Advanced Databases

Dimensional modelling conversion from ER to Star Schema

Giulia Vilone

giulia.vilone@tudublin.ie

Benefits of Data Warehouse

- **Performance** - Integer relationships, natural partitioning, single joins benefit SQL optimizer
- **Supports change management**
- **Usability/simplicity** - Easy to read, interpret, join, calculate
- **Presentation** - Consistency, taxonomy, labelling
- **Reuse** - Conformed dimensions reduce redundancy, role-plays

Example

PROBLEM: Build a Data Warehouse to investigate the effect of news and user-generated content (Twitter) sentiment on the stock market.

QUESTIONS:

- What kind of data do we need?
- Decide the GRAIN
- Where are the data?
- What kind of transformation do we need to do? What kind of cleaning?

Example – Data sources, cleaning & transformation

Data type	Data source	Cleaning & transformation
Stock market data	Yahoo Finance (csv), Bloomberg API (json)	Should be fine, check for dividends and splits adjustments. Include Volumes
Twitter data	Twitterscraper (json)	The text of the tweets does not go into the DM. We need to process the tweets and apply a sentiment library (like textblob) to get the sentiment and subjectivity of each tweet. Add IBM Watson Tone Analyzer dimensions as well (anger, fear, joy, analytical level…)
News	Reuters RSS feed, Yahoo Finance RSS feed	Assign News to stocks (entity recognition, keywords matching….)
All data	…	Assign surrogate keys

Example – The data warehouse DM

Entity Relationship (ER)

Vs.

Dimensional Modelling (DM)

Entity Relationship Modelling: Review

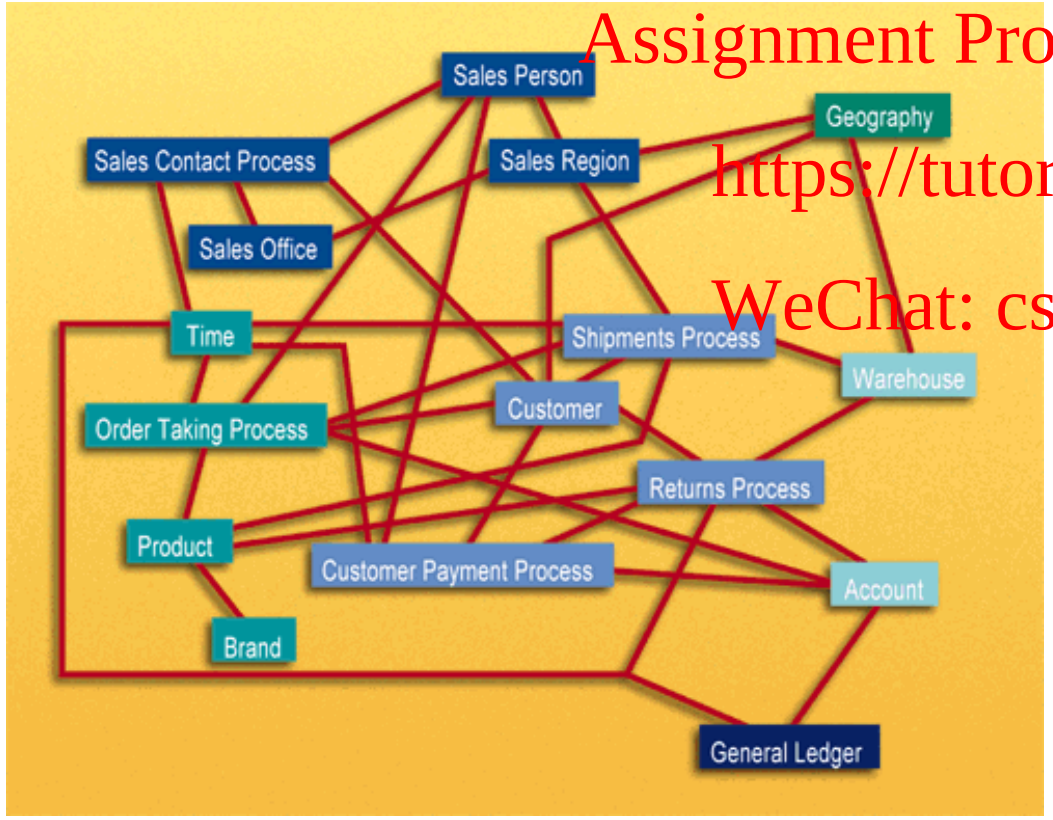
- ER modelling is a technique used to "abstract" user's data requirements into a model that can be analysed and ultimately implemented.
- The focus of ER modelling:
  - achieve processing and data storage efficiency by reducing data redundancy (storing data elements once)
  - provide flexibility and ease of maintenance
  - protect the integrity of data by storing it once
- ER modelling 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 relation databases are set to 3rd normal form

Form type	Description
1st Normal form	Tables have unique keys and no repeating groups or multi-value fields
2nd Normal form	Every attribute is dependent on 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

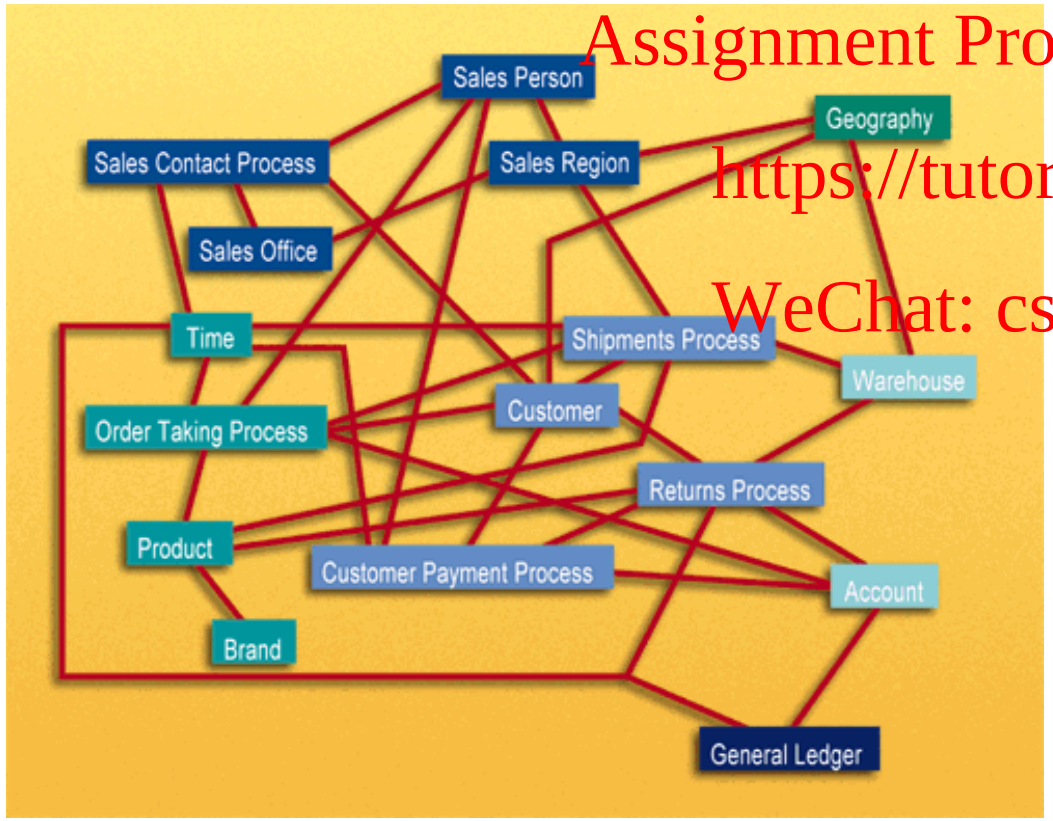
ER model issues

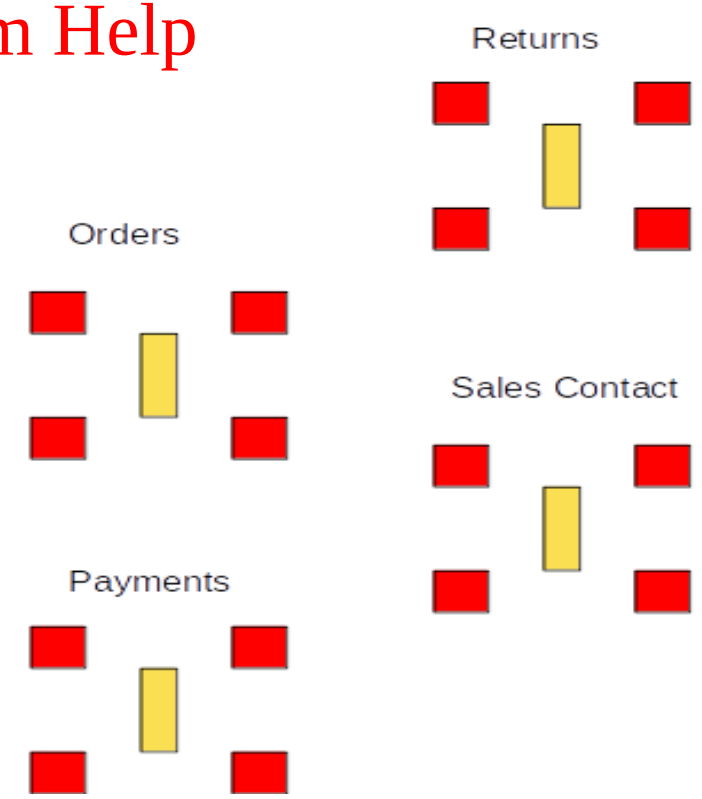
- End users cannot understand, remember or navigate an ER model.
- There is no graphical user interface (GUI) that takes a general ER model and makes it usable by end users.
- Software cannot usefully query a general ER model. Cost-based optimizers that attempt to do this are notorious for making the wrong choices, with disastrous consequences for performance.
- Use of the ER modelling 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 designate them as **fact tables**.
- Alternatively, look for **events** or **transactions** – these may also be facts.
- Denormalize all 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 DM has many important data warehouse advantages that the ER model lacks:

- First, the DM is a predictable, standard framework. Report writers, query tools, and user interfaces can all make strong assumptions about the DM to make the GUIs more understandable and the 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

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

The DM is that it is **gracefully extensible** to accommodate unexpected new data elements and new design decisions.

