Advanced Databases

Lecture 6: Dimensional Modelling, Conversion from ER to Star Schema

Dr. Pierpaolo Dondio,

Topics

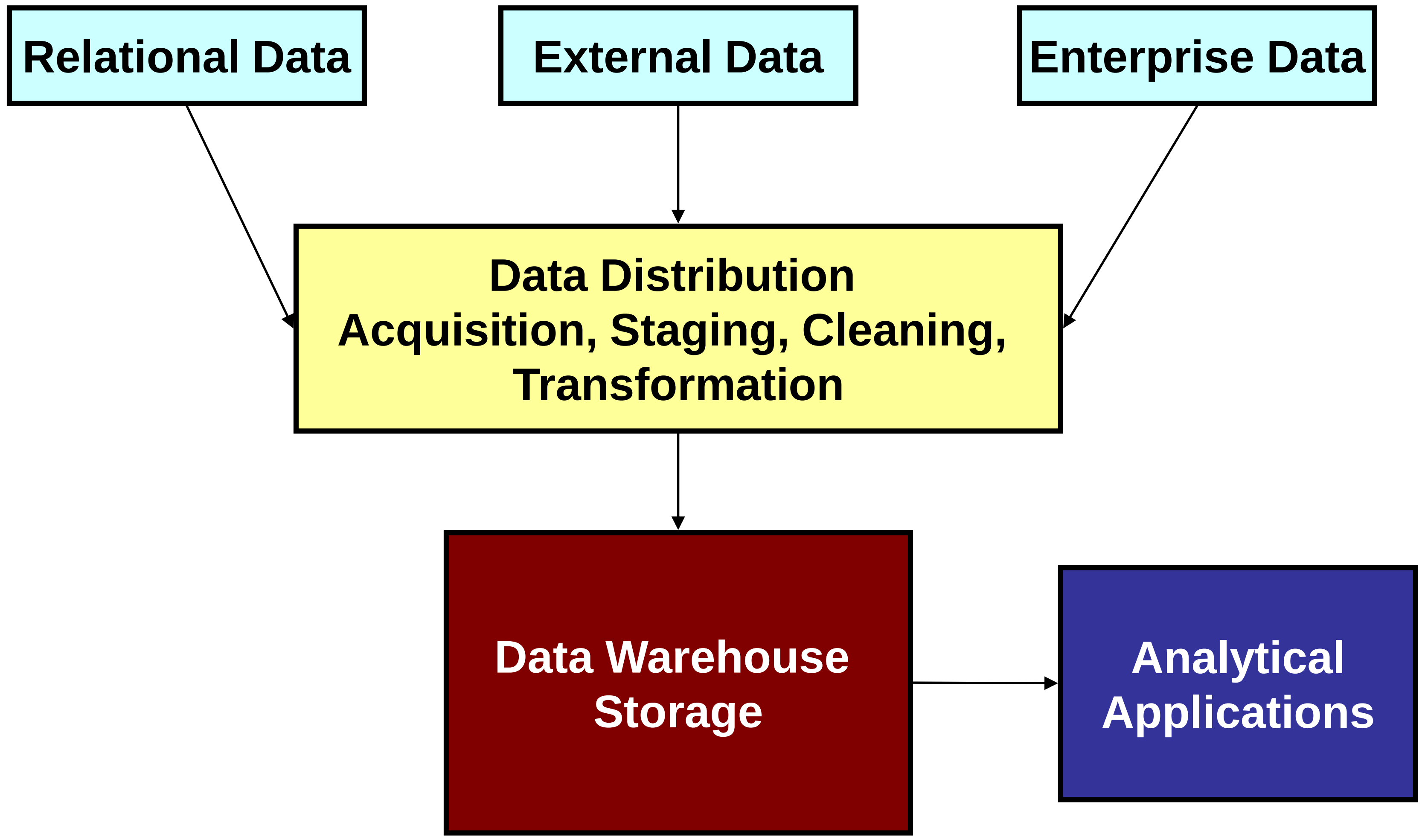
- Dimensional Modeling
- Differences with ER
- ER / DM Conversion
- Examples
- Hierarchical Dimensions

1. Introduction to Dimensional Modeling

Why Dimensional Data Warehouses?

- Business needs to analyze data so that it can:
  - Understand trends
  - Predict future behavior and needs
  - Personalize contact with customers
  - Be competitive
- All of this in a speedy manner, with the ability to do “What if’s”

Dimensional Data Warehouse Architecture



Some Key points

- The Datawarehouse is a database (and it can be built using a relational DB like Oralce)
- It is not a live / day-by-day database
- Dab-by day transactions go to another database(s), usually fully relational databases fully normalized
- A DW:
  - s updated at specific points in time
  - is mainly read-only (analytics)
  - is optimized for (read) performances
  - is a collection (integration) of different sources
  - The “yellow” box (= the staging area) is permanent and it is where data are clean and integrated

Some Definitions..

Data Warehouse: a definition

- "a subject-oriented, integrated, time-varying, non-volatile collection of data that is used primarily in organizational decision making" (Inmon, 2005)

- "a collection of decision support technologies, aimed at enabling the *knowledge worker* (executive, manager, analyst) to make better and faster decisions" (Chaudhuri, 1997)

- "a specifically prepared repository of data created to support decision making" (Wixom, 2001)

Definition – continued…

Definition

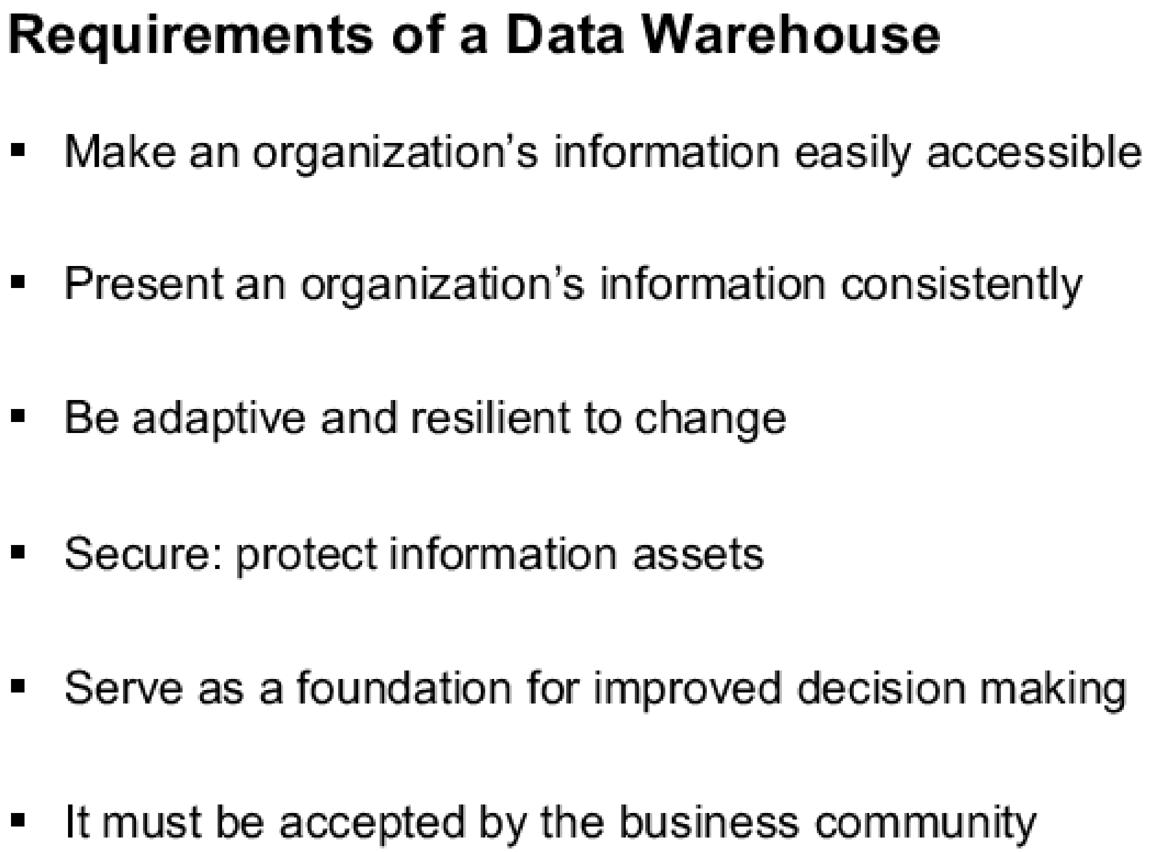
- Subject Oriented
  - Data is organized around the major subjects of the enterprise (such as customers, products, sales etc.) rather than the major application areas (invoicing, stock control etc.).
  - The aim is to store decision-support data rather than application-oriented data.

- Integrated
  - Data from different source systems comes together
  - Data may be inconsistent and presented in different formats
  - The integrated data source must be made consistent

Definition – continued…

- Time Variant
  - Data in the warehouse is only accurate and valid at some point in time or over some time interval
  - The time-variance of the data warehouse is also shown in the extended time that the data is held
  - The implicit or explicit association of time with all data, and the fact that the data represents a series of snapshots
- Non-Volatile
  - Data is not updated in real-time but is refreshed from operational systems on a regular basis
  - New data is always added as a supplement to the database rather than a replacement
  - The database continually absorbs the new data, incrementally integrating it with the previous data

DW Requirements



Requirements of a Data Warehouse

- Make an organization's information easily accessible
- Present an organization's information consistently
- Be adaptive and resilient to change
- Secure: protect information assets
- Serve as a foundation for improved decision making
- It must be accepted by the business community

Kimball 2002

DW Requirements

Requirements: accessibility

- Understandable – legible, i.e. meaningfully labeled
- Intuitive and obvious to the business user – not just developers
- Well-designed tools that are simple and easy to use in accessing data
- Tractable: minimal wait times on data operations

DW Requirements

Requirements: consistency

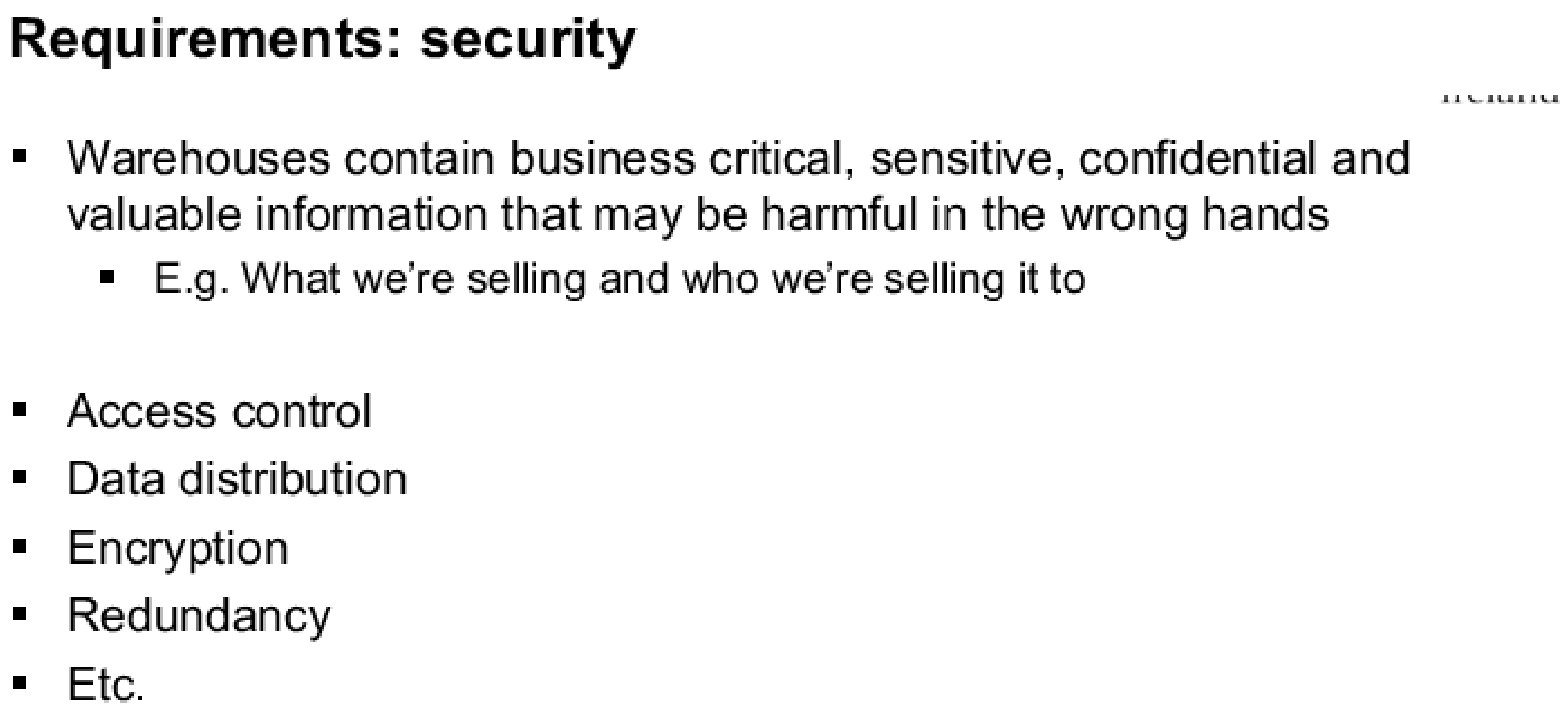
- Credible data; multiple sources of data leads to inconsistencies – data must be clean, and quality assured
- Cross Business Process Compatibility: a spade is always a spade, otherwise it should be labeled differently
- Common definitions are available for end users
- Consistent information is high quality information, that is accounted for and complete

DW Requirements

Requirements: adaptive and resilient

- Tolerant to inevitable (business) changes
- Warehouse changes should be graceful, and not invalidate existing data or applications
- New use cases or business cases should not disrupt existing applications
- If changes to descriptive data cannot be avoided appropriate measures must be in place to account for such changes

DW Requirements



Requirements: security

- Warehouses contain business critical, sensitive, confidential and valuable information that may be harmful in the wrong hands
  - E.g. What we're selling and who we're selling it to

- Access control
- Data distribution
- Encryption
- Redundancy
- Etc.

