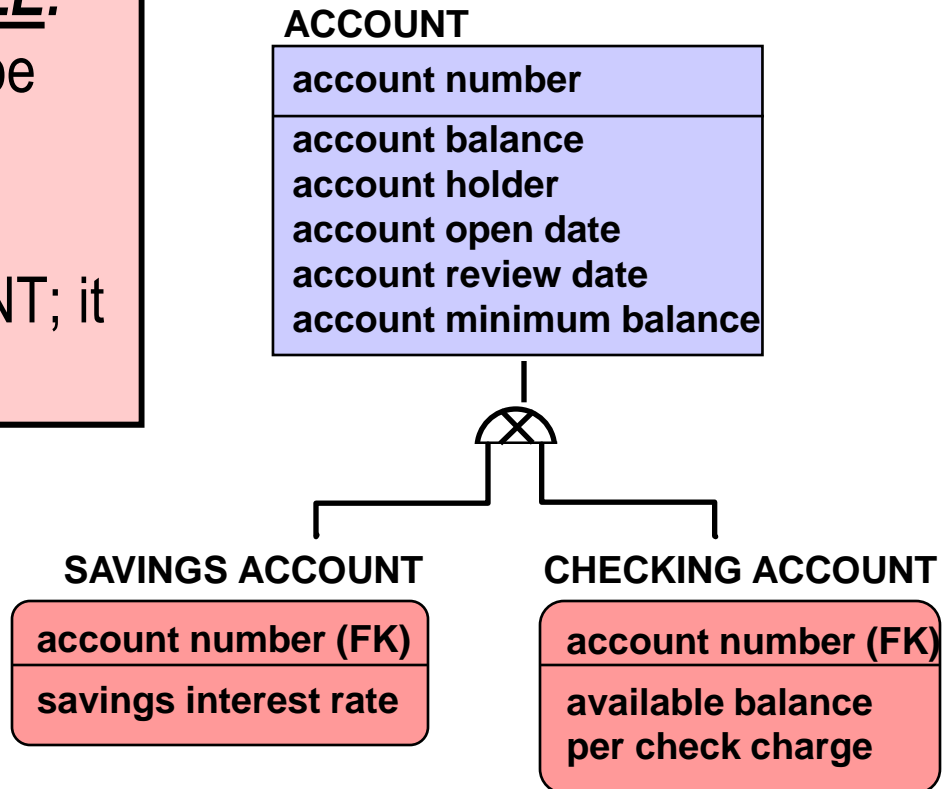


The Category Discriminator attribute can appear alongside the relationship symbol.

Exclusive Subtype Relationship

THE BUSINESS RULE:

An ACCOUNT must be either a SAVINGS ACCOUNT or a CHECKING ACCOUNT; it cannot be both.



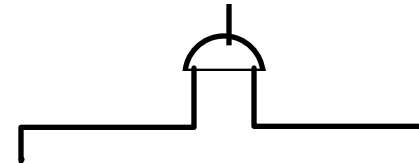
Inclusive Subtype Relationship

THE BUSINESS RULE:

An ACCOUNT may be a SAVINGS ACCOUNT or a CHECKING ACCOUNT, or it may be both.

ACCOUNT

account number
account balance
account holder
account open date
account review date
account minimum balance



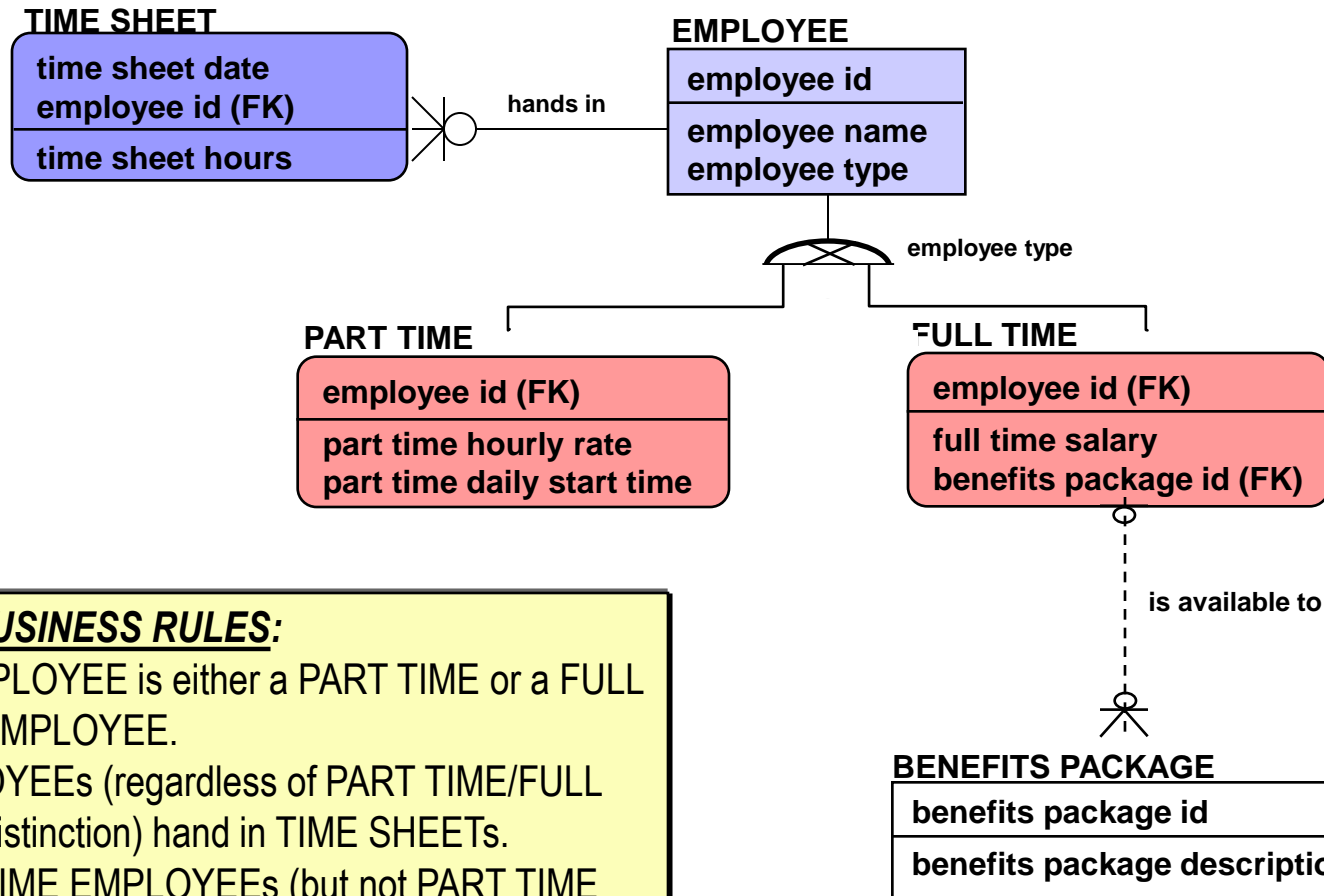
SAVINGS ACCOUNT

account number (FK)
savings interest rate

CHECKING ACCOUNT

account number (FK)
available balance
per check charge

Separate Relationships Example



THE BUSINESS RULES:

An EMPLOYEE is either a PART TIME or a FULL TIME EMPLOYEE.

EMPLOYEEs (regardless of PART TIME/FULL TIME distinction) hand in TIME SHEETs.

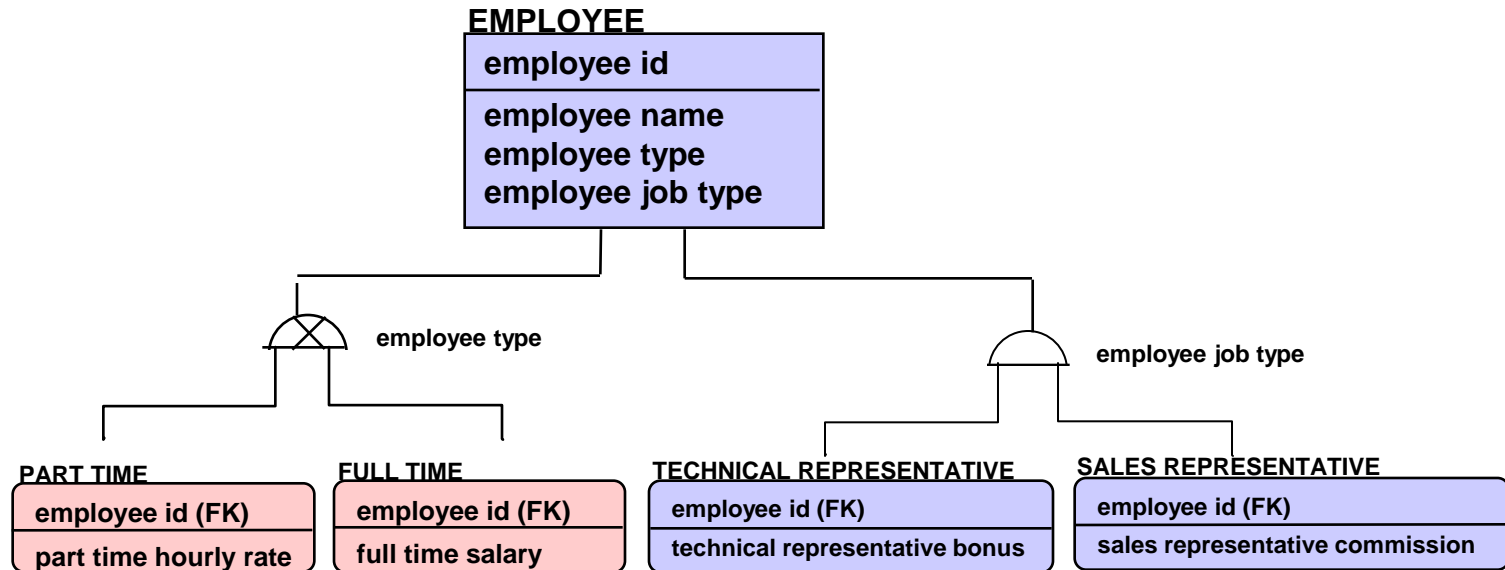
FULL TIME EMPLOYEEs (but not PART TIME EMPLOYEEs) are eligible for benefits.

Multiple Generalization Hierarchies

THE BUSINESS RULES:

EMPLOYEEs **must be** either FULL TIME **OR** PART TIME.

EMPLOYEEs **may** be either a TECHNICAL REPRESENTATIVE **OR** a SALES REPRESENTATIVE [or another Job Type]



Referential Integrity (RI)

REFERENTIAL INTEGRITY (RI)

Rules that determine what happens when a Parent or Child instance is inserted, updated or deleted.

Why Referential Integrity?

- What is the impact of:
 - ◆ Inserting, updating, or deleting a Parent Primary Key value?
 - ◆ Inserting, updating, or deleting a Child Foreign Key value?
- None of these actions should break the relationship from Child to Parent
- Options can be specified as to how the DBMS should manage these actions to maintain referential integrity

Referential Integrity Options

<u>RI Option</u>	<u>What It Does</u>
RESTRICT	Does not allow Action
CASCADE	Duplicates Action in related tables
SET NULL	Allows Action and sets the Child foreign key to Null
SET DEFAULT	Allows Action and sets the Child foreign key to the Default value
NONE	No Restriction placed on Action

RI Entity Default Matrix

- Erwin defaults RI to the matrix below.

ACTION	RELATIONSHIP TYPE			
	Identifying	Non-Identifying (Nulls Allowed)	Non-Identifying (No Nulls)	Subtypes
Child Delete	None	None	None	None
Child Insert	Restrict	Set Null	Restrict	Restrict
Child Update	Restrict	Set Null	Restrict	Restrict
Parent Delete	Restrict	Set Null	Restrict	Cascade
Parent Insert	None	None	None	None
Parent Update	Restrict	Set Null	Restrict	Cascade

Normalization

NORMALIZATION

A formal data modeling approach to examining and validating the model.