Gracefully extensible means:

- 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

There is a body of standard approaches for handling common DM situations in the business world. These modelling situations include:

- Slowly changing dimensions, where a "constant" dimension such as Product or Customer actually evolves slowly and asynchronously.

DM strengths

A final strength 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 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.

ER vs DM – Final summary

Strengths	DM	ER
Data completeness	✗	✓
Data update	✗	✓
Data analysis for business and decision-making purposes	✓	✗
Data aggregation	✓	✗
Daily operations	✗	✓
Query performance optimization	✓	✗
Standard, predictable structure	✓	✗
Integrable with a GUI	✓	✗
Easily extendible to accommodate new, unexpected data and facts	✓	✗
Adapt for a data warehouse	✓	✗

Dimensional Modelling conversion

From ER diagram to DM

Modelling 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 business 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 DMs, we’ll inevitably duplicate data with different labels and terminology.”

Identify the business process

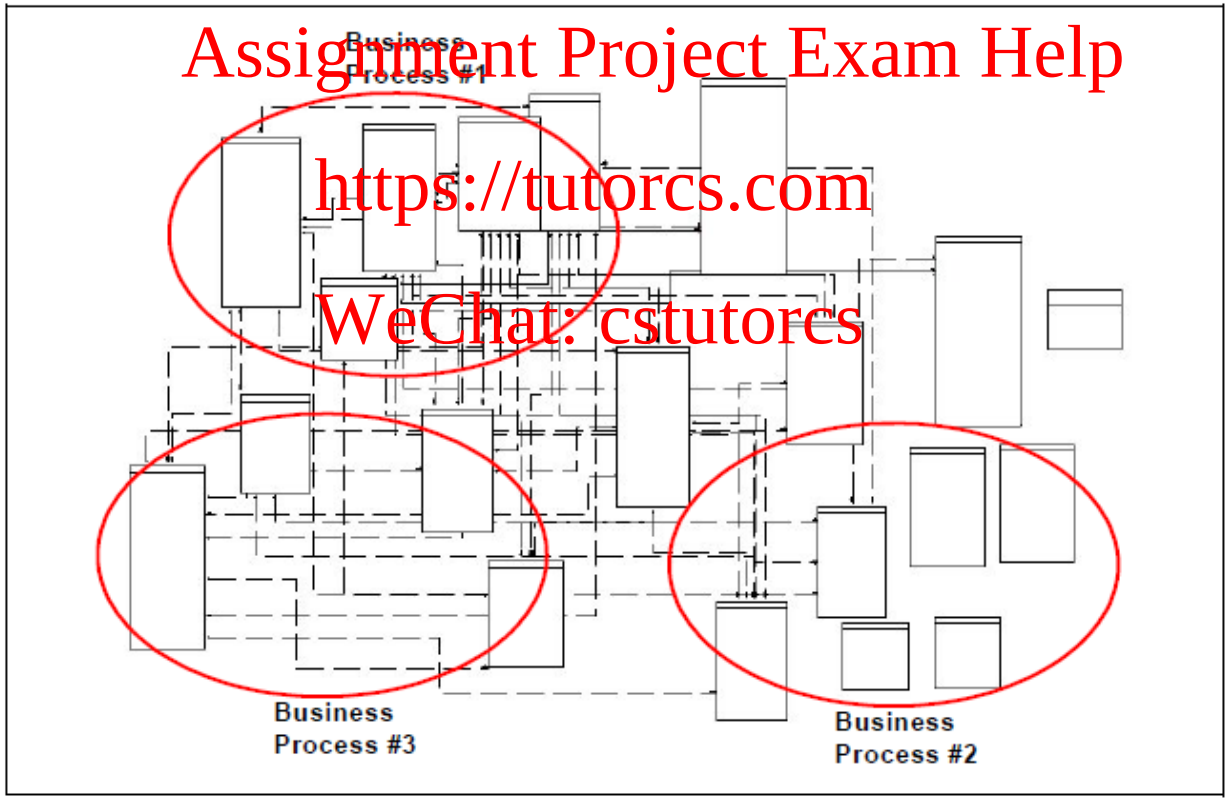


Figure 5-1 E/R model consists of several business processes

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 thought 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.

Many-to-Many

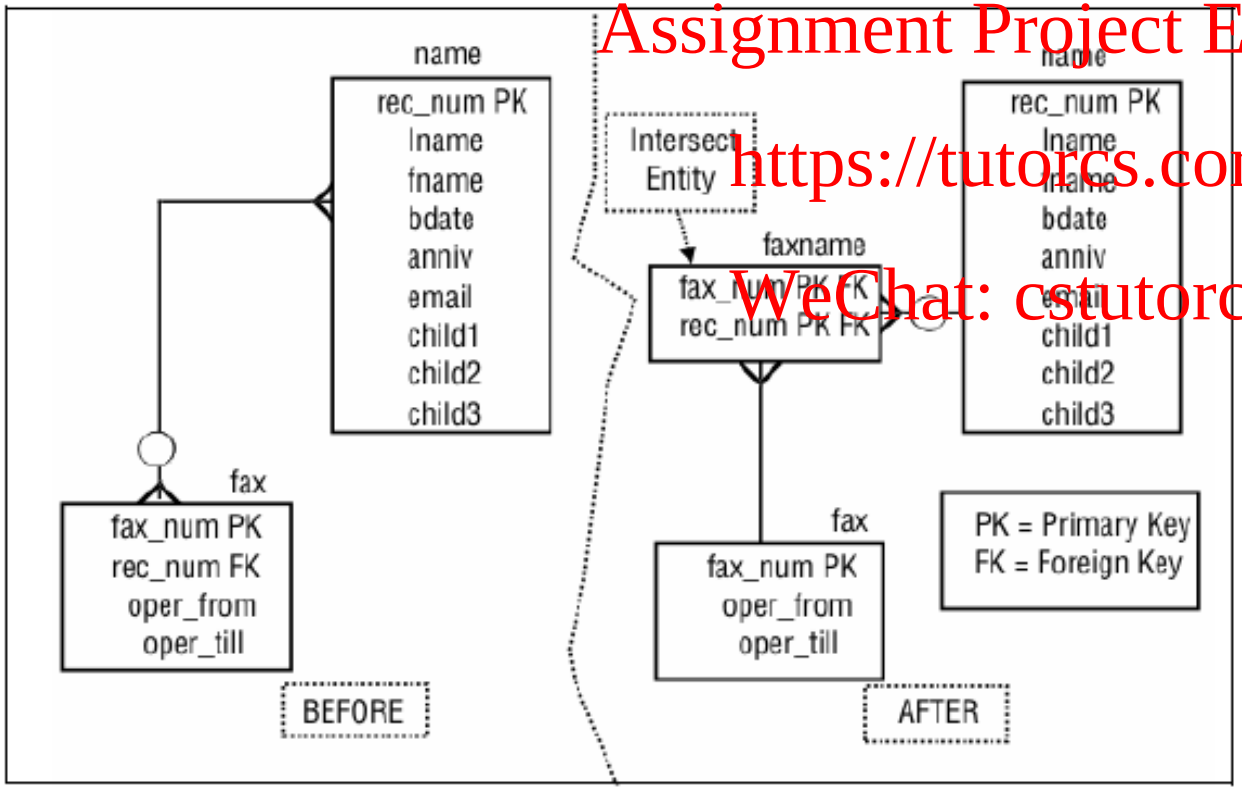


Figure 5-3 Many-to-many relationship

- They represent the link between dimensions
- They usually help to identify **transaction 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 ER model.
- Every ER 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.

Transaction / Non transaction

What are transaction tables?

- 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.
- 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.

Transaction / Non transaction

What are non-transaction based tables?

