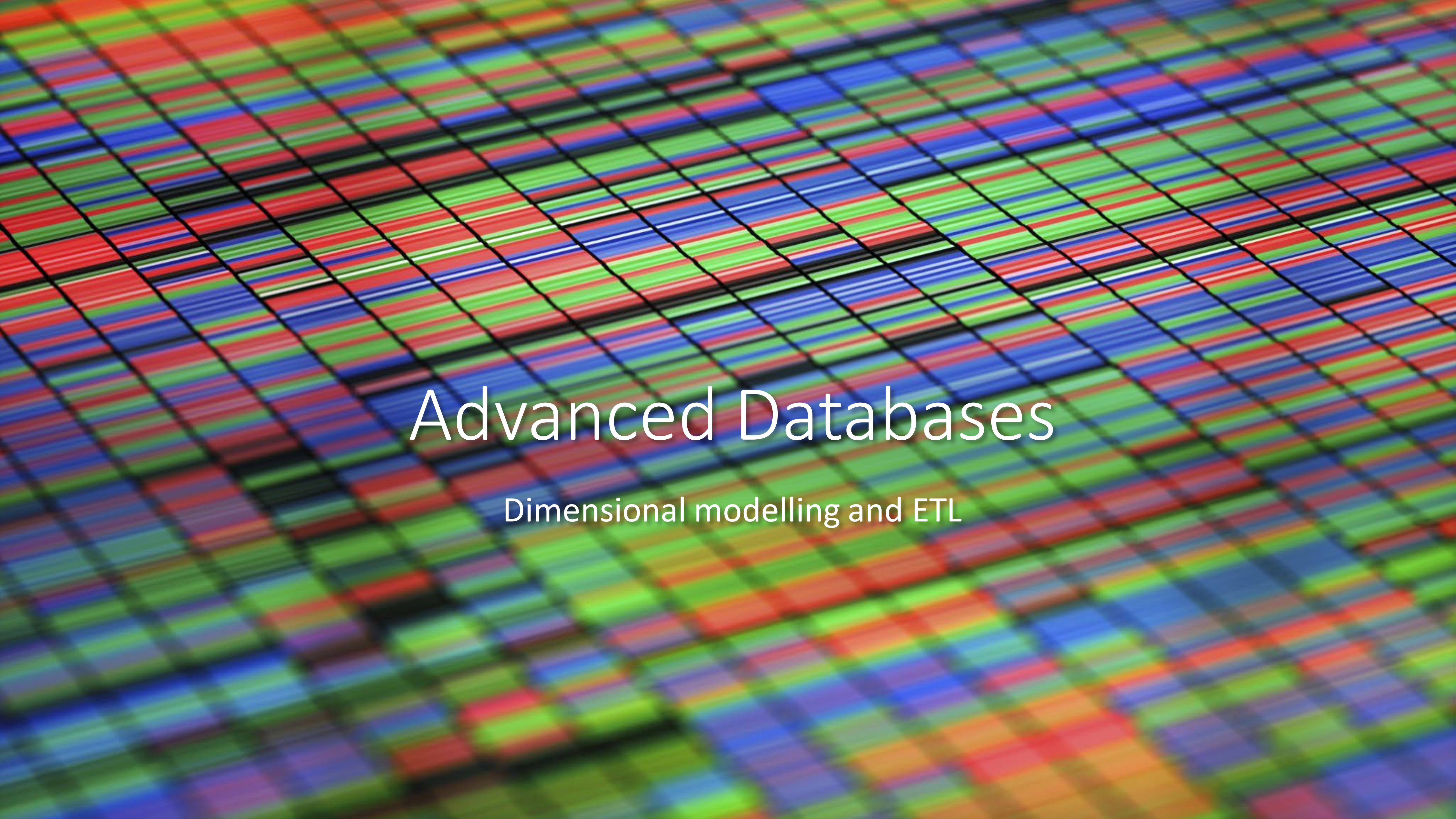
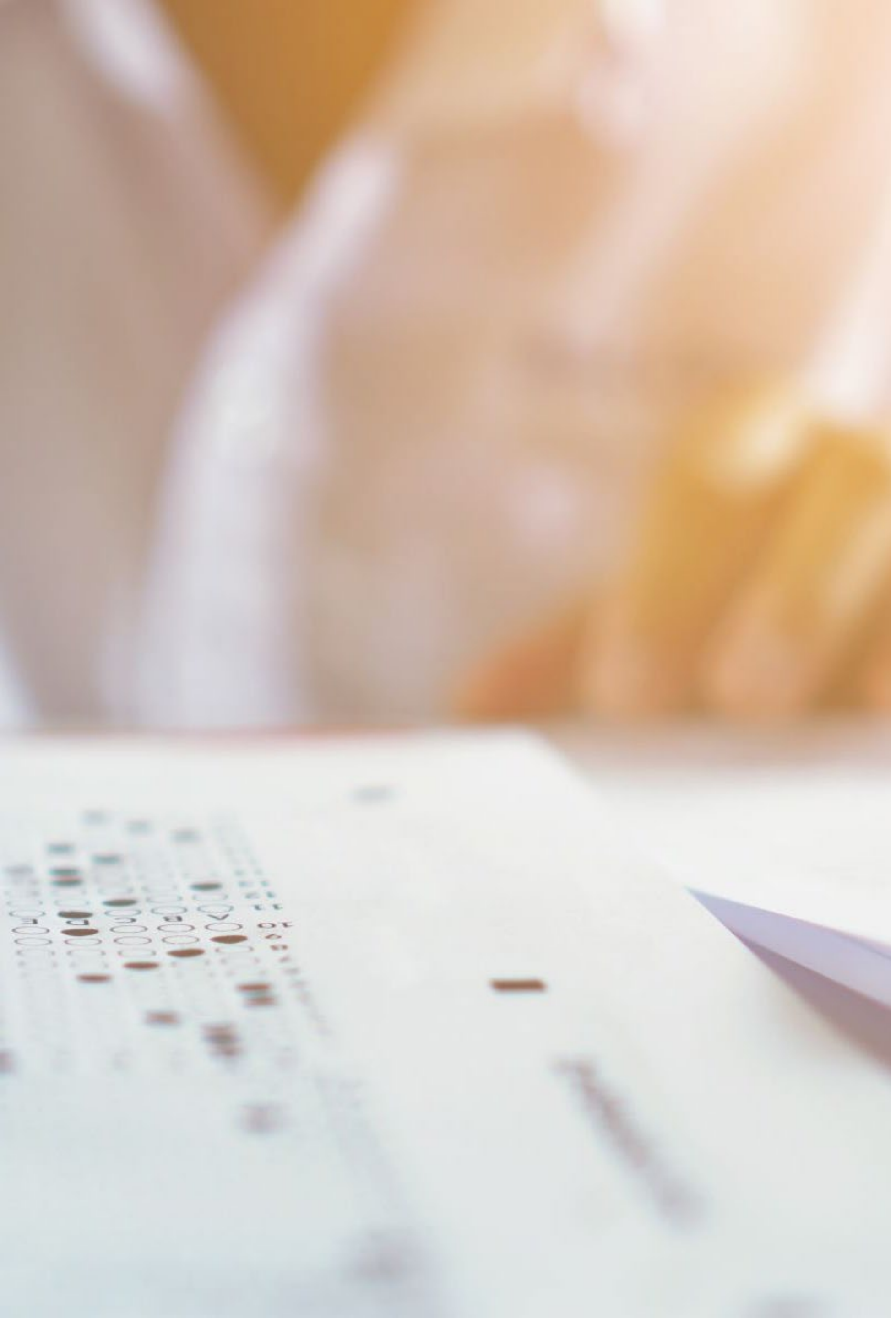


Advanced Databases

Dimensional modelling and ETL

Dimensional Modelling Steps

- Select the business process.
- Declare the grain.
- Identify the dimensions.
- Identify the facts.



Business Processes

- Operational activities performed by your organization
  - E.g. taking an order, registering students for a class, or snapshotting every account each month.
- Events generate or capture performance metrics that translate into facts in a fact table.
- Most fact tables focus on the results of a single business process.
- Choosing the process is important because it defines a specific design target and allows the grain, dimensions, and facts to be declared.
- Each business process corresponds to a row in the enterprise data warehouse bus matrix.

Grain

- Declaring the grain is the pivotal step in a dimensional design.
- The grain establishes exactly what a single fact table row represents.
- The grain declaration becomes a binding contract on the design.
- The grain must be declared before choosing dimensions or facts because every candidate dimension or fact must be consistent with the grain.
- Atomic grain refers to the lowest level at which data is captured by a given business process.
- Each proposed fact table grain results in a separate physical table; different grains must not be mixed in the same fact table.

Dimensions

- Dimensions provide the “who, what, where, when, why, and how” context surrounding a business process event.
- Dimension tables contain the descriptive attributes used by BI applications for filtering and grouping the facts.
- With the grain of a fact table firmly in mind, all the possible dimensions can be identified.
- Whenever possible, a dimension should be single valued when associated with a given fact row.
- A disproportionate amount of effort is put into the data governance and development of dimension tables because they are the drivers of the user’s BI experience.

Dimension Tables

- Every dimension table has a single primary key column.
- This primary key is embedded as a foreign key in any associated fact table
  - The dimension row’s descriptive context is exactly correct for that fact table row
- Dimension tables are usually wide, flat denormalized tables with many low-cardinality text attributes.
- While operational codes and indicators can be treated as attributes, the most powerful dimension attributes are populated with verbose descriptions.
- Dimension table attributes are the primary target of constraints and grouping specifications from queries and BI applications.
- The descriptive labels on reports are typically dimension attribute domain values.

Facts

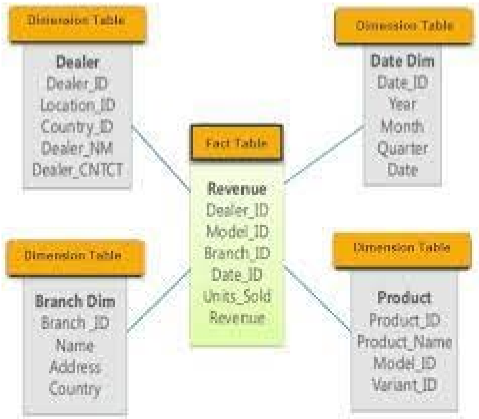
- Facts are the measurements that result from a business process event and are almost always numeric.
- A single fact table row has a one-to-one relationship to a measurement event as described by the fact table's grain.
- Thus a fact table corresponds to a physical observable event, and not to the demands of a particular report.
- Within a fact table, only facts consistent with the declared grain are allowed.
- For example, in a retail sales transaction, the quantity of a product sold and its extended price are good facts, whereas the store manager's salary is disallowed.

Fact Tables

- In addition to numeric measures, a fact table always contains foreign keys for each of its associated dimensions, as well as optional degenerate dimension keys and date/time stamps.
- Fact tables are the primary target of computations and dynamic aggregations arising from queries.