➤ Pros:

- ◆ Ensures that each attribute belongs to the entity to which it is assigned
- ◆ Redundant storage of information is minimized

➤ Cons:

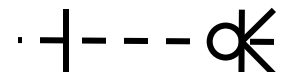
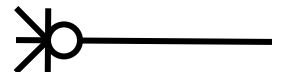
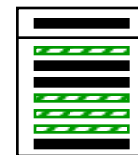
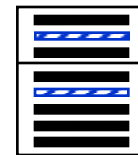
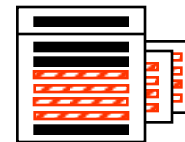
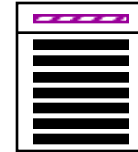
- ◆ Can adversely affect performance if rigorously implemented
- ◆ Can adversely affect deadlines if rigorously implemented

Normal Forms

- Dr. E. F. Codd identified 'normal forms' as the different states of a 'normalized relational' data model.
 - ◆ 1NF = No repeating groups
 - ◆ 2NF = No partial key dependencies
 - ◆ 3NF = No non-key interdependencies
 - ◆ 4NF = No independent multiple relationships
 - ◆ 5NF = No semantically related multiple relationships

How to Normalize an Entity

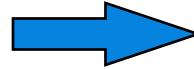
- Before you normalize an entity, identify its **Primary Key**
- Identify and resolve violations of **1NF** - make sure there are no repeating groups
- Identify and resolve violations of **2NF** - make sure that each non-key attribute depends on the entire key
- Identify and resolve violations of **3NF** - make sure that no non-key attribute depends on another non-key attribute



Identify the Primary Key

PUPPY

puppy number
puppy name
kennel name
kennel location
kennel code
trick id 1
trick name 1
trick where learned 1
trick level of difficulty 1
trick id 2
trick name 2
trick where learned 2
trick level of difficulty 2




PUPPY

puppy number
puppy name
kennel name
kennel location
kennel code
trick id 1
trick name 1
trick where learned 1
trick level of difficulty 1
trick id 2
trick name 2
trick where learned 2
trick level of difficulty 2

First Normal Form - 1NF



- Ask the following for each attribute:
 - ◆ Does this attribute occur more than once for any given instance? (NO REPEATING GROUPS)
- If yes,
 - ◆ build a new entity
 - ◆ move all 'repeating' attributes to the new entity
 - ◆ select or formulate attribute(s) for the new entity's PK
 - ◆ build an IDENTIFYING relationship FROM THE ORIGINAL entity TO THE NEW entity 

1NF Violation

PUPPY

puppy number

puppy name

kennel name

kennel location

kennel code

trick id 1

trick name 1

trick where learned 1

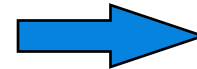
trick level of difficulty 1

trick id 2

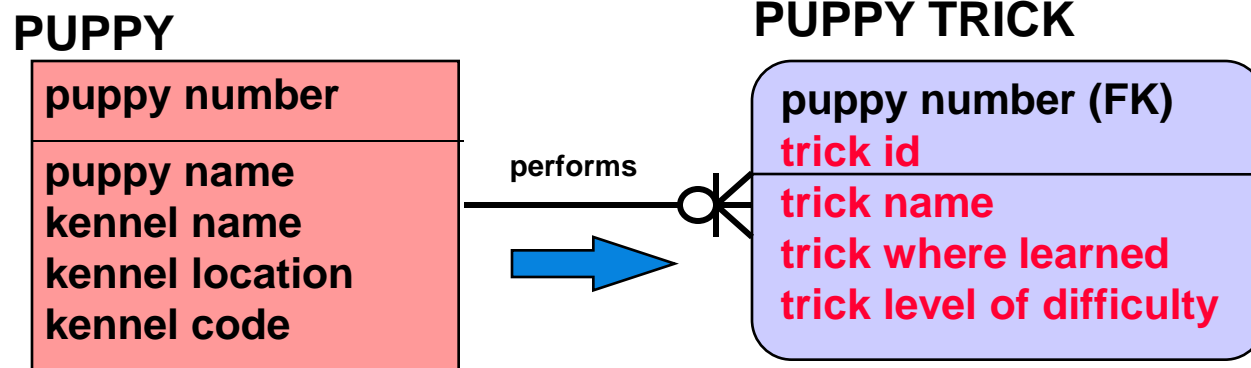
trick name 2

trick where learned 2

trick level of difficulty 2

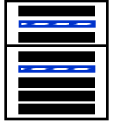


1NF - Solution



The new entity should be named to reflect its intention, given a primary key, and the inherited foreign key will be present as part of the primary key.

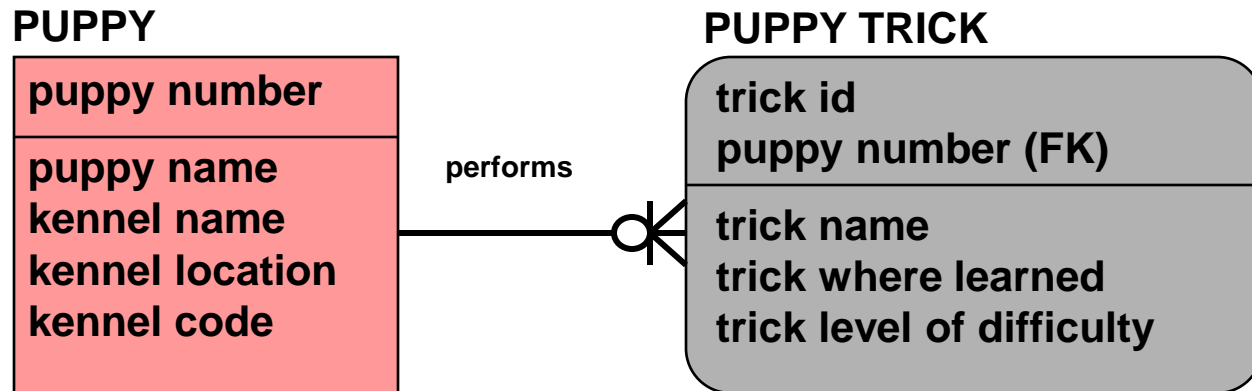
Second Normal Form - 2NF



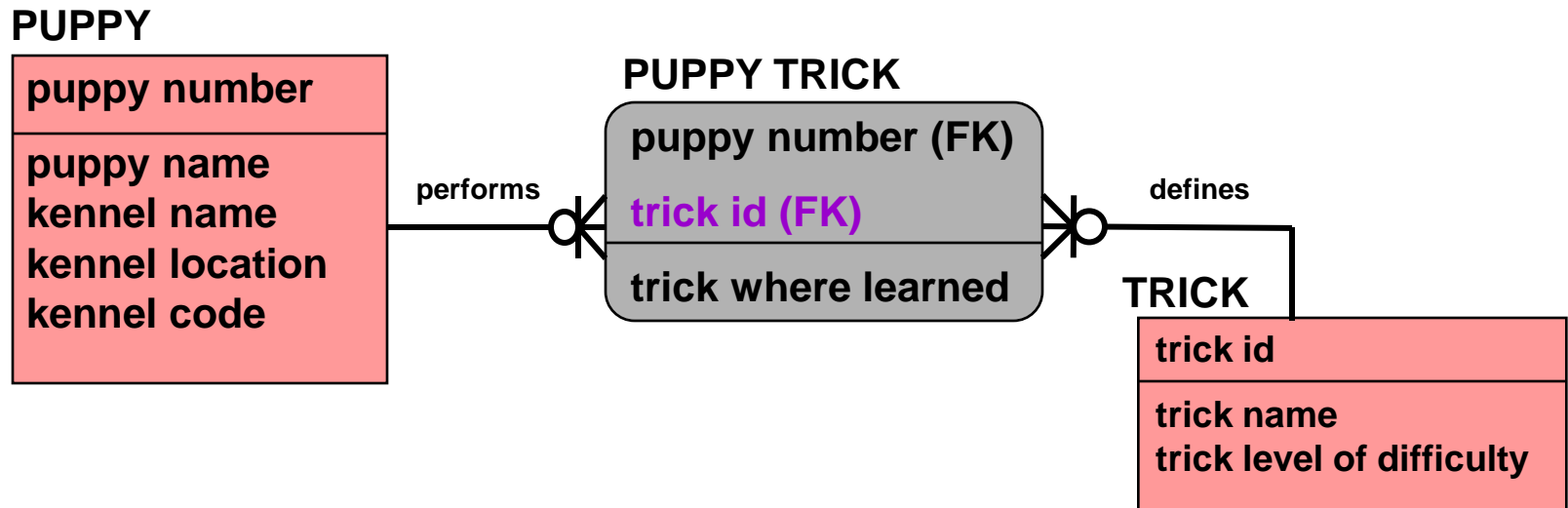
- For ONLY those entities that have a composite key, ask the following of each non-key attribute:
 - ◆ Is this attribute dependent on *part* of the primary key? (NO PARTIAL KEY DEPENDENCIES)
- If yes,
 - ◆ build a new entity
 - ◆ move all the attributes having the *same* partial key dependency to the new entity
 - ◆ use the determinant attribute as the key or determine a better PK (move the PK attribute too)
 - ◆ build and name an identifying relationship FROM NEW entity BACK TO ORIGINAL entity



2NF - Violation

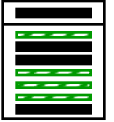


2NF - Solution

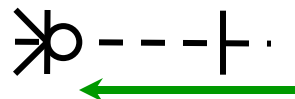


Separate the attributes that depend only on trick id (in the key of PUPPY TRICK) into their own 'TRICK' Entity; then return the borrowed attribute to reconstruct the whole key.

Third Normal Form - 3NF



- For each non-key attribute, ask the following:
 - ◆ Does this attribute depend on some other non-key attribute? (NO Non-Key INTERDEPENDENCIES)
- If yes,
 - ◆ build a new entity to contain all attributes with *same* non-key dependency
 - ◆ use determinant attribute(s) as the PK
 - ◆ build and name a non-identifying, non-mandatory relationship FROM NEW entity BACK TO ORIGINAL entity



3NF - Violation

PUPPY

puppy number
puppy name
kennel name
kennel location
kennel code

PUPPY TRICK

puppy number (FK)
trick id (FK)
trick where learned

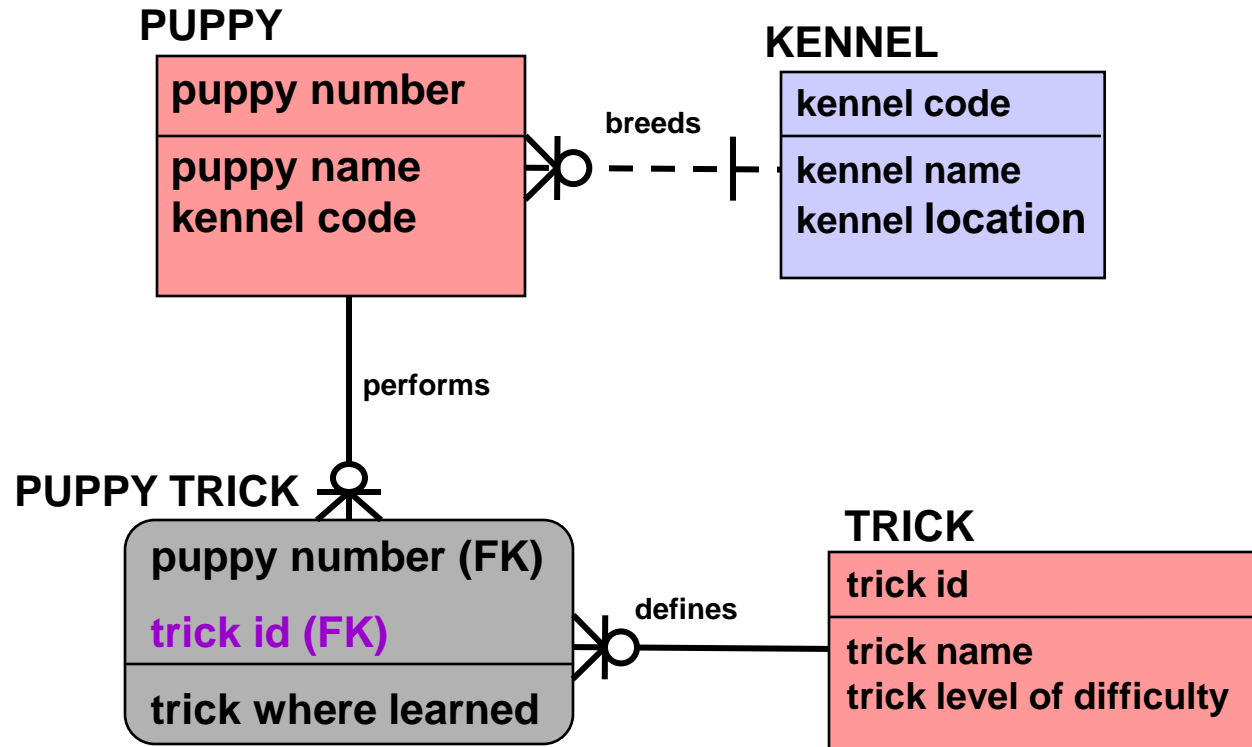
performs

defines

TRICK

trick id
trick name
trick level of difficulty

3NF - Solution



Separate the non-key attributes that depend on other non-key attributes (i.e., kennel location) into their own entity.



