Advanced Databases

Dimensional modelling and ETL

Giulia Vilone

giulia.vilone@tudublin.ie

Extraction, Transformation & Loading (ETL)

- The “plumbing” work of data warehousing
- Data are moved from source to target data bases
- A very costly, time consuming part of data warehousing

ETL Overview

- To get data out of the source and load it into the data warehouse – simply a process of copying data from one database to other

- Data is extracted from an OLTP database, transformed to match the data warehouse schema and loaded into the data warehouse database
- Many data warehouses also incorporate data from non-OLTP systems such as text files, legacy systems, and spreadsheets; such data also requires extraction, transformation, and loading
- When defining ETL for a data warehouse, it is important to think of ETL as a process, not a physical implementation

ETL Overview

- ETL is often a complex combination of process and technology that consumes a significant portion of the data warehouse development efforts and requires the skills of business analysts, database designers, and application developers

- It is not a one-time event as new data is added to the data warehouse periodically – monthly, daily, hourly

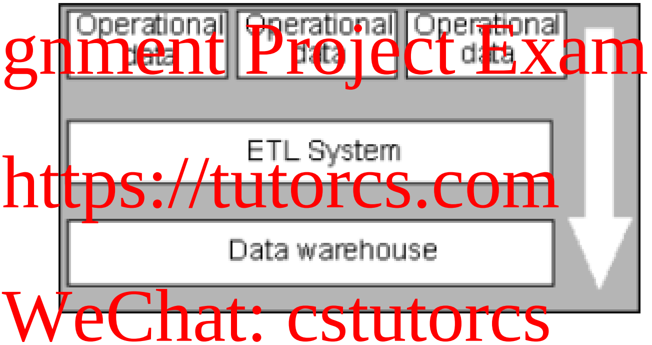
- Because ETL is an integral, ongoing, and recurring part of a data warehouse
  - Automated
  - Well documented
  - Easily changeable

Extraction

Data extraction

- Often performed by routines (not recommended because of high program maintenance and no automatically generated meta data)
- Sometimes source data is copied to the target database using the replication capabilities of standard RDMS (not recommended because of “dirty data” in the source systems)
- Increasing performed by specialized ETL software

Data extraction

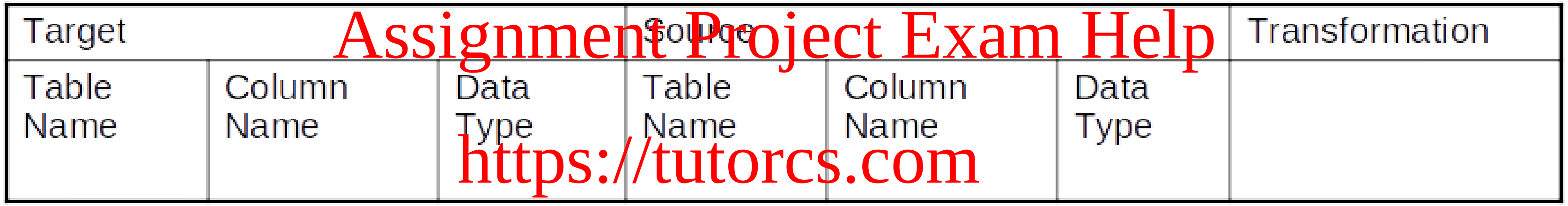


- The integration of all of the disparate systems across the enterprise is the real challenge to getting the data warehouse to a state where it is usable
- Data is extracted from heterogeneous data sources
- Each data source has its distinct set of characteristics that need to be managed and integrated into the ETL system in order to effectively extract data.

Data extraction

- ETL process needs to effectively integrate systems that have different:
  - **DBMS**
  - **Operating Systems**
  - **Hardware**
  - **Communication protocols**
- Need to have a **logical data map** before the physical data can be transformed
- The logical data map **describes the relationship** between the extreme starting points and the extreme ending points of your ETL system usually presented in a table or spreadsheet

Logical mapping



Target			Source			Transformation
Table Name	Column Name	Data Type	Table Name	Column Name	Data Type	

- The content of the logical data mapping document has been proven to be the critical element required to efficiently plan ETL processes
- The table type gives us our queue for the ordinal position of our data load processes—first dimensions, then facts.
- The primary purpose of this document is to provide the ETL developer with a clear-cut blueprint of exactly what is expected from the ETL process. This table must depict, without question, the course of action involved in the transformation process
- The transformation can contain anything from the absolute solution to nothing at all. Most often, the transformation can be expressed in SQL. The SQL may or may not be the complete statement

Transformation

Data staging

- Often used as an interim step between data extraction and later steps

- Accumulates data from asynchronous sources using native interfaces, flat files, FTP sessions, or other processes

- At a predefined cutoff time, data in the staging file is transformed and loaded to the warehouse
- There is usually no end user access to the staging file
- An operational data store may be used for data staging

Data transformation

- Transforms the data in accordance with the business rules and standards that have been established
- Example include: format changes, deduplication, splitting up fields, replacement of codes, derived values, and aggregates

Data transformation

- Main step where the ETL adds value
- Actually changes data and provides guidance whether data can be used for its intended purposes
- Performed in staging area

Data transformation

Data Quality paradigm:

- Correct
- Unambiguous
- Consistent
- Complete
- Data quality checks are run at 2 places - after extraction and after cleaning and confirming additional check are run at this point