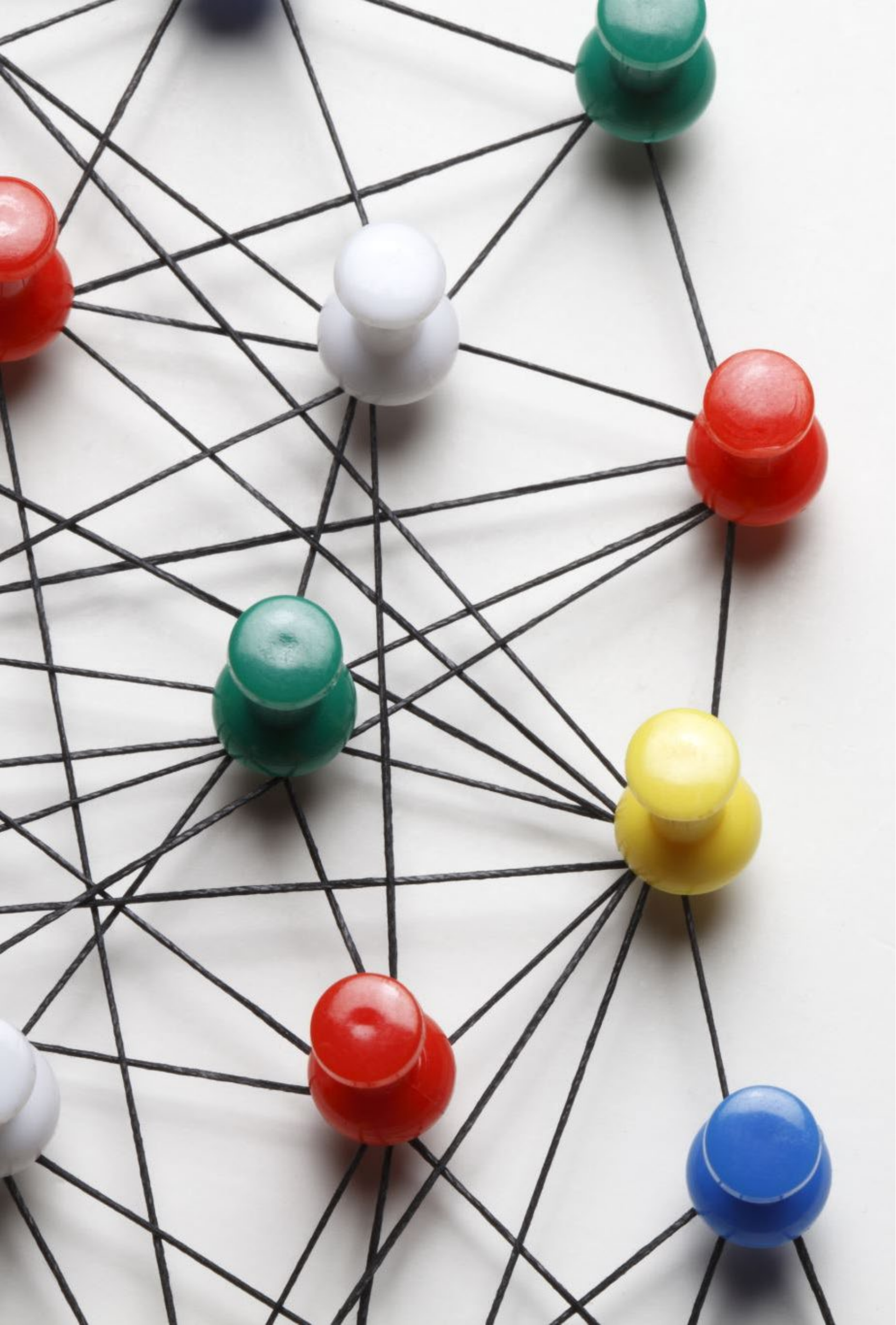
Dimension Modelling

- Focus on process measurement events, dividing data into either measurements or the “who, what, where, when, why, and how” descriptive context.
- Star schemas characteristically consist of fact tables linked to associated dimension tables via primary/foreign key relationships.



Dimensional Modelling

- A dimension table is designed with one column serving as a unique primary key.
- This cannot be the operational system’s natural key
  - Because there will be multiple dimension rows for that natural key when changes are tracked over time.
  - Plus these may be created by more than one source system, and these natural keys may be incompatible or poorly administered.
- You should create anonymous integer primary keys for every dimension.
- These dimension **surrogate keys** are simple integers, assigned in sequence, starting with the value 1, every time a new key is needed.
- The date dimension is exempt from the surrogate key rule; this highly predictable and stable dimension can use a more meaningful primary key.



Dimensional Modelling

- Single column surrogate fact keys can be useful
  - But are not required.
- Fact table surrogate keys, which are not associated with any dimension, are assigned sequentially during the ETL load process and are used
  - 1) as the single column primary key of the fact table;
  - 2) to serve as an immediate identifier of a fact table row without navigating multiple dimensions for ETL purposes;
  - 3) to allow an interrupted load process to either back out or resume;
  - 4) to allow fact table update operations to be decomposed into less risky inserts plus deletes.

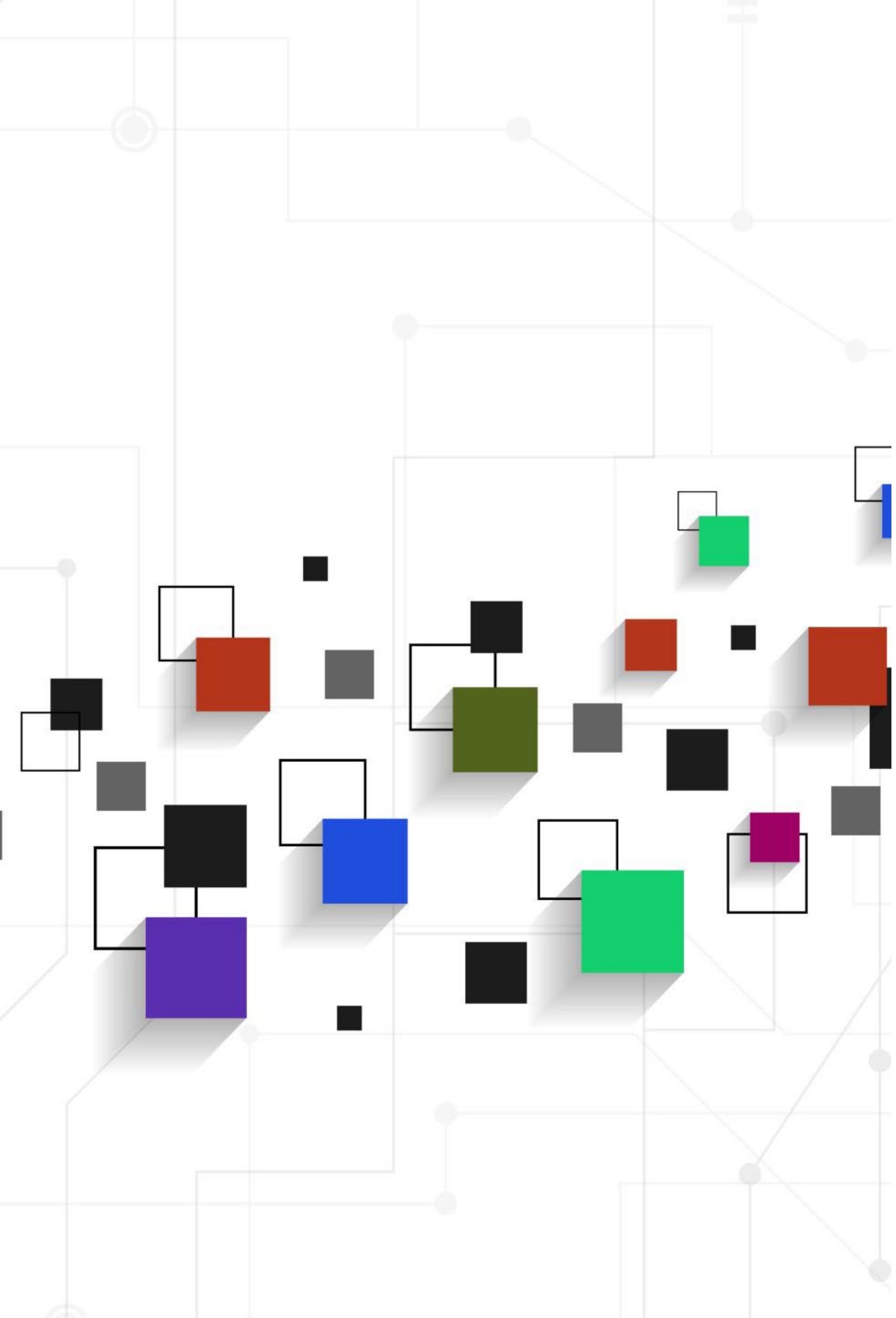
Extract, Transform and Load

ETL

Data Warehouse v Data Mart

- A data mart is a subset of a data warehouse oriented to a specific business process.
  - Marts contain repositories of summarized data collected for analysis on a specific section or unit within an organization, for example, the sales department.

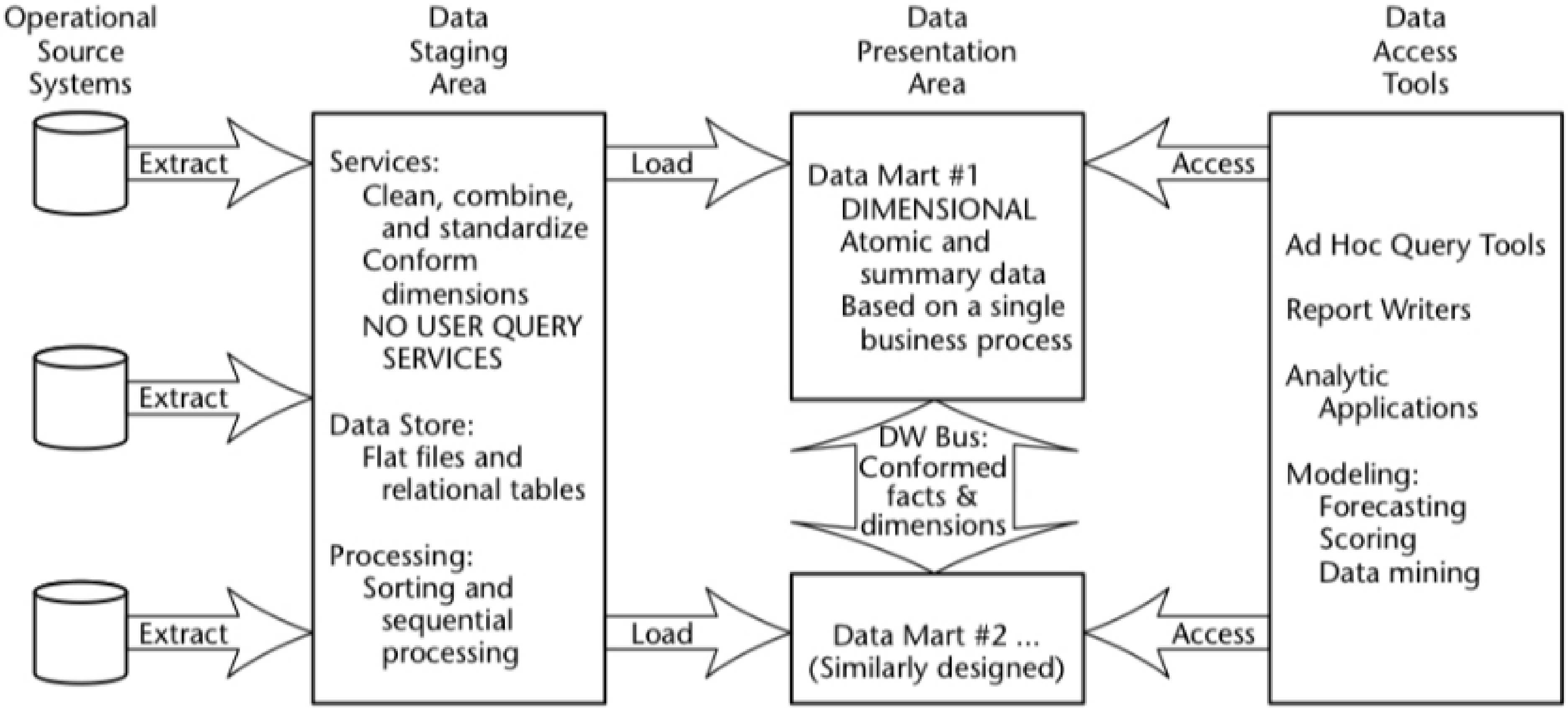
- A data warehouse is a large centralized repository of data that contains information from many sources within an organization.
  - The collated data is used to guide business decisions through analysis, reporting, and data mining tools.