Results of Normalization

- Data is easier to define
- Data interdependencies are identified
- Data ambiguities are resolved
- Data model can be more flexible
- Data model is easier to maintain
- Performance can be an issue
- The structure can be very complex



Physical Data Model

- A Physical Data Model is specific to the software and performance constraints of the specific database management system to be used in the implementation.
- Both software and data storage structures are considered and the model is often modified to meet performance or physical constraints.



Physical Data Model

Characteristics

- Dependent specific software and data storage structure
- Includes tables and columns
- Includes physical database objects (triggers, stored procedures, tablespaces)
- Includes referential integrity rules that restrict relationships between tables

Physical Representation of Data Model

Logical vs.
Physical

CUSTOMER

customer number
customer first name
customer last name
customer company name
customer street address
customer city
customer state code
customer zip code
customer area code
customer phone number
customer fax area code
customer fax number

SALES ORDER

sales order number
sales order date
sales order total
sales order status
customer number (FK)
employee number (FK)
shipment date
shipment charge
payment number (FK)

places

CUSTOMER

customer_number: int
first_name: char(15)
last_name: char(20)
customer_company_name: varchar(20)
street_address : char(30)
city: char(30)
state: char(2)
zip_code: char(9)
area_code_phone: char(3)
phone_number: char(7)
fax_area_code: char(3)
fax_number: char(7)

SALES_ORDER

sales_order_number: int
sales_order_date: datetime
sales_order_total: decimal(9,2)
sales_order_status: smallint
customer_number: int
emp_no: int
shipment_date: datetime
shipment_charge: decimal(5,2)
payment_number: int

places

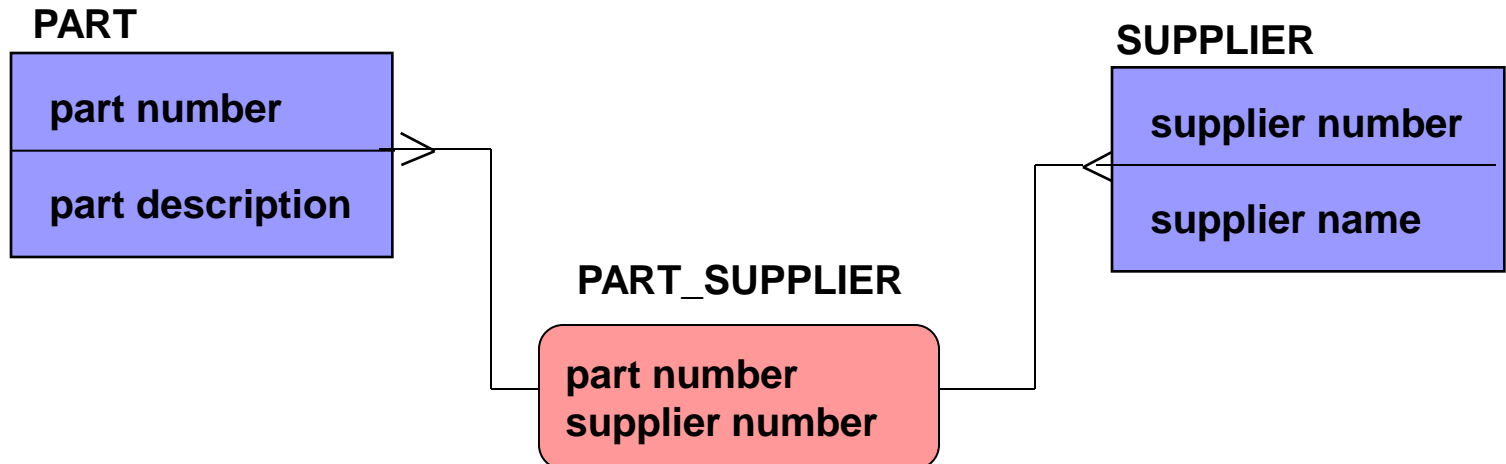
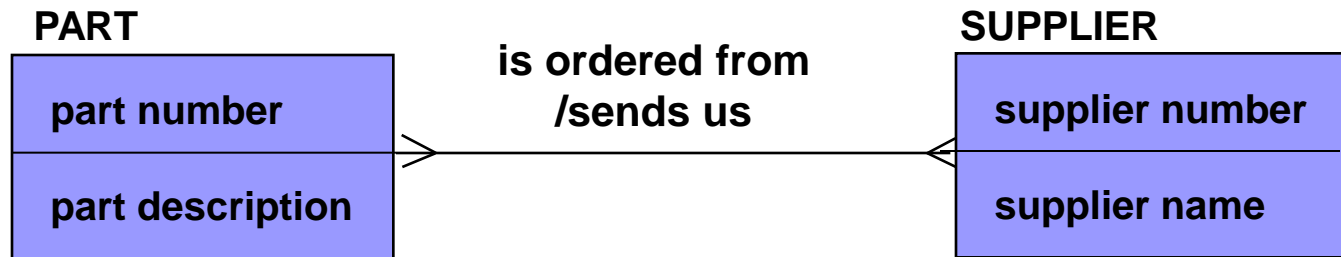


Logical vs. Physical

- Entity becomes Table
- Attribute becomes Column
- Alternate Key/Inversion Entry becomes Index
- Some objects are logical only; some are physical only

'Logical Only' Objects

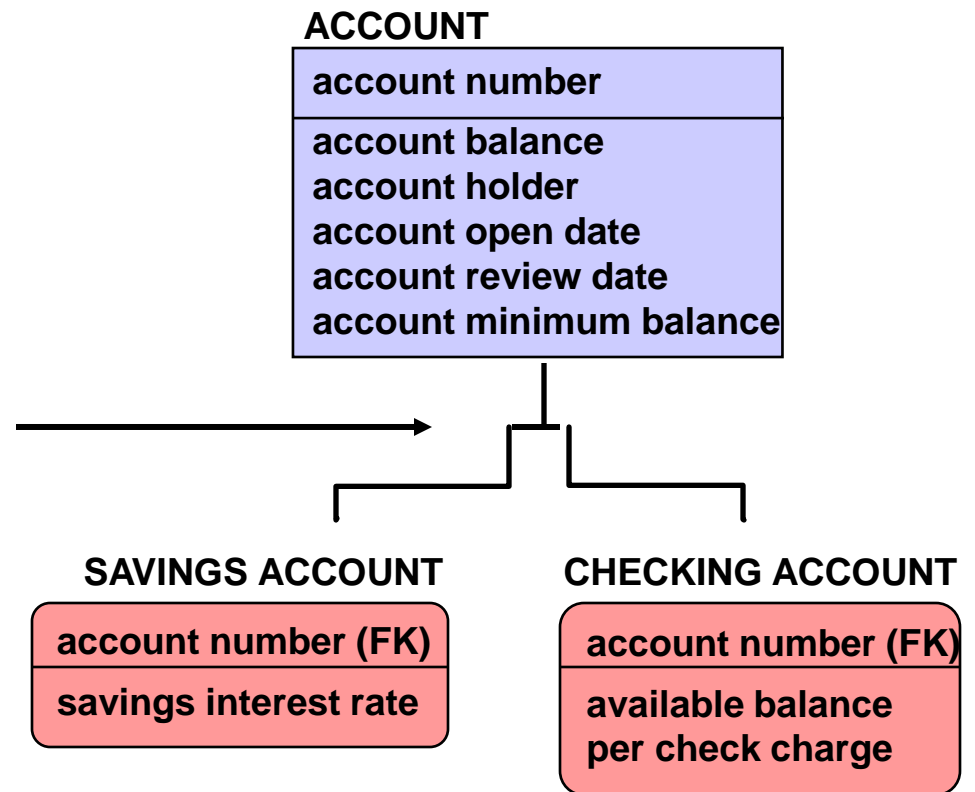
- Many-to-Many relationships are shown only in the logical Model



'Logical Only' Objects

- Generalization Hierarchies are only shown in the logical model.

Hierarchy symbol changes but tables do not (unless manually changed)



'Logical Only' Objects

- Entire entities, individual attributes or relationships may also be designated as 'logical only'
- For example, high-level entities that help explain the logical design but are not part of the physical design

'Logical Only' Objects

PERSON

person id
person first name
person last name
person street address
person city
person state code
person zip code
person area code
person phone number

CUSTOMER

customer id.person id (FK)
customer company name
customer since

ORDER

order number
sales representative (FK)
shipment method code (FK)
payment number (FK)
customer id (FK)
order date
order shipment charge
order total
order shipment date (IE4)

CUSTOMER

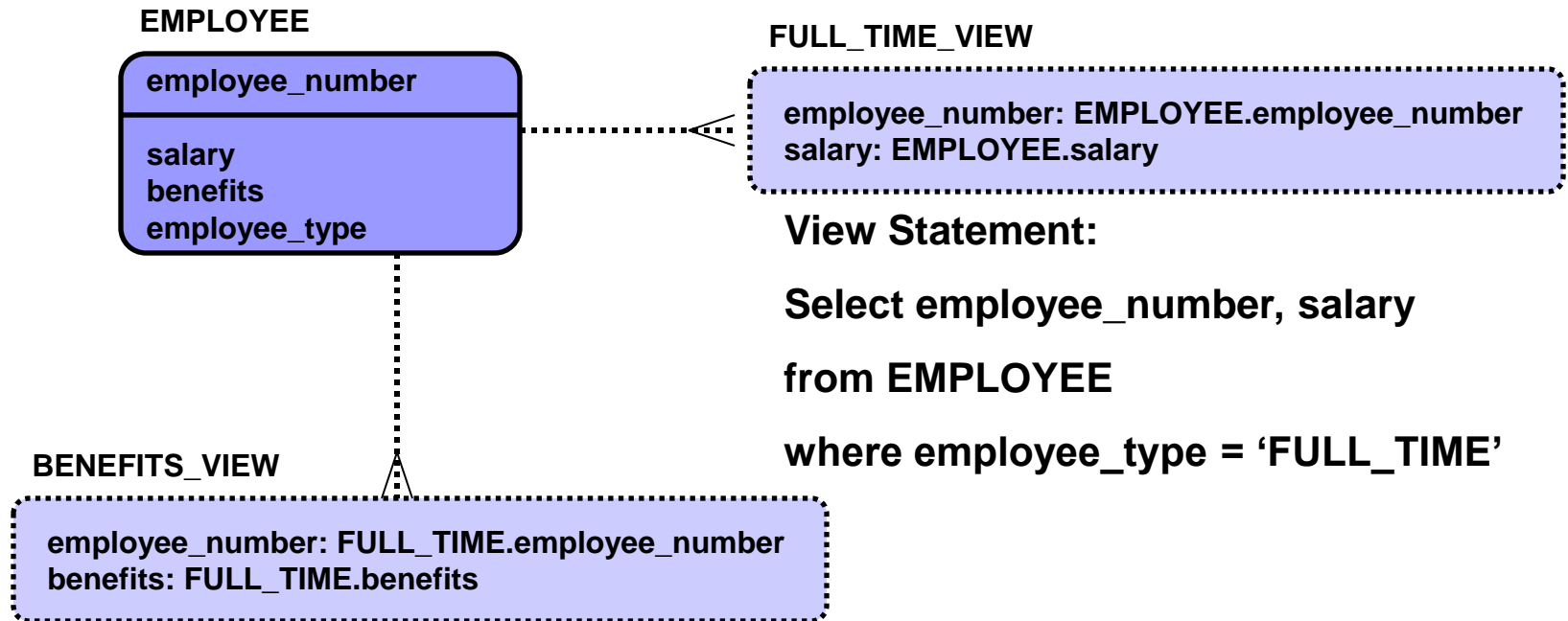
customer_id
customer_first_name
customer_last_name
customer_street_address
customer_city
customer_state_code
customer_zip_code
customer_area_code
customer_phone_number
customer_company_name
customer_since

