THE UNIVERSITY OF WESTERN AUSTRALIA

Data Warehousing

Lecture 4 Extraction, Transform and Load (ETL)

CITS3401
CITS5504

Zeyi Wen

Computer Science and Software Engineering

School of Maths, Physics and Computing

Acknowledgement: The lecture slides are based on online resources

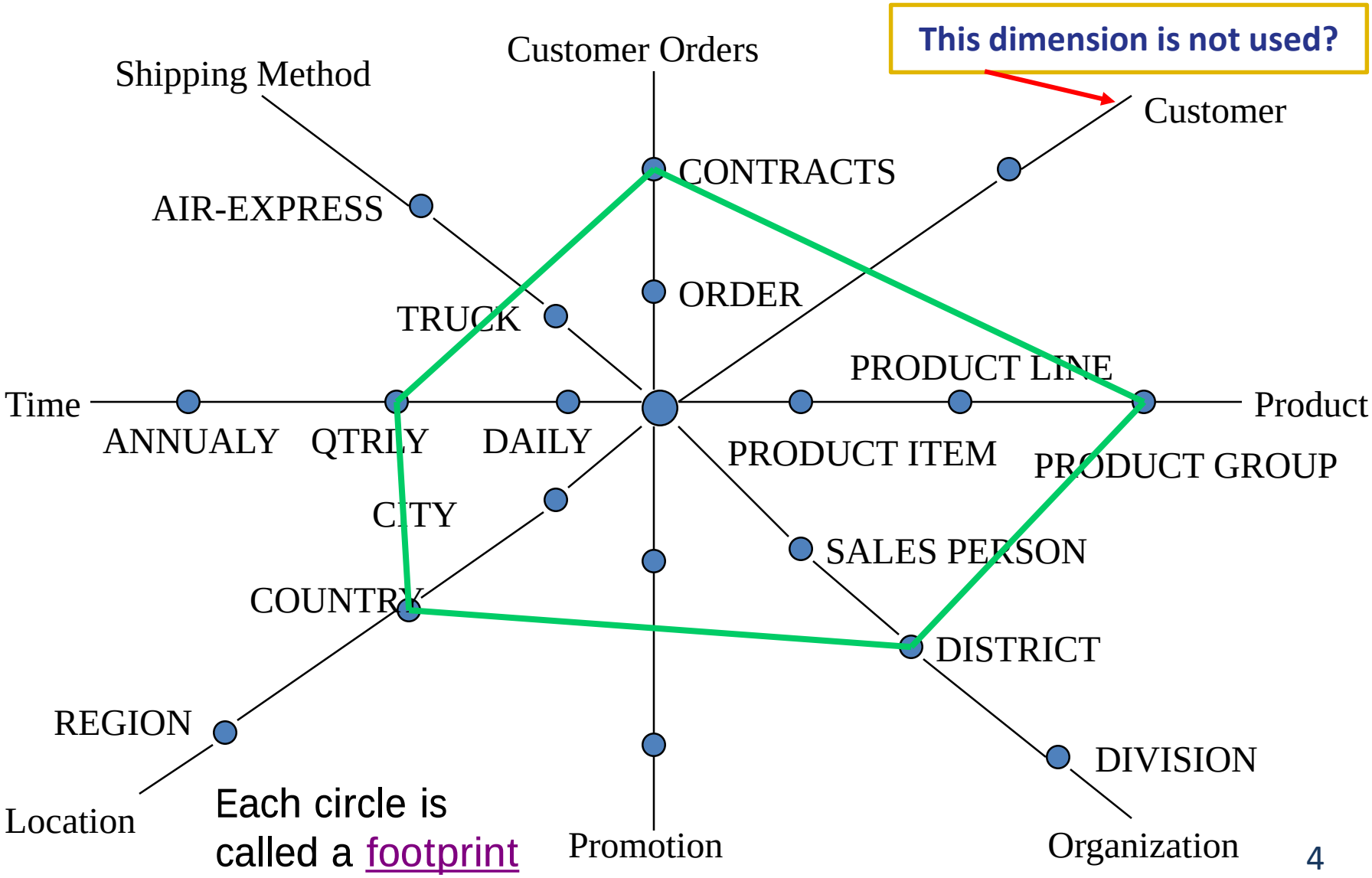
Lecture 3 and Project 1

- Correction on the number of cuboids in Lecture 3
- StarNet: each dimension has “all” in the concept hierarchy.
- ETL: data preprocessing is part of ETL
- Do I need to construct csv files from the Excel files? **Yes**.
- Week 4 Lab has an example script to import csv files to tables in SQL Server.