DW Requirements

Requirements: improved decision making

- Have the right data, visualization, and analytical tools
- There is only 1 true output: the decision made after viewing evidence from a data warehouse
- Decisions should deliver business impact and value
- Decision **support** system

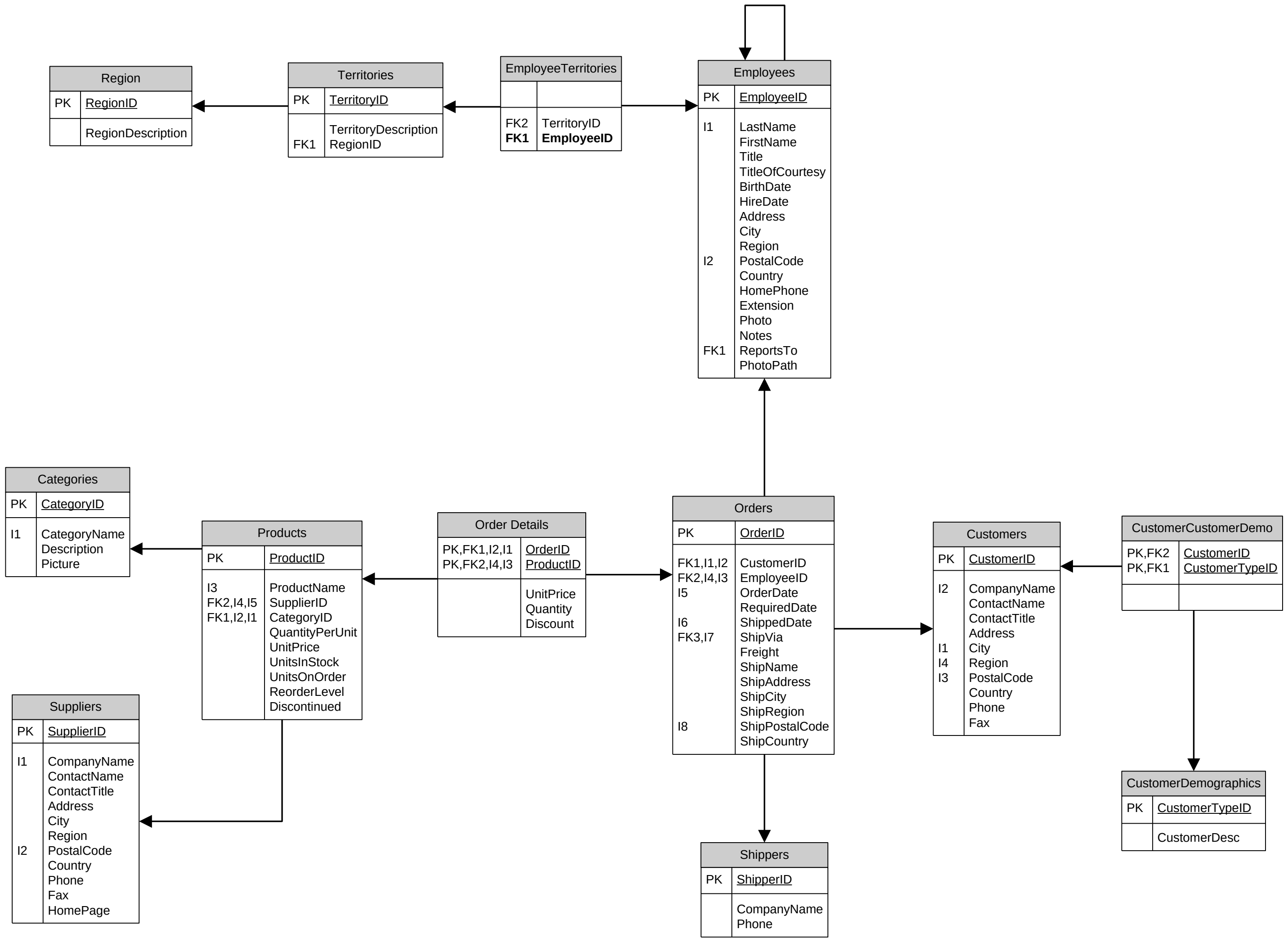
DW Requirements

Requirements: acceptability

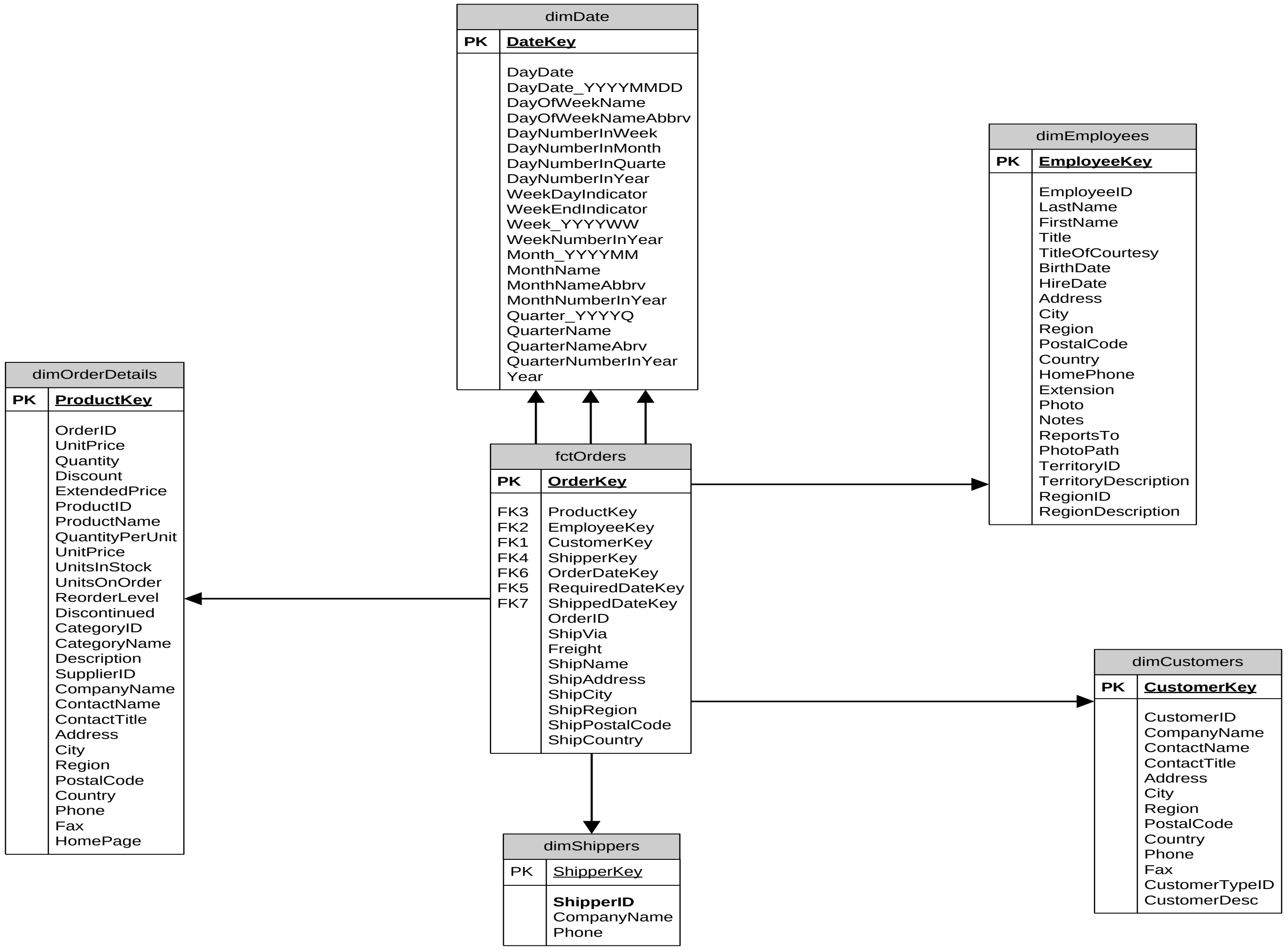
- Success is unequivocally linked to user acceptance
- Acceptance is not only at the user level, but senior management must believe in and drive the technology
  - Data warehousing is often seen as an optional extra to daily business
- Users will not accept or rely on the data if they do not trust it
- Tools must be intuitive – 6 months training is not an option

What is a Dimensional Model?

Northwind Database Model – Relational Format



Northwind Database – Star Schema – Orders

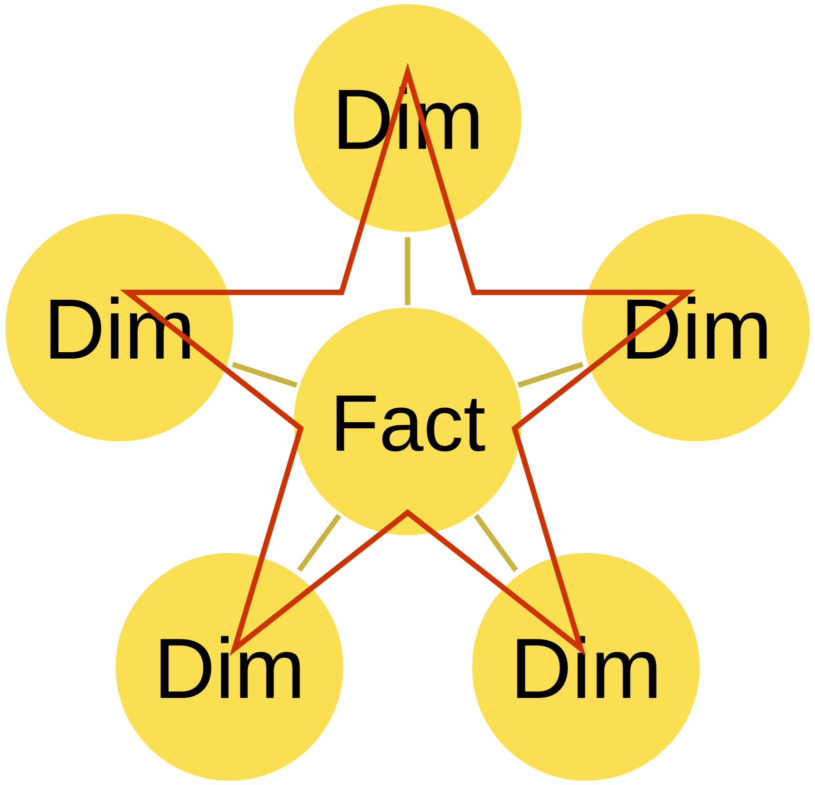


What is Dimensional Modeling?

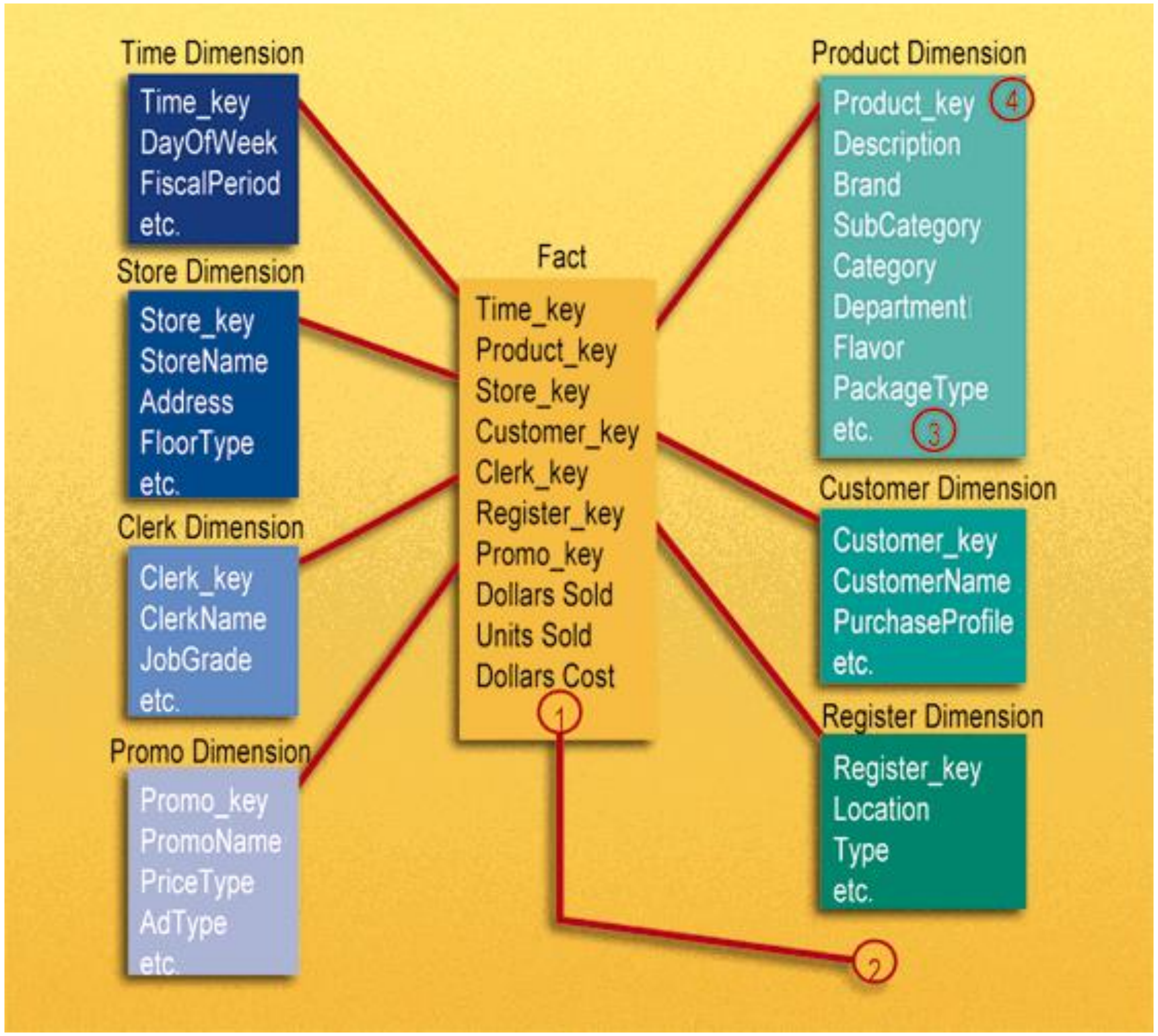
- DM is a logical design technique that seeks to present the data in a standard, intuitive framework that allows for high-performance access.
- Can be implemented using a relational or a DBMS
- Every dimensional model is composed of one table with a multipart key, called the fact table, and a set of smaller tables called dimension tables.
- Each dimension table has a single-part primary key that corresponds exactly to one of the components of the multipart key in the fact table.
- This characteristic "star-like" structure is often called a star join. The term star join dates back to the earliest days of relational databases.

Star Schema

- Singe data (fact) table surrounded by multiple descriptive (dimension) tables



Dimensional Model Example



Dimensional Schema

- Fact Tables
  - contain related measures
  - Usually the largest tables
  - Usually appended to
  - Can contain detail or summary data
  - Measures are usually additive
- Dimension Tables
  - Contain descriptors
  - Utilize business terminology
  - Textual and discrete data
  - Attributes through which the table measures are analyzed

Dimensional Model: Fact Tables

- A fact table contains information about things that an organization wants to measure.
- A fact table's key is made up from the keys of two or more parents.
- A fact always 'resolves' a many-to-many relationship between the parent, or dimension tables.
- The most useful fact tables also contain one or more numerical measures, or facts, that occur for the combination of keys that define each record.
- Example: the facts are Dollars Sold, Units Sold, and Dollars Cost.

Dimensional Model: Fact Tables

- » The most useful facts in a fact table are numeric and additive.
- » Additivity is crucial because data warehouse applications almost never retrieve a single fact table record; rather, they fetch back hundreds, thousands, or even millions of these records at a time, and often the most useful thing to do with so many records is to add them up.

Dimensional Model: Dimension Tables

- » Dimension tables contain information about how an organization wants to analyze facts:
  - » “Show me sales revenue (fact) for last week (time) for blue cups (product) in the western region (geography)
- » Dimension tables most often contain descriptive textual information ‘Blue cups’, ‘Western Region’
- » Dimension attributes are used as the source of most of the interesting ‘constraints’ in data warehouse queries., and they are virtually always the source of the row headers in the SQL answer set.

Dimensions vs Facts

Dimensions

- The time independent, textual and descriptive attributes by which users describe objects.
- Combining all the attributes including hierarchies, rollups and sub-references into a single dimension is denormalization.
- Often the “by” word in a query or report
- Not time dependent

Facts

- Business Measurements
- Most Facts are Numeric
- Additive, Semi-Additive, Non-Additive
- Built from the lowest level of detail (grain)
- Very Efficient
- Time dependent

Benefits

- » Performance (Integer relationships, natural partitioning, Single joins benefit SQL optimizer)
- » Supports Change management
- » Usability/Simplicity (easy to read, interpret, join, calculate)
- » Presentation (Consistency, Taxonomy, Labeling)
- » Reuse (Conformed dimensions reduce redundancy, Role-plays)

Example

- We need to build a DW to investigate the effect of news and user-generated content (Twitter) sentiment on the stock market.
- What kind of data do we need?
- Decide the GRAIN
- Where are the data?
- What kind of transformation do I need to do? What kind of Cleaning?