Typical ETL

- You write ETL to
- Consolidate data sources from different source systems
- Accumulate data into a staging area
- Then use an ETL tool to model data into a data presentation area.
- This data presentation area consists of multiple data marts

Basics of a Data Warehouse



Operational Source Systems

Extract

Extract

Extract

Data Staging Area

Services:
Clean, combine, and standardize
Conform dimensions
NO USER QUERY SERVICES

Data Store:
Flat files and relational tables

Processing:
Sorting and sequential processing

Load

Load

Data Presentation Area

Data Mart #1
DIMENSIONAL
Atomic and summary data
Based on a single business process

DW Bus:
Conformed facts & dimensions

Data Mart #2 ...
(Similarly designed)

Access

Access

Data Access Tools

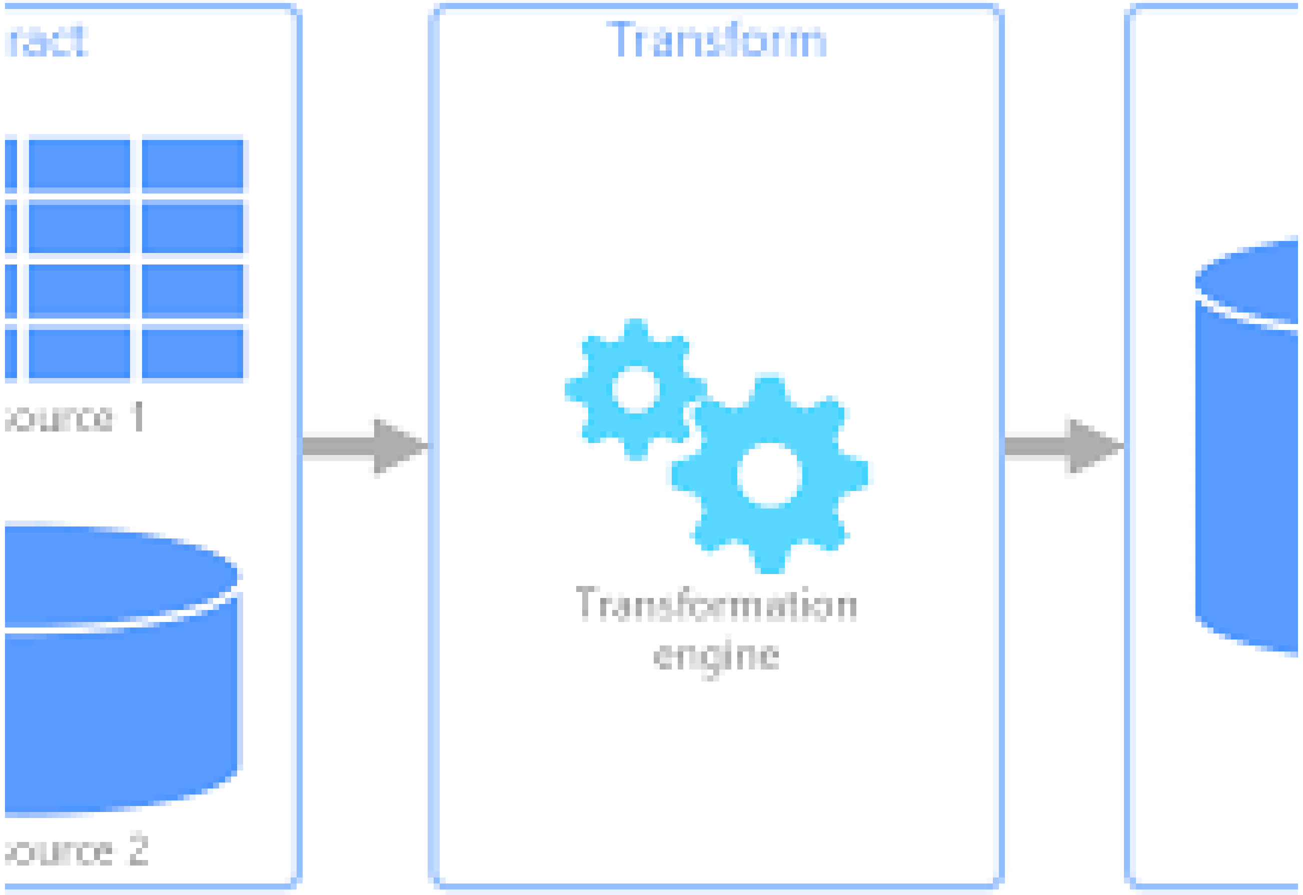
Ad Hoc Query Tools

Report Writers

Analytic Applications

Modeling:
Forecasting
Scoring
Data mining

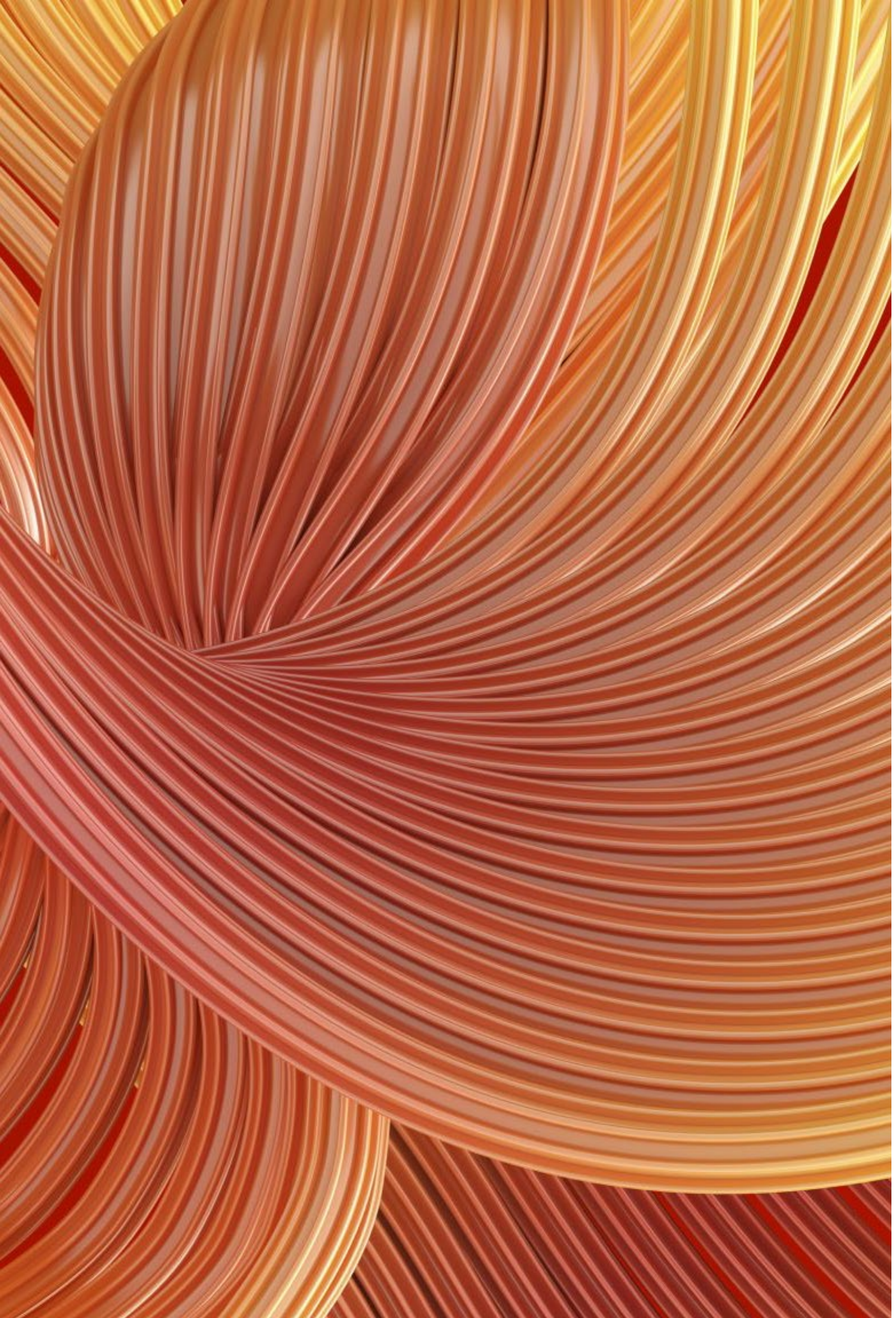
Extract, Transform and Load (ETL)



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

When defining ETL for a data warehouse

Important to think of ETL as a process, not a physical implementation

Copying data from one database to other

Data is extracted from an OLTP database, transformed to match the data 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;