Closed Cube in more detail (Lecture 3)

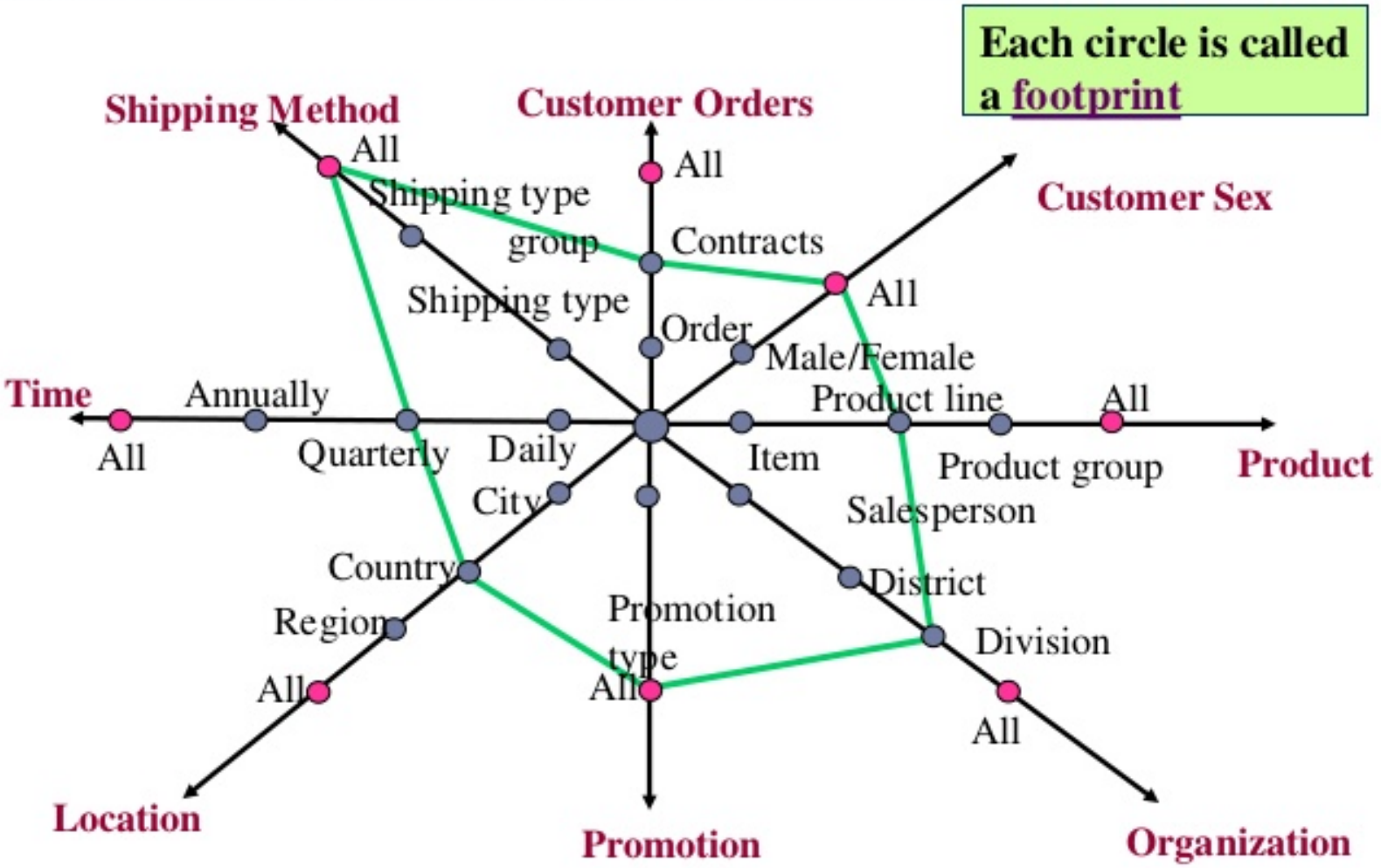
- Suppose that there are 2 base cells for a database of 100 dimensions, denoted as
  - $\{(a_1, a_2, a_3, \ldots, a_{100}, 10); (a_1, a_2, b_3, \ldots, b_{100}, 10)\}$, where each has a measure of 10.
  - The rest of the cells have a measure of 0.
- Is iceberg cube good enough in sparse cases like this? How many cuboids?
  - Apex cuboid + 1-D cuboids + 2-D cuboids + … + 99-D cuboids
    - $\binom{100}{0} + \binom{100}{1} + \binom{100}{2} + \cdots + \binom{100}{99} + 1 = 2^{100}$
  - Total number of distinct aggregate cells is $(2^{101} - 2) - 4$.
  - Ignore all the aggregate cells that can be obtained by replacing constants with $*$, only 3 cells really offer new information.
    - $\{(a_1, a_2, a_3, \ldots, a_{100}, 10), (a_1, a_2, b_3, \ldots, b_{100}, 10)), (a_1, a_2, *, \ldots, *, 20)\}$