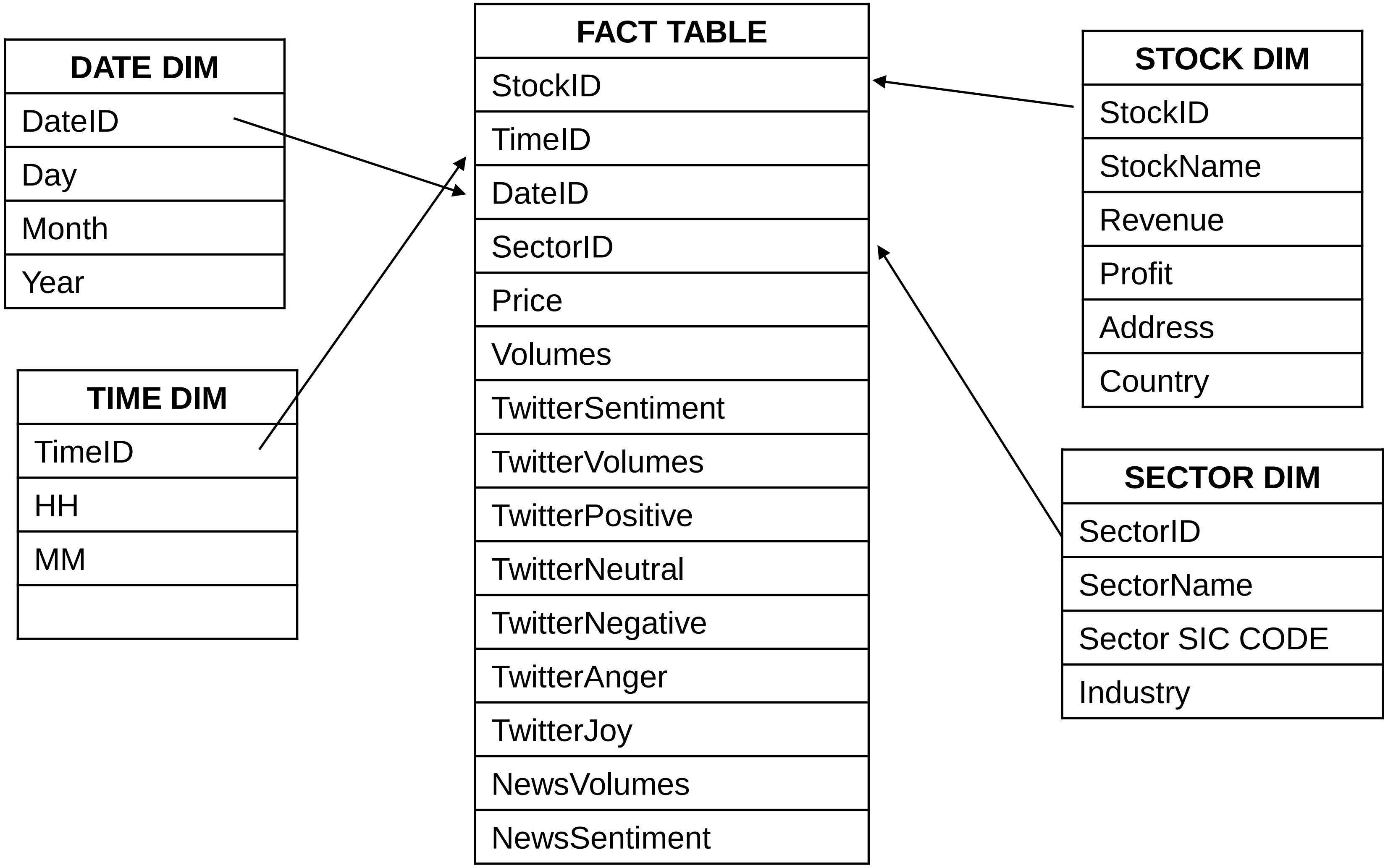
Data Sources

- Stock Market Data from Yahoo Finance (csv) or from Bloombeg API (json)
- Twitter data from twitterscraper
- News from Reuters RSS feed and Yahoo Finance RSS feed

Staging Area – Cleaning and Transformation

- Stock market price – should be fine, check for dividends and splits adjustments. Include Volumes
- Twitter data. The text of the tweets does not go into the DM! I need to process the tweets and apply a sentiment library (like textblob) to get the sentiment and subjectivity of each tweet
- I can add IBM Watson Tone Analyzer dimensions as well (anger, fear, joy, analytical level…)
- Assign News to stocks (entity recognition, keywords matching….)
- Assign Surrogate Keys

The DataWarehouse - DM



DATE DIM
DateID
Day
Month
Year

TIME DIM
TimeID
HH
MM

FACT TABLE
StockID
TimeID
DateID
SectorID
Price
Volumes
TwitterSentiment
TwitterVolumes
TwitterPositive
TwitterNeutral
TwitterNegative
TwitterAnger
TwitterJoy
NewsVolumes
NewsSentiment

STOCK DIM
StockID
StockName
Revenue
Profit
Address
Country

SECTOR DIM
SectorID
SectorName
Sector SIC CODE
Industry

Entity Relationship (ER)

Vs.

Dimensional Modeling (DM)

Entity Relationship Modeling: Review

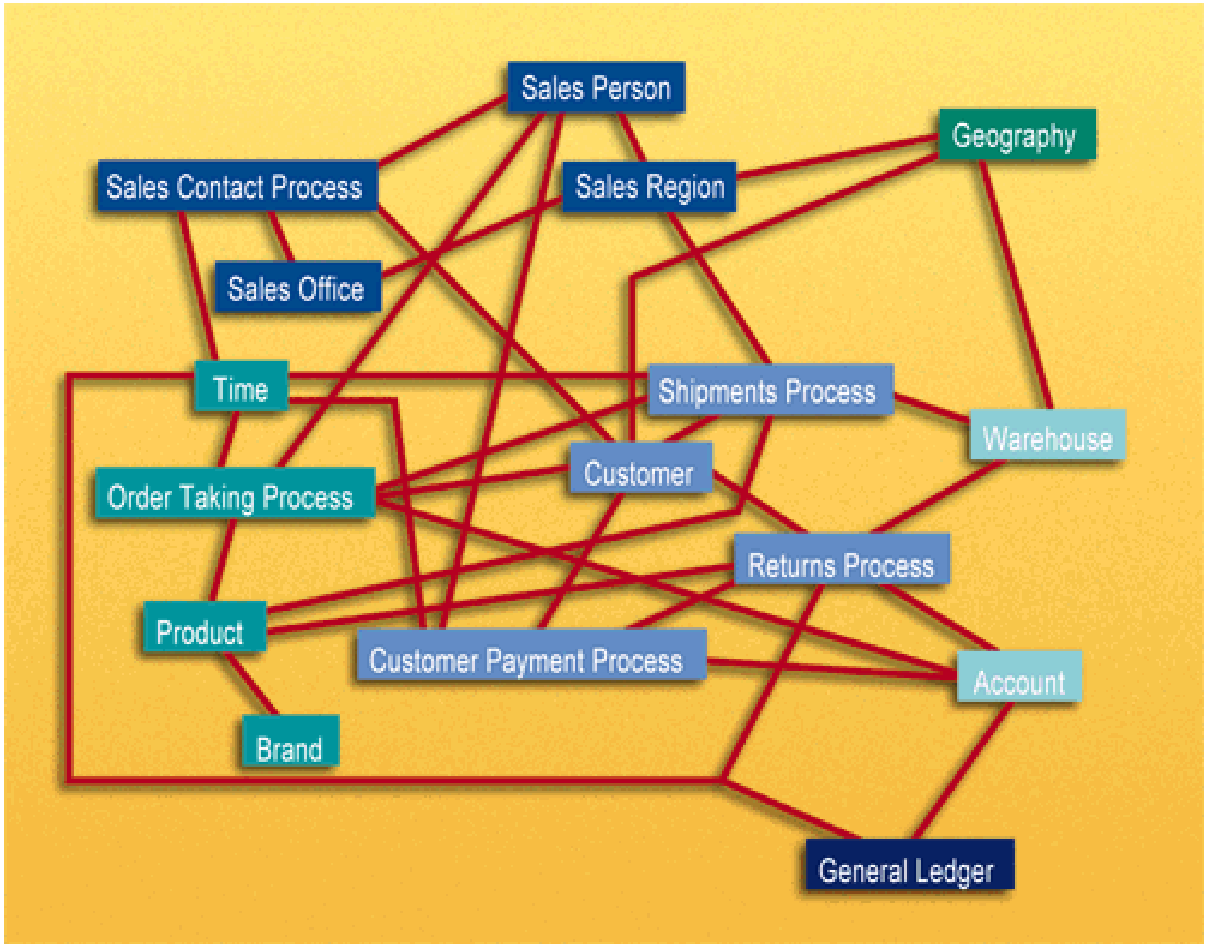
- Entity Relationship modeling is a technique used to ‘abstract’ user’s data requirements into a model that can be analyzed and ultimately implemented.
- The focus of ER modeling:
  - achieve *processing* and *data storage* efficiency by reducing data redundancy (storing data elements once)
  - provide flexibility and ease of maintenance
  - protect the integrity of data by storing it once
- ER modeling and normalization great for transaction processing as it makes transactions as simple as possible (as data stored only in one place)

Relational Normal Form

» Most relational databases are set to 3rd normal form

- ✓ 1st Normal form – Tables have unique keys and no repeating groups or multi-value fields
- ✓ 2nd Normal form – Every attribute is dependent ont the entire key of the table
- ✓ 3rd Normal form – Attributes are dependent only on the key. No derived elements

ER Model Example



Normalized databases become very complex making queries difficult and inefficient – a ‘spiderweb of joins’ is required for many queries. A database normalized for transaction processing is typically unusable for non-technical users who wish to perform queries

Drawbacks to Relational Data Structures

» Data is not structured for analytical usage

» Multiple Joins are resource intensive

» Missing data from external sources, context history, not operational sources

ER model issues

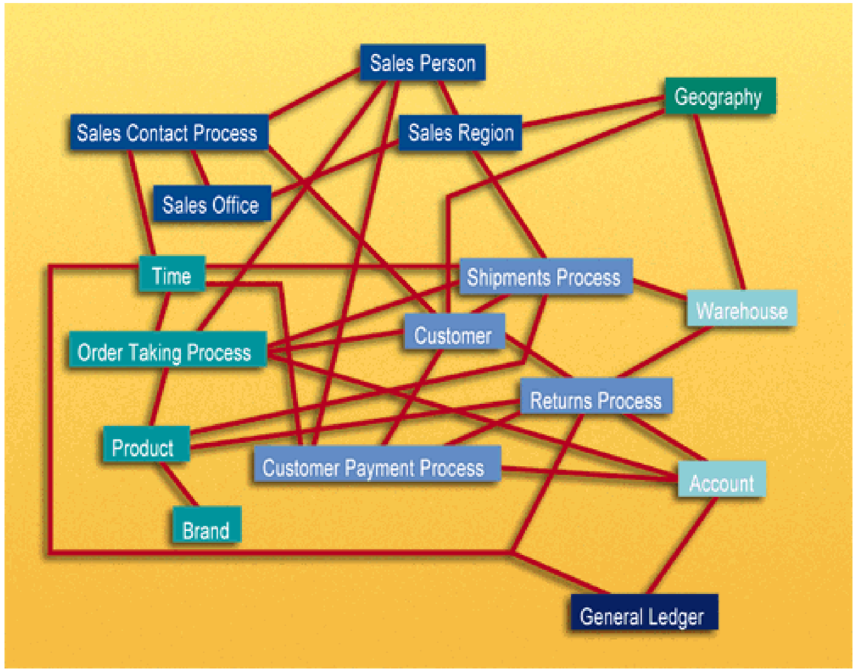
- End users cannot understand or remember an ER model. End users cannot navigate an ER model. There is no graphical user interface (GUI) that takes a general ER model and makes it usable by end users.
- Software cannot usefully query a general ER model. Cost-based optimizers that attempt to do this are notorious for making the wrong choices, with disastrous consequences for performance.
- Use of the ER modeling technique defeats the basic allure of data warehousing, namely intuitive and high-performance retrieval of data.
- The solution: the Dimensional Data Model

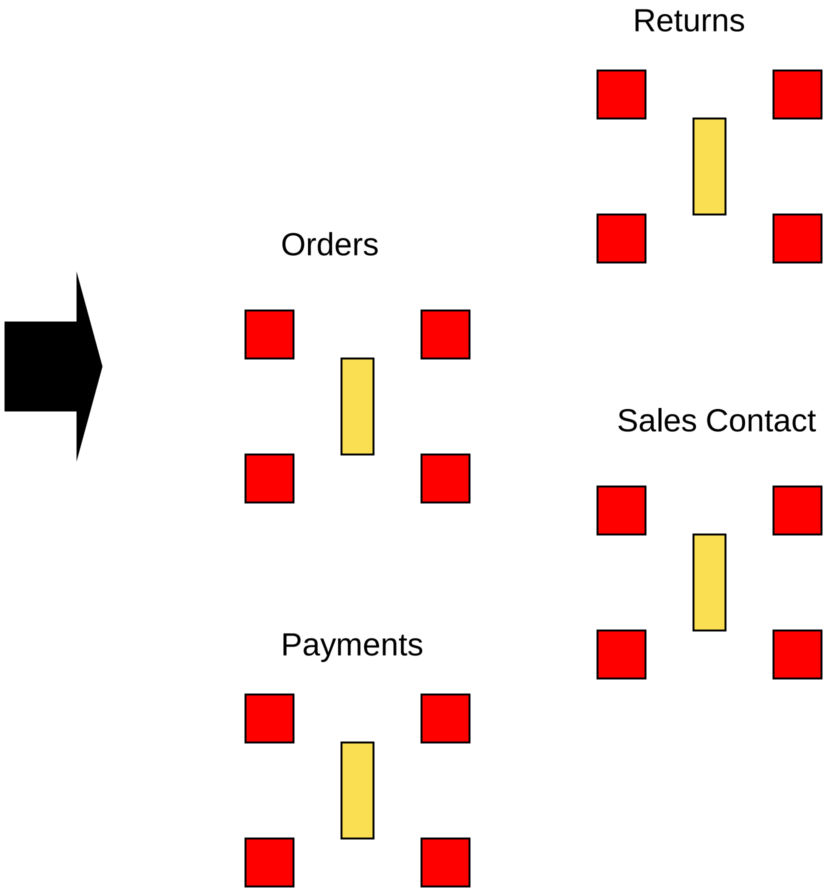
Dimensional Model vs ER model

» The key to understanding the relationship between DM and ER is that a single ER diagram breaks down into multiple DM diagrams, or ‘stars’.

» Think of a large ER diagram as representing every possible business process within an application. The ER diagram may have Sales Calls, Order Entries, Shipment Invoices, Customer Payments, and Product Returns, all on the same diagram.

Dimensional Model vs ER model





Dimensional Model vs ER model

- To create the individual ‘stars’ that exist within an application:
  - Look for many-to-many relationships in the ER model containing numeric and additive facts and to designate them as fact tables.
  - Alternatively, look for ‘events’ or ‘transactions’ – these may also be facts
  - Denormalize all of the remaining tables into flat tables with single-part keys that connect directly to the fact tables. These tables become the dimension tables.
  - In cases where a dimension table connects to more than one fact table, we represent this same dimension table in both schemas, and we refer to the dimension tables as "conformed" between the two dimensional models.

DM Strengths

- The dimensional model has a number of important data warehouse advantages that the ER model lacks.
  - First, the dimensional model is a predictable, standard framework. Report writers, query tools, and user interfaces can all make strong assumptions about the dimensional model to make the user interfaces more understandable and to make processing more efficient.
  - Rather than using a cost-based optimizer, a database engine can make very strong assumptions about first constraining the dimension tables and then "attacking" the fact table all at once with the Cartesian product of those dimension table keys satisfying the user's constraints.