Data cleansing

- Source systems contain "dirty data" that must be cleansed
- ETL software contains rudimentary data cleansing capabilities
- Specialized data cleansing software is often used. Important for performing name and address correction and householding functions
- Leading data cleansing vendors include Vality (Integrity), Harte-Hanks (Trillium), and Firstlogic (i.d.Centric)

Reasons for "dirty" data

- Dummy Values
- Absence of Data
- Multipurpose Fields
- Cryptic Data
- Contradicting Data
- Inappropriate Use of Address Lines
- Violation of Business Rules
- Reused Primary Keys
- Non-Unique Identifiers
- Data Integration Problems

Steps in data cleansing

- Parsing
- Correcting
- Standardizing
- Matching
- Consolidating

Parsing

- Parsing locates and identifies individual data elements in the source files and then isolates these data elements in the target files.
- Examples include parsing the first, middle, and last name; street number and street name; and city and state.

Correcting

- Corrects parsed individual data components using sophisticated data algorithms and secondary data sources.
- Example include replacing a vanity address and adding a zip code.

Standardizing

- Standardizing applies conversion routines to transform data into its preferred (and consistent) format using both standard and custom business rules.
- Examples include adding a pre name, replacing a nickname, and using a preferred street name.

Matching

- Searching and matching records within and across the parsed, corrected and standardized data based on predefined business rules to eliminate duplications.
- Examples include identifying similar names and addresses.

Consolidating

- Analysing and identifying relationships between matched records and consolidating/merging them into ONE representation.

Data transformation examples

- Selecting only certain columns to load (or selecting null columns not to load)
- Translating coded values (e.g., if the source system stores 1 for male and 2 for female, but the warehouse stores M for male and F for female), this calls for automated data cleansing; no manual cleansing occurs during ETL
- Encoding free-form values (e.g., mapping "Male" to "1" and "Mr" to M)
- Deriving a new calculated value (e.g., sale_amount = qty * unit_price)
- Sorting
- Joining data from multiple sources (e.g., lookup, merge)
- Aggregation (for example, rollup — summarizing multiple rows of data — total sales for each store, and for each region, etc.)
- Transposing or pivoting (turning multiple columns into multiple rows or vice versa)
- Splitting a column into multiple columns (e.g., putting a comma-separated list specified as a string in one column as individual values in different columns)
- Disaggregation of repeating columns into a separate detail table (e.g., moving a series of addresses in one record into single addresses in a set of records in a linked address table)
- Applying any form of simple or complex data validation. If validation fails, it may result in a full, partial or no rejection of the data, and thus none, some or all the data is handed over to the next step, depending on the rule design and exception handling.

Loading

Loading dimensions

Loading facts

Data loading

- Data are physically moved to the data warehouse.
- The loading takes place within a “load window”.
- The trend is to near real time updates of the data warehouse as the warehouse is increasingly used for operational applications.

Loading dimensions

- The primary key is a single field containing meaningless unique integer – Surrogate Keys.
- The DW owns these keys and never allows any other entity to assign them.
- De-normalized flat tables – all attributes in a dimension must take on a single value in the presence of a dimension primary key.
- Should possess one or more other fields (attributes).

Loading dimensions

- The data loading module consists of all the steps required to administer slowly changing dimensions (SCD) and write the dimension to disk as a physical table in the proper dimensional format with correct primary keys, and final descriptive attributes.

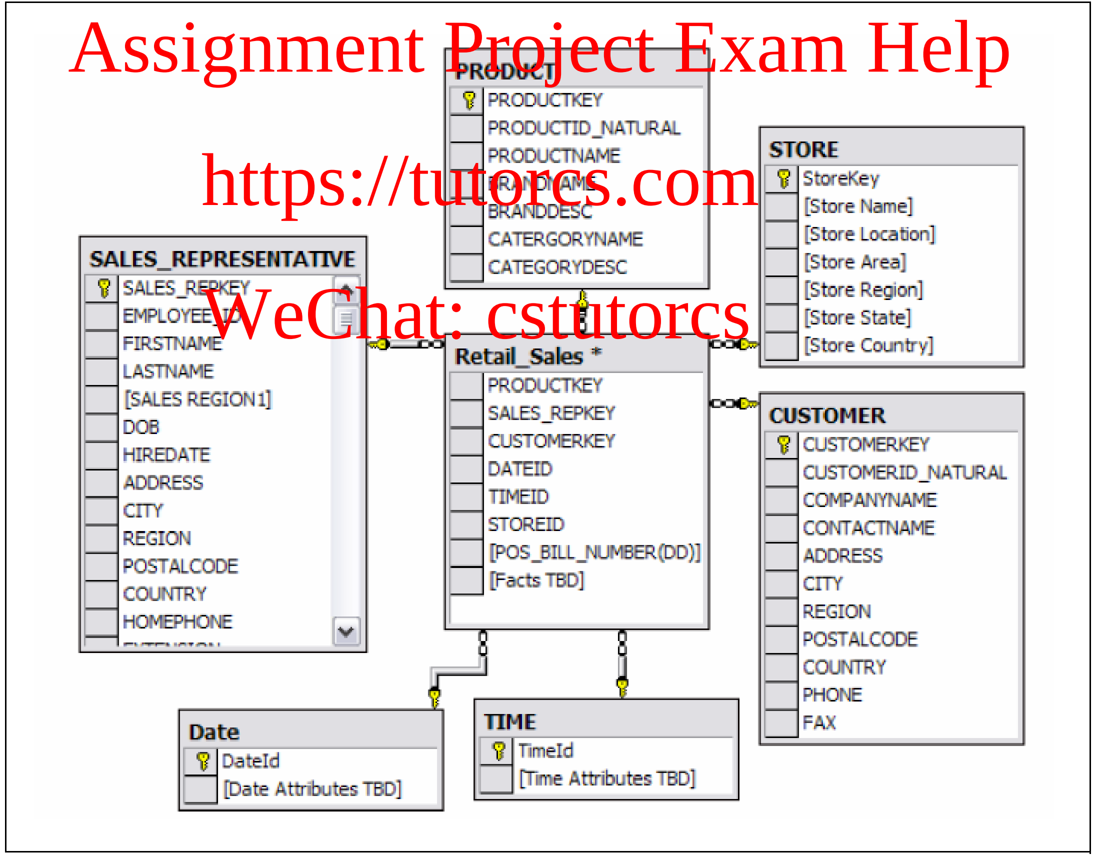
- Creating and assigning the surrogate keys occur in this module.