This data also requires extraction, transformation, and loading

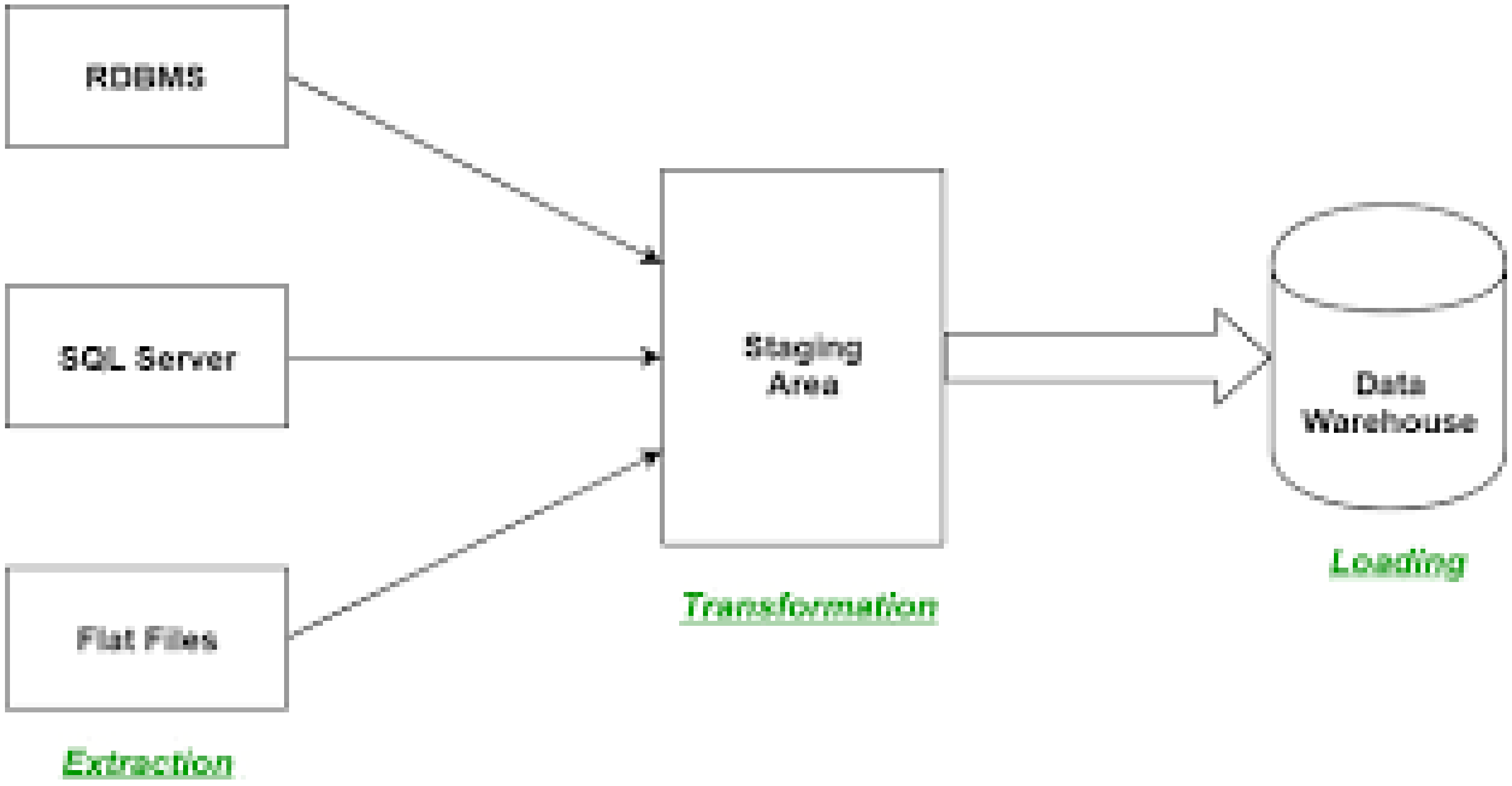
ETL Overview

- Often a complex combination of process and technology
- Often consumes a significant portion of the data warehouse development efforts
- 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

ETL Overview

- ETL is an integral, ongoing, and recurring part of a data warehouse
- Therefore it should be:
  - Automated
  - Well documented
  - Easily changeable

ETL Process


RDBMS
SQL Server
Flat Files
Extraction
Staging Area
Transformation
Data Warehouse
Loading

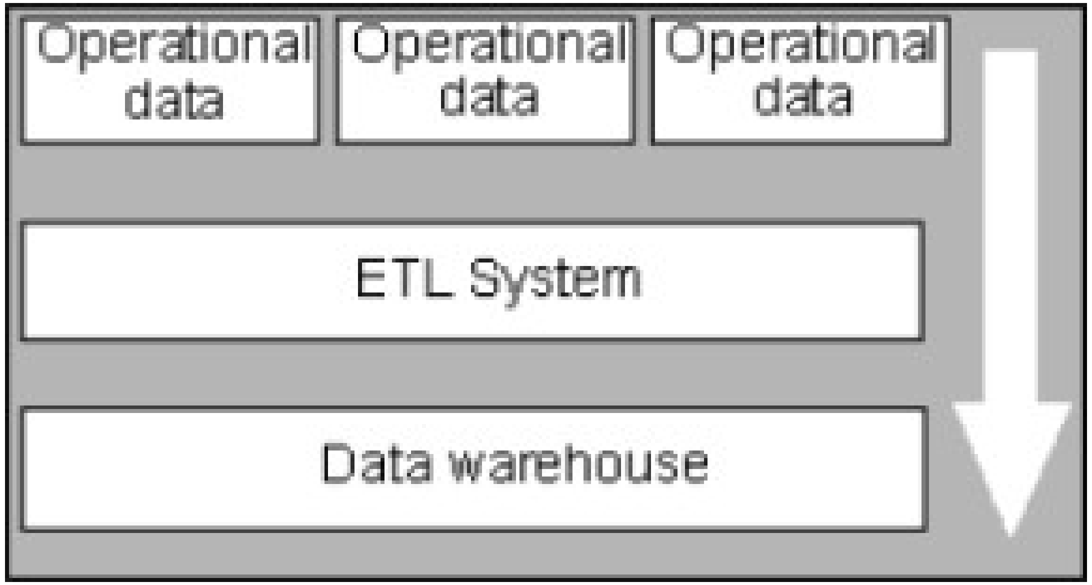
Extraction

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
- 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

- 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.



ETL process needs to effectively integrate systems that have different:

DBMS

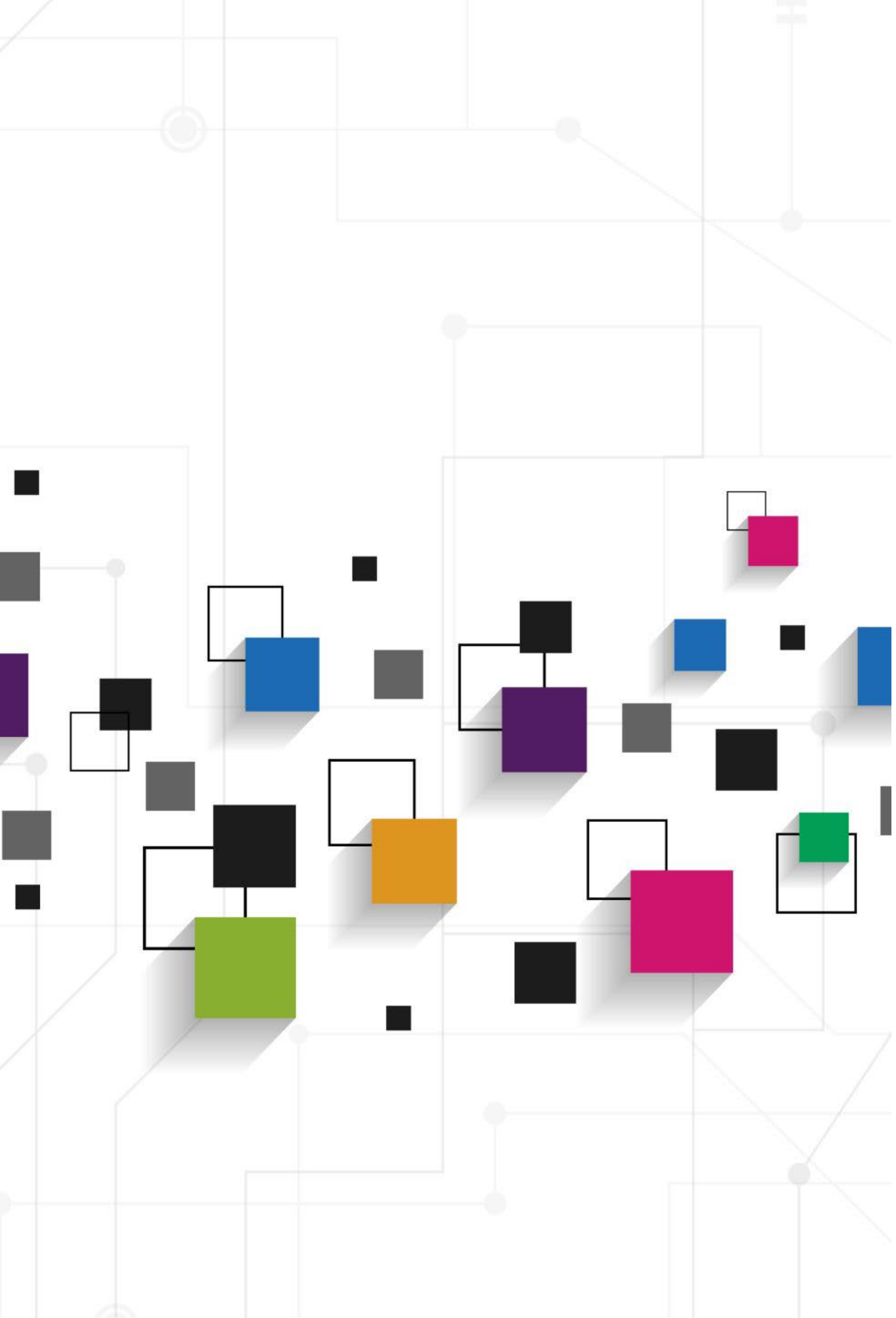
Operating Systems

Hardware

Communication protocols

Need to have a logical data map before the physical data can be transformed

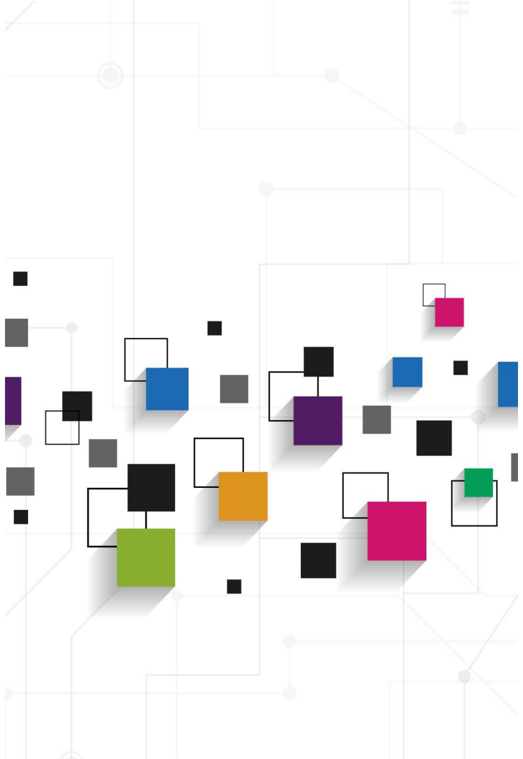
Data Extraction



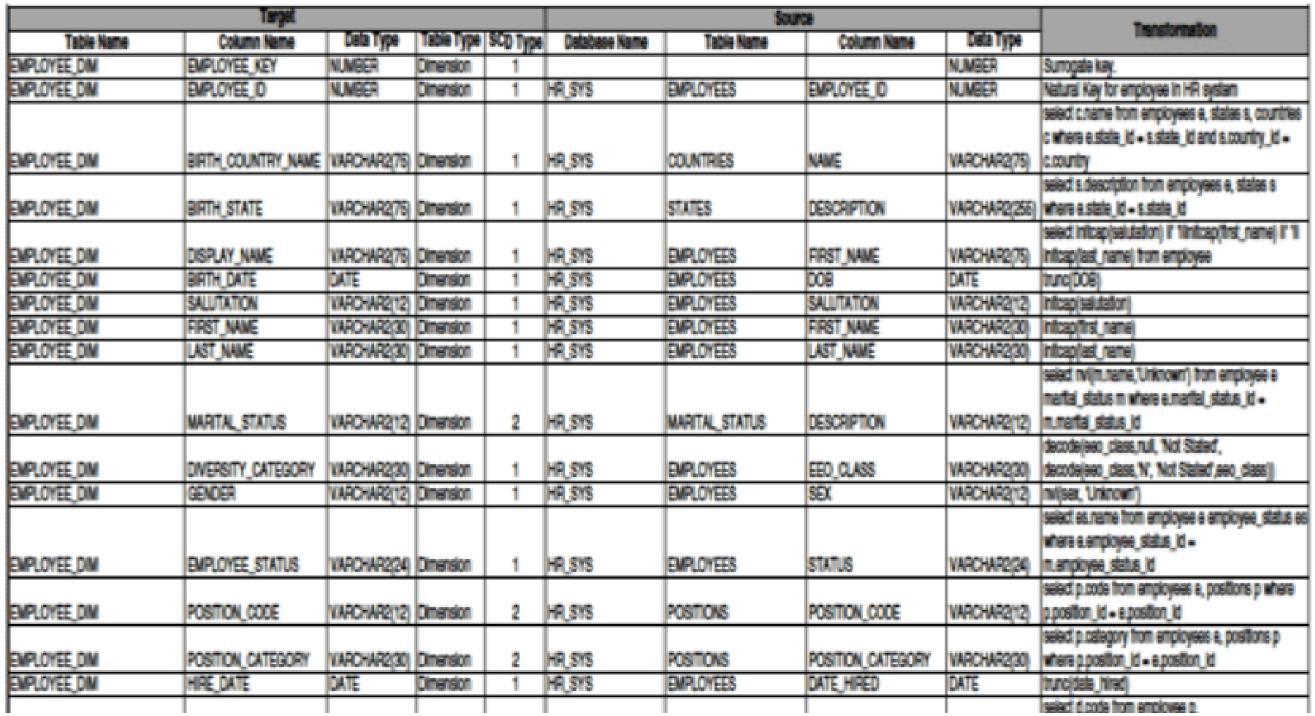
Logical Data Map

- **Describes the relationship** between the extreme starting points and the extreme ending points of your ETL system
- Usually presented in a table or spreadsheet
- Contains
  - The data definition for the data warehouse source systems
  - The target data warehouse data model
  - The exact manipulation of the data required to transform data from its original format to its destination format