DM Strengths

» A second strength of the dimensional model is that the predictable framework of the star join schema withstands unexpected changes in user behavior. Every dimension is equivalent. All dimensions can be thought of as symmetrically equal entry points into the fact table. The logical design can be done **independent of expected query patterns**. The user interfaces are symmetrical, the query strategies are symmetrical, and the SQL generated against the dimensional model is symmetrical.

DM Strengths

- A third strength of the dimensional model is that it is ‘gracefully extensible’ to accommodate unexpected new data elements and new design decisions.
- Gracefully extensible:
  - all existing tables (both fact and dimension) can be changed in place by simply adding new data rows in the table, or the table can be changed in place with a SQL alter table command.
  - Data should not have to be reloaded.
  - No query tool or reporting tool needs to be reprogrammed to accommodate the change.
  - Old applications continue to run without yielding different results. Adding new unanticipated facts (that is, new additive numeric fields in the fact table), as long as they are consistent with the fundamental grain of the existing fact table

DM Strengths

- A fourth strength of the dimensional model is that there is a body of standard approaches for handling common modeling situations in the business world. These modeling situations include:
  - Slowly changing dimensions, where a "constant" dimension such as Product or Customer actually evolves slowly and asynchronously.

DM Strengths

- » A final strength of the dimensional model is the management of aggregates.
- » Aggregates are summary records that are logically redundant with base data already in the data warehouse, but they are used to enhance query performance.
- » A comprehensive aggregate strategy is required in every medium- and large-sized data warehouse implementation.
- » All of the aggregate management software packages and aggregate navigation utilities depend on a very specific single structure of fact and dimension tables that is absolutely dependent on the dimensional model.

ER vs DM – Final Points

» ER models are not appropriate for Data Warehouses. ER modeling does not really model a business; rather, it models the micro relationships among data elements.

» ER models are wildly variable in structure. As such, it is extremely difficult to optimize query performance.

Dimensional Modeling

Conversion

From ER Diagram to Dimensonal Model

Modeling Design Process

1. Identify the Business Process
   - » Source of “measurements”
2. Identify the Grain
   - » What does 1 row in the fact table represent or mean?
3. Identify the Dimensions
   - » Descriptive context, true to the grain
4. Identify the Facts
   - » Numeric additive measurements, true to the grain

Step 1 - Identify the B. Process

- » This is a business activity typically tied to a source system.
- » Not to be confused with a business department or function. An Orders dimensional model should support the activities of both Sales and Marketing.
- » “If we establish departmentally bound dimensional models, we’ll inevitably duplicate data with different labels and terminology.”

Step 2 - Identify the Grain

- The level of detail associated with the fact table measurements.
- A critical step necessary before steps 3 and 4.
- Preferably it should be at the most atomic level possible.
- “How do you describe a single row in the fact table?”

Step 3 - Identify the Dimensions

» The list of all the discrete, text-like attributes that emanate from the fact table.

» They are the "by" words used to describe the requirements.

» Each dimension could be though of as an analytical "entry point" to the facts.

» "How do business people describe the data that results from the business process?"

Step 4 - Identify the Facts

- Must be true to the grain defined in step 2.
- Typical facts are numeric additive figures.
- Facts that belong to a different grain belong in a separate fact table.
- Facts are determined by answering the question, “What are we measuring?”
- Percentages and ratios, such as gross margin, are non-additive. The numerator and denominator should be stored in the fact table.

Identify BP

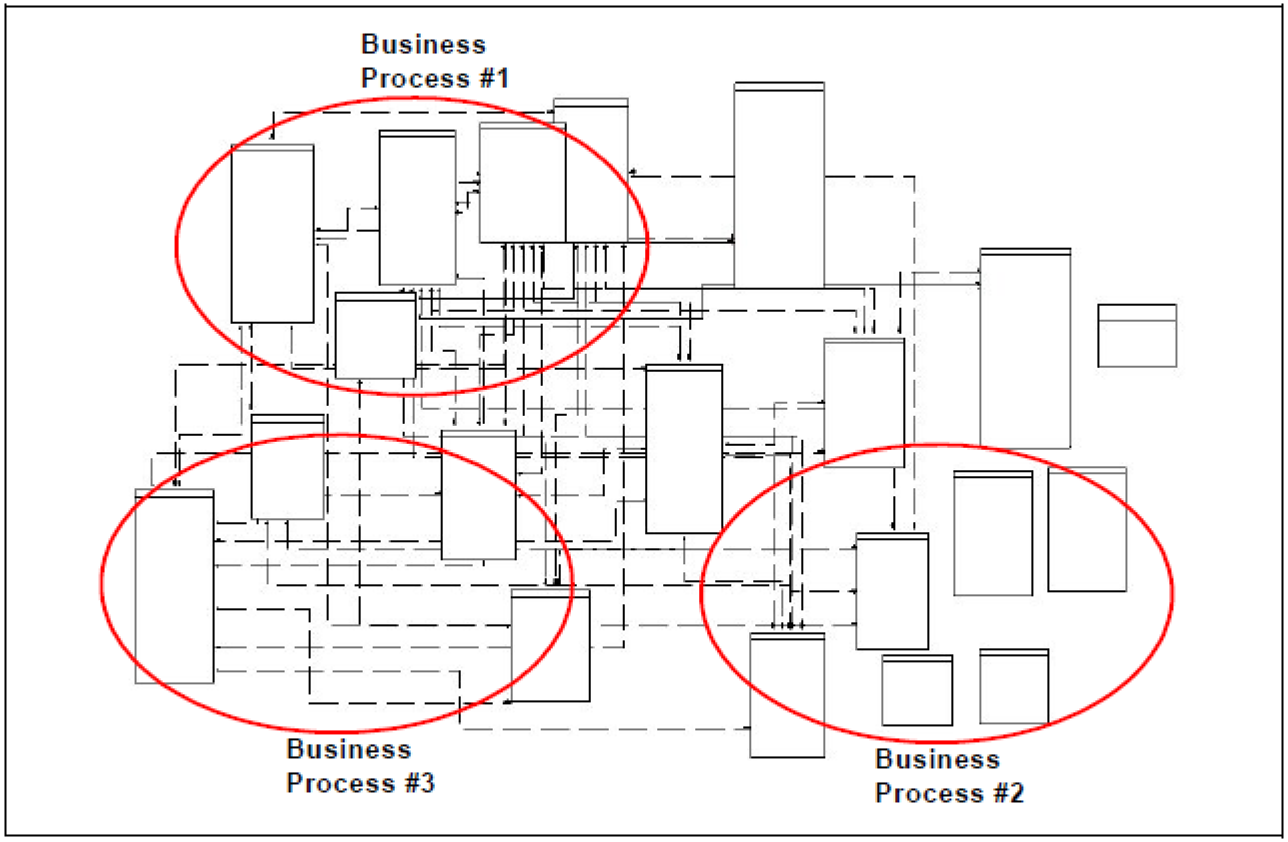


Figure 5-1 E/R model consists of several business processes

Many-to-Many

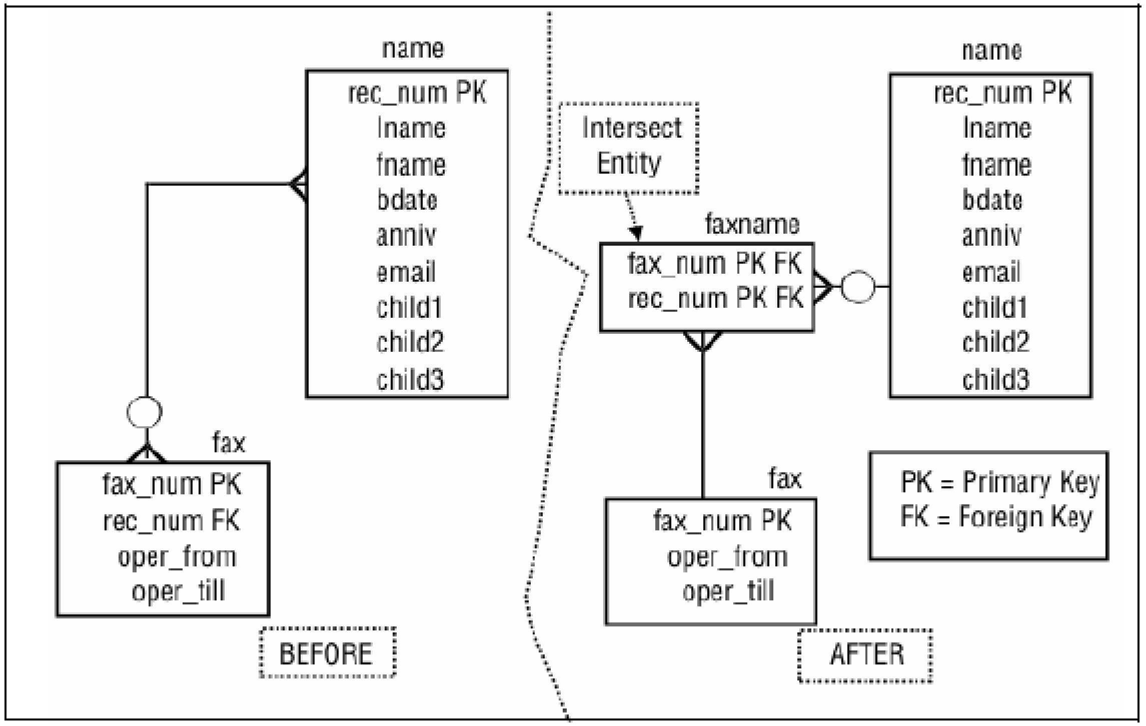


Figure 5-3 Many-to-many relationship

» They represent the link between dimensions
» They usually help to identify transactional tables

Transaction Tables

- The idea behind this step is to identify the transaction-based tables that serve to express many-to-many relationships inside an E/R model.
- Every E/R model consists of transaction-based tables which constantly have data inserted, or are updated with data, or have data deleted from them.
- For example, in an ERP database, there are transaction tables, such as Invoice and Invoice_Details, which are constantly inserted and updated because they are transaction-based tables.
- Tables such as Employee and Products in an E/R model may be fairly static.

Transactional / Non Transactional

What are a transaction-based table?

- » generally involved in storing facts and measures about the business.
- » They generally store foreign keys and facts, such as quantity, sales price,
- » profit, unit price, and discount.
- » In transaction tables, records are usually inserted, updated, and deleted as and when the transactions occur.
- » Such tables represent many-to-many relationships between non-transaction-based tables.
- » larger in volume and grow in size much faster than the non-transaction-based tables.

What are a non-transaction-based tables?

- » generally involved in storing descriptions about the business.
- » They describe entities such as products, product category, product brand..
- » In non-transaction tables, records are usually inserted and there
- » are fewer updates and deletes.
- » Such tables are far smaller in volume and grow very slowly in size, compared to the transaction-based tables.

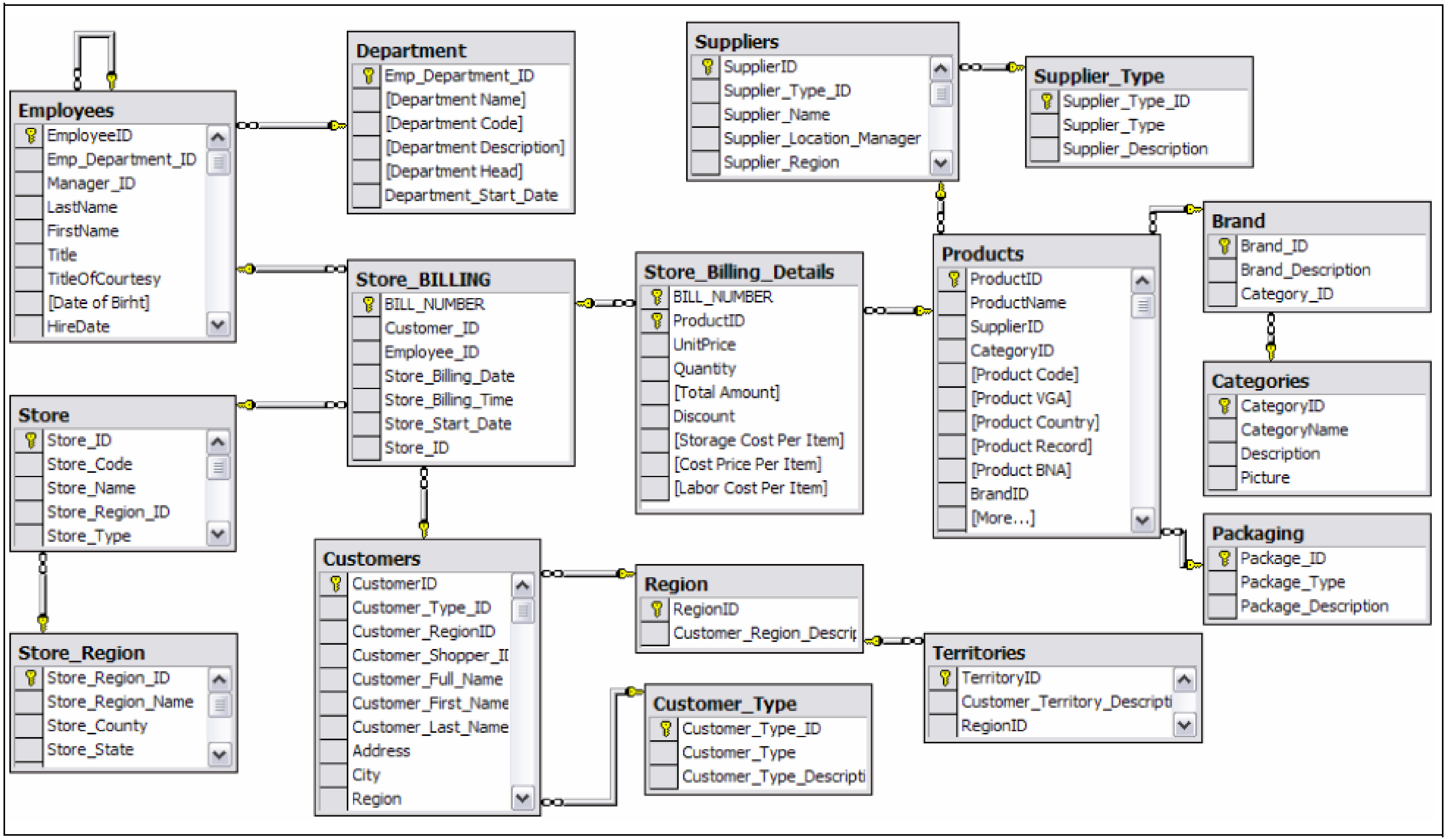
Denormalization

- » taking the remaining tables in the E/R model and denormalizing them into dimension tables for the dimensional model.
- » The primary key of each of the dimensions is made a surrogate (non-intelligent,
- » integer) key.
- » This surrogate key connects directly to the fact table