Loading slow changing dimensions

Loading dimensions that change

- When DW receives notification that an existing row in dimension has changed it gives out 3 types of responses:
  - **Type 1**
  - **Type 2**
  - **Type 3**

Example



PRODUCT
- PRODUCTKEY
- PRODUCTID_NATURAL
- PRODUCTNAME
- BRANDNAME
- BRANDDESC
- CATERGORYNAME
- CATEGORYDESC

STORE
- StoreKey
- [Store Name]
- [Store Location]
- [Store Area]
- [Store Region]
- [Store State]
- [Store Country]

SALES_REPRESENTATIVE
- SALES_REPKEY
- EMPLOYEE_ID
- FIRSTNAME
- LASTNAME
- [SALES REGION1]
- DOB
- HIREDATE
- ADDRESS
- CITY
- REGION
- POSTALCODE
- COUNTRY
- HOMEPHONE
- EXTENSION

Retail_Sales *
- PRODUCTKEY
- SALES_REPKEY
- CUSTOMERKEY
- DATEID
- TIMEID
- STOREID
- [POS_BILL_NUMBER(DD)]
- [Facts TBD]

CUSTOMER
- CUSTOMERKEY
- CUSTOMERID_NATURAL
- COMPANYNAME
- CONTACTNAME
- ADDRESS
- CITY
- REGION
- POSTALCODE
- COUNTRY
- PHONE
- FAX

Date
- DateId
- [Date Attributes TBD]

TIME
- TimeId
- [Time Attributes TBD]

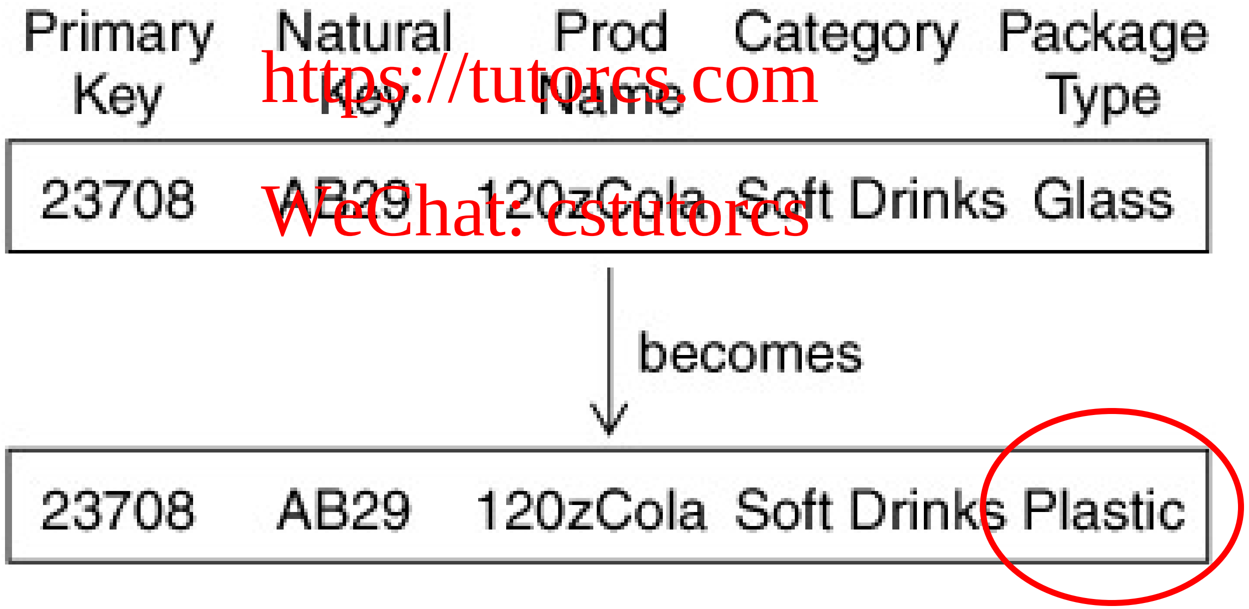
Type 1 – Overwriting the history

- A Type-1 approach overwrites the existing dimensional attribute with new data, and therefore no history is preserved.