Logical Data Map



The logical data map



Target					Source				Transformation
Table Name	Column Name	Data Type	Table Type	SCD Type	Database Name	Table Name	Column Name	Data Type	
EMPLOYEE_DIM	EMPLOYEE_KEY	NUMBER	Dimension	1				NUMBER	Surrogate key.
EMPLOYEE_DIM	EMPLOYEE_ID	NUMBER	Dimension	1	HR_SYS	EMPLOYEES	EMPLOYEE_ID	NUMBER	Natural Key for employee in HR system
EMPLOYEE_DIM	BIRTH_COUNTRY_NAME	VARCHAR2(75)	Dimension	1	HR_SYS	COUNTRIES	NAME	VARCHAR2(75)	select c.name from employees e, states s, countries c where e.state_id = s.state_id and s.country_id = c.country
EMPLOYEE_DIM	BIRTH_STATE	VARCHAR2(75)	Dimension	1	HR_SYS	STATES	DESCRIPTION	VARCHAR2(255)	select s.description from employees e, states s where e.state_id = s.state_id
EMPLOYEE_DIM	DISPLAY_NAME	VARCHAR2(75)	Dimension	1	HR_SYS	EMPLOYEES	FIRST_NAME	VARCHAR2(75)	select initcap(salutation) || ' '||initcap(first_name) || ' '|| initcap(last_name) from employee
EMPLOYEE_DIM	BIRTH_DATE	DATE	Dimension	1	HR_SYS	EMPLOYEES	DOB	DATE	trunc(DOB)
EMPLOYEE_DIM	SALUTATION	VARCHAR2(12)	Dimension	1	HR_SYS	EMPLOYEES	SALUTATION	VARCHAR2(12)	initcap(salutation)
EMPLOYEE_DIM	FIRST_NAME	VARCHAR2(30)	Dimension	1	HR_SYS	EMPLOYEES	FIRST_NAME	VARCHAR2(30)	initcap(first_name)
EMPLOYEE_DIM	LAST_NAME	VARCHAR2(30)	Dimension	1	HR_SYS	EMPLOYEES	LAST_NAME	VARCHAR2(30)	initcap(last_name)
EMPLOYEE_DIM	MARITAL_STATUS	VARCHAR2(12)	Dimension	2	HR_SYS	MARITAL_STATUS	DESCRIPTION	VARCHAR2(12)	select nvl(m.name,'Unknown') from employee e marital_status m where e.marital_status_id = m.marital_status_id
EMPLOYEE_DIM	DIVERSITY_CATEGORY	VARCHAR2(30)	Dimension	1	HR_SYS	EMPLOYEES	EEO_CLASS	VARCHAR2(30)	decode(eeo_class,null,'Not Stated', decode(eeo_class,'N','Not Stated',eeo_class))
EMPLOYEE_DIM	GENDER	VARCHAR2(12)	Dimension	1	HR_SYS	EMPLOYEES	SEX	VARCHAR2(12)	nvl(sex,'Unknown')
EMPLOYEE_DIM	EMPLOYEE_STATUS	VARCHAR2(24)	Dimension	1	HR_SYS	EMPLOYEES	STATUS	VARCHAR2(24)	select es.name from employee e employee_status es where e.employee_status_id = m.employee_status_id
EMPLOYEE_DIM	POSITION_CODE	VARCHAR2(12)	Dimension	2	HR_SYS	POSITIONS	POSITION_CODE	VARCHAR2(12)	select p.code from employees e, positions p where p.position_id = e.position_id
EMPLOYEE_DIM	POSITION_CATEGORY	VARCHAR2(30)	Dimension	2	HR_SYS	POSITIONS	POSITION_CATEGORY	VARCHAR2(30)	select p.category from employees e, positions p where p.position_id = e.position_id
EMPLOYEE_DIM	HIRE_DATE	DATE	Dimension	1	HR_SYS	EMPLOYEES	DATE_HIRED	DATE	trunc(date_hired)
									select d.code from employee e,

Logical mapping

Target				Source			
Table Name	Column Name	Table Type	SCD Type	Database	Table Name	Column Name	Transformation

- This is blueprint of what is expected from the ETL process
- 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

Logical mapping

Target				Source			
Table Name	Column Name	Table Type	SCD Type	Database	Table Name	Column Name	Transformation

- The table type gives us our queue for the ordinal position of our data load processes
  - first dimensions, then facts.

Logical Map

- Target table name.
  - The physical name of the table as it appears in the data warehouse
- Target column name.
  - The name of the column in the data warehouse table
- Table type.
  - Indicates if the table is a fact, dimension, or sub dimension (outrigger)
- SCD (slowly changing dimension) type.
  - For dimensions, this component indicates a Type-1, -2, or -3 slowly changing dimension approach. This indicator can vary for each column in the dimension.

Slowing Changing Dimension Type (SCD)

- Used to enable the historic aspect of data in a data warehouse
- Dimension attributes are modified over time
  - E.g. Customer and product are dimension of Sales if that is our fact table
- In operational systems, we may overwrite the modified attributes as we may not need the historical aspects of data
- In the data warehouse, we need to maintain the history
  - We may not be able to simply overwrite the data
  - We need to implement special techniques to maintain the history considering analytical and volume aspects of the data warehouse

Logical Map

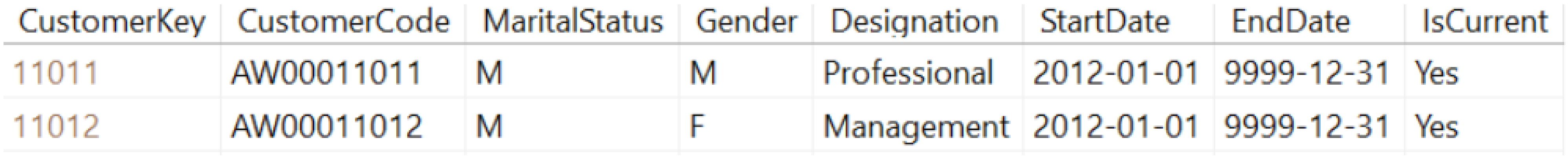
- Source database.
  - The name of the instance of the database where the source data resides.
  - Usually the connect string required to connect to the database. It can also be the name of a file as it appears in the file system. In this case, the path of the file would also be included.
- Source table name.
  - The name of the table where the source data originates.
  - There will be many cases where more than one table is required. In those cases, simply list all tables required to populate the relative table in the target data warehouse.
- Source column name.
  - The column or columns necessary to populate the target.
  - Simply list all of the columns required to load the target column. The associations of the source columns are documented in the transformation section.
- Transformation.
  - The exact manipulation required of the source data so it corresponds to the expected format of the target. This component is usually notated in SQL or pseudo-code.

Slowing Changing Dimension Type (SCD)

- SCD type.
  - Type 0
    - Ignore any changes and audit the changes.
    - E.g. HireDate for an employee
  - Type 1
    - Overwrite the changes
    - E.g. a situation where you only had partial information when first setting up the warehouse, empty columns can be updated based on other data
  - Type 2 (most popular)
    - History will be added as a new row.
    - Need to add additional columns to indicate when each set of data was relevant and which one is current

Slowing Changing Dimension Type (SCD)

- Type 2 (most popular)

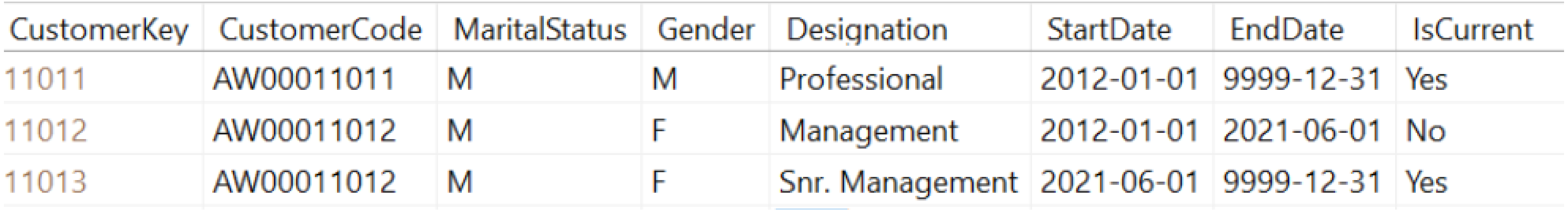


CustomerKey	CustomerCode	MaritalStatus	Gender	Designation	StartDate	EndDate	IsCurrent
11011	AW00011011	M	M	Professional	2012-01-01	9999-12-31	Yes
11012	AW00011012	M	F	Management	2012-01-01	9999-12-31	Yes

- Suppose the customer whose CustomerCode is AW00011012, has been promoted to Senior Management.
- If we update the record in the data warehouse you will not be able to see previous records.
- Solution: create a new record in the data warehouse with a new **CustomerKey** and a new Designation.
  - Other attributes will be remaining the same.
  - We have added three columns startdate, enddate and IsCurrent to our staging area to be able to manage which data is current

Slowing Changing Dimension Type (SCD)

- Type 2 (most popular)



CustomerKey	CustomerCode	MaritalStatus	Gender	Designation	StartDate	EndDate	IsCurrent
11011	AW00011011	M	M	Professional	2012-01-01	9999-12-31	Yes
11012	AW00011012	M	F	Management	2012-01-01	2021-06-01	No
11013	AW00011012	M	F	Snr. Management	2021-06-01	9999-12-31	Yes

- A new record has been added with a new customer key
- It is for the customer AW00011012
- The previous record’s end date has been changed and the indicator IsCurrent is set to No
- The new record has IsCurrent set to Yes

Slowing Changing Dimension Type (SCD)

- SCD type.
  - Type 3
    - History will be added as a new column.
  - Type 4
    - A new dimension will be added
  - Type 6
    - Combination of Type 2 and Type 3

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

- 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

- 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

- Data Quality paradigm:
  - Correct
  - Unambiguous
  - Consistent
  - Complete
- Data quality checks are run at 2 places
  - After extraction and
  - After cleaning
    - Additional confirmation checks are run at this point

Data Cleansing (Data Cleaning/Data Scrubbing)

- Process of fixing incorrect, incomplete, duplicate or otherwise erroneous data in a data set
  - 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

Types of "dirty" data addressed

- Typos and invalid or missing data.
  - For example, misspellings and other typographical errors, wrong numerical entries, syntax errors and missing values, such as blank or null fields that should contain data.
- Inconsistent data.
  - Names, addresses and other attributes are often formatted differently from system to system.
  - For example, one data set might include a customer's middle initial, while another doesn’t.
  - Data elements such as terms and identifiers may also vary.