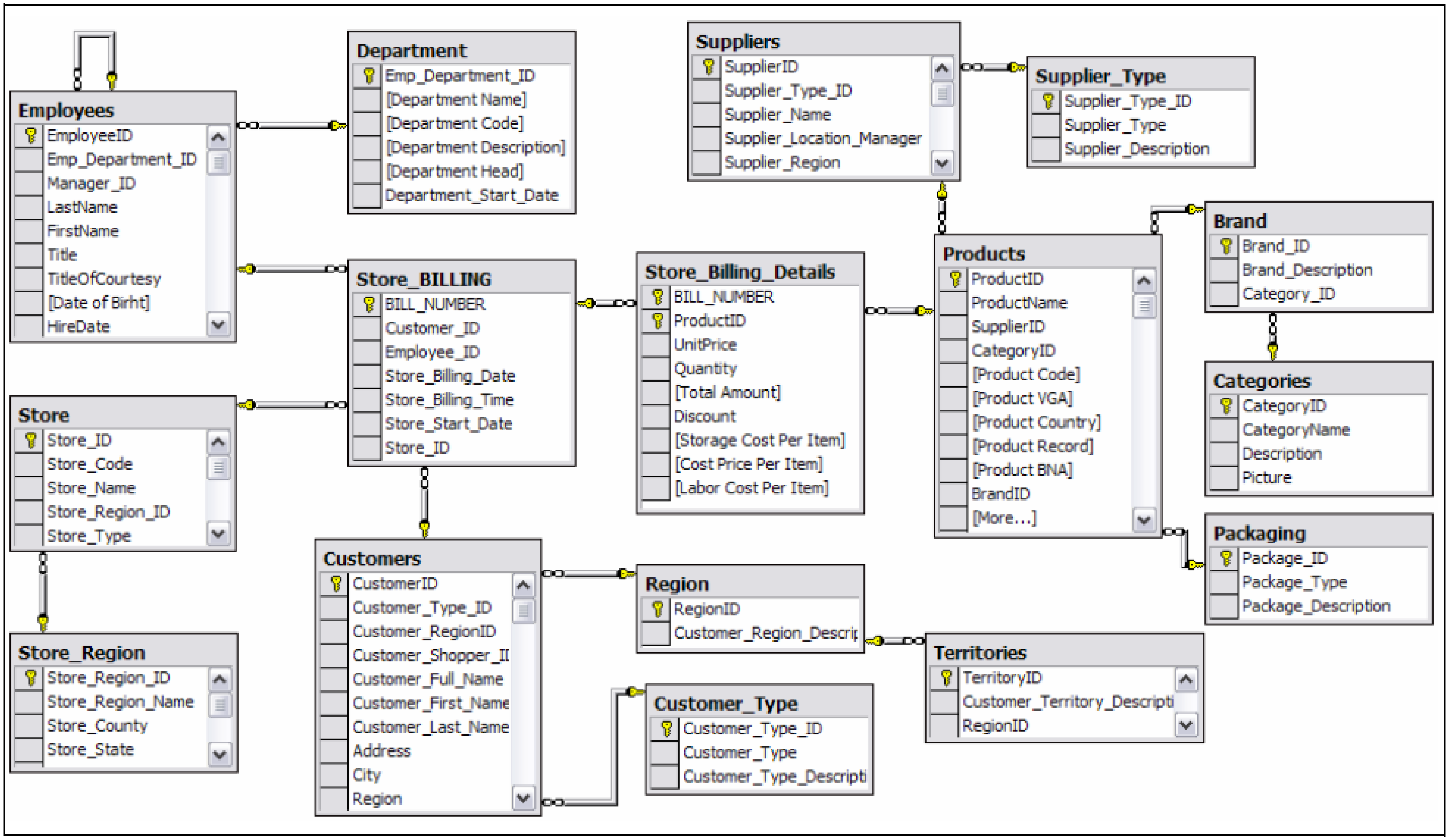
Date and Time

- » The last step generally involves identifying the date and time dimension.
- » Dates are generally stored in the form of a date timestamp column inside the E/R
- » model.
- » date and time-related columns are generally found in the transaction-based tables.

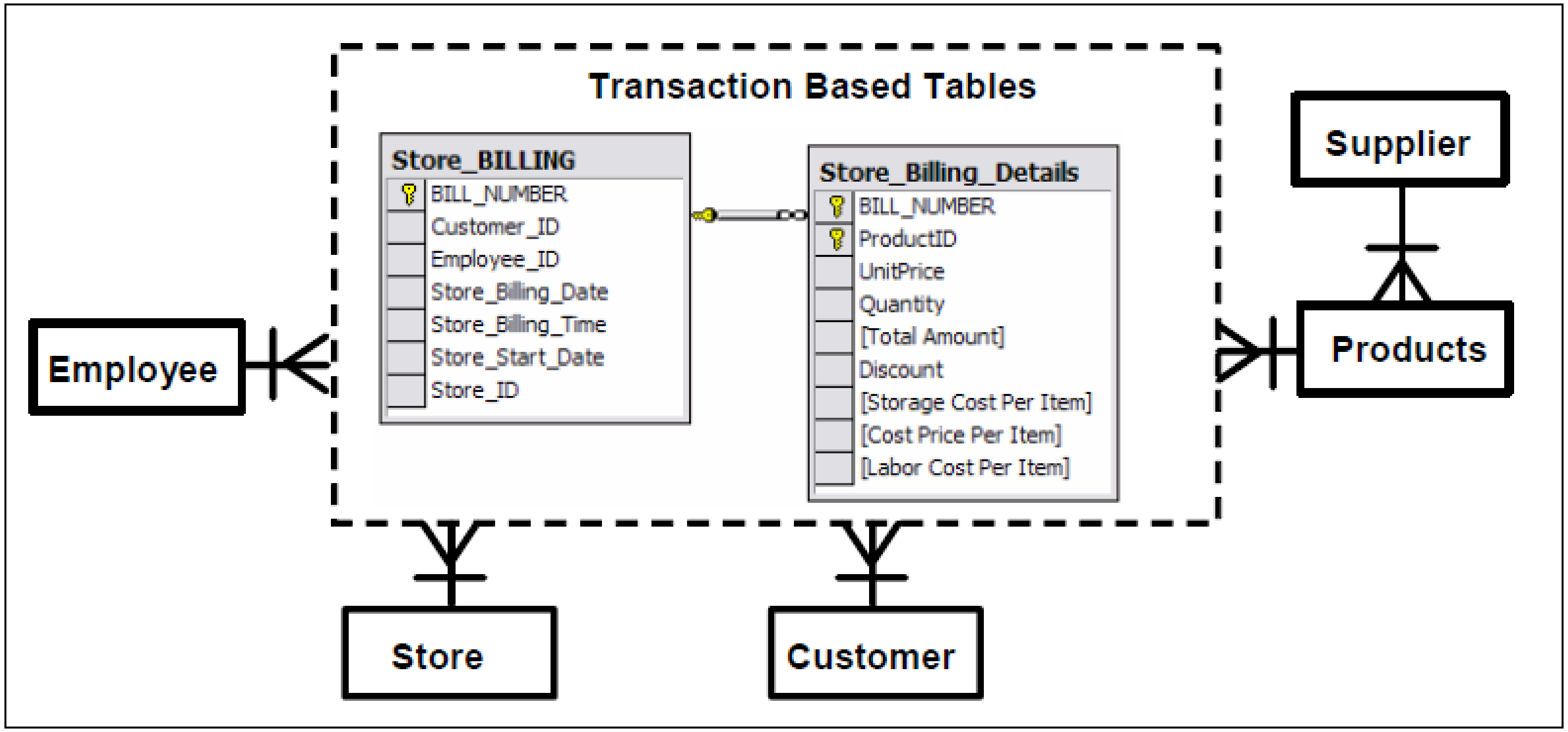
Example of ER/DM Conversion



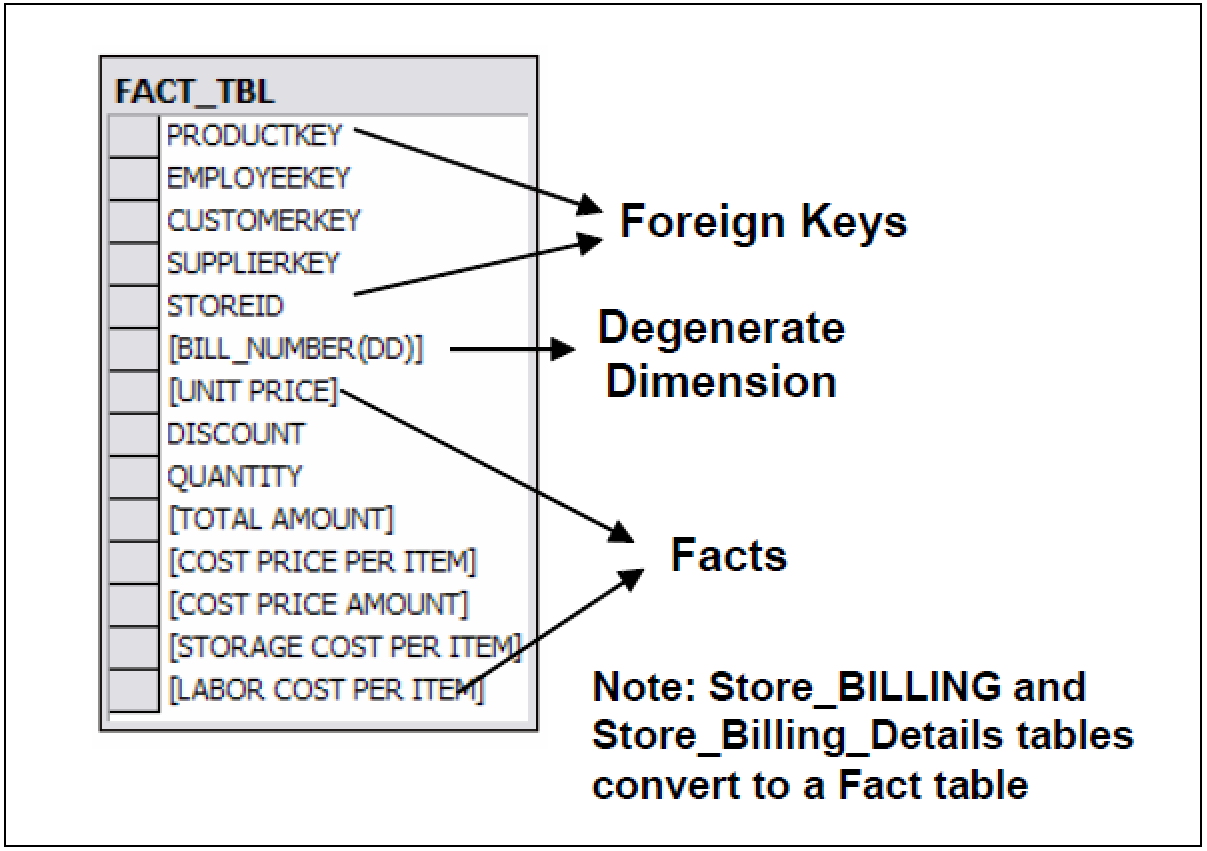
Identify BP – Retail Sales



Transaction-Based Table, Many to Many

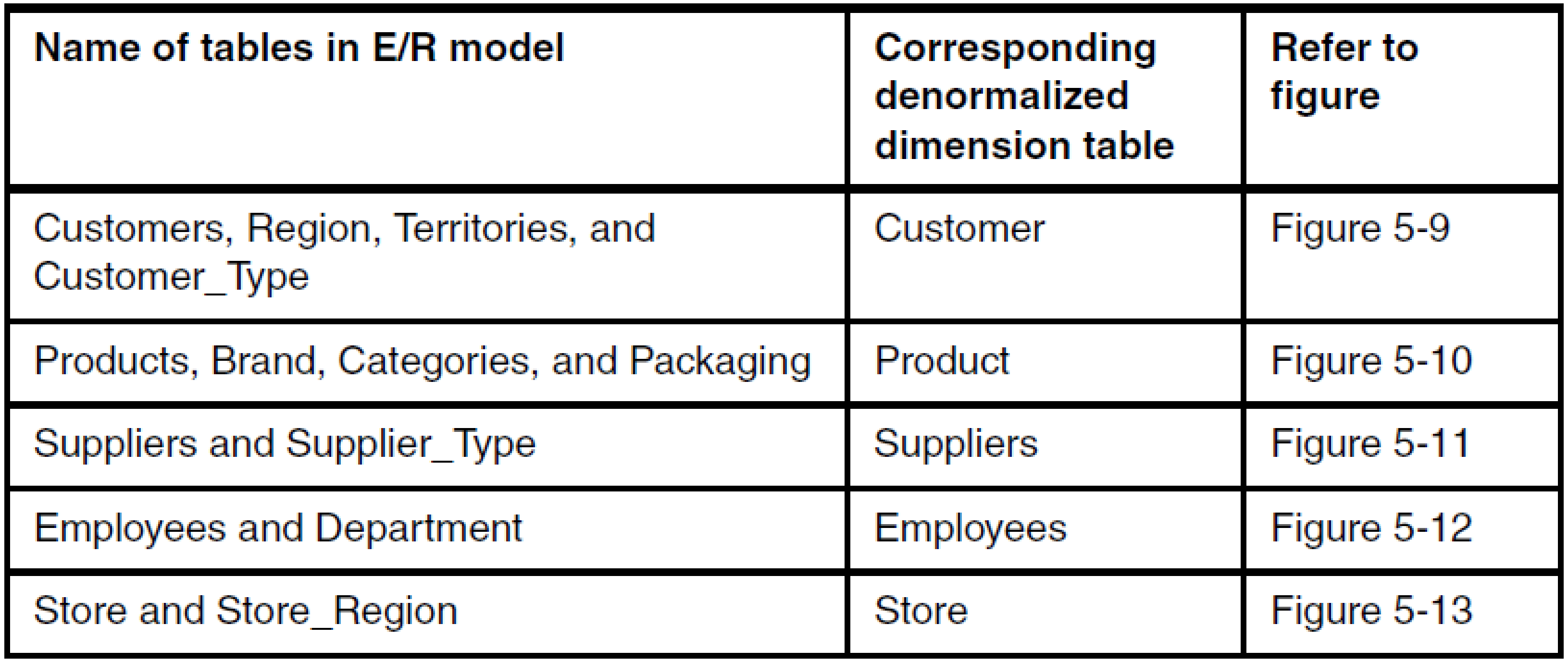


Fact Table



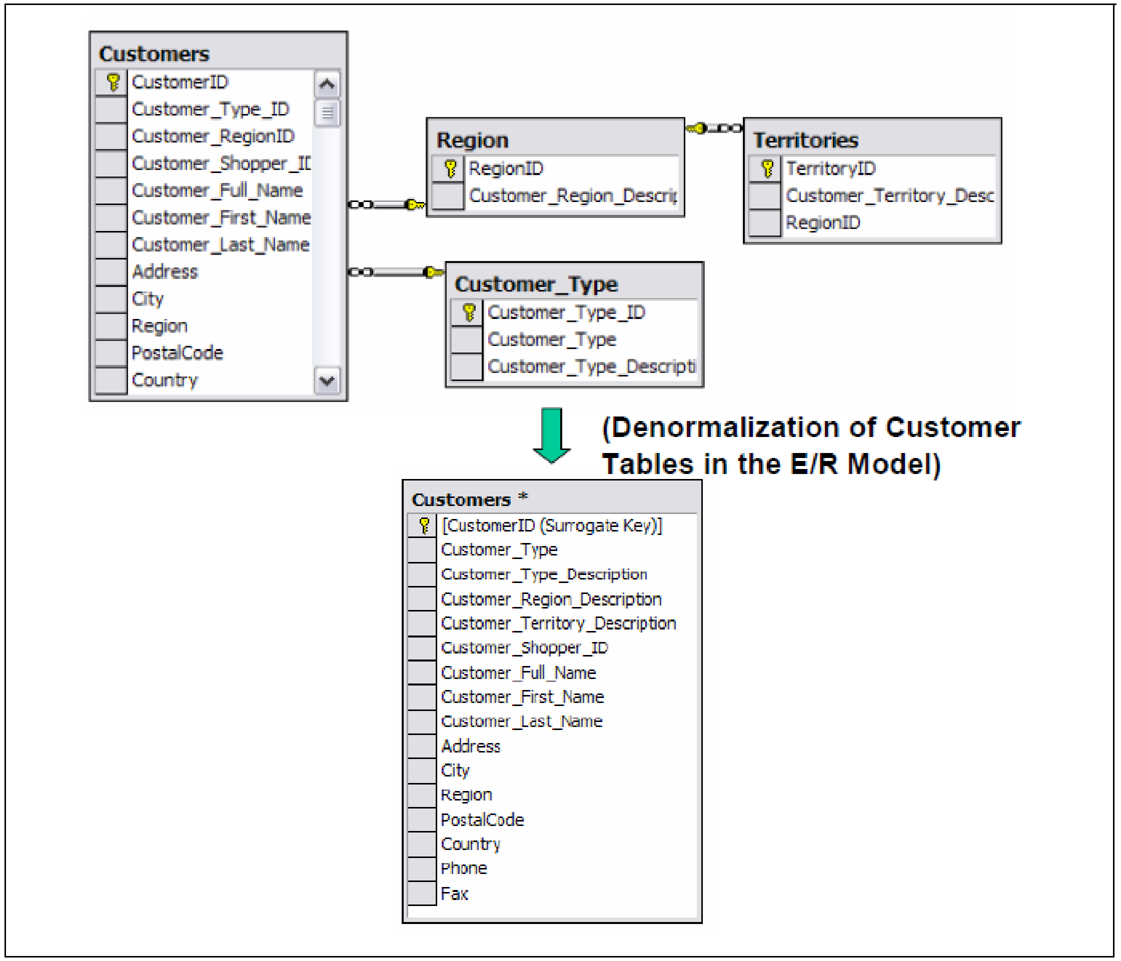
Note: Store_BILLING and Store_Billing_Details tables convert to a Fact table