SKEY	EMPL_ID	LASTNAME	FIRSTNAME	REGION
976735	DF134A	Seles	Monnica	California

- Spelling error (it makes sense).
- From California to New Jersey?

Type 1 dimension



Primary Key	Natural Key	Prod Name	Category	Package Type
23708	AB29	120zCola	Soft Drinks	Glass

becomes

23708	AB29	120zCola	Soft Drinks	Plastic

Type 1 – Summary

Type-1 approach	Description
When to use the Type-1 change handling approach	• This may be the best approach to use if the attribute change is simple, such as a correction in spelling. And, if the old value was wrong, it may not be critical that history is not maintained. • It is also appropriate if the business does not need to track changes for specific attributes of a particular dimension.
Advantages of the Type-1 change handling approach	• It is the easiest and most simple to implement. • It is extremely effective in those situations requiring the correction of bad data. • No change is needed to the structure of the dimension table.
Disadvantages of the Type-1 change handling approach	• All history may be lost if this approach is used inappropriately. It is typically not possible to trace history • All previously made aggregated tables need to be rebuilt.
Impact on existing dimension table structure	• No impact. The table structure does not change.
Impact on pre-existing aggregations	• Any pre-existing aggregations based on the old attribute value will need to be rebuilt. For example, when correcting the spelling of the FIRSTNAME of Monika (correct name), any pre-existing aggregations based on the incorrect value Monnica will need to be rebuilt.
Impact on database size	• No impact on database size.

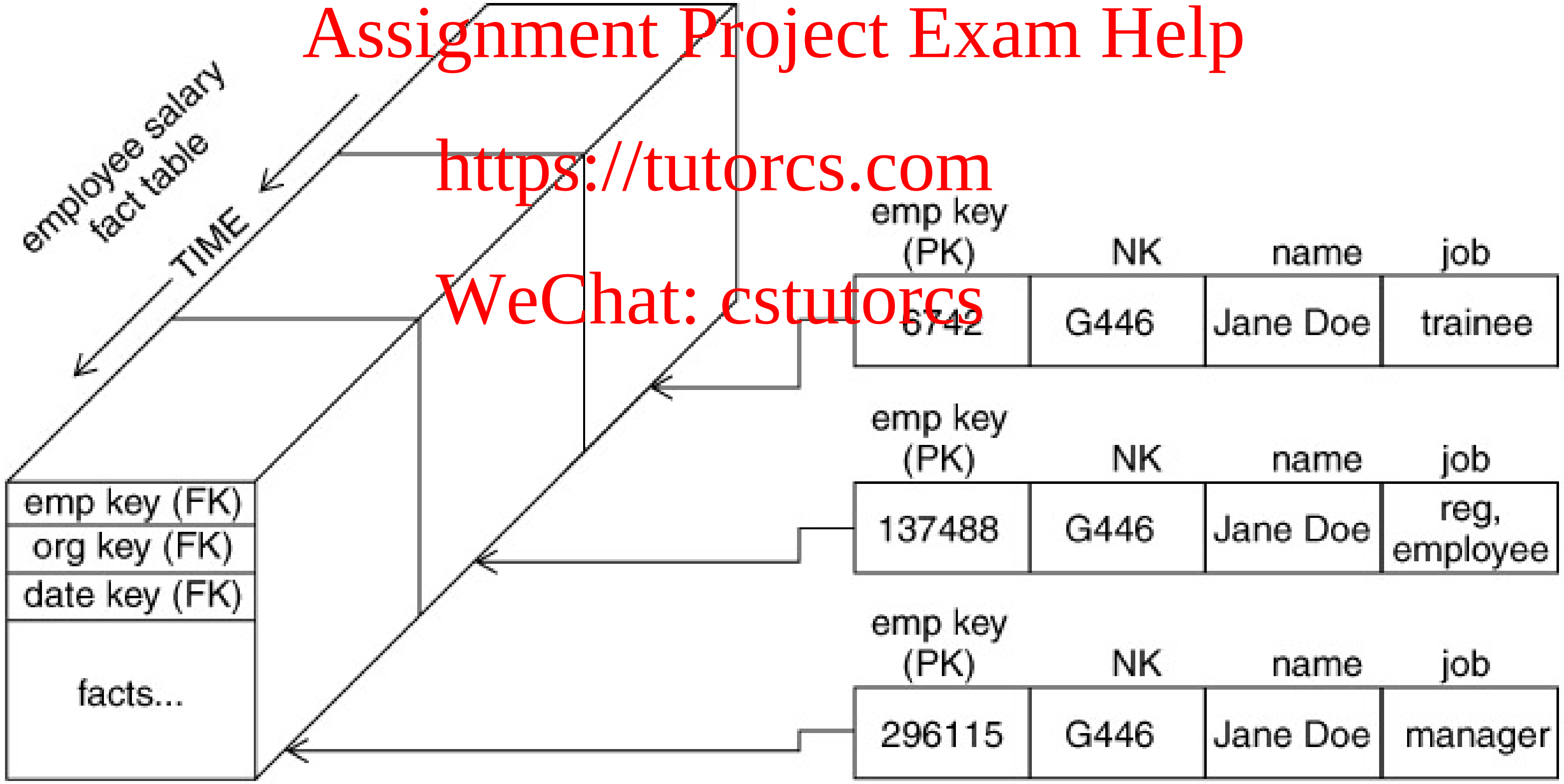
Type 2 – Preserving history

- Type-2 adds a new dimension row.