ORDERs

order_number
sales_representative
shipment_method_code
payment_number
customer_id
order_date
order_shipment_charge
order_total
order_shipment_date

'Physical Only' Objects

- Views are only shown in the physical model



View Statement:

```
Select employee_number, salary
from EMPLOYEE
```

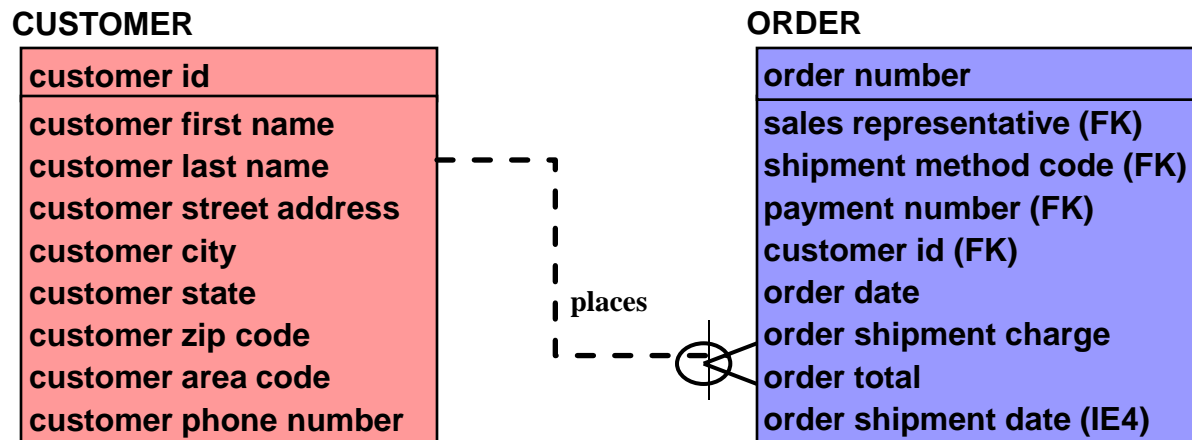
```
where employee_type = 'FULL_TIME'
```

View Statement:

```
Select employee_number, benefits
from EMPLOYEE
```

'Physical Only' Objects

- Entire tables, columns or relationships may also be designated as 'physical only'
 - ◆ Code Tables
 - ◆ Columns for Auditing Purposes



'Physical Only' Objects

- STATE is a code ('look-up') table
- 'Last_Update' fields in CUSTOMER

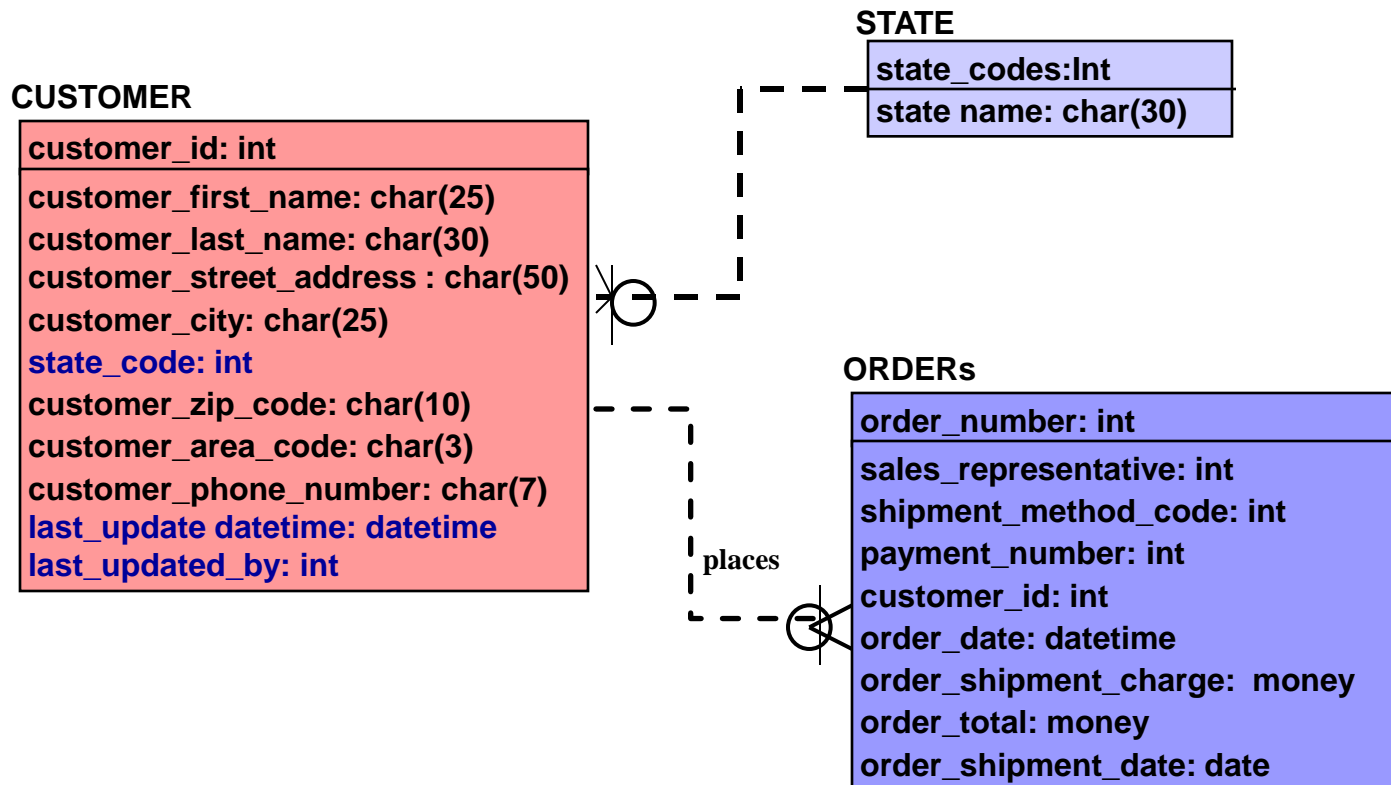


Table Properties

TABLE PROPERTY

A property that can be assigned to a physical table and which is supported by the specified target DBMS.

A property that can be assigned to a table in a client application and which is supported by the specified target client tool.

- The physical model should include table characteristics including table spaces, segments, and other 'physical only' characteristics.
- The name of the table should be changed to reflect the database's naming standards.

Column Properties

COLUMN PROPERTY

A property that can be assigned to a physical table column and which is supported by the specified target DBMS.

A property that can be assigned to a column in a client application and which is supported by the specified target client tool.

- The physical model shows column data types and field lengths in the context of the specified database.
- Column names should be changed to reflect the database's naming standards.

INDEX

A structure associated with a table to make searching faster.

- There is a trade-off between the number of indexes and performance.
 - ◆ More indexes = faster searching and better query performance.
 - ◆ More indexes = slower inserts and updates.
- This is important for dimensional modeling!



Other Physical Properties

- Other physical objects include (availability will vary based on DBMS):
 - ◆ Segments/Table spaces/Databases
 - ◆ Stored Procedures
 - ◆ Triggers
 - ◆ Scripts



Dimensional Data Models

- Dimensional Data Models
 - ◆ Developed top-down
 - ◆ Depicts a business process through its relevant facts and dimensions
 - ◆ Groups data into categories of business measure and characteristics
 - ◆ More suitable for analytical applications where the focus is querying large sets of data

FACT

A measurement, generally additive in nature, of the organization.

- Facts are typically described as the performance measures of the business.
- Usually numerical and represent counts, dollar amounts, percentages or ratios.
- Examples are sales, revenue, expenses, policies, and claims.

Dimensions

DIMENSION

An entity by which the business views the measures (facts).

- Dimensions are groupings of similar data into a larger category or subject area.
- Dimensions may be hierarchical in nature, like Time - hours roll into days, days into months, etc.
- Dimensions may not be hierarchical, such as a vehicle dimension with model, year, color, etc.

ATTRIBUTE

*A distinct characteristic of an **DIMENSION** for which data is maintained.*

- Fields within a dimension that describe the item associated with a dimension.
- Acts as a source of query constraints.
- Dimensions are made up of attributes.



Characteristics

- A Dimensional Model has the following characteristics:
 - ◆ The structure is organized around subject areas.
 - ◆ The structure is integrated.
 - ◆ The structure has a standard set of keys, naming conventions, and field formats.
 - ◆ Connection paths are clearly defined.



OLAP vs. OLTP

- Normalized models are the standard for transactional systems (OLTP).
- OLTP systems are highly volatile, meaning they perform thousands of transactions (inserts, updates, or deletes) in minutes or seconds.
- They must complete these transactions as quickly as possible to maximize response time of the applications using the DBMS.



OLAP vs. OLTP

- Dimensional models are the standard for data warehouses and DSS (OLAP).
- OLAP systems must query large amounts of data, perform summarization tasks, and present the result set to the user.
- They must be able to handle extremely large result sets and return that information to the user quickly without bogging the source transaction systems down.



Opposite Philosophies

- Since OLTP is focused on fast transactions, it has characteristics of a normalized relational model:
 - ◆ Minimal redundancy (normalization)
 - ◆ Limited index use
 - ◆ Efficient use of storage space
 - ◆ Elimination of inconsistent data
 - ◆ Few maintenance concerns



Opposite Philosophies

- Since OLAP is focused on query performance, it has characteristics of a dimensional model:
 - ◆ Increased redundancy (de-normalization)
 - ◆ Increased index use
 - ◆ Increased storage space
 - ◆ Consolidation of inconsistent data
 - ◆ Increased maintenance issues (history)



Terminology

- **Grain** - Level of detail in fact and dimension tables.
- **Hierarchy** - Represents the organizational structure of like data groupings. Implies levels and rollup.
- **Fact** - Performance measure of the business.
- **Dimension** - Descriptions, or views, of facts.
- **Attribute** - Descriptions of dimensions or the data elements on the dimension tables.



Terminology

- **Atomic Layer** - Dimensions and facts at the lowest level of detail (think ODS).
- **Summary Layer** - Dimensions and facts aggregated to intermediate values.
- **Presentation Layer** - Dimensions, facts, and other tables altered specifically for presentation tool limitations.
- **Reporting Layer** - Dimensions, facts, and other tables created or altered to improve reporting capabilities and performance.



Dimensional Modeling Process

Step 1: Choose the grain of each fact table.

- Granularity defines the level of detailed data.
- It must be determined prior to going forward in the modeling process.
- Typical grains are individual transactions, time-based aggregation, and/or aggregations along a commonly used dimension.



Dimensional Modeling Process

Step 2: Choose the dimension attributes.

- For example, what should our time dimension look like? Should it have just 'January for month', or also 'Jan' and '1'?
- Should we store the code and the description, just the code, or just the description?
- What values will our users need to filter or report on?



Dimensional Modeling Process

Step 3: Identify dimensional hierarchies.

- A dimension such as time may have days rolling into months and then quarters, as well as days rolling into weeks which may cross months and quarters.
- Sales geography may differ from physical geography.
- Zip codes can cross city boundaries and cities are made up of multiple zip codes.