Types of "dirty" data addressed

- Duplicate data.
    - Need to identify duplicate records in data sets and either remove or merge them through the use of deduplication measures.
    - For example, when data from two systems is combined, duplicate data entries can be reconciled to create single records.
- Irrelevant data.
    - Some data -- outliers or out-of-date entries, for example -- may not be relevant to analytics applications and could skew their results.
    - Should remove redundant data from data sets, which streamlines data preparation and reduces the required amount of data processing and storage resources.

Steps in Data Cleansing

- Parsing
- Correcting
- Standardizing
- Matching
- Consolidating

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 replacing a nickname or 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 uses a numerical code for gender, but the warehouse uses an alphabetic code
  - This calls for automated data cleansing
- 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

Data transformation examples

- Sorting
- Joining data from multiple sources
  - e.g. lookup, merge
- 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

Data transformation examples

- Aggregation
  - E.g. rollup — summarizing multiple rows of data
    - total sales for each store, for each region, etc.
- 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

Data transformation examples

- 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
  - Therefore none, some or all the data will be handed over to the next step (load)
    - Depends 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

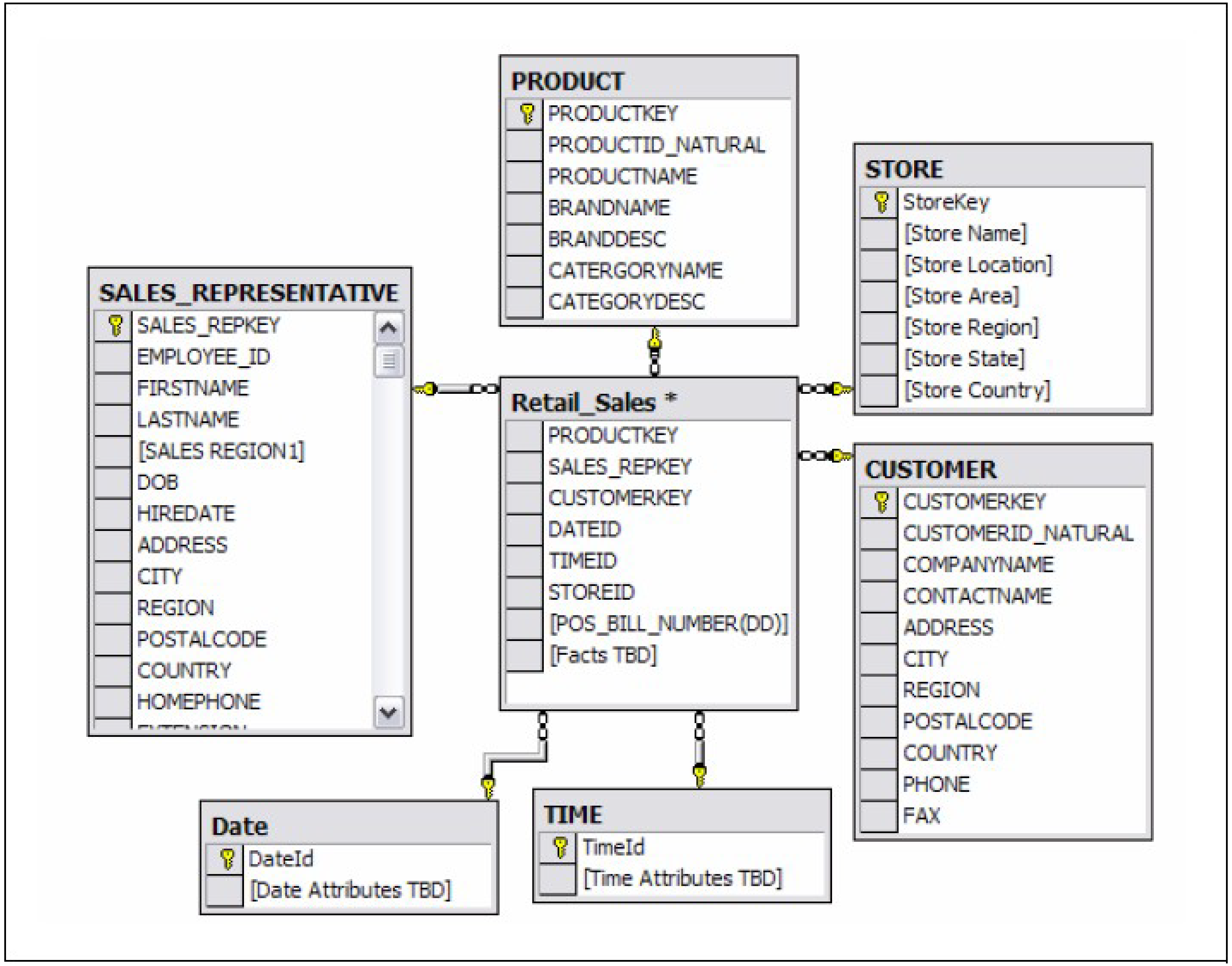
- The data loading module consists of all the steps required to administer slowly changing dimensions (SCD) and write the dimension to disk as a physical table in the proper dimensional format with correct primary keys, and final descriptive attributes.
- Creating and assigning the surrogate keys occur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

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

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

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

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

Date
- 🔑 DateId
- [Date Attributes TBD]

TIME
- 🔑 TimeId
- [Time Attributes TBD]

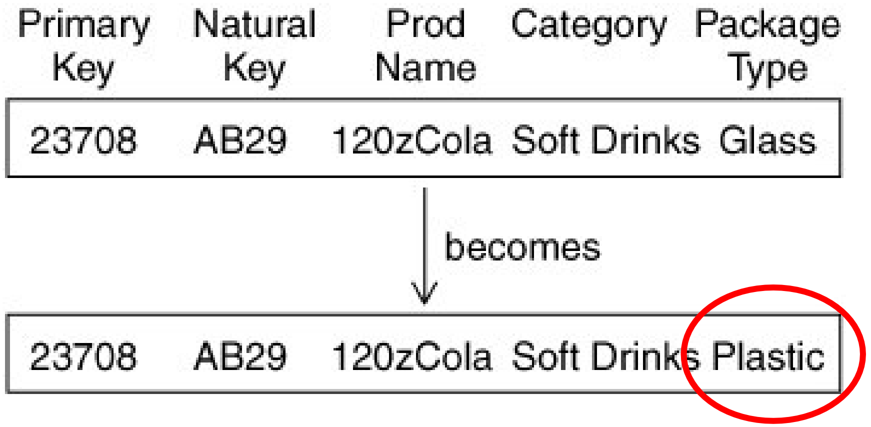
Type 1 – Overwriting the history

- 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

Primary Key	Natural Key	Prod Name	Category	Package Type
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.

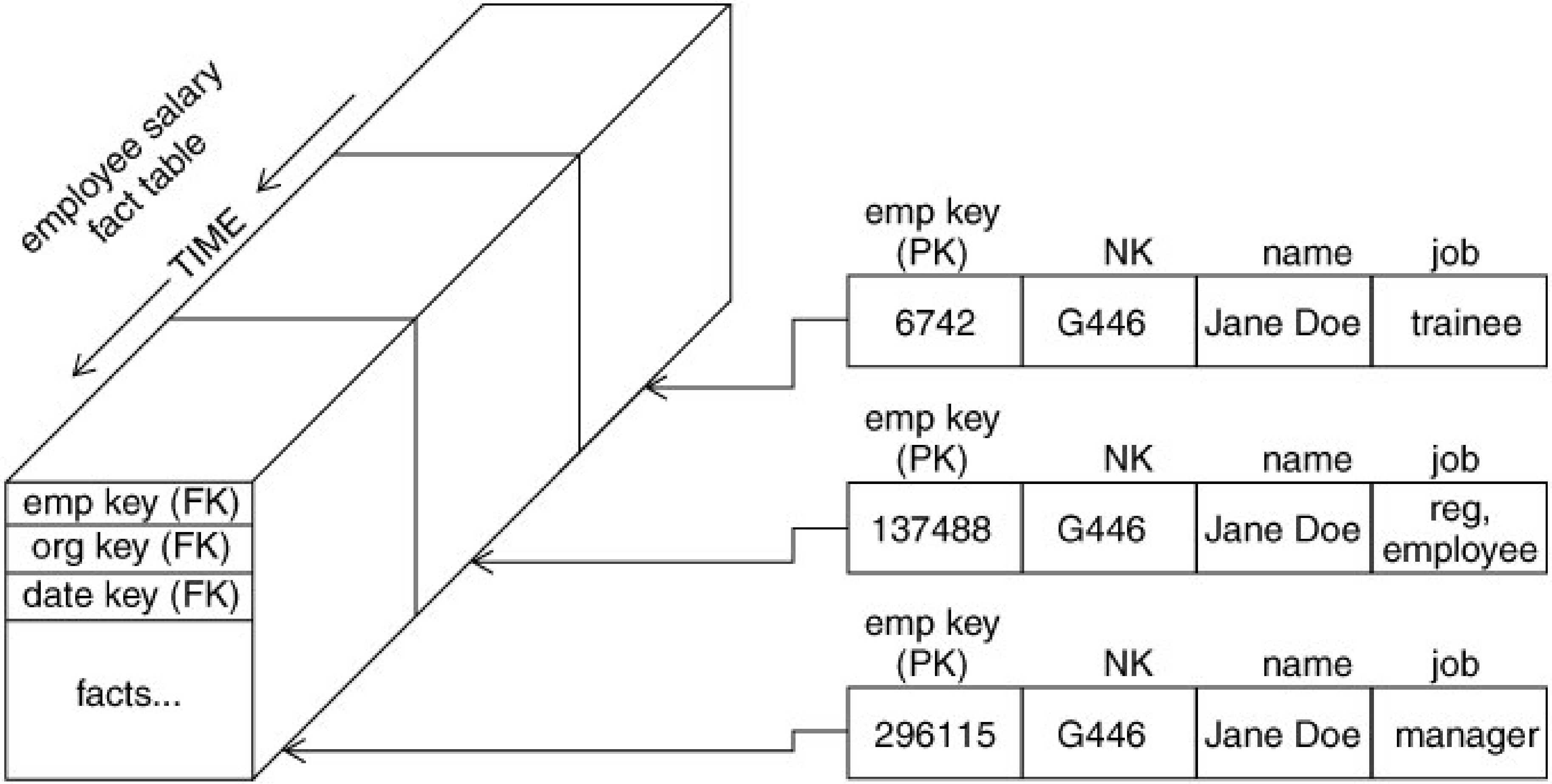
Type 2 – Preserving history

- 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 partitioned.
- 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

- 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.
- There is no need to include effective date attributes inside the dimension table
  - 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.
- Granularity is important

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.