SKEY	EMPL_ID	LASTNAME	FIRSTNAME	REGION
976735	DF134A	Seles	Monika	California
976736	DF134A	Seles	Monika	New Jersey

- Facts related to Monica are now <u>partitioned.</u>
- Note the use of Surrogate Key.
- The problem of expiration date (when the change happened?).

Type-2 dimensions



emp key (PK)	NK	name	job
6742	G446	Jane Doe	trainee

emp key (PK)	NK	name	job
137488	G446	Jane Doe	reg, employee

emp key (PK)	NK	name	job
296115	G446	Jane Doe	manager

Expiration date

- There is no need to include effective date attributes inside the dimension
- The date/time dimension will find the right data
- Granularity is important
- Effective and expiration date attributes are necessary in the staging area because we need to know which surrogate key is valid when loading historical fact records.
- You may still use them as helpful extras that are not required for the basic partitioning of history.
- You can get the same answer by partitioning history using the date or time dimension of your dimensional designs.

Type 2 – Summary

Type-2 approach	Description
When to use the Type-2 change handling approach	• When there is need to track an unknown number of historical changes to dimensional attributes.
Advantages of the Type-2 change handling approach	• Enables tracking of all historical information accurately and for an *infinite* number of changes.
Disadvantages of the Type-2 change handling approach	• Causes the size of the dimension table to grow fast. In cases where the number of rows being inserted is very high, then storage and performance of the dimensional model may be affected. • Complicates the ETL process needed to load the dimensional model. ETL-related activities that are required in the type-2 approach include maintenance of effective and expiration date attributes in the staging area.
Impact on existing dimension table structure	• No changes to dimensional structure needed. • Additional columns for effective and expiration dates are not needed in the dimension table.
Impact on pre-existing aggregations	• There is no impact on the pre-aggregated tables. The aggregated tables are not required to be rebuilt as with the type-1 approach.

Type 2 – Summary

Type-2 approach	Description
Impact on database size	• Yes, accelerates the dimensional table growth because with each change in a dimensional attribute, a new row is inserted into the dimension table.
Adding effective and expiration dates to dimension tables	No. This is not necessary in the dimension tables. However, effective and expiration attributes are needed in the staging area because we need to know which surrogate key is valid when we are loading historical fact rows. In the dimension table, we stated that the effective and expiration dates are not needed though you may still use them as helpful extras that are not required for the basic partitioning of history. It is important to note that in case you add the effective and expiration date attributes in the dimension table, then there is no need to constrain on the effective and expiration date in the dimension table in order to get the right answer. You could get the same answer by partitioning history using the date or time dimension of your dimensional designs.