- Generally involved in storing descriptions about the business.
- They describe entities such as products, product category, product brand.
- 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 ER 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

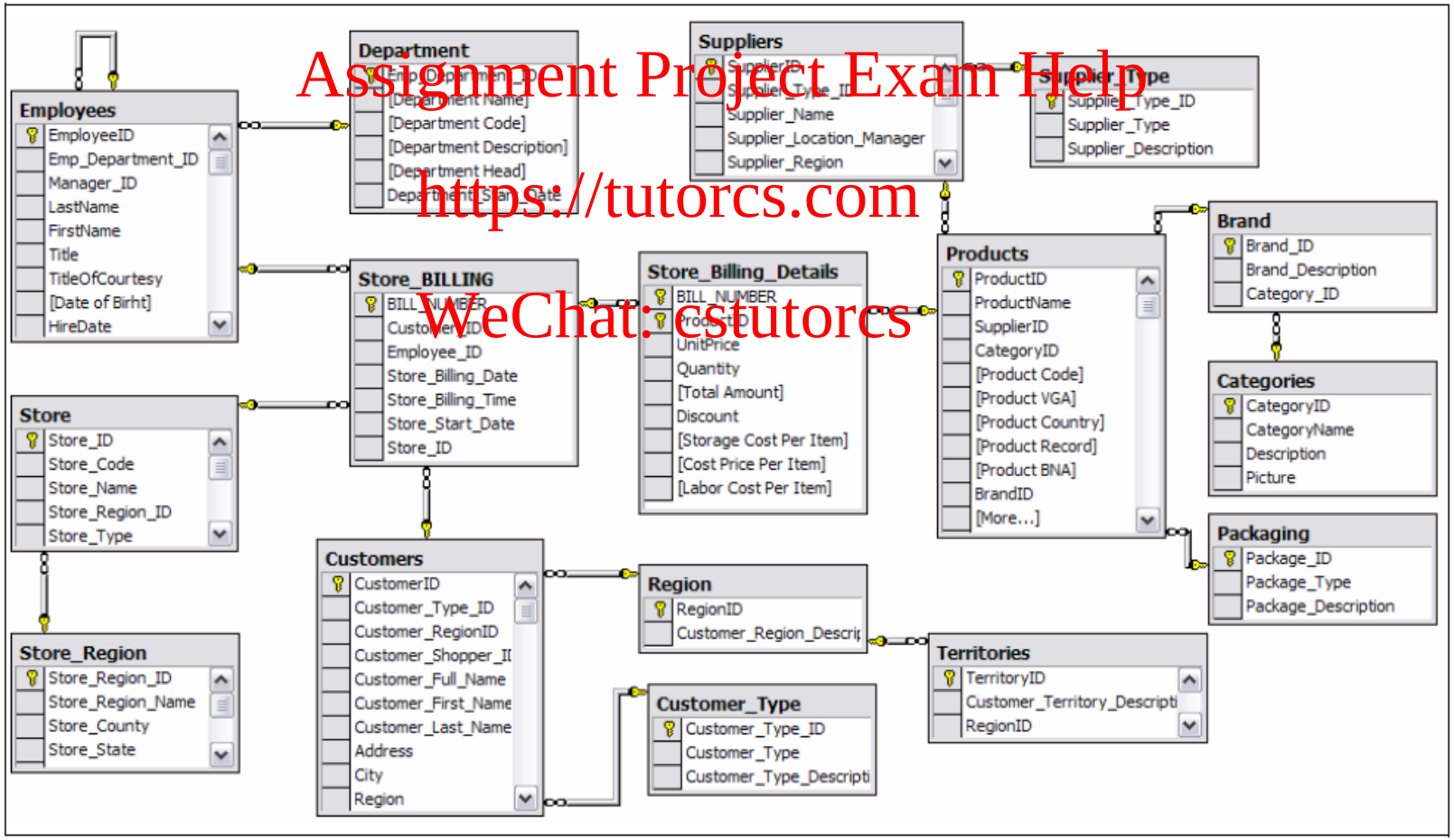
Date and Time

- The last step generally involves identifying the date and time dimension.

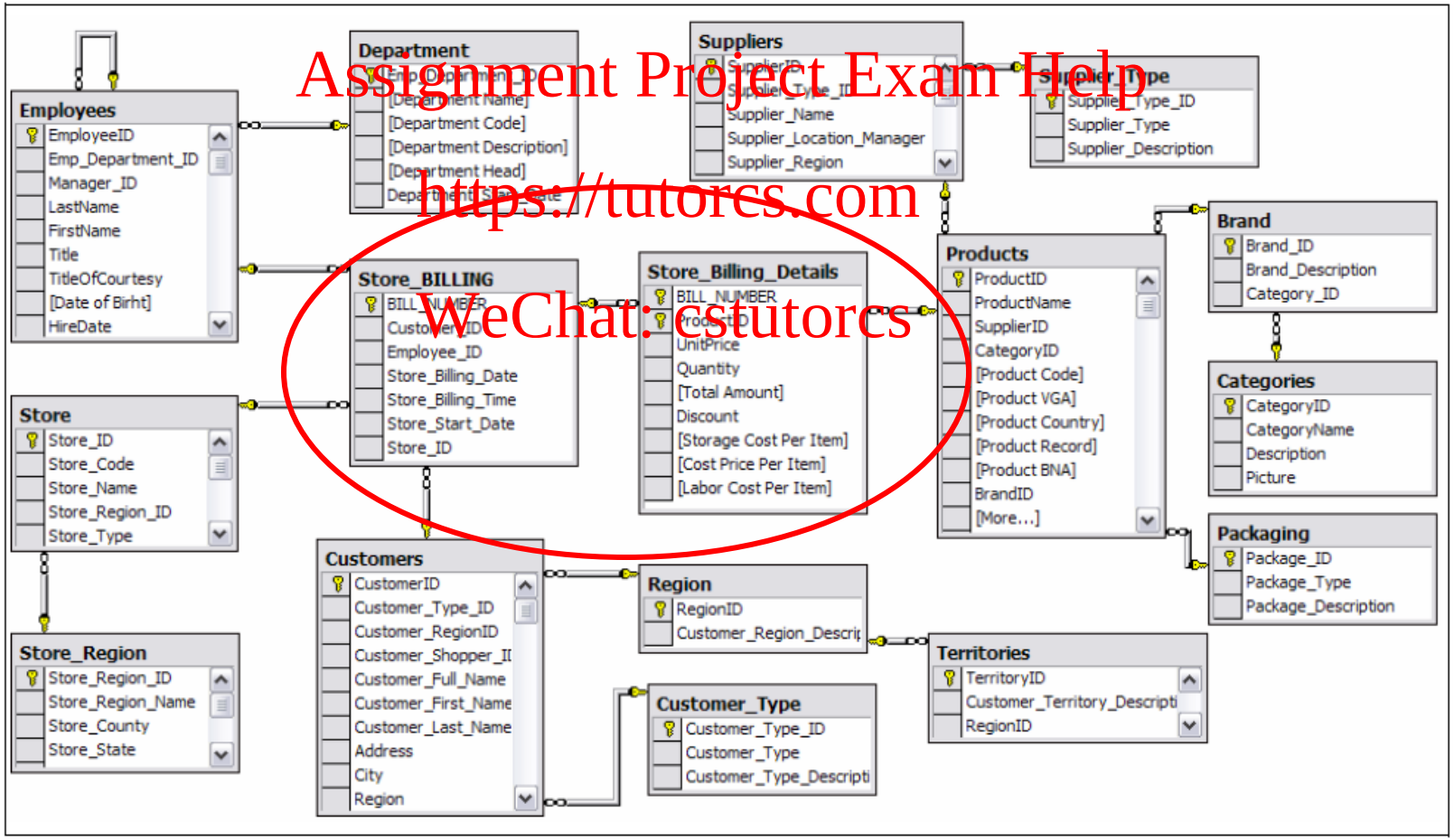
- Dates are generally stored in the form of a date timestamp column inside the ER model.

- Date and time-related columns are generally found in the transaction-based tables.