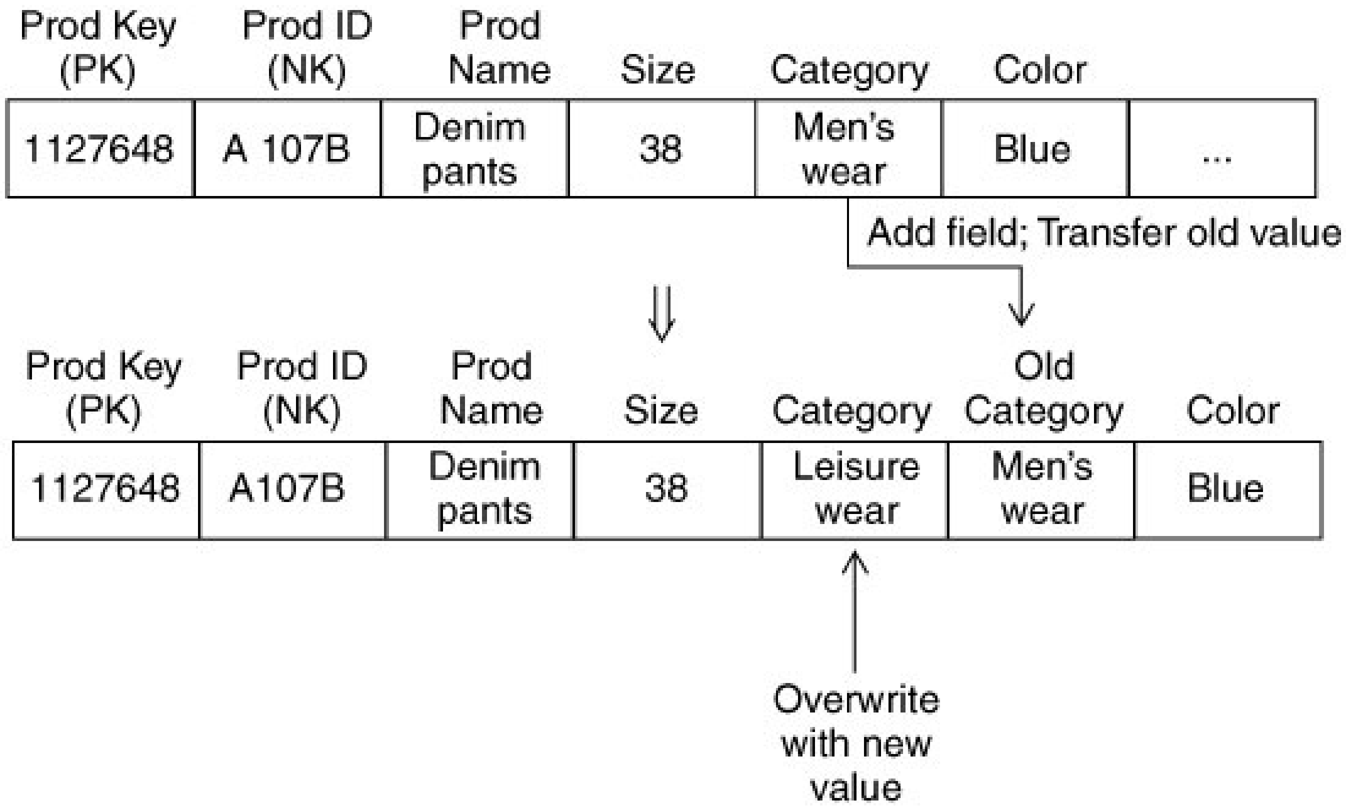
Type 3 – Preserving limited history

- 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 changes their name.

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

- 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.

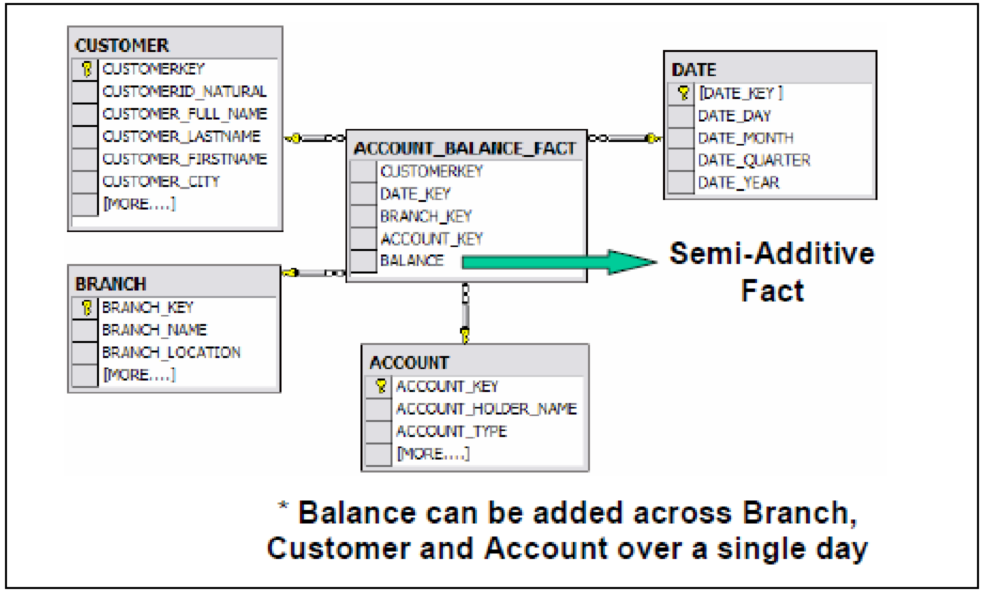
Type of facts

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
- E.g Account Balances:
  - The balance is not additive across time
  - It is obviously possible across customers
- E.g. if you have the number of items in the warehouse for each day, you can sum up the items for each day (total warehouse of the day), but it make no senso to sum up in the year.



Semi-Additive Fact

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

Additive

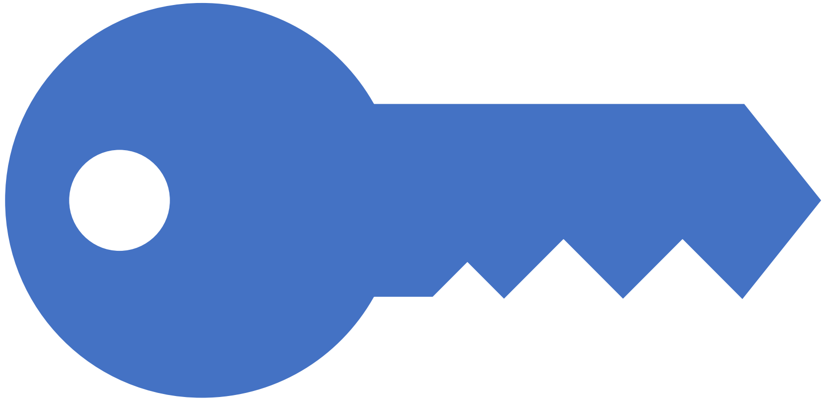
- Most flexible and useful facts
    - E.g. total sales

Loading Facts

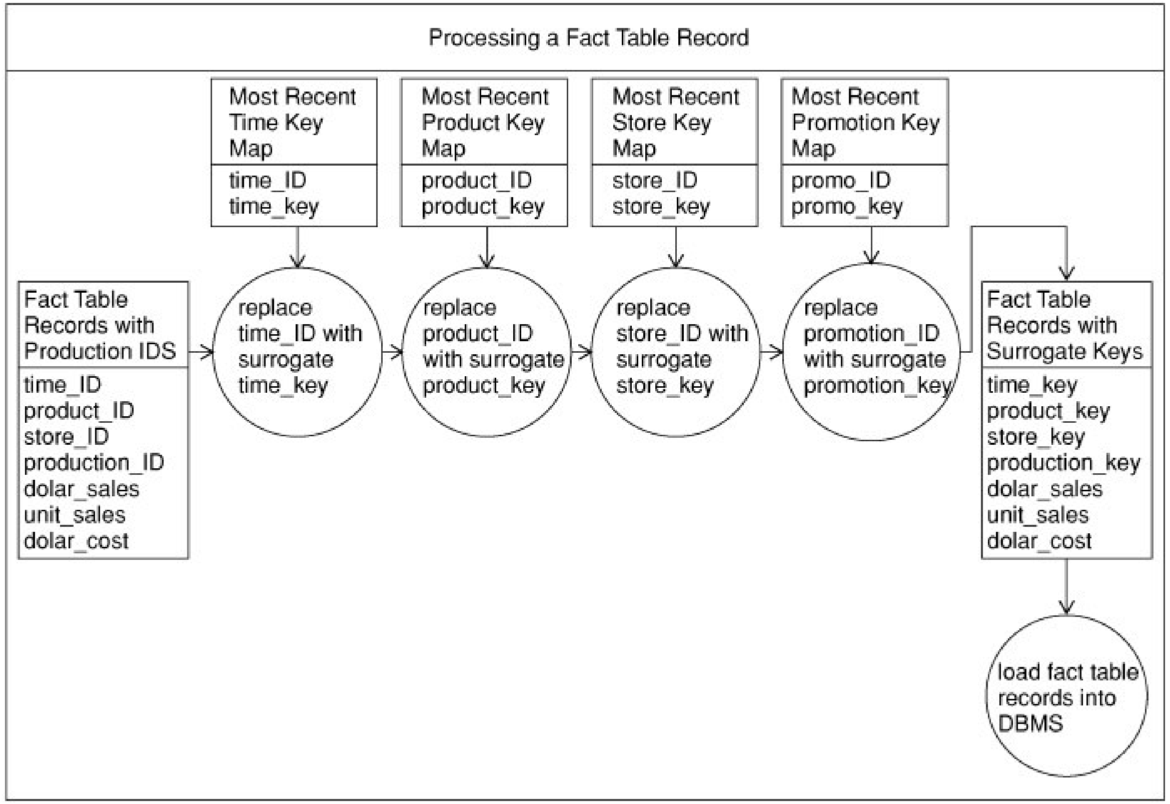
- Fact tables hold the measurements of an enterprise.
- The relationship between fact tables and measurements is extremely simple.
- If a measurement exists, it can be modelled as a fact table row. If a fact table row exists, it is a measurement.