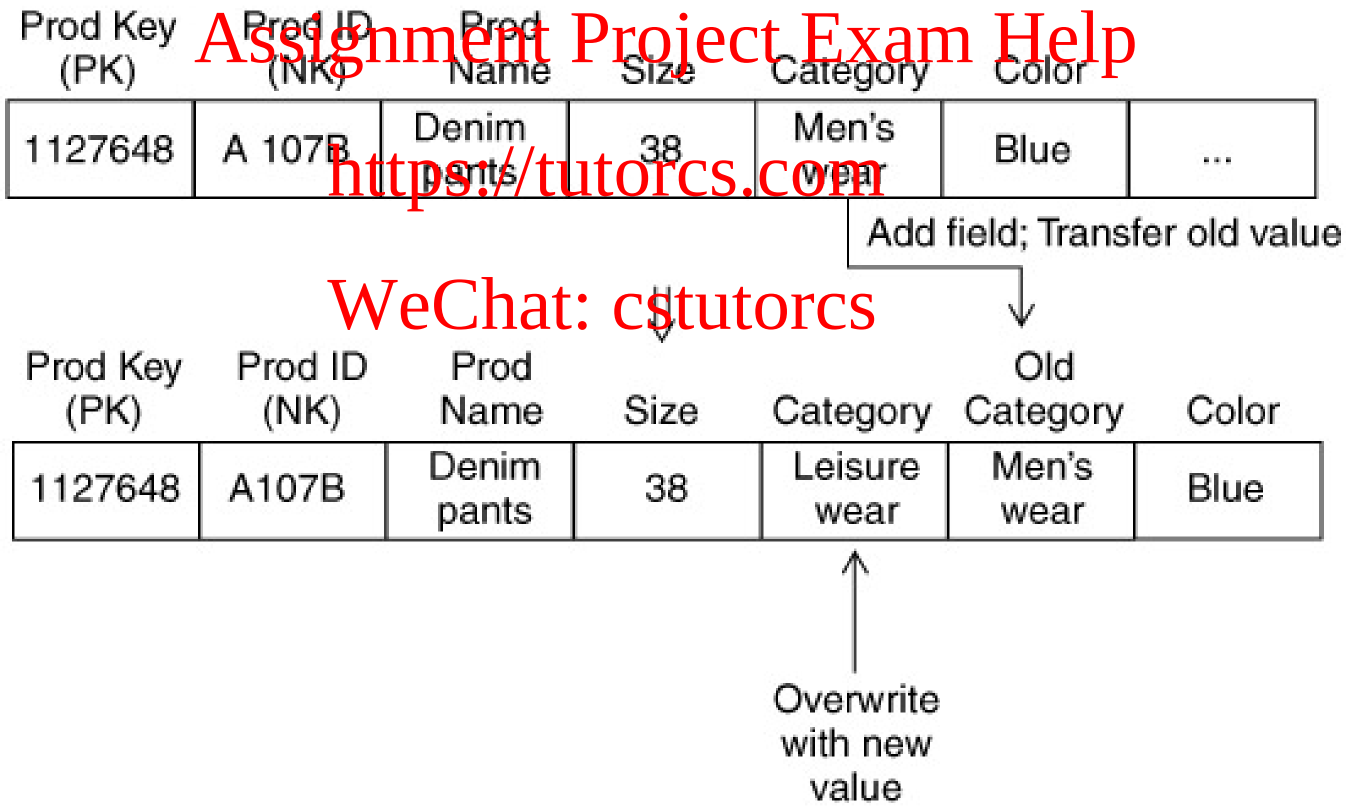
Type 3 – Preserving limited history

- The type-3 approach is typically used only if there is a limited need to preserve and accurately describe history. An example is when someone gets married and there is a need to retain the original surname of the person.

SKEY	EMPL_ID	OLDLNAME	NEWLNAME	FIRSTNAME	REGION
976735	DF134A	Seles	Sampras	Monika	California

- With this design I only keep one change, I need to add more fields to track more changes.
- The type-3 approach enables us to see new and historical fact in the table rows by either the new or prior attribute values.

Type-3 dimensions



Prod Key (PK)	Prod ID (NK)	Prod Name	Size	Category	Color	
1127648	A 107B	Denim pants	38	Men's wear	Blue	...

Add field; Transfer old value

Prod Key (PK)	Prod ID (NK)	Prod Name	Size	Category	Old Category	Color
1127648	A107B	Denim pants	38	Leisure wear	Men's wear	Blue

Overwrite with new value

Type 3 – Summary

Type-2 approach	Description
When to use the Type-3 change handling approach	• Should only be used when it is necessary for the data warehouse to track historical changes, and when such changes will only occur for a finite number of times. If the number of changes can be predicted, then the dimension table can be modified to place additional columns to track the changes.
Advantages of the Type-3 change handling approach	• Does not increase the size of the table as compared to the type-2 approach, since new information is updated. • Allows us to keep part of history. This is equivalent to the number of changes we can predict. Such prediction helps us modify the dimension table to accommodate new columns.
Disadvantages of the Type-3 change handling approach	• Does not maintain all history when an attribute is changed more often than the number in the predicted range, because the dimension table is designed to accommodate a finite number of changes. • If we designed a dimension table assuming a fixed number of changes, then needed more, then we would have to redesign or risk losing history.

Type 3 – Summary

Type-2 approach	Description
Impact on existing dimension table structure	• The dimension table is modified to add columns. • The number of columns added depends on the number of changes to be tracked.
Impact on pre-existing aggregations	• You may be required to rebuild the pre-aggregated tables.
Impact on database size	• No impact is there since data is only updated.

Type of facts

Non-additive

They cannot be added meaningfully:

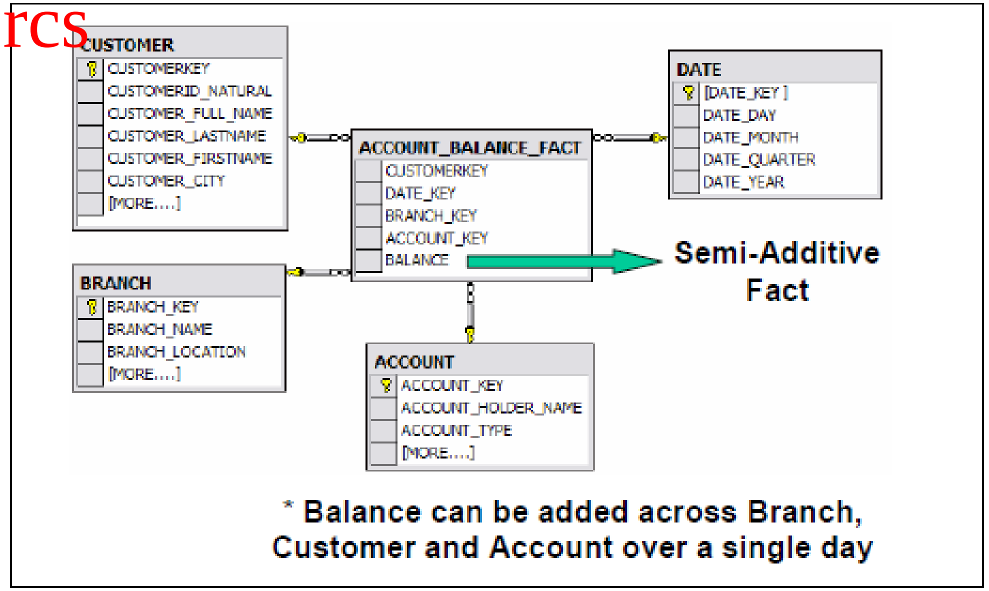
- **Textual facts**: cannot add, but count
- **Per-Unit Prices**: can add only if you know number of units sold
- **Percentage and Ratios**: non additive, keep denominator and numerator when possible
- **Measure of Intensity**: room temperature
- **Averages**
- **Degenerate Numbers**: order numbers….