Dimensional Modeling Process

Step 4: Choose the dimensions that apply to each fact table.

- Typical dimensions include time, product, policyholder, agent, and geography.
- Remember to evaluate granularity when applying dimensions to facts.



Dimensional Modeling Process

Step 5: Choose the measured facts, including precalculated facts.

- Each aggregated and derived fact will need to be evaluated for inclusion in the model or calculation in the application.
- Trade-offs include storage and indexing and must be weighed against the access requirements.



Dimensional Modeling Process

Step 6: Determine slowly changing dimensions

- These are the dimensions that change over time.
- If tracking these changes is important, the method must be decided.
- Options: overwrite the existing record, store all records with effective dates, or a historical and current value tables.



Dimensional Model Schemas

- Dimensional Data Models fall into three types of models:
 - ◆ Star Schema
 - ◆ Snowflake Schema
 - ◆ Multi-dimensional Schema
- Several factors influence schema choice:
 - ◆ Presentation restrictions
 - ◆ Inconsistency of data
 - ◆ Complex queries and analysis

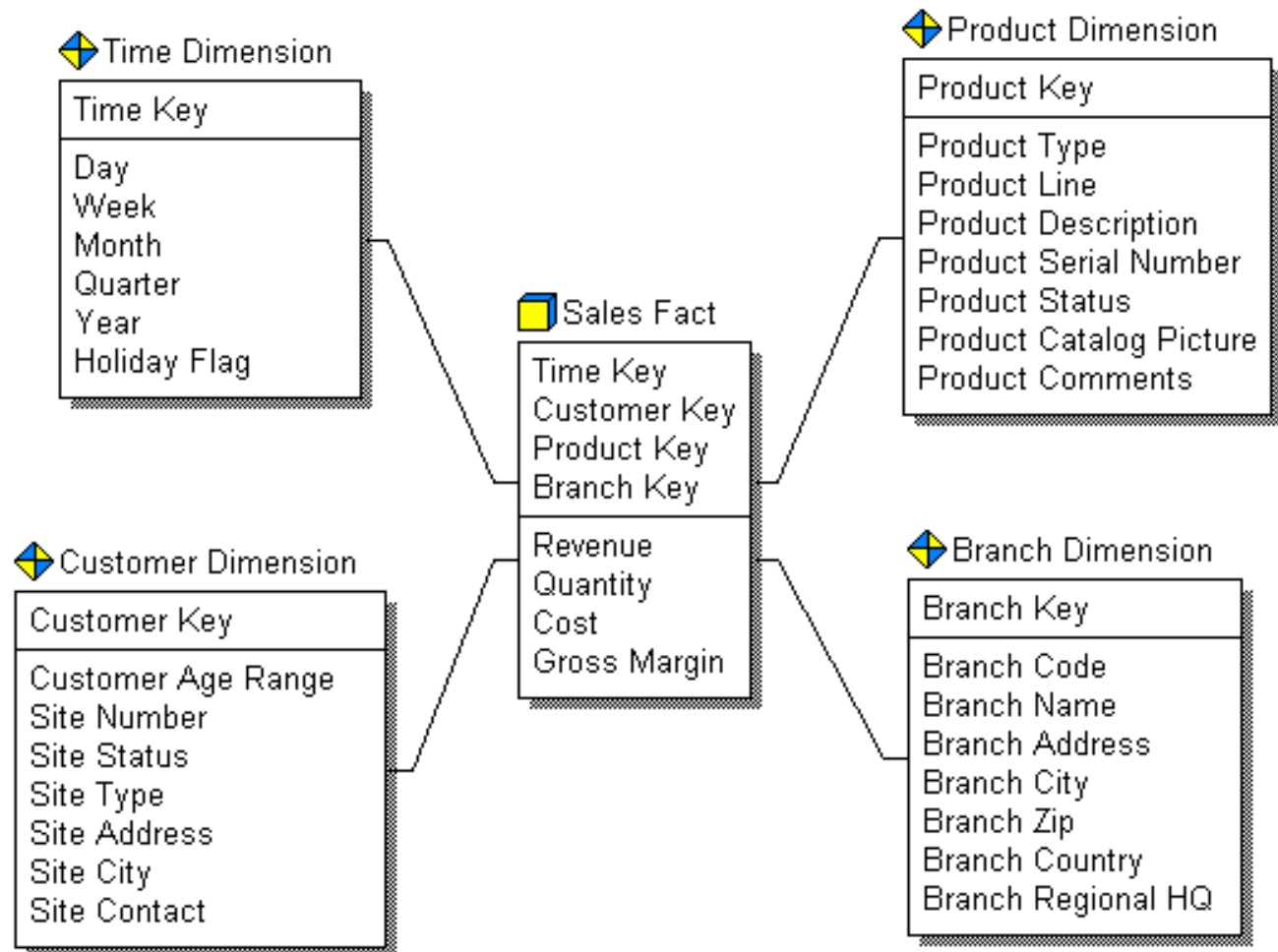
Star Schema

STAR SCHEMA

A database design that stores a central fact table surrounded by multiple dimension tables.

- Star schema represents a compromise between the fully normalized model and the denormalized model.
- Descriptive 'dimension' information is maintained in a set of denormalized dimension tables.

Star Schema



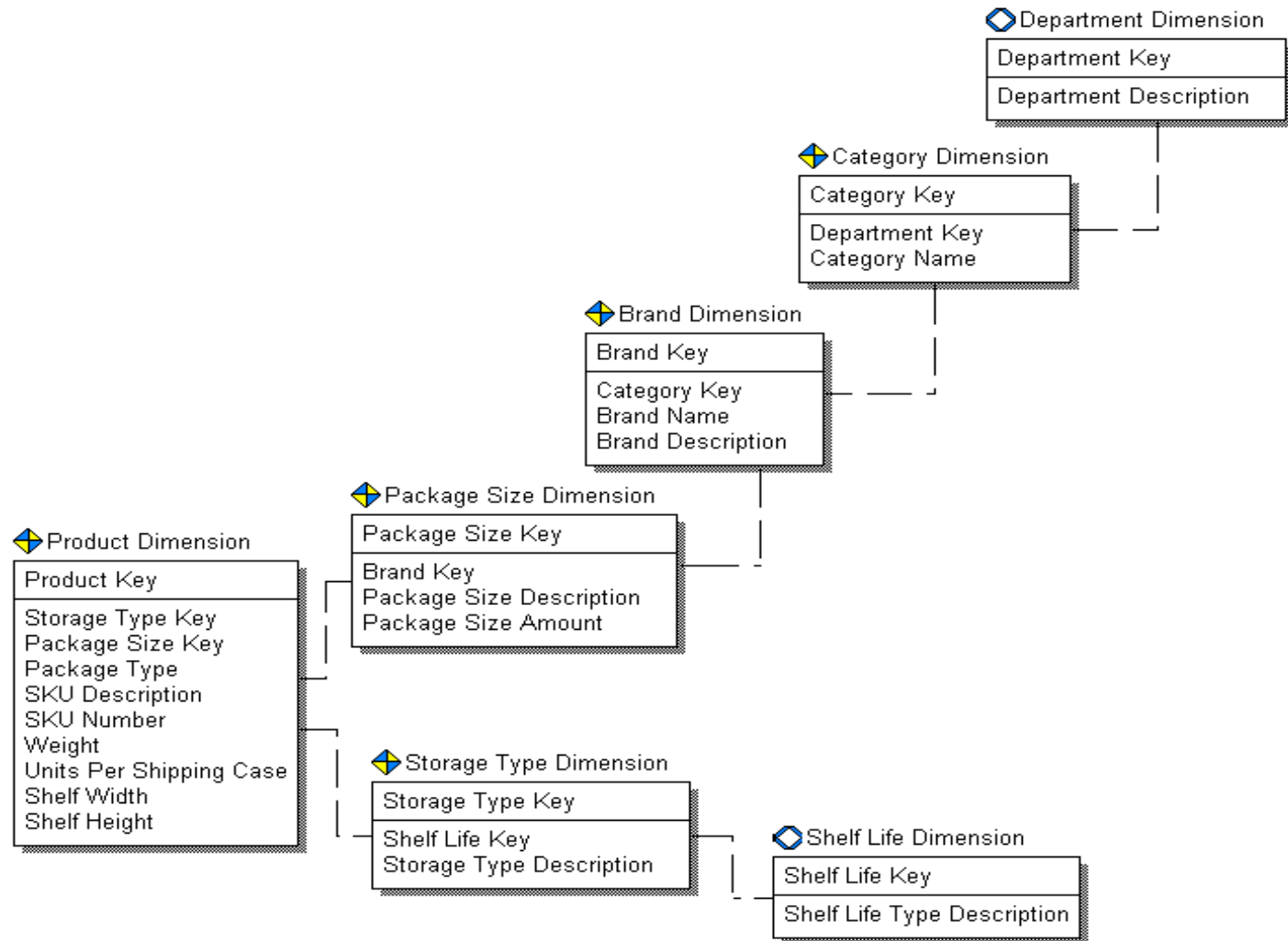
Snowflake Schema

SNOWFLAKE SCHEMA

A database design that stores a central fact table surrounded by multiple dimension tables decomposed or normalized into one or more hierarchies.

- Snowflake schemas are most often used when dealing with large hierarchies that are static.
- Snowflaked tables (look-up tables) may increase the speed of queries depending on the presentation tool (i.e. MicroStrategy)

Snowflake Schema





Star vs. Snowflake

- Star Schemas:
 - ◆ Easier to navigate
 - ◆ Improved performance
 - ◆ Supports 'browsing' dimensions
 - ◆ More widely used
 - ◆ Storage saved with snowflake becomes negligible



Star vs. Snowflake

➤ Snowflake Schemas:

- ◆ Hierarchical structure can be visualized
- ◆ Easier for data modelers and DBA's to use because it is closer to normalized model
- ◆ Some tools don't support snowflakes
- ◆ Destroys 'browsing' speed and flexibility

Multi-Dimensional Schemas

MULTI-DIMENSIONAL SCHEMA

Hierarchical databases that consists of only one structure - a multi-dimensional array - that contains all the summarized data at higher levels in the array.

- Also known as MOLAP databases
- Stores and aggregates data at multiple levels in a hierarchy.
- Utilizes drill-up and drill-down to move around the hierarchy.



Multi-Dimensional Schemas

- Multi-Dimensional Schemas:
 - ◆ Provide user with a cross-dimensional perspective allowing analysis across dimensions
 - ◆ Specialized programmer must create database
 - ◆ Data explosion becomes an issue because each additional dimension results in an exponential increase in the number of dimension intersections (cells).

MDM's and Sparsity

- Sparsity relates to the unpopulated cells in a table.
- It results from every combination of attributes not having a value or an entry associated with it.
- May be reduced if users are satisfied with more summarized than atomic level data.
- Addressing the issue may be tool-dependent.



Logical to Dimensional

- Moving from a logical relational model to a dimensional model requires following the dimensional steps.
- How do we take a relational table and 'dimensionalize' it?
- What issues are we concerned with when modeling new 'dimensions' from our existing logical model?

Adding Attributes

- Remember 'physical only' attributes:
 - ◆ Audit Columns
 - ◆ Code Table Foreign Keys
- Some attributes will need to be added to dimensions to handle cases that would not occur in an OLTP model.
 - ◆ Attributes denoting an 'event' in time
 - ◆ Demographical attributes that allow users to categorize customers by age, location, preferences

Generalized Keys

- If we are tracking history, each new record must have a new primary key in order to preserve referential integrity.
- Remember 'surrogate' keys:
 - ◆ A contrived, non-intelligent, single-attribute key used to replace a long composite key or an id attribute if there might be multiple history records for the same id.
- Customer Key might be added to the Customer dimension to capture this information.



Indicators and Flags

- Indicators/Flags are frequently used to identify special events.
- Events can also be captured in their own dimension table.
- Holidays may influence our sales or accident rates and some are not consistent dates.
- ‘Holiday Flag’, ‘Store Event’, and ‘Promotion’ are attributes that might be added to the Time dimension to capture this information.