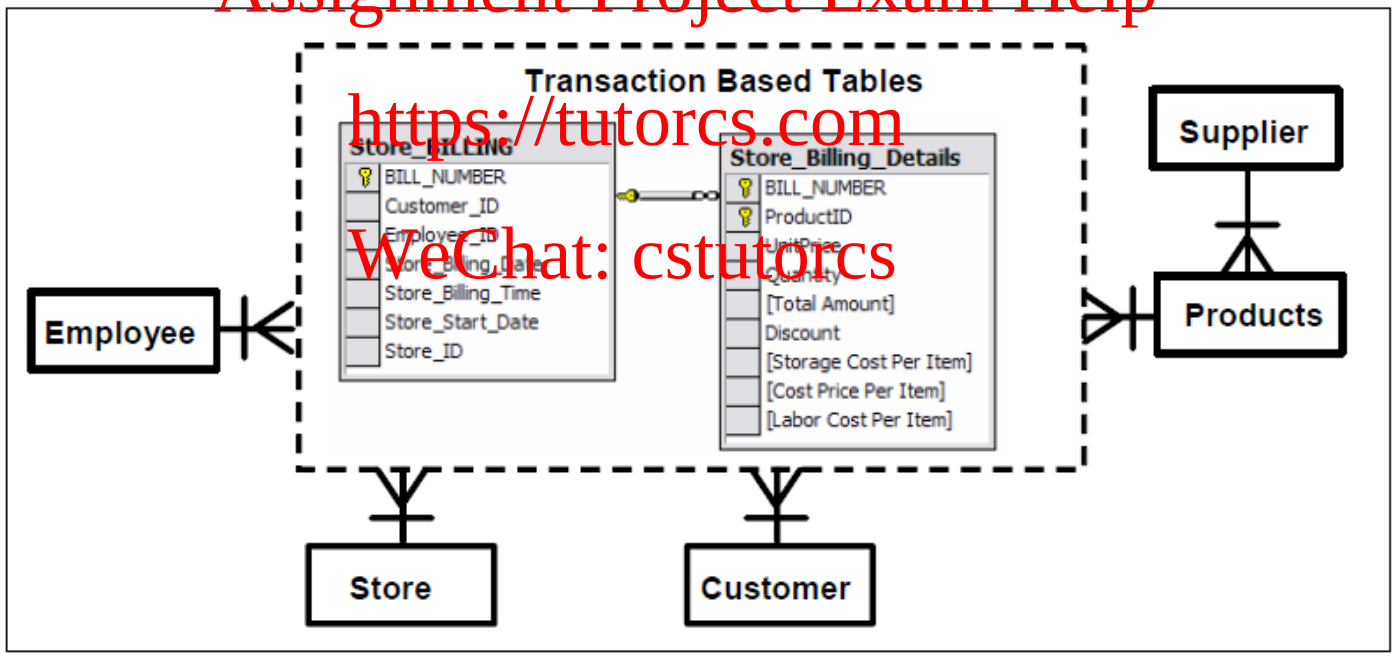
Example of ER/DM conversion



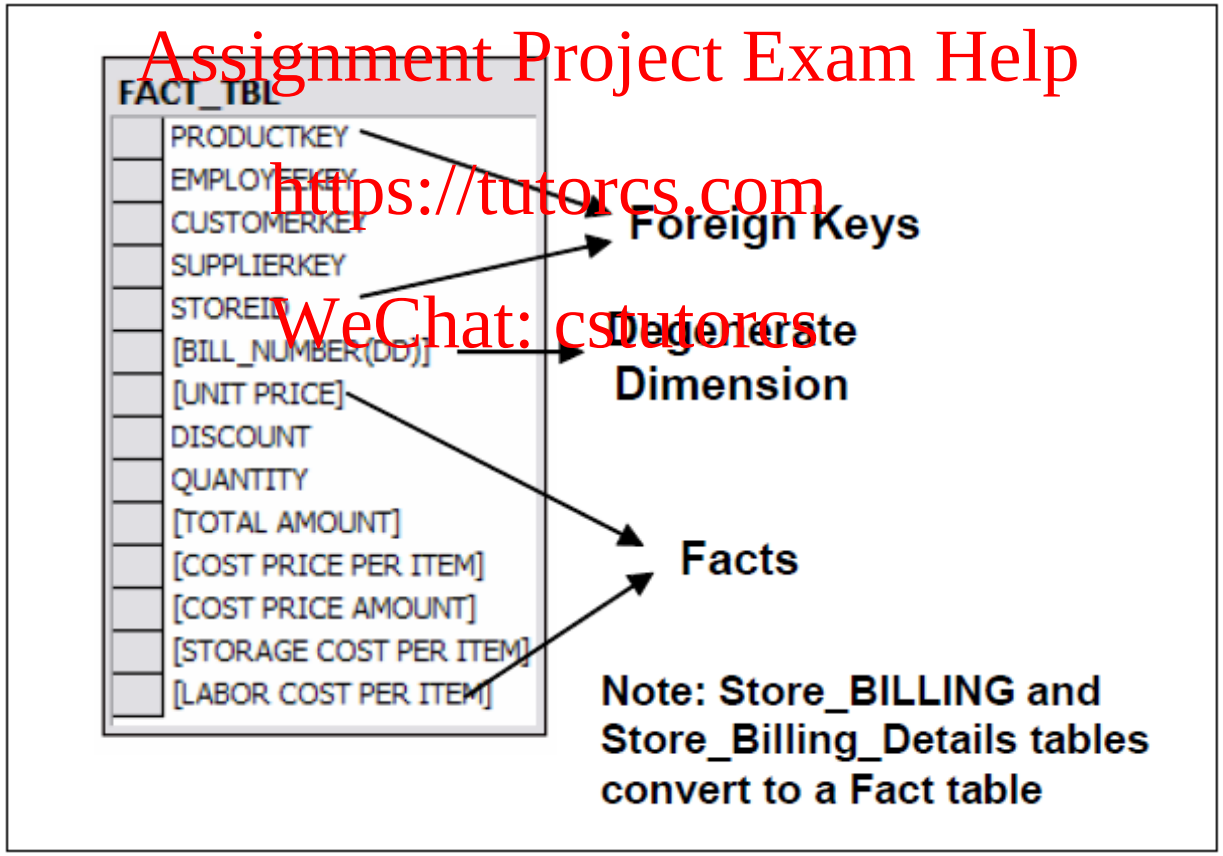
Identify business process – Retail sales