Dimensions Denormlization



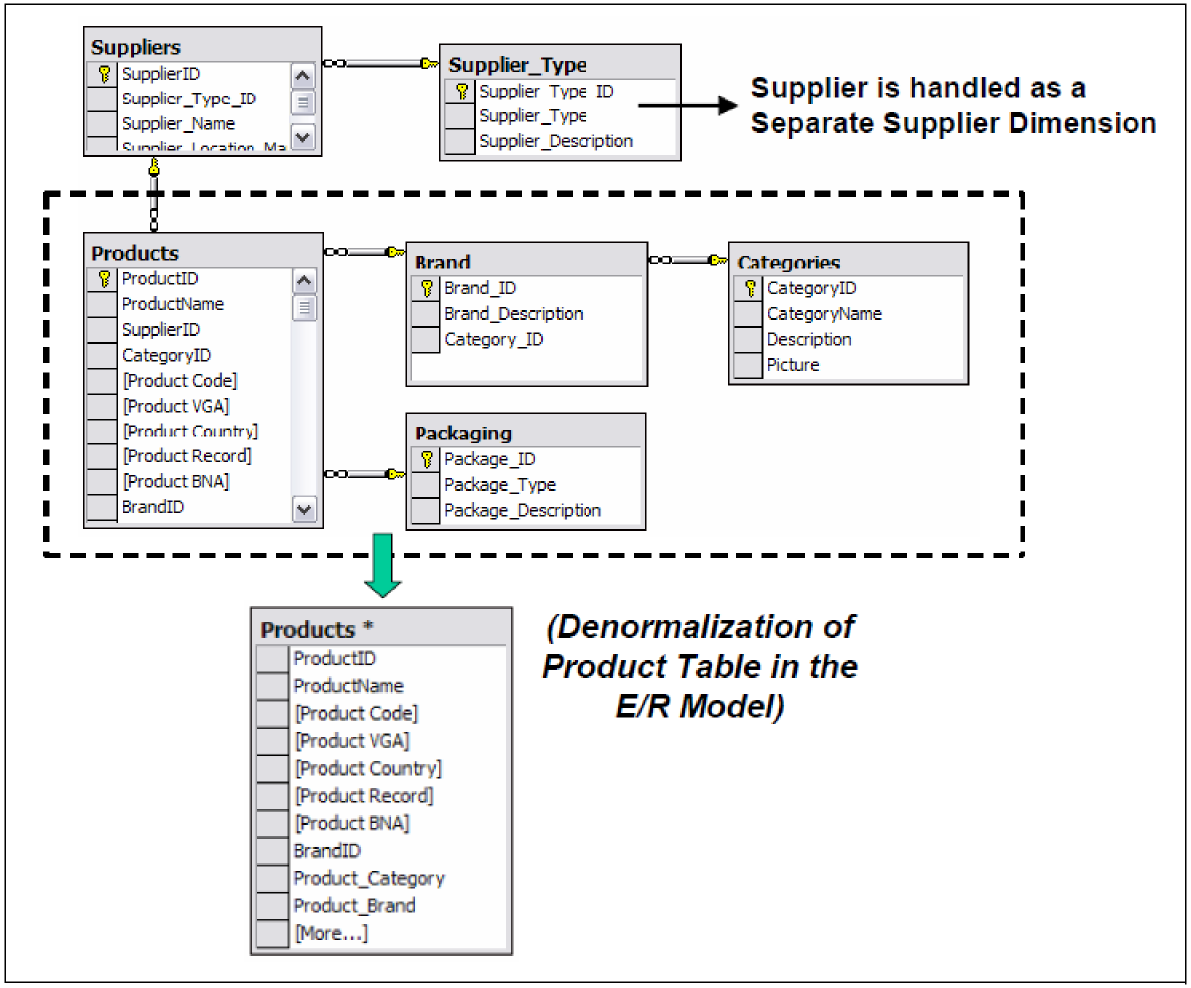
Name of tables in E/R model	Corresponding denormalized dimension table	Refer to figure
Customers, Region, Territories, and Customer_Type	Customer	Figure 5-9
Products, Brand, Categories, and Packaging	Product	Figure 5-10
Suppliers and Supplier_Type	Suppliers	Figure 5-11
Employees and Department	Employees	Figure 5-12
Store and Store_Region	Store	Figure 5-13

Customers



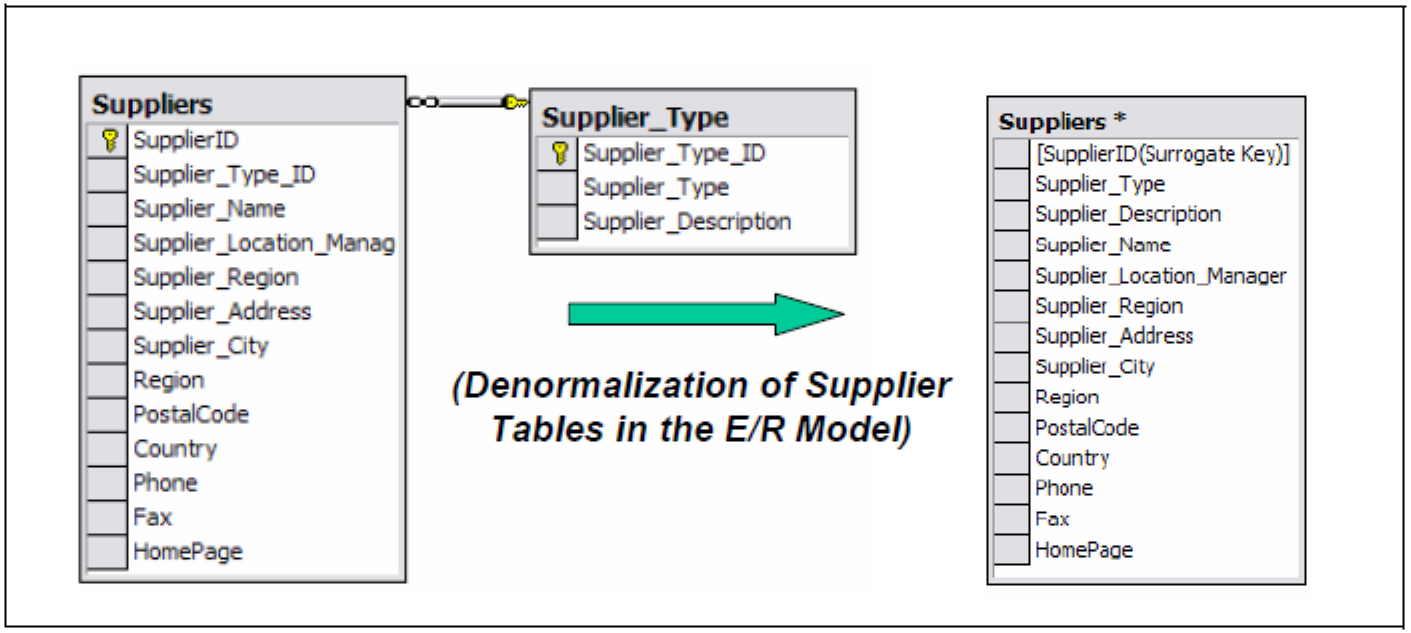
(Denormalization of Customer Tables in the E/R Model)

Product



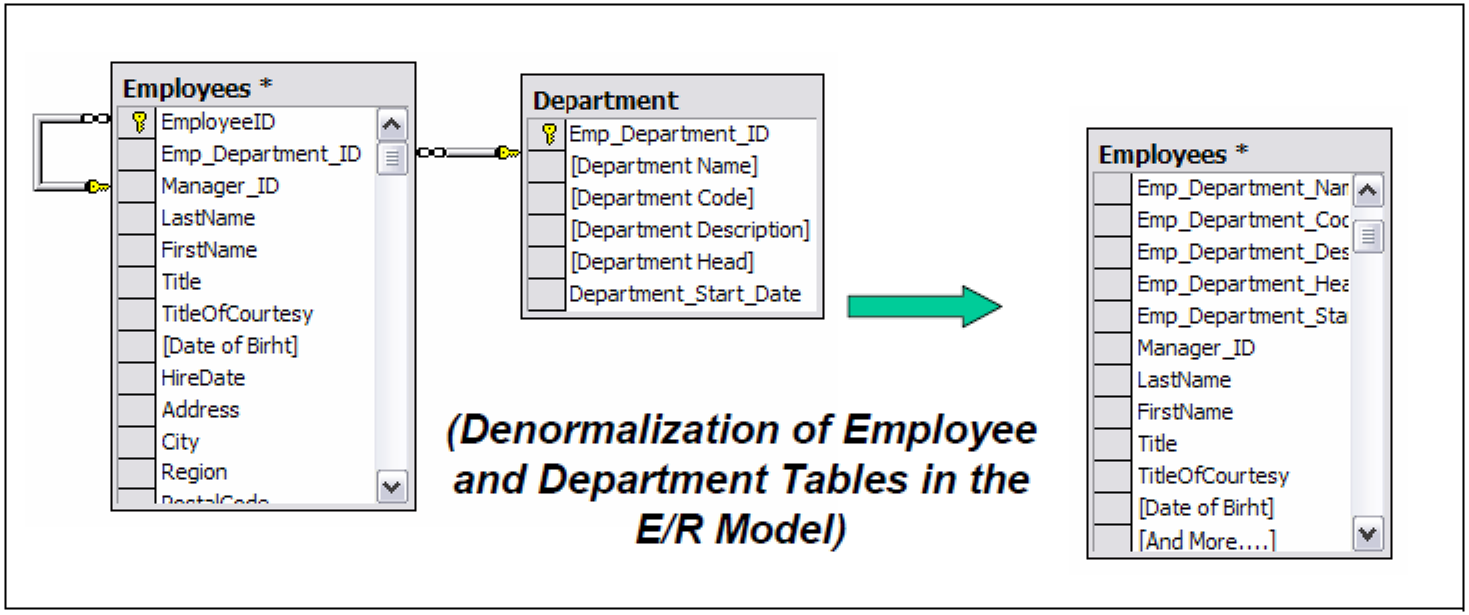
Suppliers
SupplierID
Supplier_Type_ID
Supplier_Name
Supplier_Location_Ma
Supplier_Type
Supplier_Type_ID
Supplier_Type
Supplier_Description
Supplier is handled as a Separate Supplier Dimension
Products
ProductID
ProductName
SupplierID
CategoryID
[Product Code]
[Product VGA]
[Product Country]
[Product Record]
[Product BNA]
BrandID
Brand
Brand_ID
Brand_Description
Category_ID
Categories
CategoryID
CategoryName
Description
Picture
Packaging
Package_ID
Package_Type
Package_Description
Products *
ProductID
ProductName
[Product Code]
[Product VGA]
[Product Country]
[Product Record]
[Product BNA]
BrandID
Product_Category
Product_Brand
[More...]
(Denormalization of Product Table in the E/R Model)

Suppliers



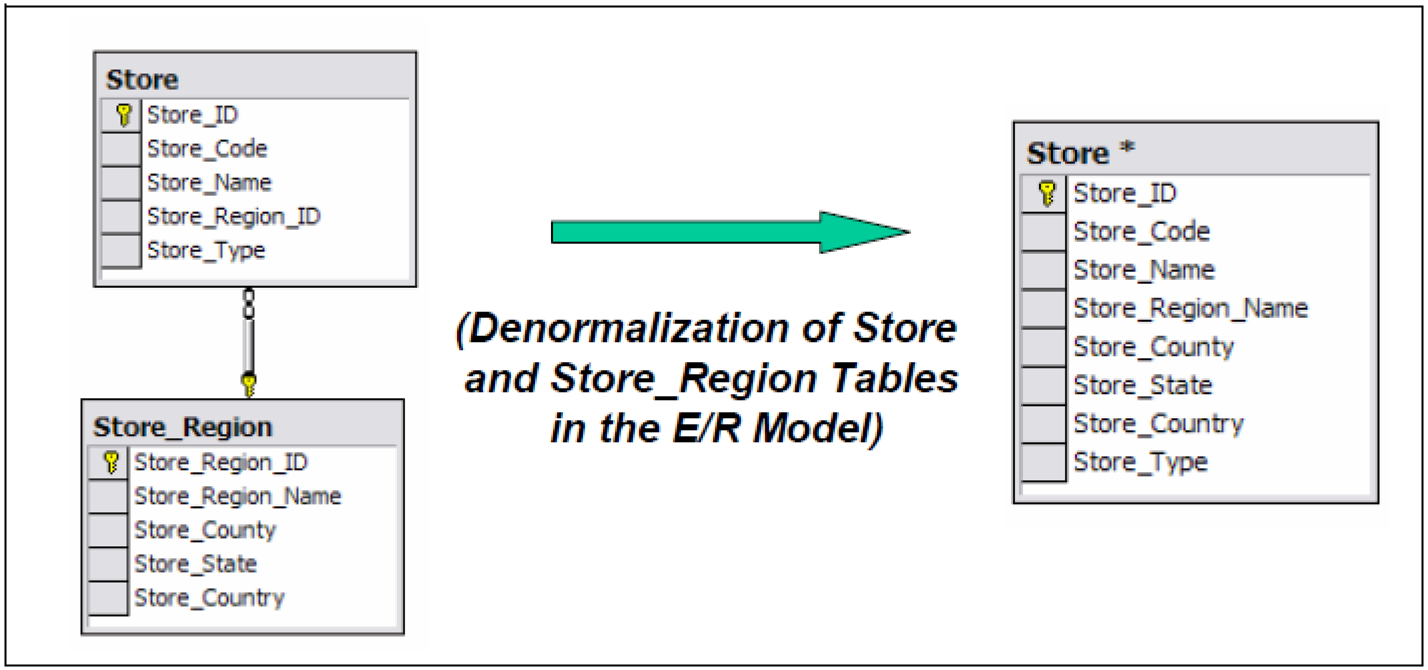
(Denormalization of Supplier Tables in the E/R Model)