Ranges

- Marketing analysts may need to browse data by age range (i.e. 21-30) for groups of customers.
- Other examples include income, marital status, level of education.
- Demographic attributes may be added to the Customer dimension to capture this information.



Auxiliary Date Attributes

- Issues arise around how to handle data that changes over time.
- For example, price records for a product may change based on promotions or seasonal factors.
- Effective dates and active row indicators may be added to a product or other slowly changing dimension to capture values changing over time.



Types of Dimensions

- Slowly Changing Dimensions
 - ◆ Type 1, 2 or 3
- Rapidly Changing or Volatile Dimensions
- Huge Dimensions and Mini-Dimensions
- Causal Dimensions
- Dirty Dimensions
- Degenerate Dimensions



Slowly-Changing Dimensions

- Most dimensions change over time.
 - ◆ Products change offered coverage or limits and deductibles.
 - ◆ Employees are promoted, fired, or change departments.
 - ◆ Customers change names and addresses.
- What are our choices for tracking these changes over time?



Slowly-Changing Dimensions

- There are three types of slowly changing dimensions:
 - ♦ **Type 1:** *Overwrites the old data* for a record with new data. This eliminates the ability to track history over time.
 - ♦ **Type 2:** *Creates a new record* with the new data at the type of the change. Accurately tracks history, but requires generalized key.
 - ♦ **Type 3:** *Tracks new and original values* in separate fields at time of change. Intermediate values are lost.


Type 1 - Overwrite Old Values

◆ Customer Dimension

Customer Key
Customer First Name
Customer Last Name
Customer Marital Status
Customer Age Range
Customer Education Level
Customer Street Address
Customer City
Customer State
Customer Account Number

- Customer Lynnette Groves is changing her name to ?
- If there is no value in tracking this change, we will overwrite the First Name and Last Name fields with the new values.
- 'UPDATE' statement; 1 record is maintained.

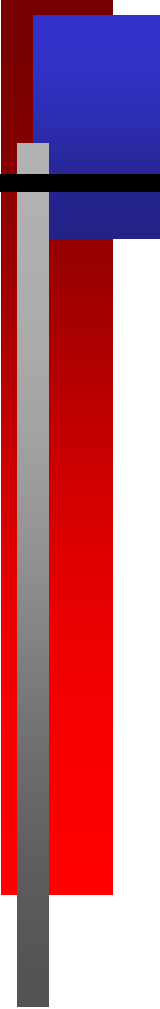
Type 2 - Create New Record

 Customer Dimension

Customer Key
Customer First Name
Customer Last Name
Customer Marital Status
Customer Age Range
Customer Education Level
Customer Street Address
Customer City
Customer State
Customer Account Number
Last Update Timestamp
Active Row Indicator

- Lynnette Groves is changing her name and we want to track both values
- Add a second record with a new Customer Key and make it the active row
- 'INSERT' statement for new, 'UPDATE' for active; 2 records are maintained
- New record for each change up to n records

Type 3 - Original and Current



Customer Dimension

Customer Key
Customer Current First Name
Customer Current Last Name
Customer Original First Name
Customer Original Last Name
Customer Marital Status
Customer Age Range
Customer Education Level
Customer Street Address
Customer City
Customer State
Customer Account Number
Last Update Timestamp

- We decide that no matter how many times she changes her name, we only want to track the original and the current.
- Before any changes, original and current are the same. Any name change updates 'current' fields.
- UPDATE' statement; 1 record is maintained



Volatile Dimensions

- What if a dimension's values change frequently?
- Price would naturally be an attribute of product and would change semi-frequently.
- Few products have prices that remain constant over many months or years.
- To capture these changes over time, we can capture these values in the fact table rather than treating it as a slowly changing dimension.



A General Rule...

- Fact tables contain counts, amounts, and other numerical information.
- Dimensions describe the business with textual fields and dates in time.
- As a general rule, one should question numerical information that occurs in the dimension tables as well as textual and data fields that occur in the fact table.



Huge Dimensions and Mini-Dimensions

- Product and Customer dimensions with millions and tens of millions of entries are not unusual for retailers, telecommunications companies, insurance companies, or financial service institutions.
- These dimensions can have hundreds of attributes and complex, multiple hierarchies that can exist simultaneously.

Huge Dimensions

HUGE DIMENSIONS

Dimensions with millions or tens of millions of entries, such as customer, that take too long to browse among relationships due to volume.

- The customer dimension in financial institutions, telecommunications companies, and catalog retailers hold data for customers on an individual basis.
- Over time, these can grow to tens of millions of rows.



Huge Dimensions and Mini-Dimensions

- The heavily-used fields in the Customer dimension consist of demographic information: age, sex, number of children, income level, education level, and other purchasing behavior information.
- These fields are also compared together to select an interesting subset of the market base for analysis.



Huge Dimensions and Mini-Dimensions

- The most effective technique for handling this situation is to separate one or more sets of these attributes into demographic mini-dimensions.
- If five or six of the demographic variables are isolated into a separate table, we need only to store the distinct combinations of information that actually occur.

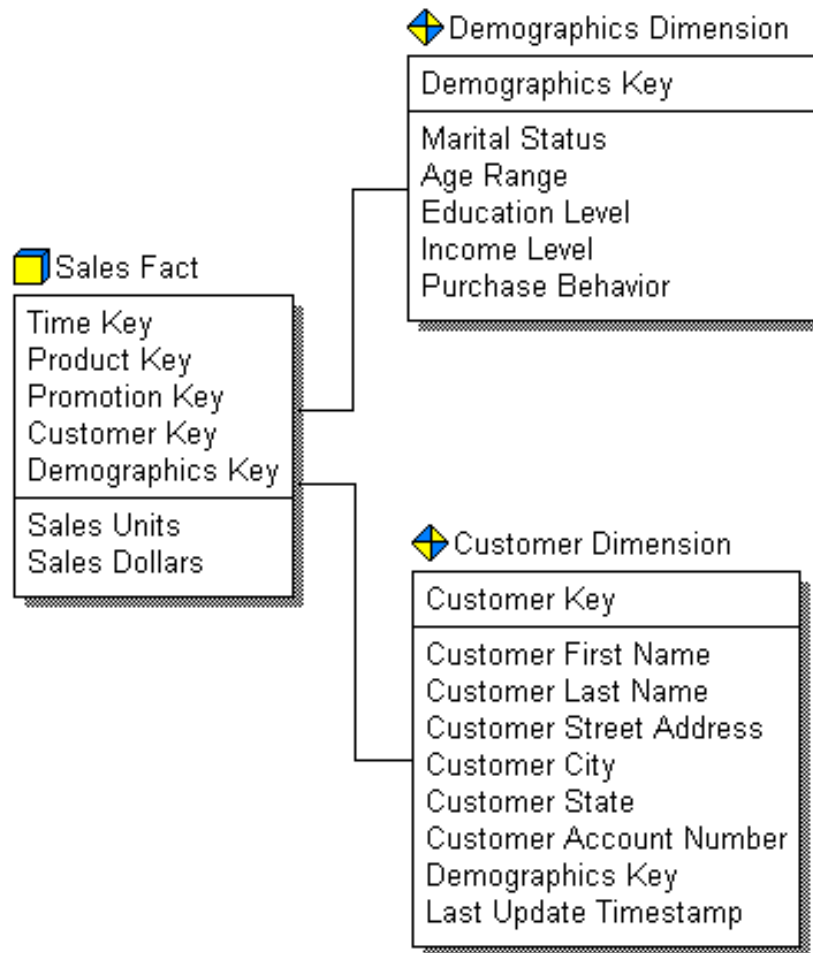
Mini-Dimensions

MINI-DIMENSIONS

Groups of related attributes separated into separate dimensions that create significant gains in performance and decreased volatility in the parent dimension.

- Typically, demographic information changes at a different rate than other customer information.
- Marketing can analyze different segments of the customer base for purchasing habits and other information.

Huge Dimensions and Mini-Dimensions



- Demographic dimension can be joined directly to the fact itself or 'snowflaked' to the customer dimension.
- Demographics Key is included in the Customer Dimension to browse data interactively.

Causal Dimensions

CAUSAL DIMENSIONS

Causal dimensions describe factors that are thought to cause a change in the performance of a measure or fact, such as advertising or promotion.

- Causal Dimensions track conditions that may influence sales, counts, or revenue.
- Promotions, holidays, and weather conditions may influence the behavior of fact data.



Causal Dimensions

- Causal dimensional attributes can be placed in a single dimension table or separated into different tables by subject.
- A 'Promotion' dimension could include price reduction type, ad type, display type, and promo start and end dates.
- The trade-offs include efficient browsing vs. more understandable tables for the user community.



Causal Dimensions

- Single table design:
 - ◆ What type of conditions are being tracked
 - ◆ Generalizes all conditions into one table
 - ◆ Multiple causal conditions may need to be stored on the same record.
- Multiple table design:
 - ◆ Different dimensions for holidays, marketing campaigns, and weather conditions.
 - ◆ Increases sparsity of fact when all conditions do not apply.

Dirty Dimensions

DIRTY DIMENSIONS

Dimensional information that may contain duplicate or extraneous entries due to inconsistent legacy data.

- Financial institutions might have a poor account-to-account correlation of individual's names.
- Insurance companies may not make a serious attempt to identify previous instances of an insured party or other policies.



Dirty Dimensions

- Some cleaning can be done in ETL process.
- Will influence fact data accuracy.
- All tools that access the data will need to take the possible inaccuracy of data into account. Some tools are designed to alleviate some of the problem to 80% accuracy.
- Level of inaccuracy may influence design of dimensions and facts so that it may be minimized.

Degenerate Dimensions

DEGENERATE DIMENSIONS

Dimensions that are so small and have no attributes of their own that they have been added to the fact table.

- Certain attributes are tracked that don't necessarily belong in their own dimension - orphan attributes.
- This may occur when fact tables are designed to reflect the actual working document.

Degenerate Dimensions

- Examples include 'order_number', 'bill_of_lading_num', and 'invoice_number'.
- While these fields seem very transaction oriented, they are helpful in grouping things such as all line items on an invoice.
- Including these fields on the fact table amounts to denormalizing the attribute due to the granularity of the fact table being the document itself or a line item of the document.

Heterogeneous Products

HETEROGENEOUS PRODUCTS

Situation that arises when product attributes and their measured facts are very different.

- The OLTP source system may keep bank products- savings accounts, checking accounts, or credit card accounts- in different tables because each product may have different relationships.
- The user wants to measure the business on the basis of all products.



Heterogeneous Products

- There are two main options when dealing with disparate products:
 - ◆ Option 1: Group all attributes into one large product dimension.
 - ◆ Option 2: Core Dimension with common attributes and product type that join to custom facts and dimensions.

Heterogeneous Products - Option 1

 Product Dimension

Product Key
Product Name
Product Description
Product Class
Product Type
Many Checking Attributes...
Many Savings Attributes...
Many Credit Card Attributes...