Transaction table, many-to-many



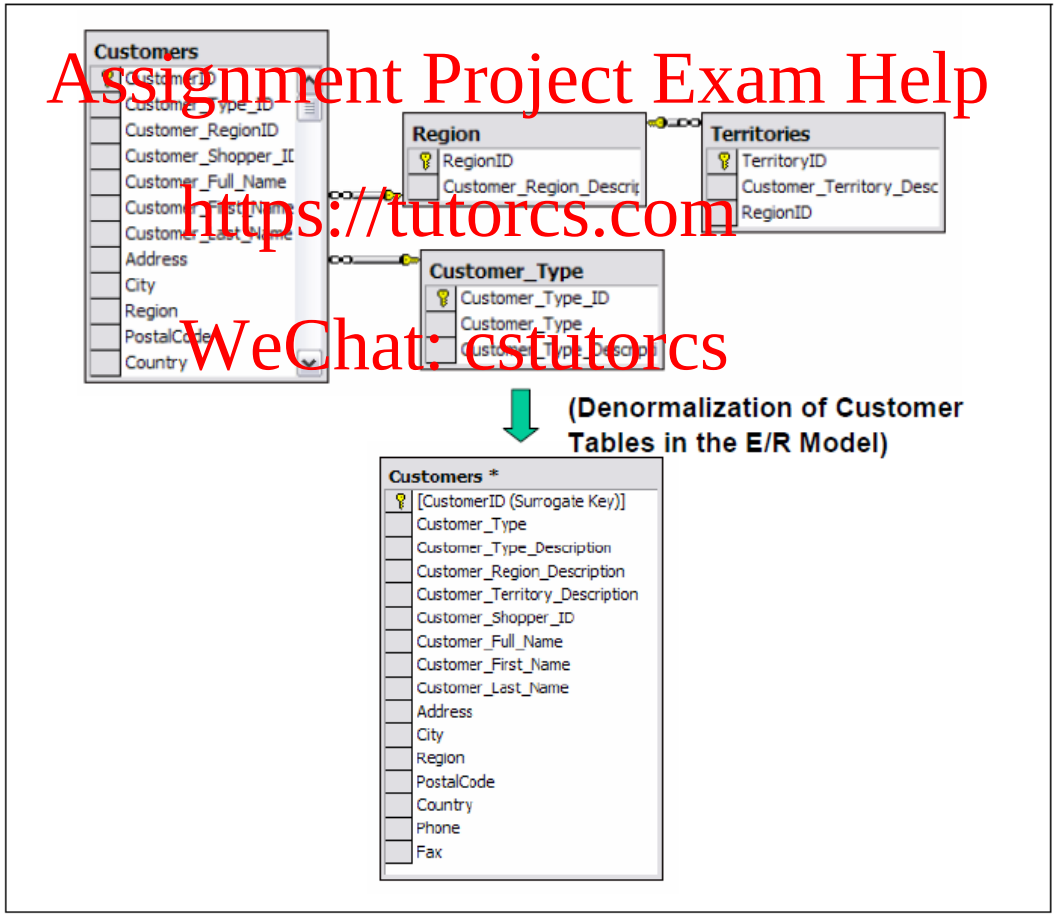
Fact table



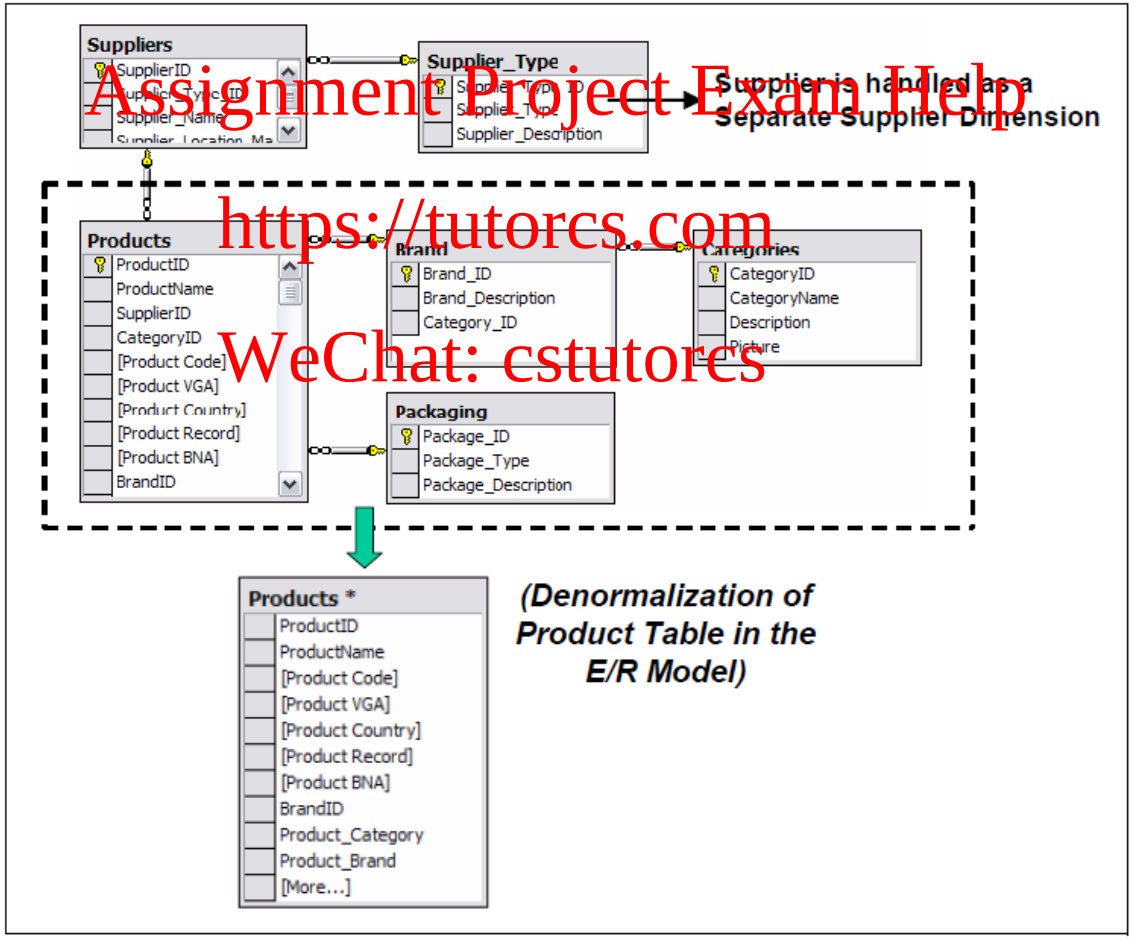
Dimension denormalization

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

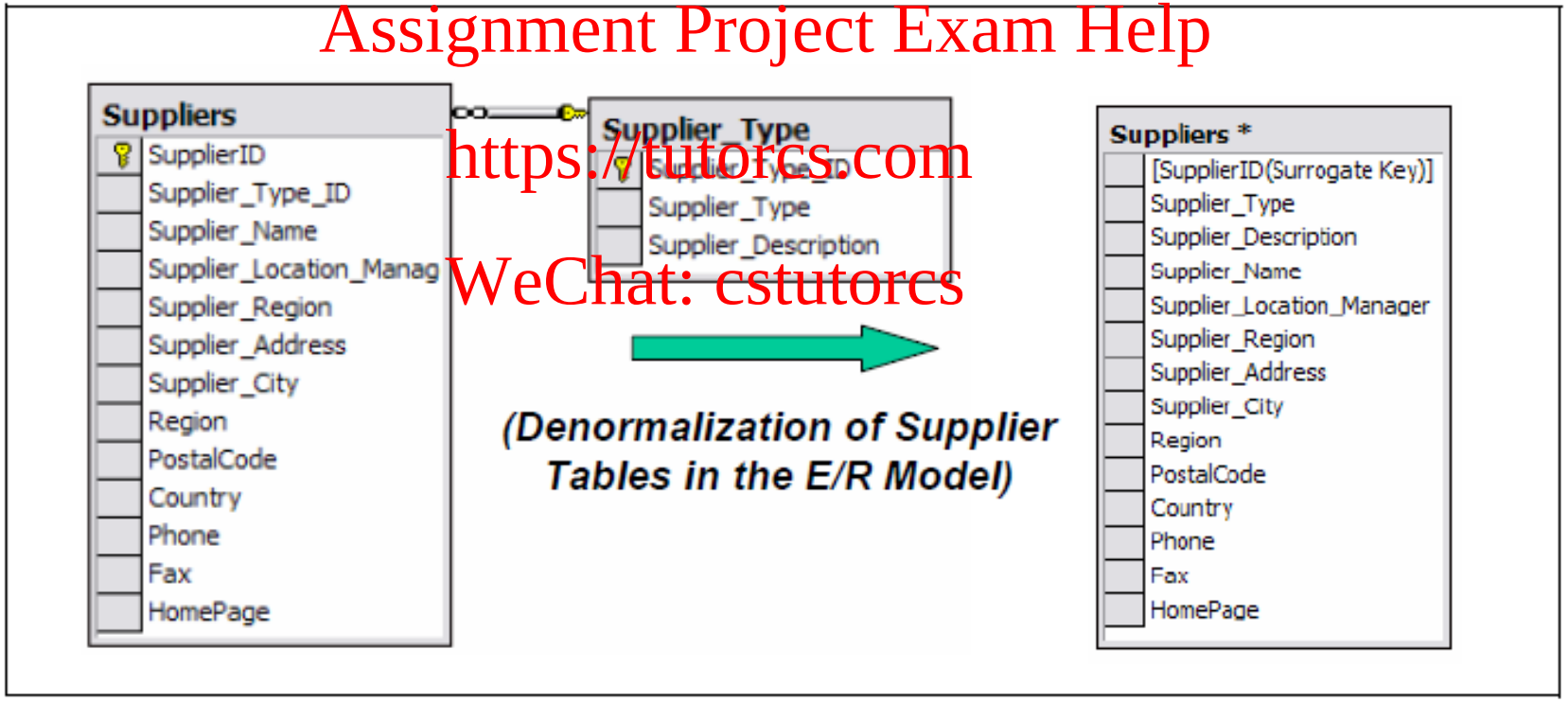