Employee



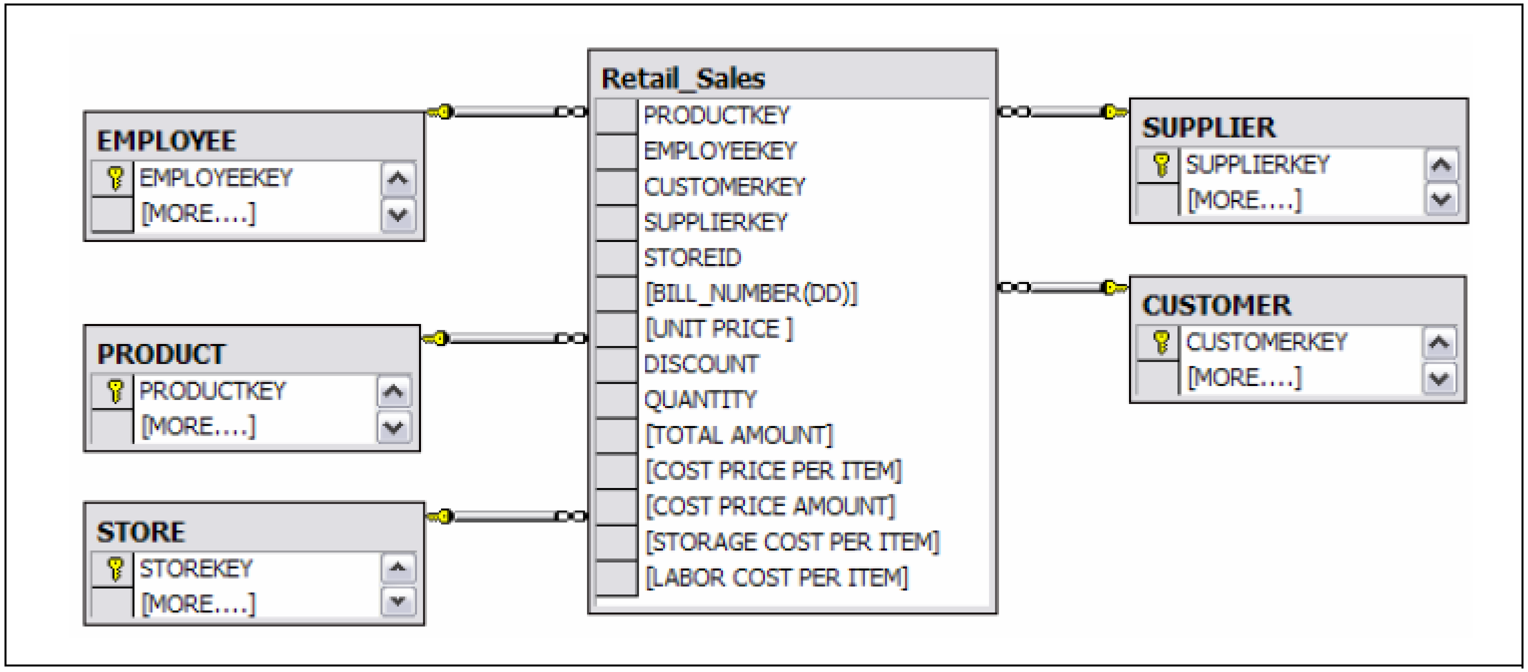
(Denormalization of Employee and Department Tables in the E/R Model)

Store

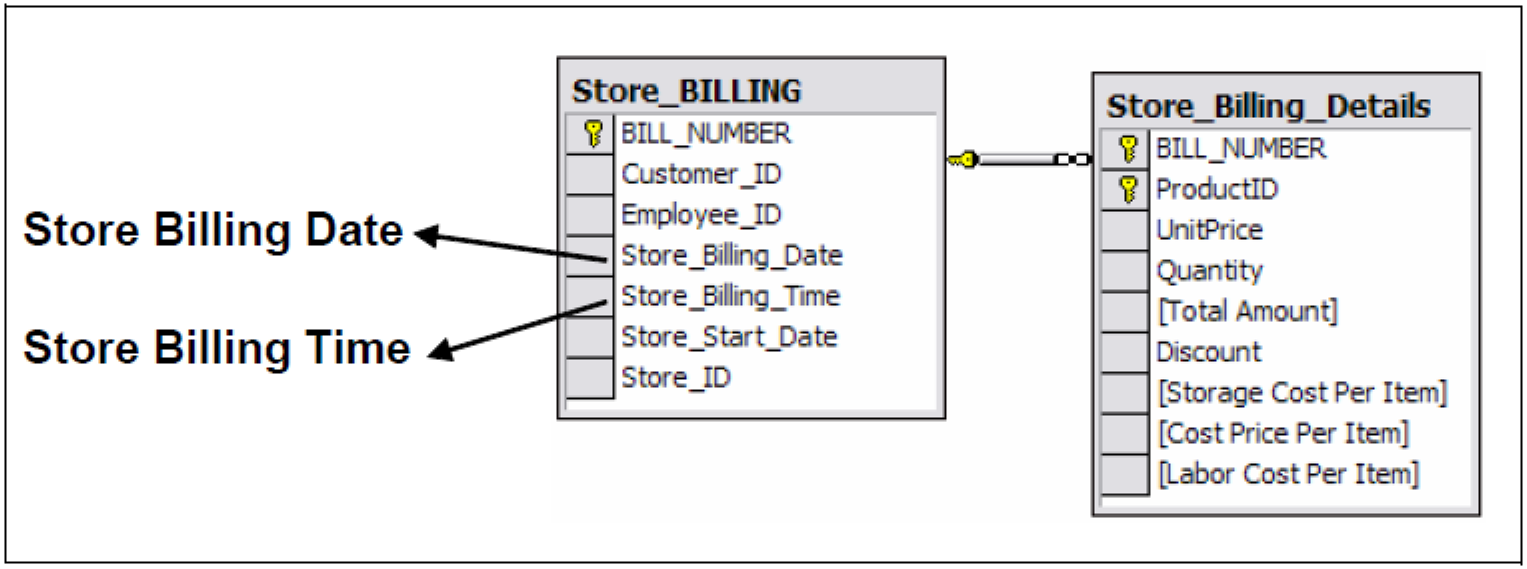


(Denormalization of Store and Store_Region Tables in the E/R Model)

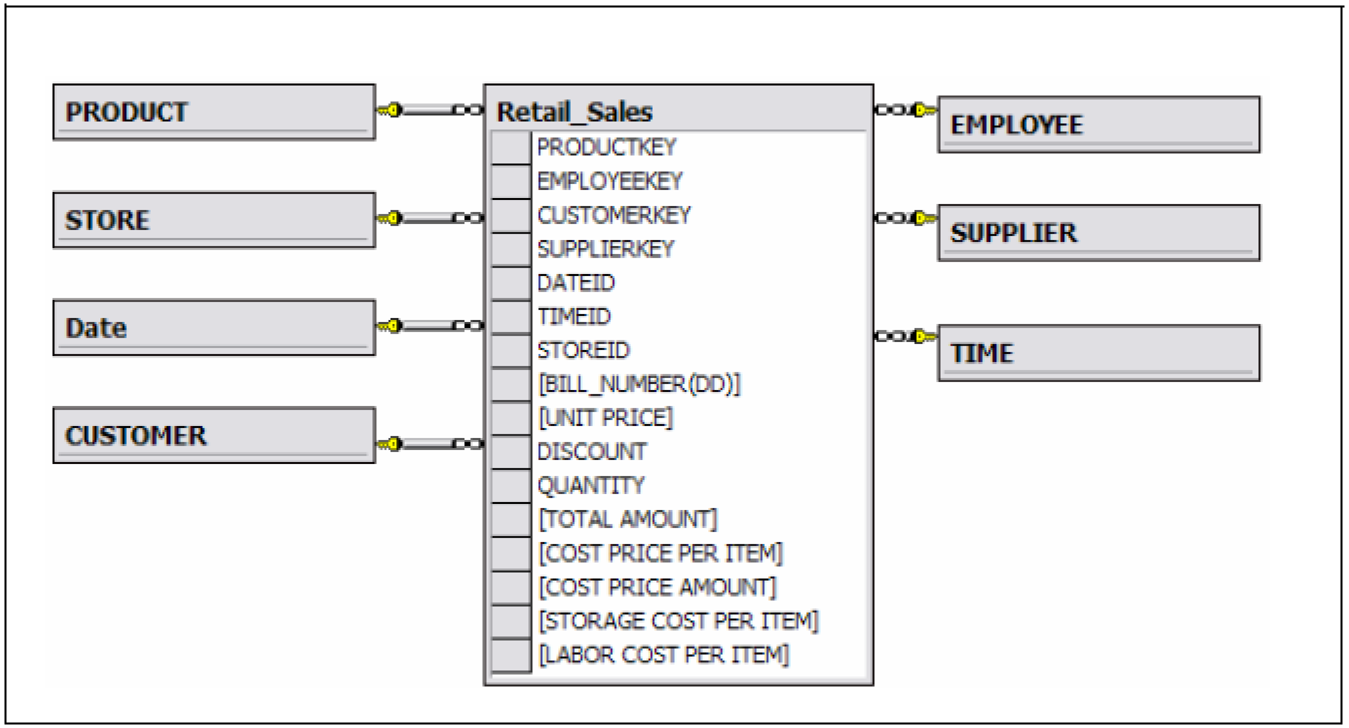
Final Star Schema



Date and Time Identification



Final Schema



Grain

» Different Grain for Different Facts

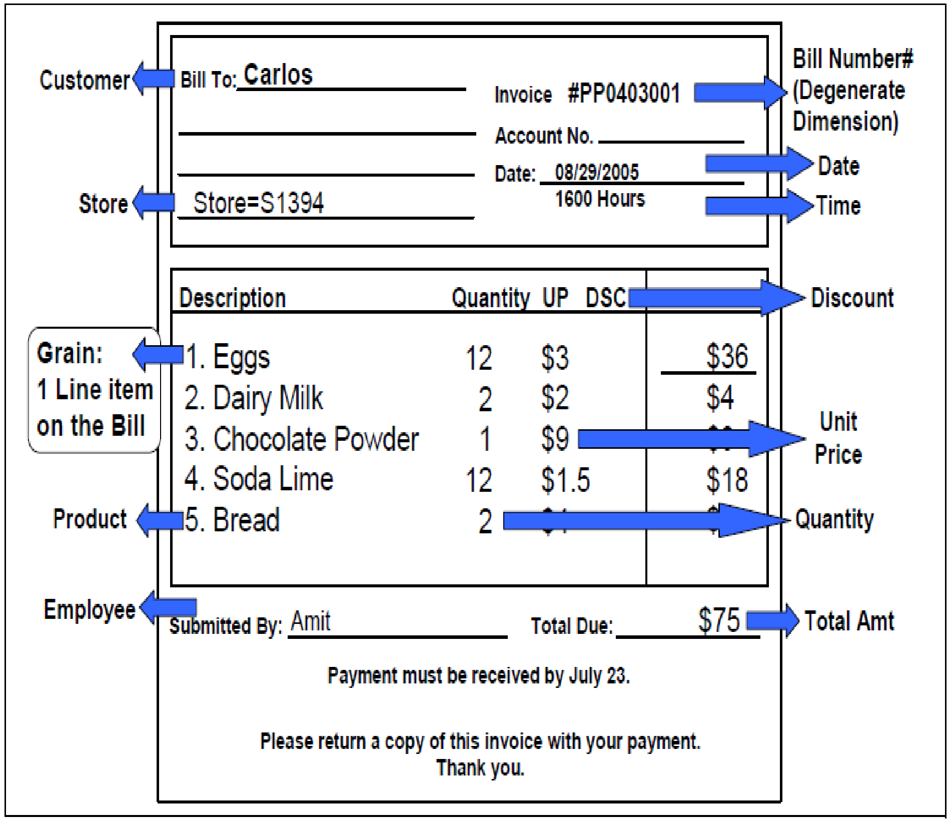
» 3 Types of Facts

» Comparisons

» How it affects the performance of the Databases

Grain

- » The lowest level of data represented in a fact table is defined as grain
- » It is important to have the grain defined at the most detailed, or atomic, level.
- » When data is defined at a very detailed level, the grain is said to be high. When there is less detailed data, the grain is said to be low.
- » For example, for date, a grain of year is a low grain, and a grain of day is a high grain.
- » In general, there are separate separate grains for a single business process.



Bill To: Carlos

Store=S1394

Invoice #PP0403001

Account No.

Date: 08/29/2005

1600 Hours

Description	Quantity	UP	DSC	
1. Eggs	12	$3		$36
2. Dairy Milk	2	$2		$4
3. Chocolate Powder	1	$9		[illegible]
4. Soda Lime	12	$1.5		$18
5. Bread	2	[illegible]		[illegible]

Submitted By: Amit

Total Due: $75

Payment must be received by July 23.

Please return a copy of this invoice with your payment.
Thank you.

Grain and DB Size

- The granularity of the fact table determines how much storage space will be required for the database.
- For example, consider the following possible granularities for a fact table:
  - Product by day by region
  - Product by month by region
- The size of a database that has a granularity of product by day by region would be much greater than a database with a granularity of product by month by region because the database contains records for every transaction made each day as opposed to a monthly summary of the transactions. You must carefully determine the granularity of your fact table because too fine a granularity could result in a huge database. Conversely, too coarse a granularity could mean the data is not detailed enough for users to perform meaningful queries.

One or many fact tables?

- When designing the dimensional model for a business process(es), one or more fact tables can be created.
- Here are guidelines to consider:
  1. Facts that are not true (valid) to any given grain should not be forced into the dimensional model. Often facts that are not true to a grain definition belong to a separate fact table with its own grain definition.
  2. Dimensions that are not true (valid) to any given grain should not be forced into the dimensional model. Often such dimensions belong to a separate dimensional model with its own fact table and grain.
  3. Separate fact tables (dimensional models) should always be created for each unique business process.

- A single business process may consist of more than one dimensional model. Do not force fit the different facts and dimensions which belong to different dimensional models into a single star schema. We strongly recommend that separate grains are identified for a business process when you are not able to fit facts or dimensions in a single star model.

Different Fact Tables

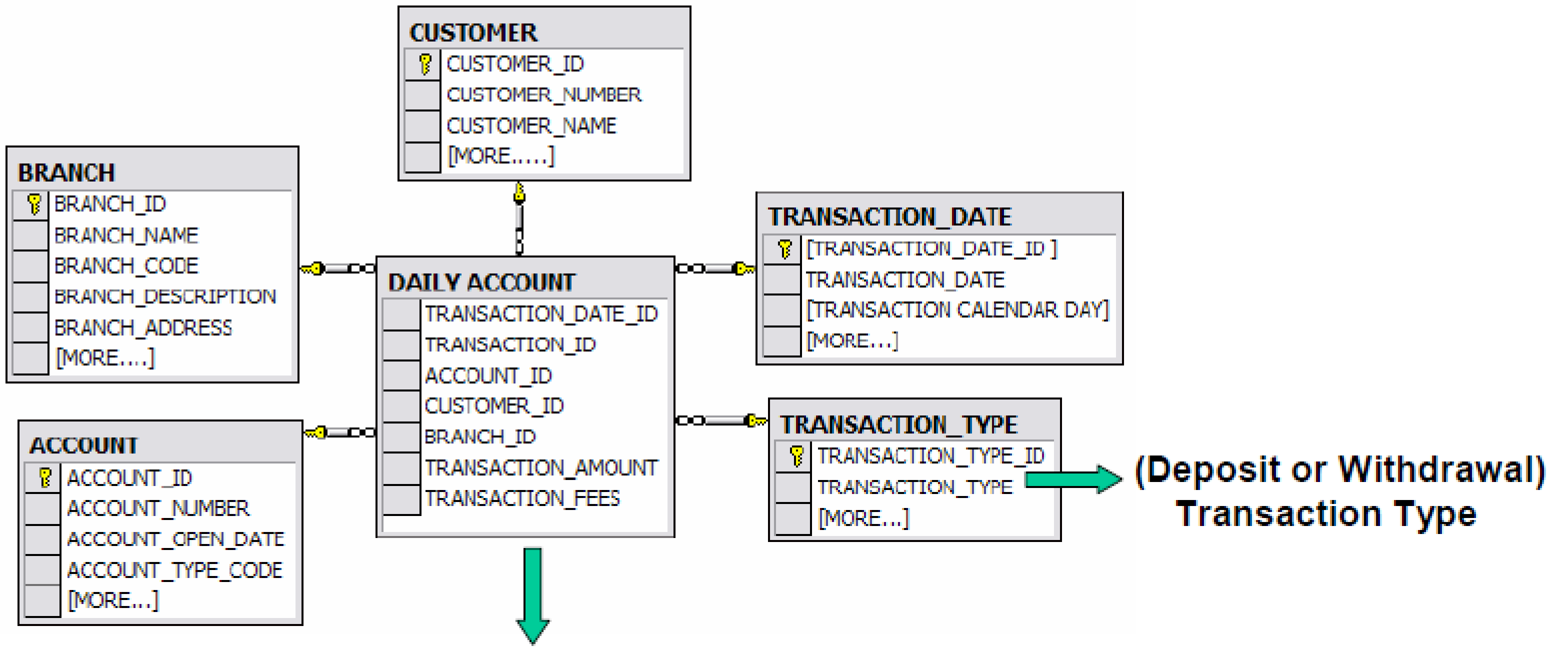
- Transaction Fact Table
- Periodic Fact Table
- Accumulating Fact Table

- Different Grains
  - Transaction and Period: only insert
  - Accumulating: insert and updated

Transaction Fact Table

- A transaction-based fact table is a table that records one row per transaction
  - *Money Withdrawn: $400, Date: August 2, 2005, Time: 4:00AM*
  - *Money Deposited: $300, Date: August 4, 2005, Time: 3:00AM*
  - *Money Withdrawn: $600, Date: August 5, 2005, Time: 2:00PM*
  - *Money Withdrawn: $900, Date: August 6, 2005, Time: 9:00PM*
- A single row is inserted for each transaction.
- Typically, the date and time dimensions are represented at the lowest level of detail.
- The transaction fact table is known to grow very fast as the number of transactions increases.