Granularity of viewing the data warehouse (Lecture 2)



Each circle is called a footprint

Every Dimension has "All" in the hierarchy



Lecture Outline

THE UNIVERSITY OF WESTERN AUSTRALIA

- **ETL Overview**
- Data Staging
- Data Extraction and Transformation
- Loading Dimension and Fact Tables
- Handling Data Changes

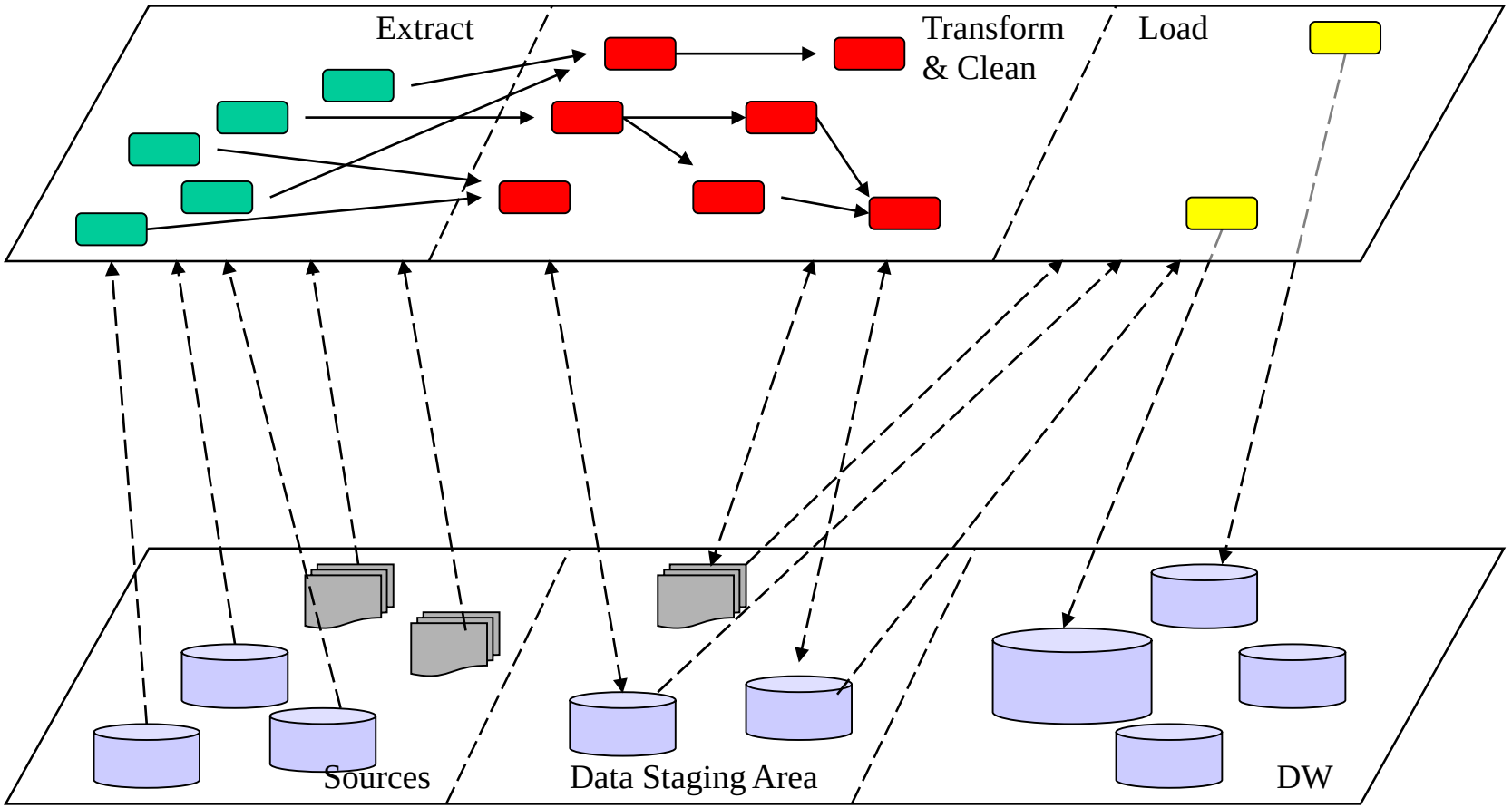
Extract, Transform, and Load (ETL)