Customers



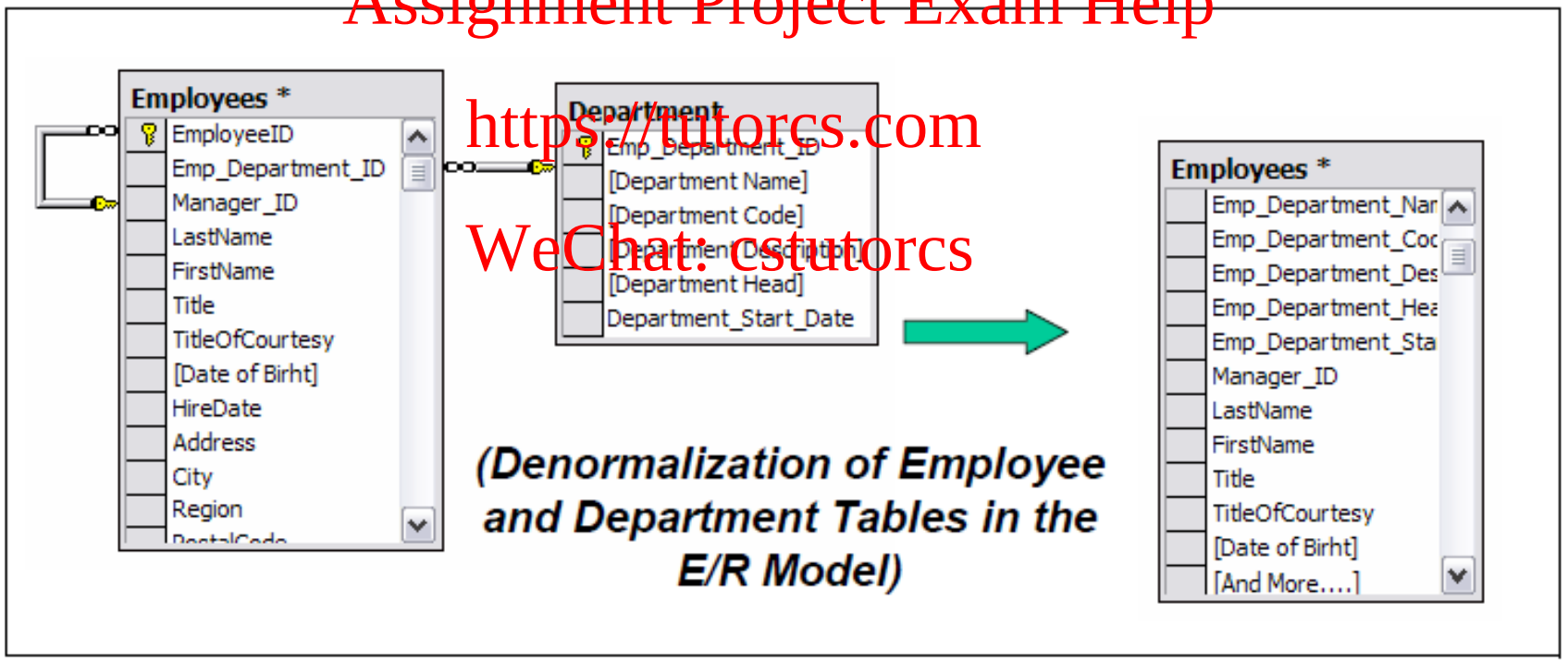
Product



Suppliers

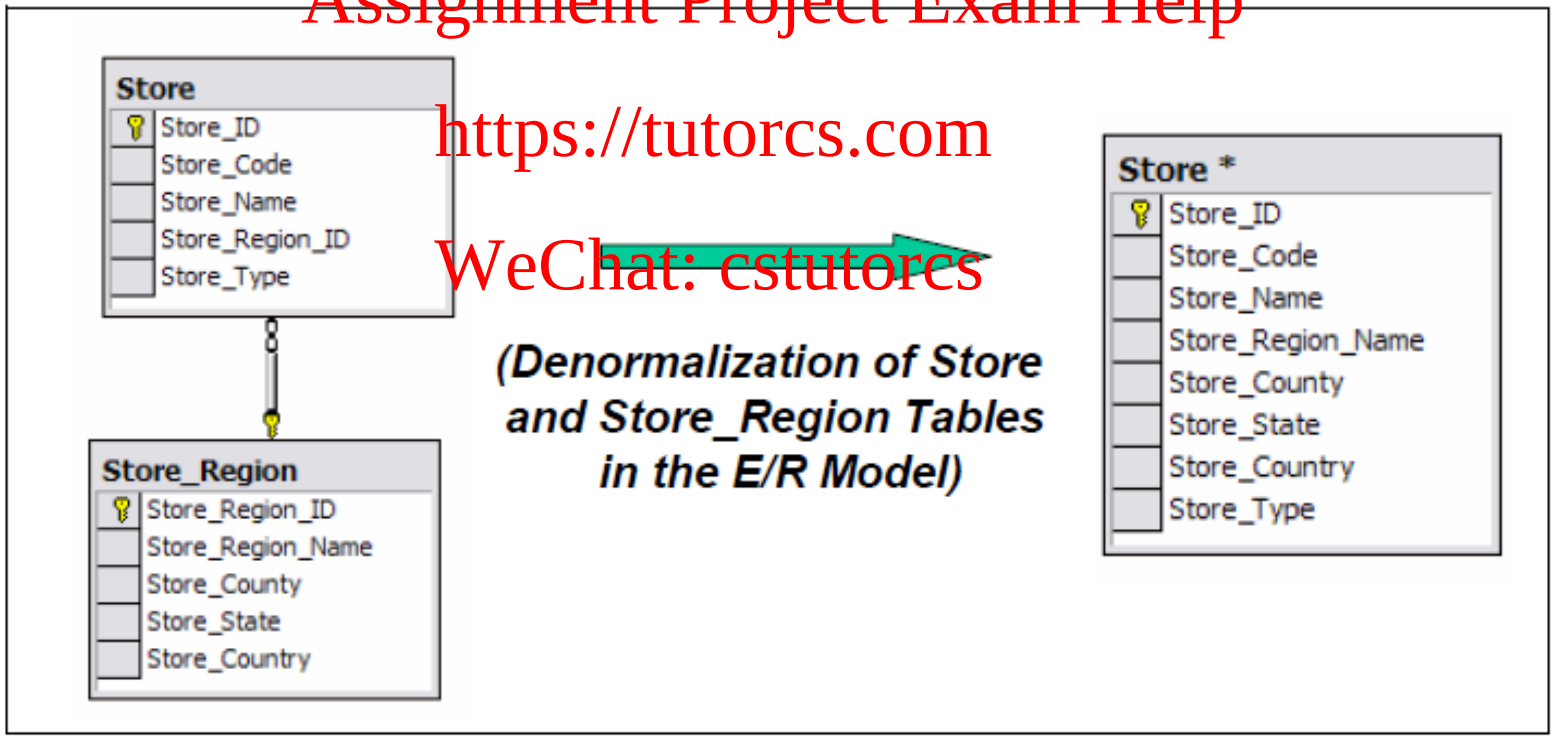


Employee

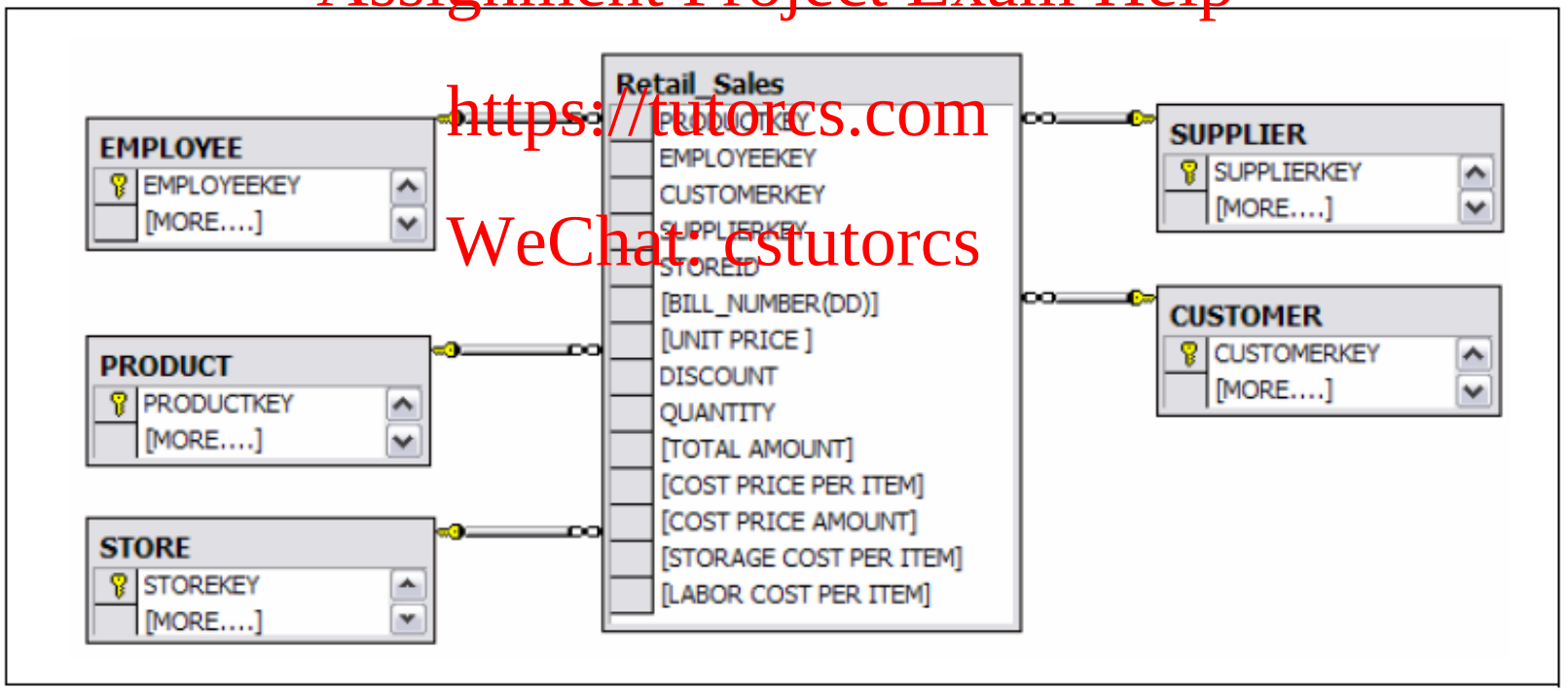


(Denormalization of Employee and Department Tables in the E/R Model)