Transaction Table Example



6 Rows inserted (1 row for each Transaction)

Row 1: Withdrawal: $400, Date: 2nd August 2,2005, Time: 4:00AM

Row 2: Deposit: $300, Date: 4th August 4,2005, Time: 3:00AM

Row 3: Withdrawal: $600, Date: 5nd August 5,2005, Time: 2:00PM

Row 4: Withdrawal: $900, Date: 6th August 6,2005, Time: 9:00PM

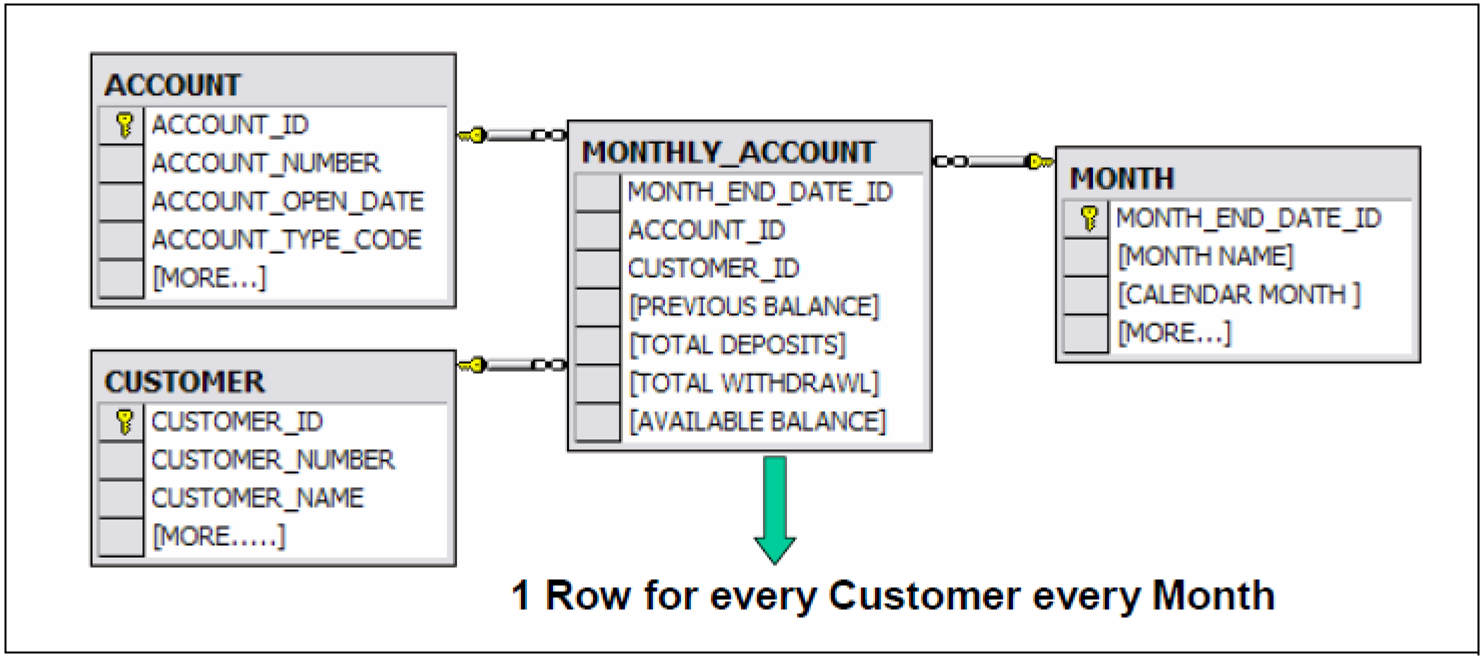
Row 5: Deposit: $900, Date: 18th August 18,2005, Time: 7:00AM

Row 6 :Deposit: $800, Date: 23rd August 23,2005, Time: 1:00AM

Periodic Fact Table

- A periodic fact table stores one row for a group of transactions made over a period of time
- A single row is inserted for each set of activities over a period of time.
- Typically, the date and time dimensions are represented at the higher level of detail.
- The periodic fact table is known to grow comparatively slowly in comparison
- to the transaction fact table.

Periodic Fact Table Example



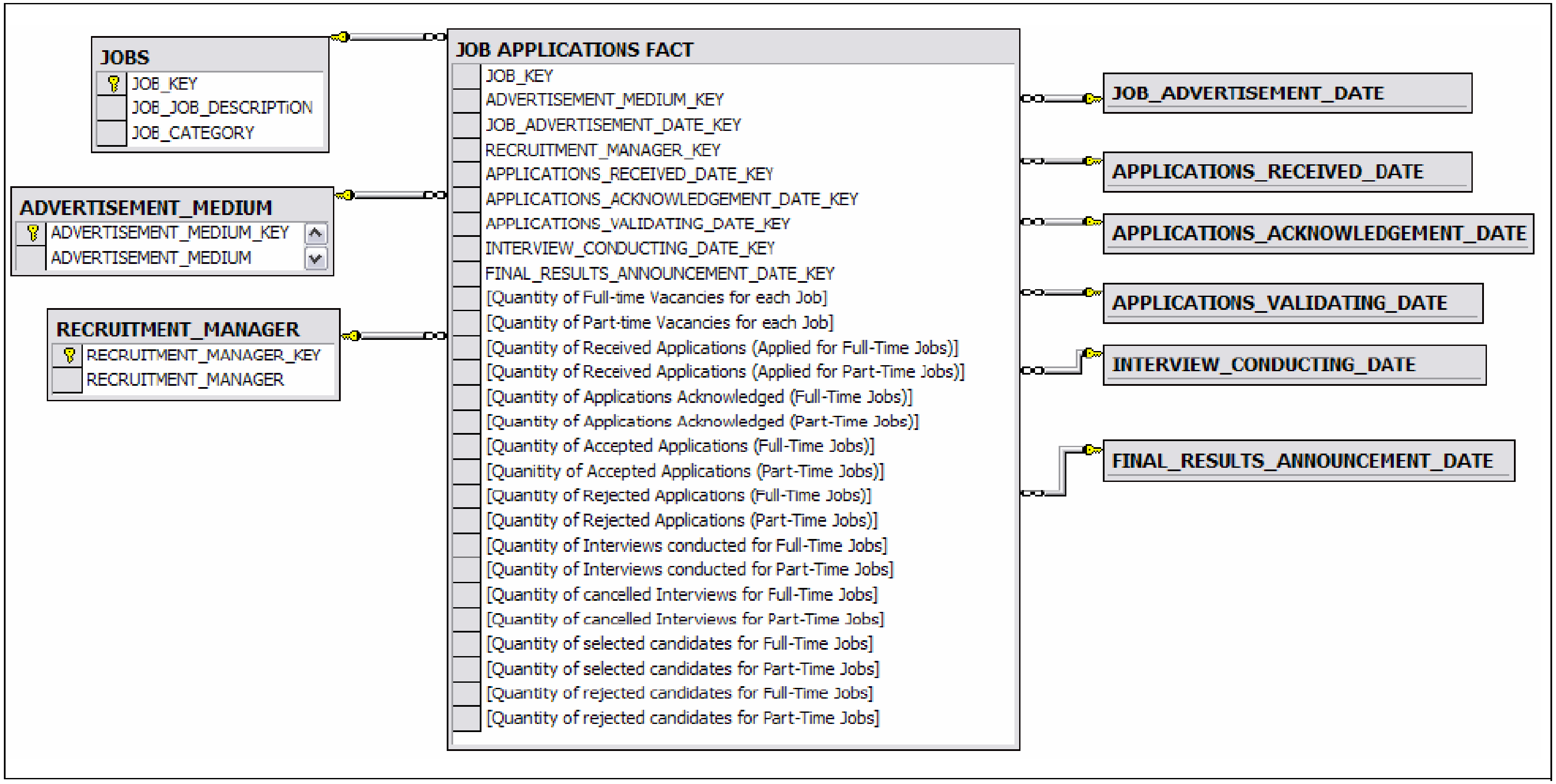
Accumulating Fact Table

- » An accumulating fact table stores one row for the entire lifetime of an event.
- » For example, from the lifetime of a credit card application being sent to the time it is accepted.
- » Accumulating fact tables are typically used for short-lived processes

Accumulating Fact Table / Example

» Consider that a big recruitment company advertises vacancies in many jobs relating to software, hardware, networking, apparel, marketing, sales, food, carpentry, plumbing, housing, house repairs, mechanical, teaching high school, teaching college, senior management, and working in restaurants. About 100 000 vacancies are advertised, in all major newspapers every month. The recruitment company senior management wants to better understand how efficiently their recruitment staff works in matching potential job candidates with the jobs they seek. The senior management wants to understand how long it takes for a prospective candidate to get a job from the time the resume is sent for a particular job vacancy.

Accumulating Fact Table / Example



JOBS
JOB_KEY
JOB_JOB_DESCRIPTION
JOB_CATEGORY
ADVERTISEMENT_MEDIUM
ADVERTISEMENT_MEDIUM_KEY
ADVERTISEMENT_MEDIUM
RECRUITMENT_MANAGER
RECRUITMENT_MANAGER_KEY
RECRUITMENT_MANAGER
JOB APPLICATIONS FACT
JOB_KEY
ADVERTISEMENT_MEDIUM_KEY
JOB_ADVERTISEMENT_DATE_KEY
RECRUITMENT_MANAGER_KEY
APPLICATIONS_RECEIVED_DATE_KEY
APPLICATIONS_ACKNOWLEDGEMENT_DATE_KEY
APPLICATIONS_VALIDATING_DATE_KEY
INTERVIEW_CONDUCTING_DATE_KEY
FINAL_RESULTS_ANNOUNCEMENT_DATE_KEY
[Quantity of Full-time Vacancies for each Job]
[Quantity of Part-time Vacancies for each Job]
[Quantity of Received Applications (Applied for Full-Time Jobs)]
[Quantity of Received Applications (Applied for Part-Time Jobs)]
[Quantity of Applications Acknowledged (Full-Time Jobs)]
[Quantity of Applications Acknowledged (Part-Time Jobs)]
[Quantity of Accepted Applications (Full-Time Jobs)]
[Quanitity of Accepted Applications (Part-Time Jobs)]
[Quantity of Rejected Applications (Full-Time Jobs)]
[Quantity of Rejected Applications (Part-Time Jobs)]
[Quantity of Interviews conducted for Full-Time Jobs]
[Quantity of Interviews conducted for Part-Time Jobs]
[Quantity of cancelled Interviews for Full-Time Jobs]
[Quantity of cancelled Interviews for Part-Time Jobs]
[Quantity of selected candidates for Full-Time Jobs]
[Quantity of selected candidates for Part-Time Jobs]
[Quantity of rejected candidates for Full-Time Jobs]
[Quantity of rejected candidates for Part-Time Jobs]
JOB_ADVERTISEMENT_DATE
APPLICATIONS_RECEIVED_DATE
APPLICATIONS_ACKNOWLEDGEMENT_DATE
APPLICATIONS_VALIDATING_DATE
INTERVIEW_CONDUCTING_DATE
FINAL_RESULTS_ANNOUNCEMENT_DATE

Comparison of Fact Tables /1

Feature	**Transaction type fact table**	**Periodic type fact table**	**Accumulating type fact table**
Grain definition of the fact table	One row per transaction. For example one row per line item of a grocery bill.	One row per period. For example, one row per month for a single product sold in a grocery store.	One row for the entire lifetime of an event. For example, the lifetime of a credit card application being sent to the time it is accepted.
Dimensions	Involves date dimension at the lowest granularity.	Involves date dimension at the end-of-period granularity.This could be end of day, end-of week, end-of month, or end-of quarter.	This type of fact table involves multiple date dimensions to show the achievement of different milestones.

Feature	Transaction type fact table	Periodic type fact table	Accumulating type fact table
Total number of dimensions involved	More than periodic fact type.	Less than transaction fact type.	Highest number of dimensions when compared to other fact table types. Generally this type of fact table is associated with several date dimension tables which are based on a single date dimension implemented using a concept of role-playing. This is discussed in 5.3.9, "Role-playing dimensions" on page 179.
Conformed Dimensions	Uses shared conformed dimensions.	Uses shared conformed dimensions.	Uses shared conformed dimensions.
Facts	Facts are related to transaction activities.	Facts are related to periodic activities. For example, inventory amount at end of day or week.	Facts are related to activities which have a definite lifetime. For example, the lifetime of a college application being sent to the time it is accepted by the college.
Conformed Facts	Uses shared conformed facts.	Uses shared conformed facts.	Uses shared conformed dimensions.
Database size	Transaction-based fact tables have the biggest size. If the grain of the transaction is chosen at the most detailed level, these tables tend to grow very fast.	The size of a Periodic fact table is smaller than the Transaction fact table because the grain of the date and time dimension is significantly higher than lower level date and time dimensions present in the transaction fact table.	Accumulating fact tables are the smallest in size when compared to the Transaction and Periodic fact tables.

Feature	Transaction type fact table	Periodic type fact table	Accumulating type fact table
Performance	Performance is typically good. However, the performance improves if you chose a grain above the most detailed because the number of rows decreases.	Performance for Periodic fact table is higher than other fact table types because data is stored at lesser detailed grain and therefore this table has fewer rows.	Performance is typically good. The select statements often require differences between two dates to see the time period in days/weeks/months between any two or more activities.
Insert	Yes	Yes	Yes
Update	No	No	Yes. Only when a milestone is reached for a particular activity.
Delete	No	No	No
Fact table growth	Very fast.	Slow in comparison to transaction-based fact table.	Slow in comparison to the transaction and periodic fact table.
Need for aggregate tables	High need (This is primarily because the data is stored at a very detailed level.)	None or very few (This is primarily because the data is already stored at a highly aggregated level.)	Medium need (This is primarily because the data is stored mostly at the day level. However, the data in accumulating fact tables is less than the transaction level.)