Semi-additive

- Additive across some dimensions but not on all of them

- Account Balances:
  - The balance is not additive across time
  - It is obviously possible across customers

Quantity on hand!
Date no, Product Store yes



CUSTOMER
CUSTOMERKEY
CUSTOMERID_NATURAL
CUSTOMER_FULL_NAME
CUSTOMER_LASTNAME
CUSTOMER_FIRSTNAME
CUSTOMER_CITY
[MORE....]

ACCOUNT_BALANCE_FACT
CUSTOMERKEY
DATE_KEY
BRANCH_KEY
ACCOUNT_KEY
BALANCE

DATE
[DATE_KEY]
DATE_DAY
DATE_MONTH
DATE_QUARTER
DATE_YEAR

BRANCH
BRANCH_KEY
BRANCH_NAME
BRANCH_LOCATION
[MORE....]

ACCOUNT
ACCOUNT_KEY
ACCOUNT_HOLDER_NAME
ACCOUNT_TYPE
[MORE....]

Semi-Additive Fact

* Balance can be added across Branch, Customer and Account over a single day

Loading facts

Fact tables hold the measurements of an enterprise. The relationship between fact tables and measurements is extremely simple. If a measurement exists, it can be modelled as a fact table row. If a fact table row exists, it is a measurement.

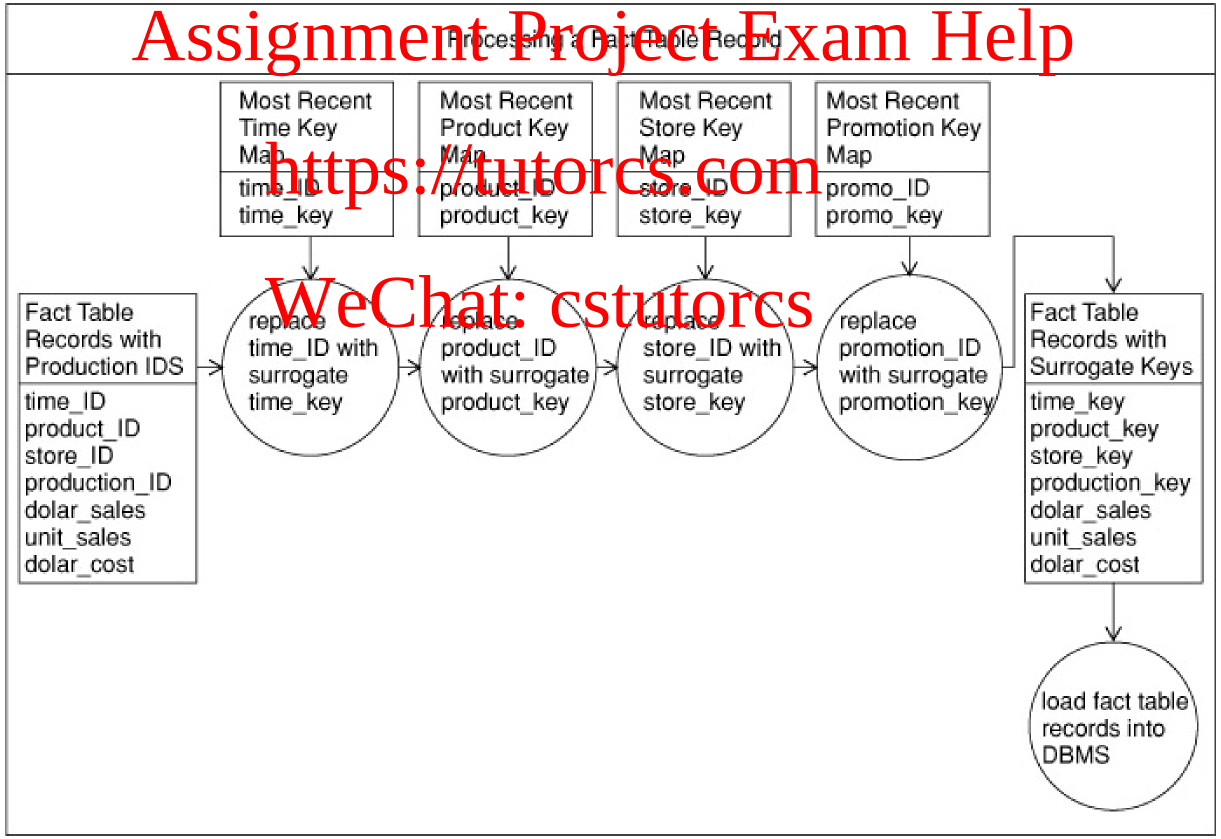
Key building process - Facts

- When building a fact table, the final ETL step is converting the natural keys in the new input records into the correct, contemporary surrogate keys.

- ETL maintains a special surrogate key lookup table for each dimension. This table is updated whenever a new dimension entity is created and whenever a **Type 2** change occurs on an existing dimension entity.

- All the required lookup tables should be pinned in memory so that they can be randomly accessed as each incoming fact record presents its natural keys. This is one of the reasons for making the lookup tables separate from the original data warehouse dimension tables.