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

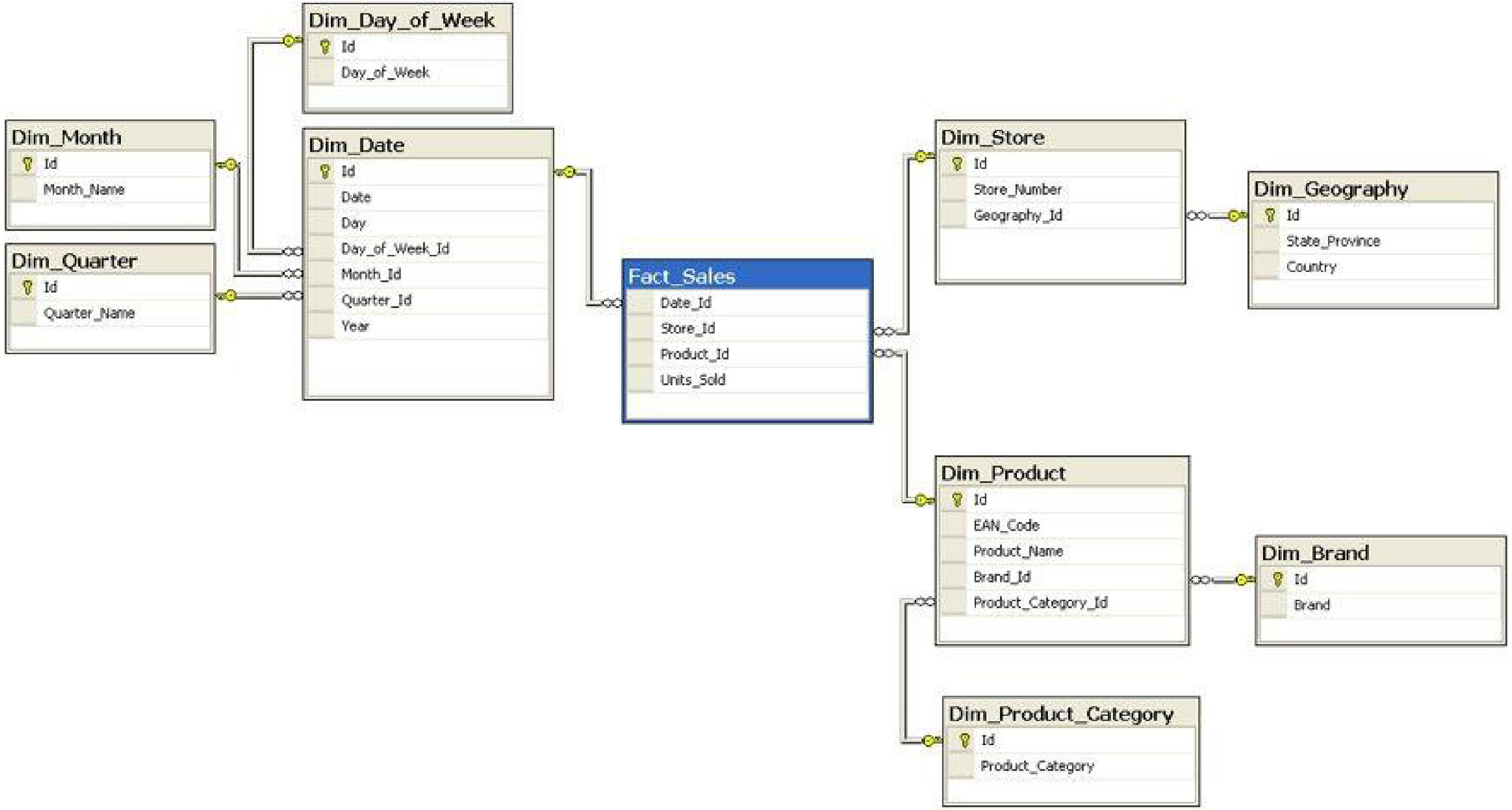
replace store_ID with surrogate store_key

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



Dim_Day_of_Week
Id
Day_of_Week

Dim_Month
Id
Month_Name

Dim_Quarter
Id
Quarter_Name

Dim_Date
Id
Date
Day
Day_of_Week_Id
Month_Id
Quarter_Id
Year

Fact_Sales
Date_Id
Store_Id
Product_Id
Units_Sold

Dim_Store
Id
Store_Number
Geography_Id

Dim_Geography
Id
State_Province
Country

Dim_Product
Id
EAN_Code
Product_Name
Brand_Id
Product_Category_Id

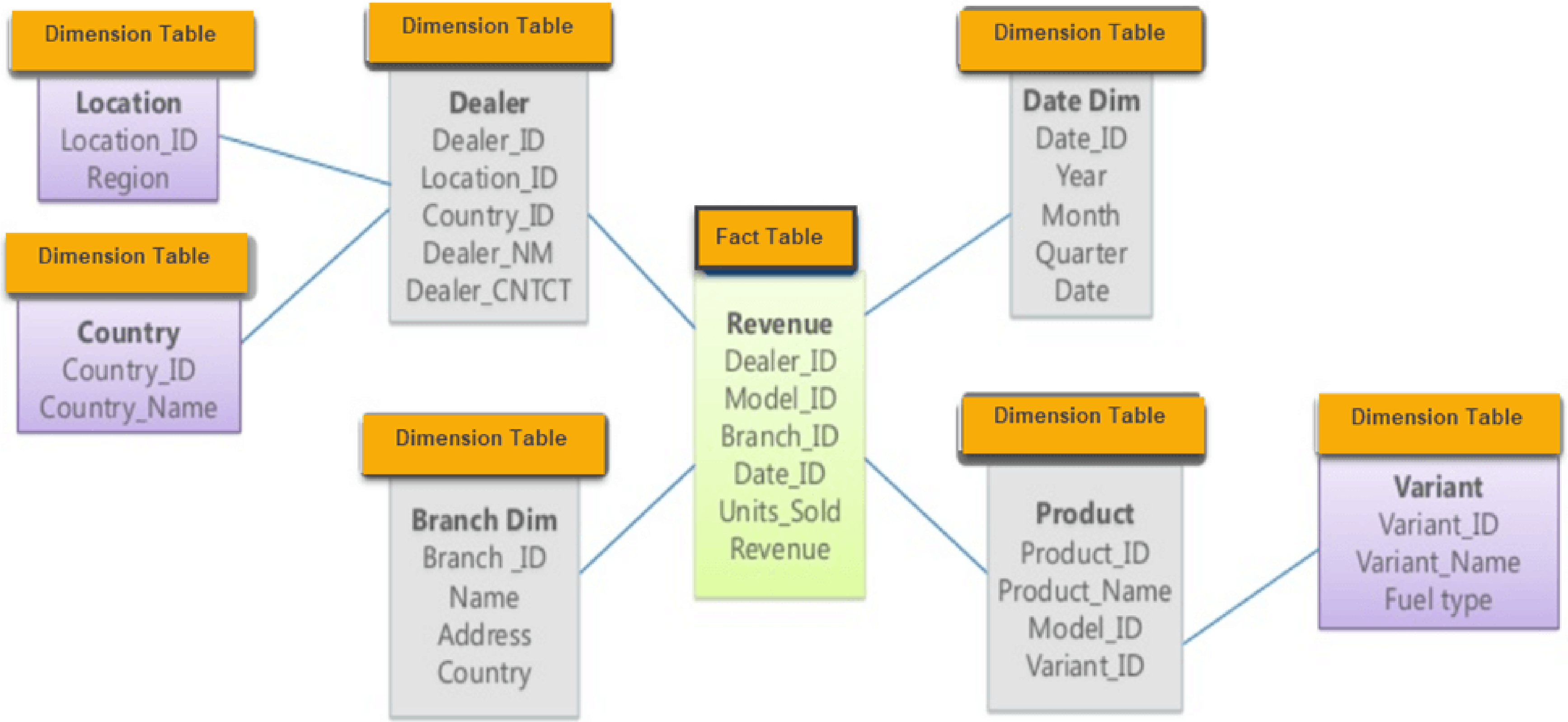
Dim_Brand
Id
Brand

Dim_Product_Category
Id
Product_Category

Query Example

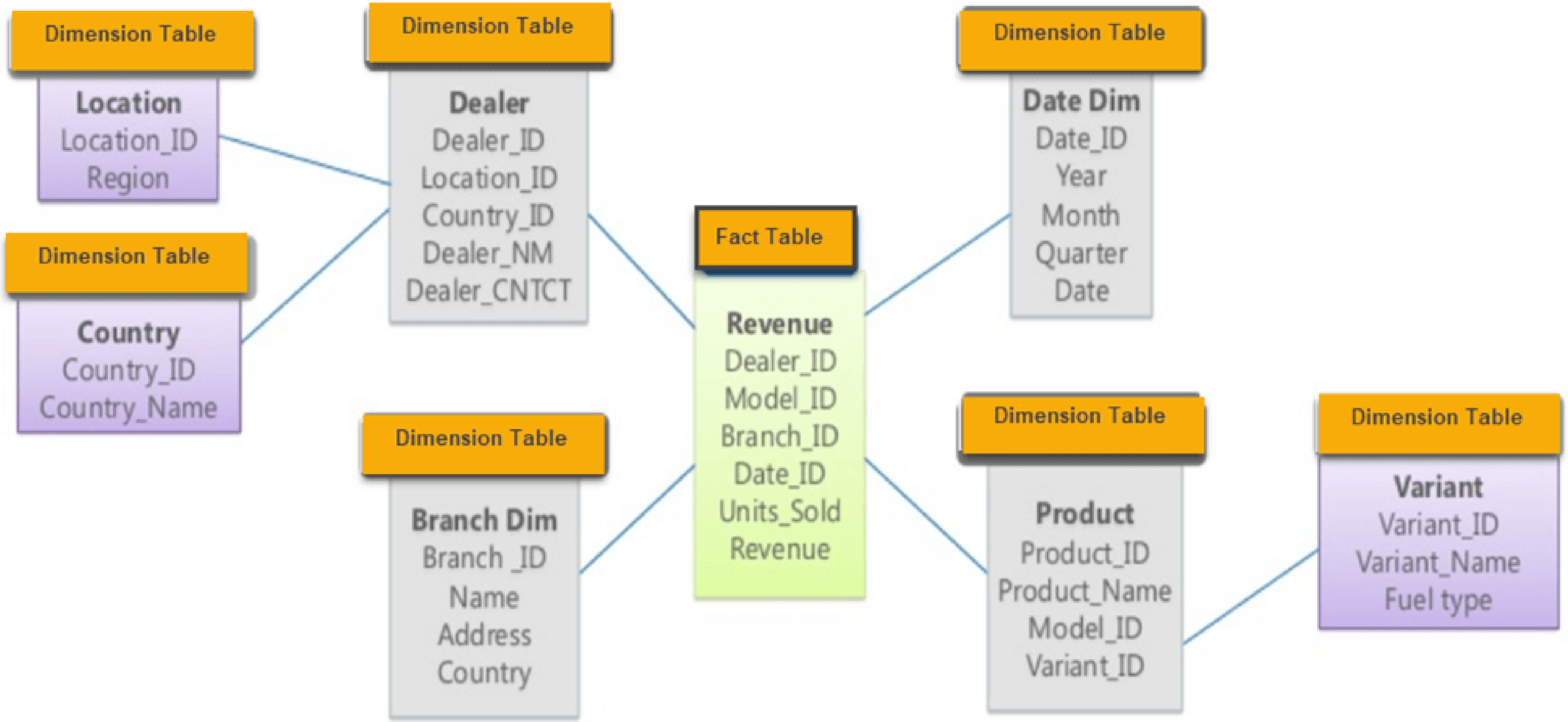
```
SELECT B.Brand, G.Country, SUM (F.Units_Sold)
FROM Fact_Sales F
INNER JOIN Dim_Date D ON F.Date_Id = D.Id
INNER JOIN Dim_Store S ON F.Store_Id = S.Id
INNER JOIN Dim_Geography G ON S.Geography_Id = G.Id INNER JOIN Dim_Product P
ON F.Product_Id = P.Id  INNER JOIN Dim_Brand B ON P.Brand_Id = B.Id
WHERE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

Snowflake Schema



- A logical arrangement of tables in a multidimensional database
- An extension of a Star Schema
- It adds additional dimensions.
- The dimension tables are normalized which splits data into additional tables

Snowflake Schema



- Example of when to use
- Date
  - Day
  - Month
  - Year
  - Quarter
- Location
  - Region
  - Country