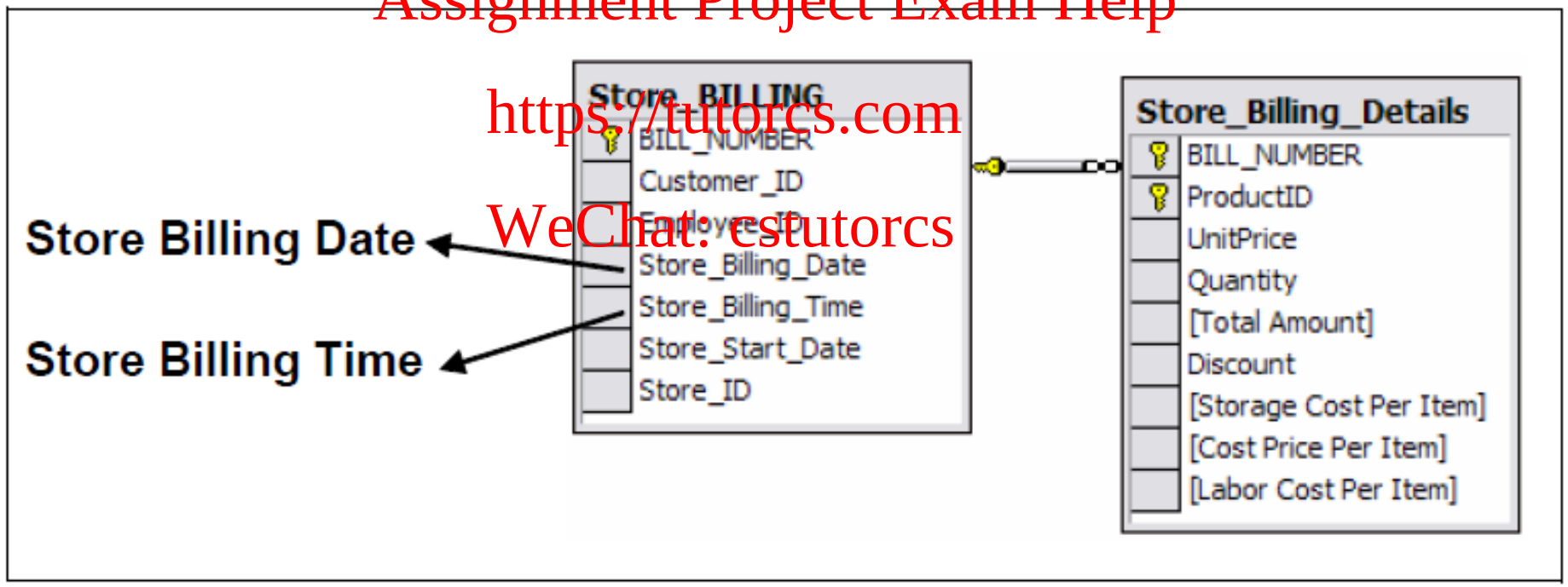
Store



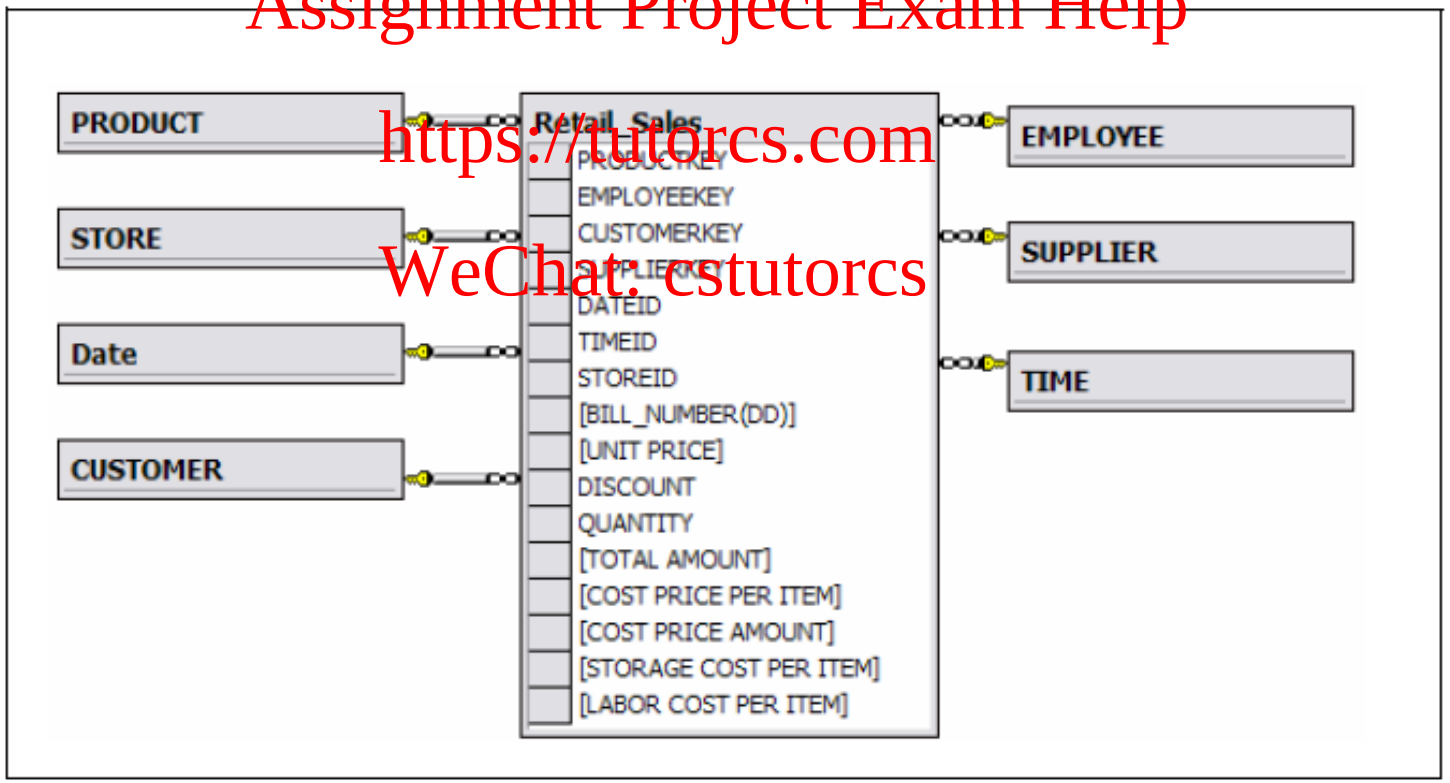
Final star schema



Date and time identification



Final schema

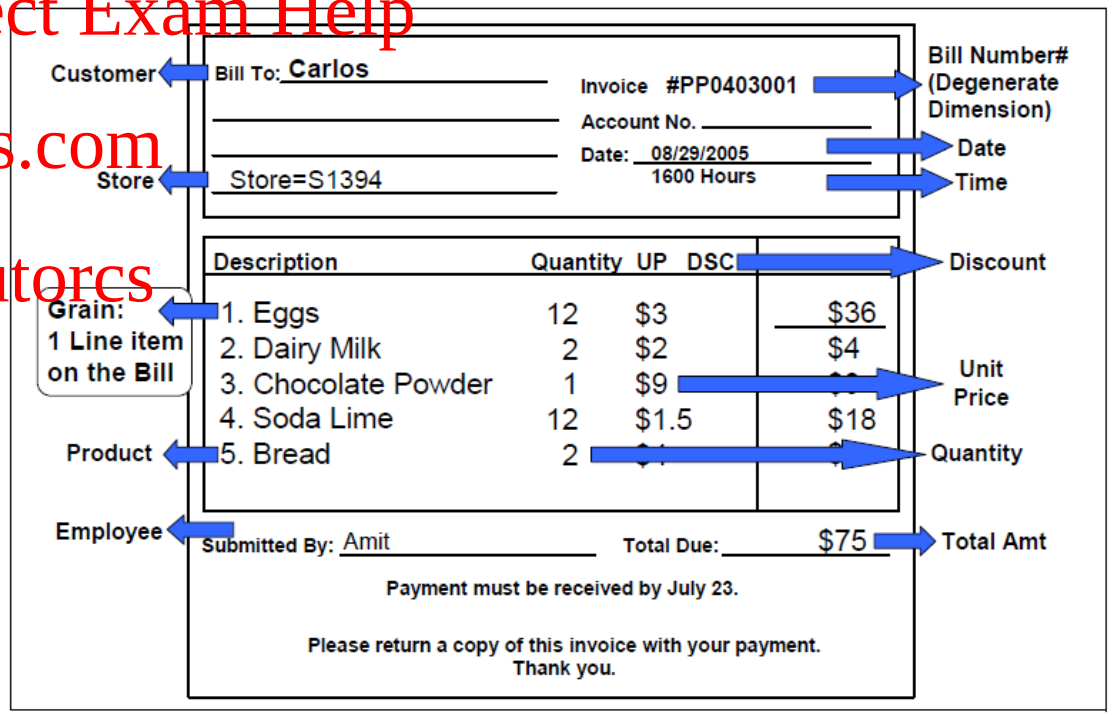


GRAIN

- 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 grains for a single business process



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 DM 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 DM. Do not force fit the different facts and dimensions which belong to different DMs into a single star schema. It is strongly recommended to separate different grains 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

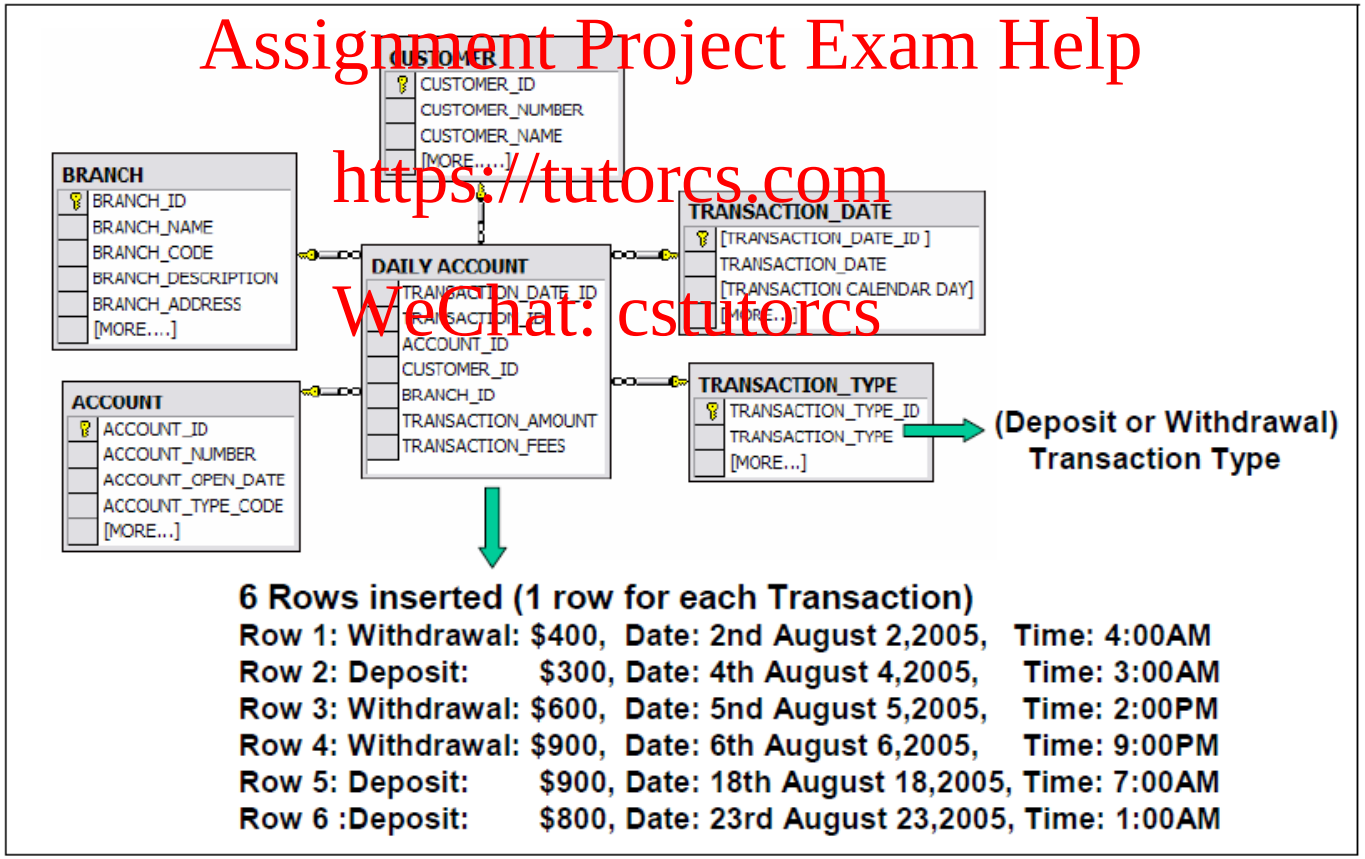
Transaction fact tables

- 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.