Key building process



Processing a Fact Table Record

Most Recent Time Key Map
time_ID
time_key

Most Recent Product Key Map
product_ID
product_key

Most Recent Store Key Map
store_ID
store_key

Most Recent Promotion Key Map
promo_ID
promo_key

Fact Table Records with Production IDS
time_ID
product_ID
store_ID
production_ID
dolar_sales
unit_sales
dolar_cost

replace time_ID with surrogate time_key

replace product_ID with surrogate product_key

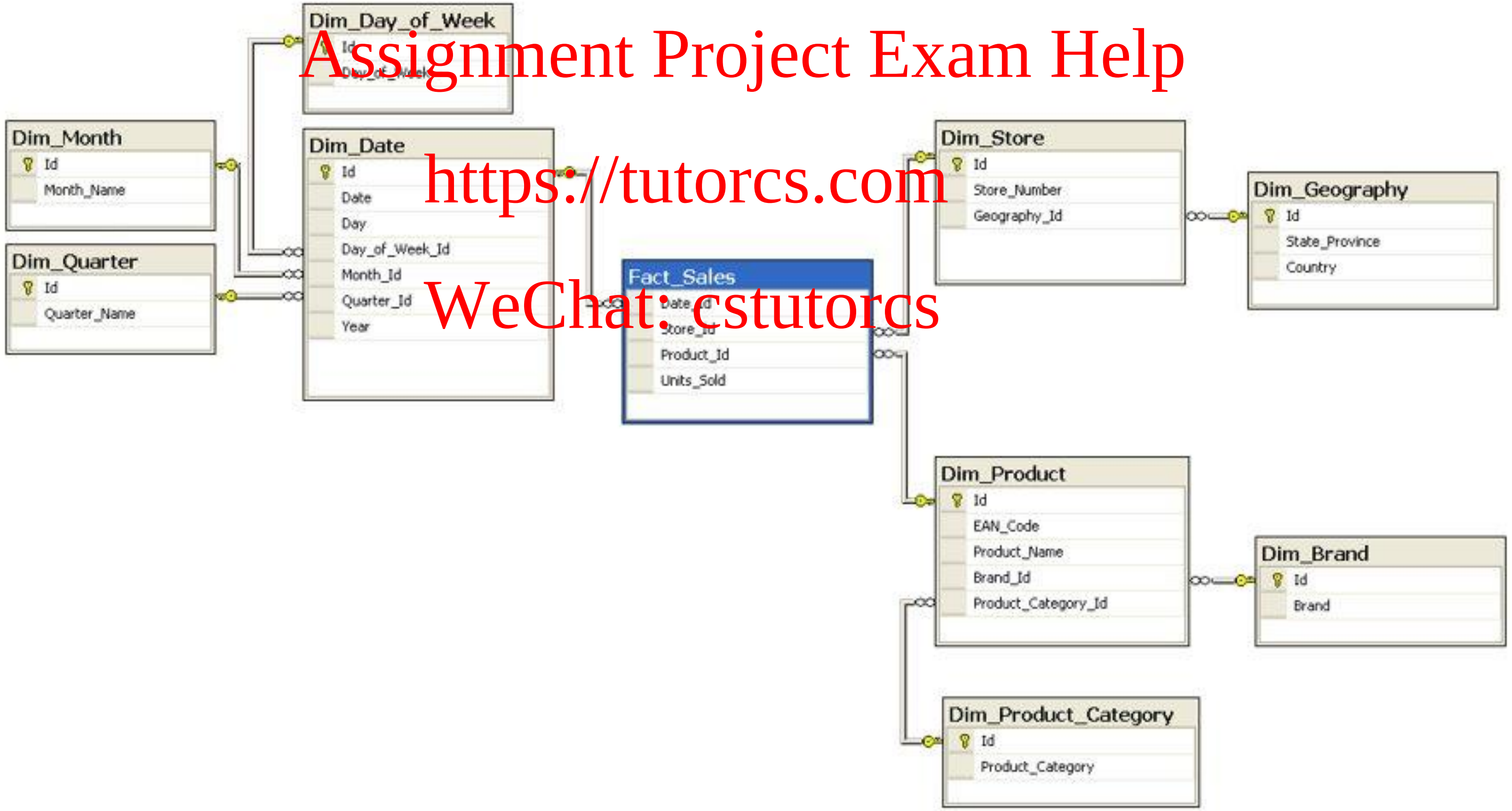
replace store_ID with surrogate store_key

replace promotion_ID with surrogate promotion_key

Fact Table Records with Surrogate Keys
time_key
product_key
store_key
production_key
dolar_sales
unit_sales
dolar_cost

load fact table records into DBMS

Facts and dimensions



Query examples

```sql
SELECT B.Brand, G.Country, SUM(F.Units_Sold)
FROM Fact_Sales F
INNER JOIN Dim_Date D ON F.Date_Id = D.Id
INNER JOIN Dim_Store S ON F.Store_Id = S.Id
INNER JOIN Dim_Geography G ON S.Geography_Id = G.Id
INNER JOIN Dim_Product P ON F.Product_Id = P.Id
INNER JOIN Dim_Brand B ON P.Brand_Id = B.Id
WHERE D.Year = 1997 AND C.Product_Category = 'tv'
GROUP BY B.Brand, G.Country
```