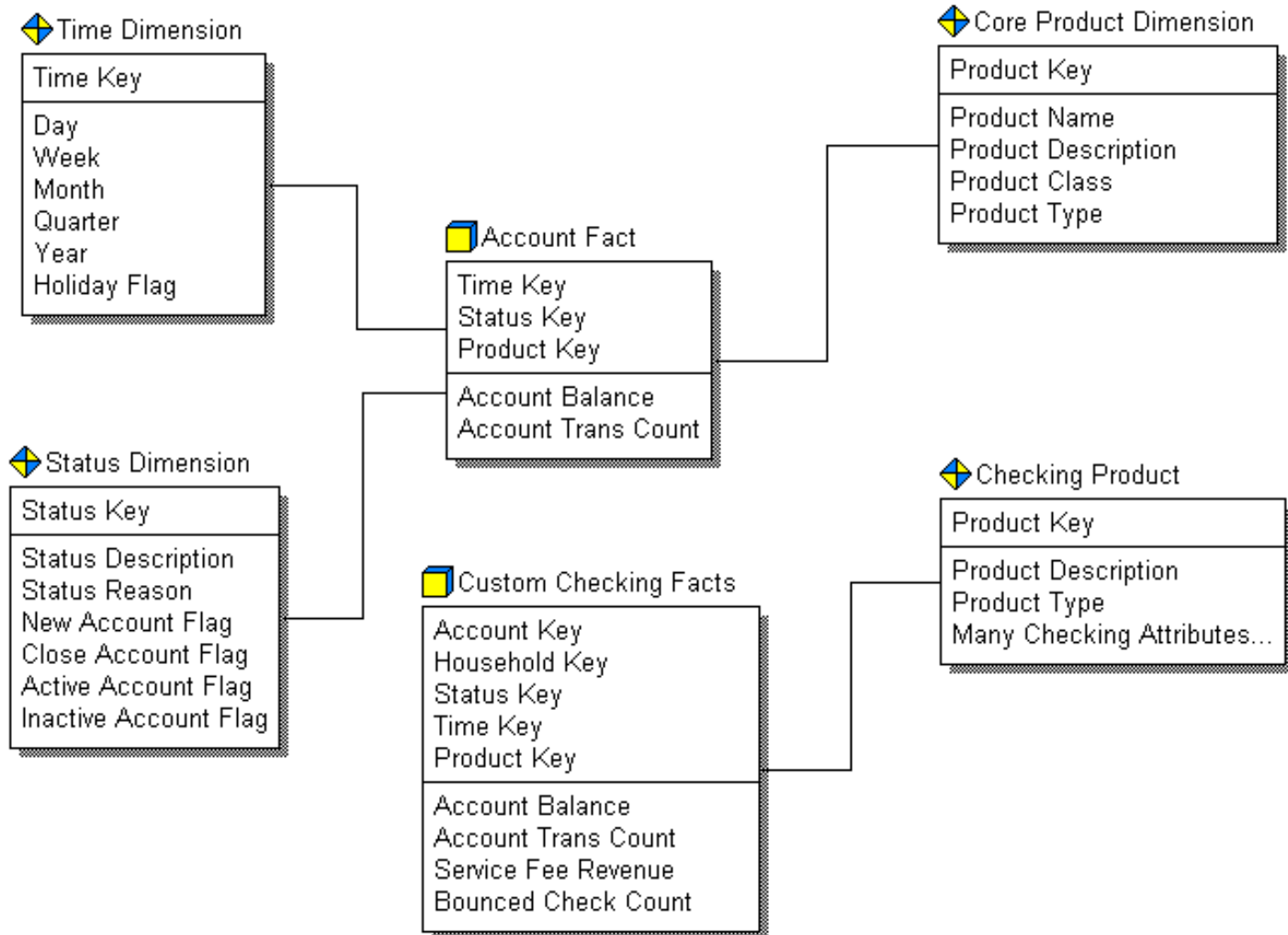
- If we have distinct attributes for each product, we can denormalize them all into one dimension.
- This increases the sparsity of the dimension because only one group of attributes will be populated for any fact row.
- Reduces the number of table joins and facts while preserving the distinctness of each product.



Heterogeneous Products - Option 2

- Another option is to maintain separate product dimensions (core dimensions) that reference separate fact tables (custom facts).
- Each core or custom dimension will contain the attributes for a specific product.
- Duplicate the core facts in every custom fact table to eliminate the need to link the two large facts together.

Heterogeneous Products - Option 2



Factless Fact Tables

FACTLESS FACT TABLES

Tables that seem like fact tables but are used to represent data or events for which there are no measured facts.

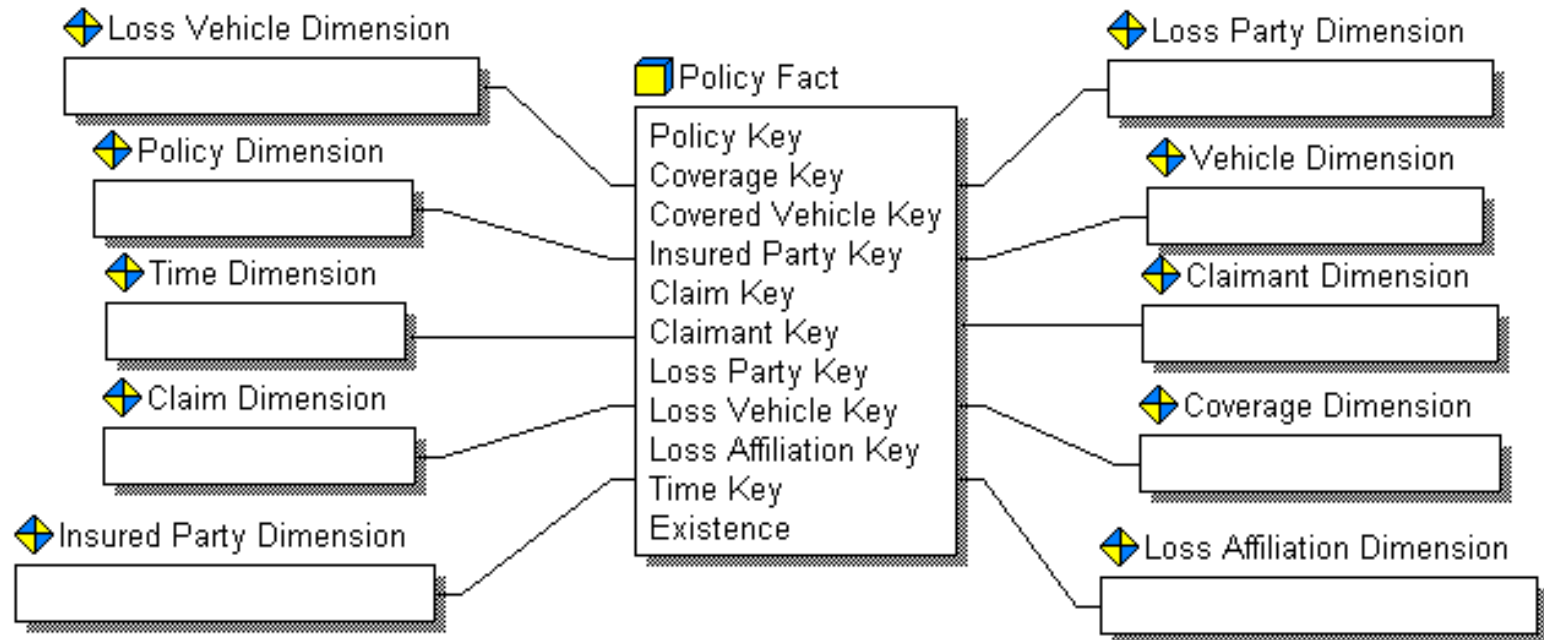
- These tables are used to track events as the simultaneous coming together of a number of dimensions.
- Two major variations: Event Tracking and Coverage tables.



Event Tracking Tables

- An insurance company needs to register all the many-to-many correlations between all the people involved in an accident (loss parties) and all the vehicles involved in an accident (loss items).
- The purpose of this schema would be to collect in one place all the miscellaneous people involved in a complex claim regardless of their role in the accident.

Example - Accident Parties Schema

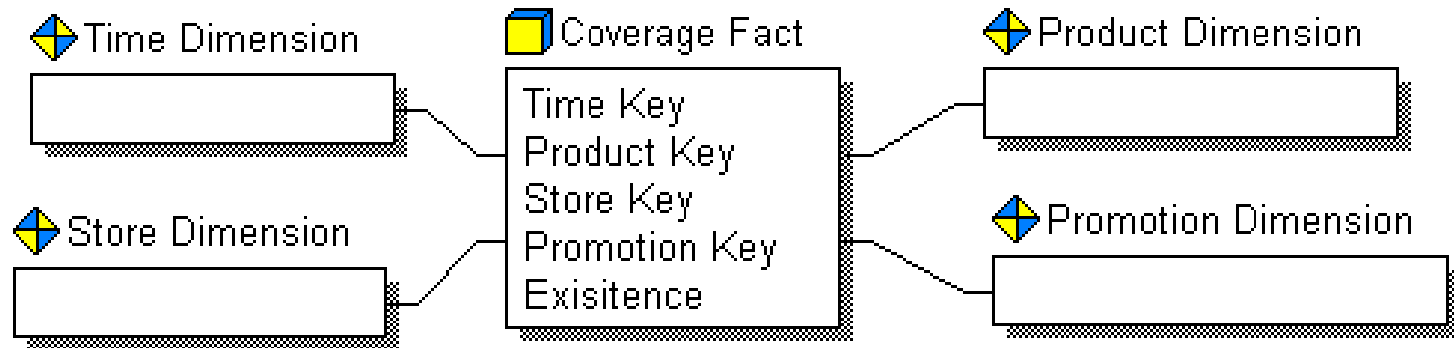


- Attribute 'existence' denotes participation in the event.
- The combination of dimensions denotes the 'occurrence' or the nature of the event and its participants.
- We can count these occurrences for analysis.

Coverage Tables

- How do we track items that were on promotion but didn't sell?
- In order to record the complex many-to-many relationships involved in promotion coverage, we must build a table to contain records of items that were on promotion in which stores at which times.
- Coverage tables are often tables of events that didn't actually happen and are factless in the same way as event tracking tables.

Example - Grocery Store Promotion



- A coverage table is an inventory snapshot table for a specially chosen subset of the inventory.
- Care should be taken that coverage table does not balloon out of control because products change promotions often.

Types of Facts

FACT

A measurement, generally additive in nature, of the organization.

- We use facts to measure performance based on business questions.
- This data is numeric in nature and is contained in our fact tables by subject and granularity.



Types of Facts

- Understanding which facts can be added across which dimensions is an important data design issue.
- Three Types of Facts:
 - ◆ Additive
 - ◆ Non-Additive
 - ◆ Semi-Additive

Additive Facts

ADDITIVE FACTS

Measurements in a fact table that can be added across all dimensions.

- Since aggregation is a key element in the usefulness of the dimensional model, its best utilized for facts that are additive, numeric values.
- We can add revenue, cost, and quantity sold for all products, all stores, and any time period.

Semi-Additive Facts

SEMI-ADDITIVE FACTS

Measurements in a fact table that can be added across some dimensions but not others.

- We cannot add risk exposure at the coverage level to get the number of policy level exposures.
- We can add coverage level exposures across the customer dimension to determine exposure by gender or age range.

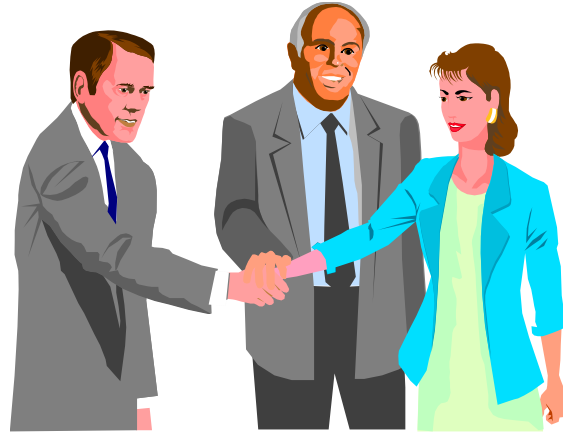
Non-Additive Facts

NON-ADDITIVE FACTS

Measurements in a fact table that cannot be added across any dimensions, like ratios.

- A new value will need to be calculated at each level for each level or for each set of data.
- It should be determined, at what levels, if any, the fact should be stored. Some values may need to be pre-calculated.

Dimensional Insurance Model



- Example Insurance Company is a property and casualty insurer for auto, fire, and personal liability.
- It has two main sources of data: all transactions relating to policy and all transactions related to claims.



Business Case

- Business users want to:
 - ◆ Examine coverage profitability over time by item type, state and county, demographics, sales broker/region, and event/catastrophe.
 - ◆ Understand the life of a policy; especially when a claim is filed and payment to a claimant or claimants is made.



Issues to Consider

- Insurance companies typically have dirty customer dimensions, which will influence our design.
- We typically carry demographical information on our customers.
- We will also be dealing with heterogeneous products.
- We will have at least two facts: Policy and Claims.
- We will model Policy first, then Claims.



Dimensional Modeling Process

Step 1: Choose the grain of each fact table.