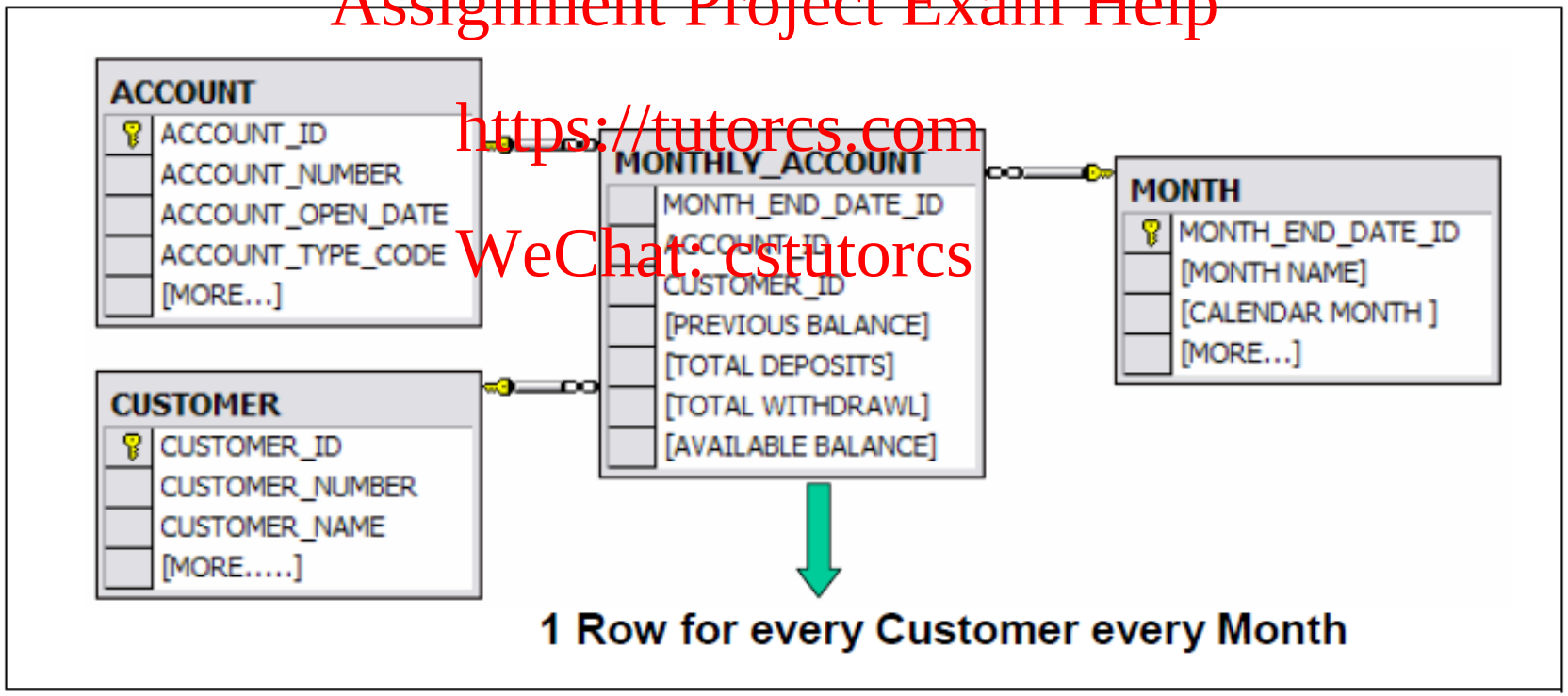
Transaction fact table example



Periodic fact tables

- 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



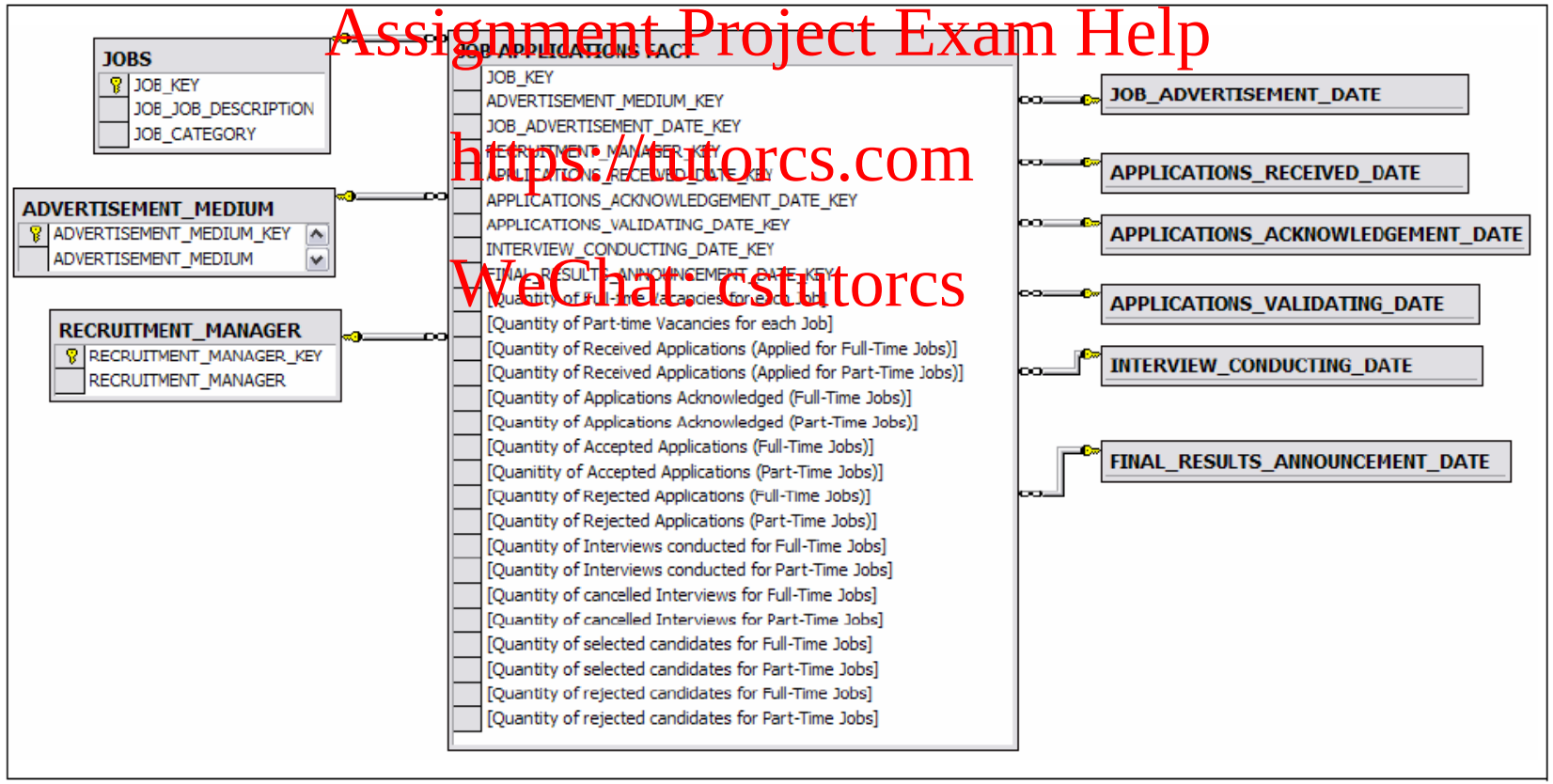
Accumulating fact tables

- 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



Comparison of fact tables

Feature	Transaction fact table	Periodic fact table	Accumulating 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	Involved date dimension at the **lowest granularity**.	Involves date dimension at the **end-of-period granularity**. This could be end of day/week/month/quarter.	Involves **multiple date dimensions** to show the achievement of different milestones.
Conformed dimensions	Uses shared conformed dimensions.	Uses shared conformed dimensions.	Uses shared conformed dimensions.
Total number of dimensions involved	More than periodic fact type.	Less than transaction fact type.	**Highest number of dimensions** when compared to other fact types. Generally, this type of fact table is associated with several date dimension tables based on a single date dimension implemented using a concept of role-playing.

Comparison of fact tables

Feature	Transaction fact table	Periodic fact table	Accumulating fact table
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 facts.
Database size	**Has the biggest size**. If the grain of the transaction is chosen at the most detailed level, these tables tend to grow very fast.	**Smaller than the transaction type** fact table because the grain of the date and time dimension is significantly higher.	**The smallest size** when compared to the other two types of fact tables.
Performance	**Performance is typically good**. However, the performance improves if you chose a grain above the most detailed one because the number of rows decreases.	**Performance is higher than other fact types** of fact table because data is stored at lesser detailed grain. Therefore, this table has fewer rows.	**Performance is typically good**. The selected statements often require differences between two dates to see the time period in days/weeks/months between any two or more activities.

Comparison of fact tables

Feature	Transaction fact table	Periodic fact table	Accumulating fact table
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 fact tables.	Slow in comparison to the transaction and periodic fact tables.
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.)