- What level of detail do we need to determine profitability of coverage?
- What level of detail do we need to determine expense of claims?
- **Policy Fact** will have a **transaction** grain.
- **Claim Fact** will have a **transaction** grain.

Dimensional Modeling Process

Step 2: Choose the dimension attributes.

- Business requirements specify the following dimensions for **Policy**:

Insured Party Dimension

Insured Party Key
Name
Address
Type
Demographics Attributes...

Coverage Dimension

Coverage Key
Coverage Description
Market Segment
Line of Business
Annual Statement Line

Covered Item Dimension

Covered Item Key
Covered Item Description
Covered Item Type

Transaction Dimension

Transaction Key
Transaction Description
Reason

Time Dimension

Date Key
Day of Week
Fiscal Period

Policy Dimension

Policy Key
Risk Grade

Employee Dimension

Employee Key
Name
Employee Type
Department

Dimensional Modeling Process

Step 2: Choose the dimension attributes.

➤ Business requirements specify the following dimensions for **Claim**:

Policy Dimension

Policy Key

Risk Grade

Time Dimension

Date Key

Day of Week

Fiscal Period

Employee Dimension

Employee Key

Name

Employee Type

Department

Claim Dimension*

Claim Key

Claim Description

Claim Type

Claimant Dimension*

Claimant Key

Claimant Name

Claimant Address

Claimant Type

Insured Party Dimension

Insured Party Key

Name

Address

Type

Demographics Attributes...

Third Party Dimension*

Third Party Key

Third Party Name

Third Party Address

Third Party Type

Coverage Dimension

Coverage Key

Coverage Description

Market Segment

Line of Business

Annual Statement Line

Covered Item Dimension

Covered Item Key

Covered Item Description

Covered Item Type



Dimensional Modeling Process

Step 3: Identify dimensional hierarchies.

- Time: Days/Months/Quarters/Years
- Coverage: Annual Statement Line/Line of Business/Market Segment
- Any other hierarchies will need to be defined by the business community so those attributes can be added to the dimensions.

Dimensional Modeling Process

Step 4: Choose the dimensions that apply to each fact table.

 Policy Fact

Date Key
Employee Key
Covered Item Key
Transaction Key
Insured Party Key
Coverage Key
Policy Key

Amount

 Claim Fact

Date Key
Employee Key
Covered Item Key
Claimant Key
Third Party Key
Insured Party Key
Coverage Key
Policy Key
Claim Key
Transaction Key

Amount



Reality Check

We'll pause here before we do steps 5 and 6 to examine our schema thus far.

- The two schemas we have developed are very useful for answering a whole range of questions.
- However, the volume of transactions makes it difficult to quickly determine the status of a policy or claim at any given point in time.

Should we change our granularity?

- We need to answer some questions at the transaction level, however some other level of aggregation would be best approach to view some aspects of data.
- We create a 'snapshot' schema for both the policy and claim facts.
- This means that every month we roll forward all transactions and take a monthly snapshot of the data to be used as a base for incremental updates at a monthly level.

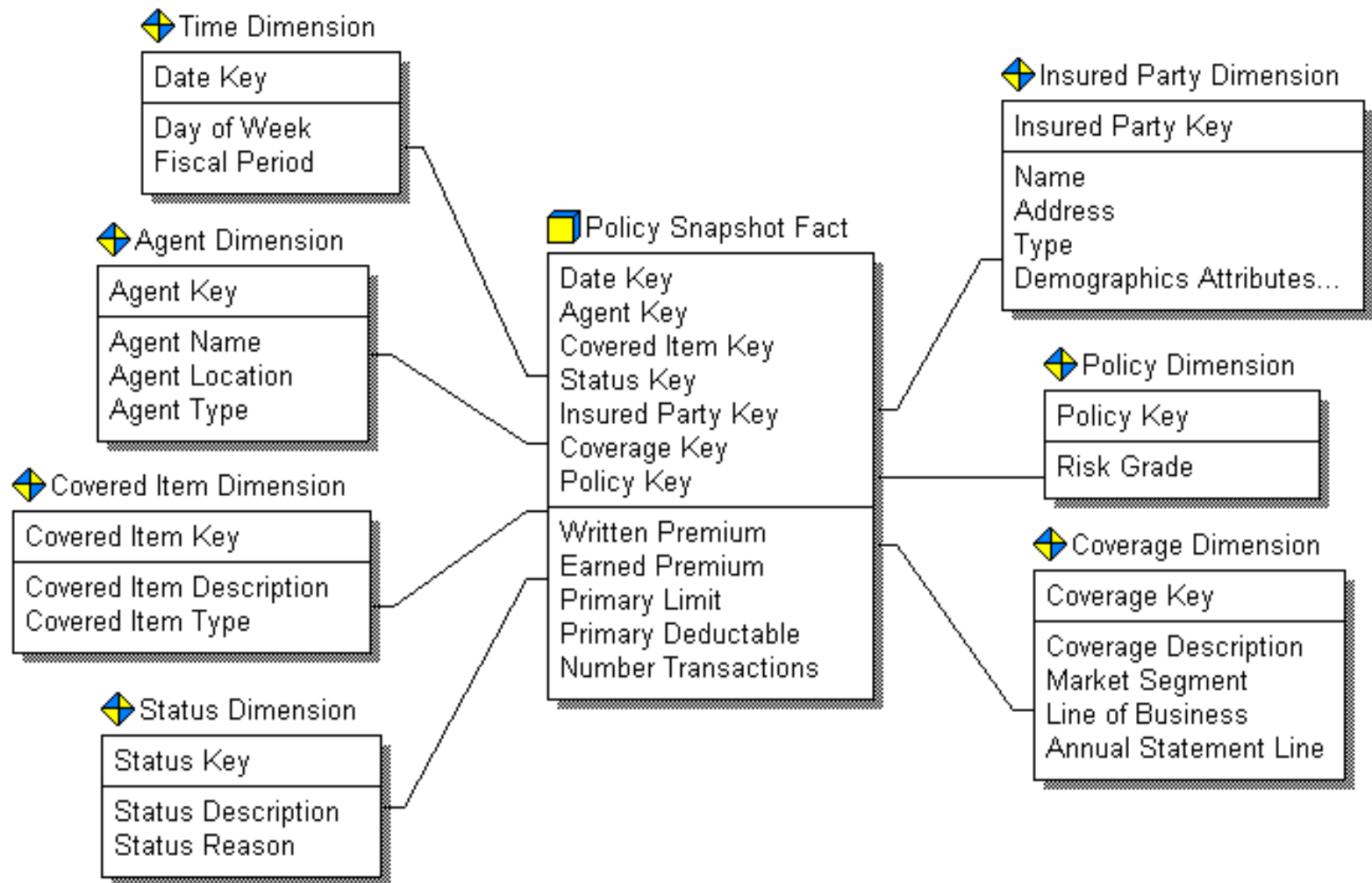
Snapshot Fact Tables

SNAPSHOT FACTS

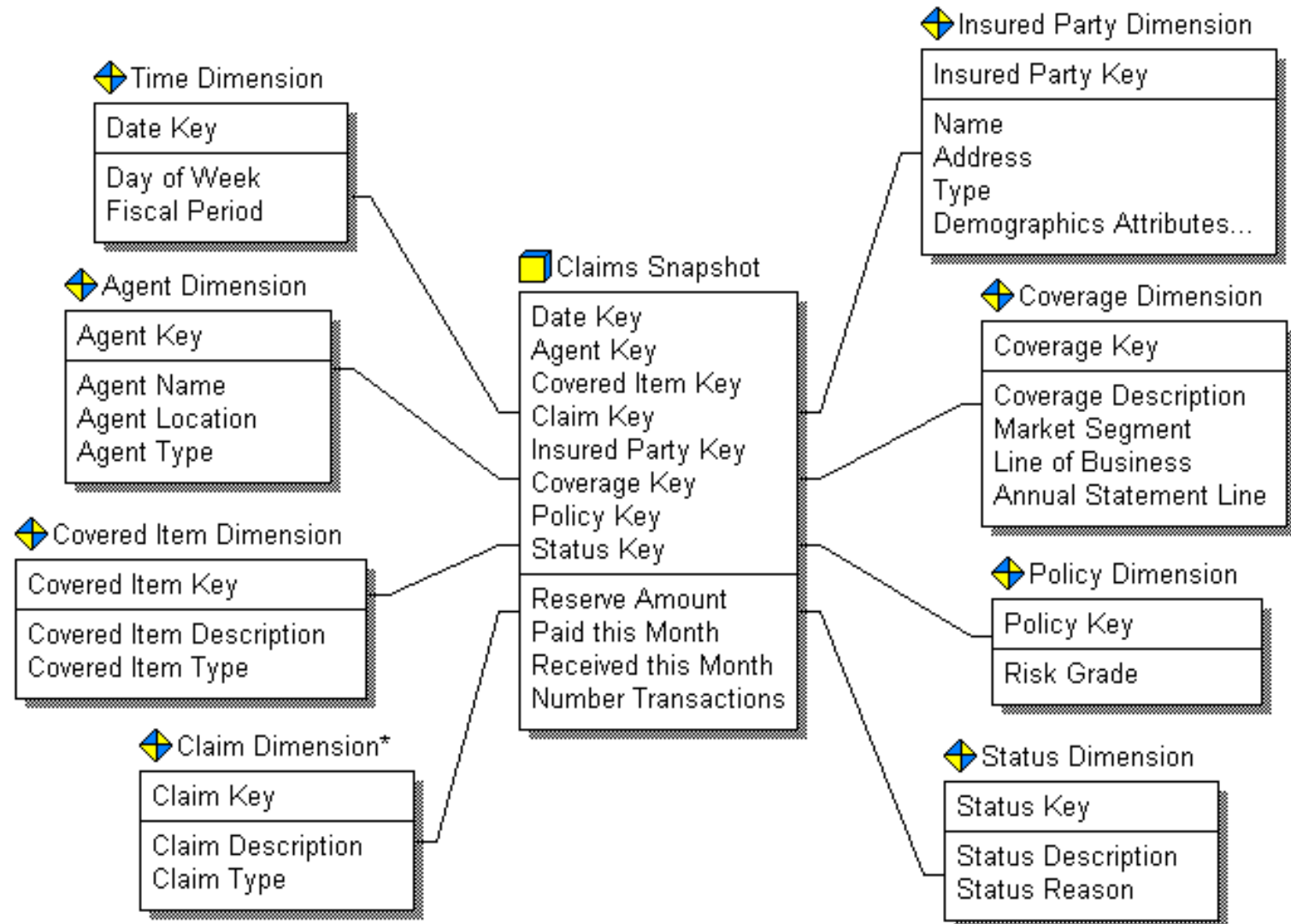
Tables created to handle pay-in-advance or multiple transaction data in a more aggregated manner.

- Snapshot fact tables for policies and claims are systematic derivations from the transaction fact tables.
- Time becomes month.
- Transaction specific dimensions like employee, claimant, third party, and transaction are suppressed.
- Status dimension is introduced to call attention to coverages or claims in specific months.
- Amount is replaced with a long list of summary facts.

Policy Snapshot Fact



Claim Snapshot Fact



Heterogeneous Products

- Remember the two options?
 - ◆ Core Dimension/Fact
 - ◆ Disparate attributes within one table
- Determine how many attributes and what the tool limitations are.
- Each case may differ.



Dimensional Modeling Process

Step 5: Choose the measured facts, including precalculated facts.

- Each aggregated and derived fact will need to be evaluated for inclusion in the model or calculation in the application.
- Trade-offs include storage and indexing and must be weighed against the access requirements.



Dimensional Modeling Process

Step 6: Determine slowly changing dimensions

- These are the dimensions that change over time.
- If tracking these changes is important, the method must be decided.
- Options: overwrite the existing record, store all records with effective dates, or a historical and current value tables.



Insurance Model - Final Tips

- Until you have a stable schema, your model will change as data inconsistencies are uncovered.
- While insurance is relatively static, the ways that business users want to look at data may change.
- Keep the future in mind and remember to not make tool-specific structures part of the atomic or summary layers.



That's it?

Questions, Comments, or Other?



Thanks!