- **Extract:**
  - the process of fetching **relevant** data from the source systems
- **Transform:**
  - the process in which we apply transformations such as
    - aggregation, joining with other datasets, applying rules, splitting or merging results, lookup, join, pivot, and
    - applying the data reduction techniques, and many others.
- **Load:**
  - the process of loading data into the data warehouse destination tables such as fact and dimensions
  - building additional structures to improve system performance

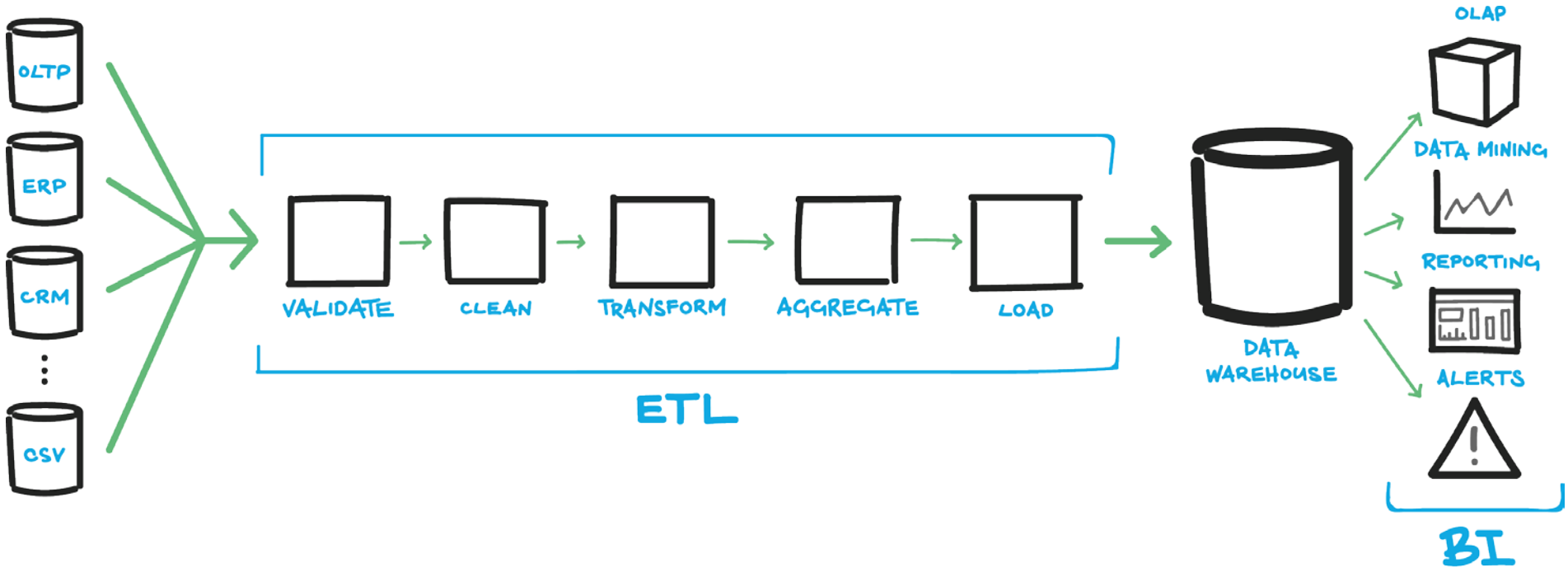
The source data should be loaded into the data warehouse in a consistent way and be reliable.

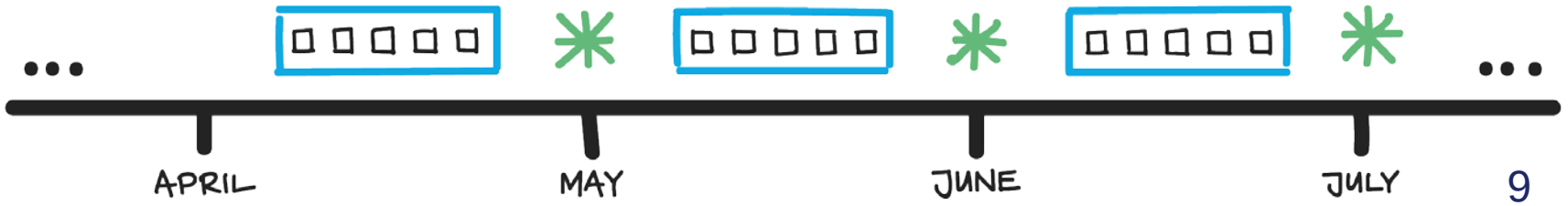
ETL Overview

- Architecture of data warehousing:
  Data sources ⇒ Data staging area ⇒ Data warehouse



ELT Lifecycle



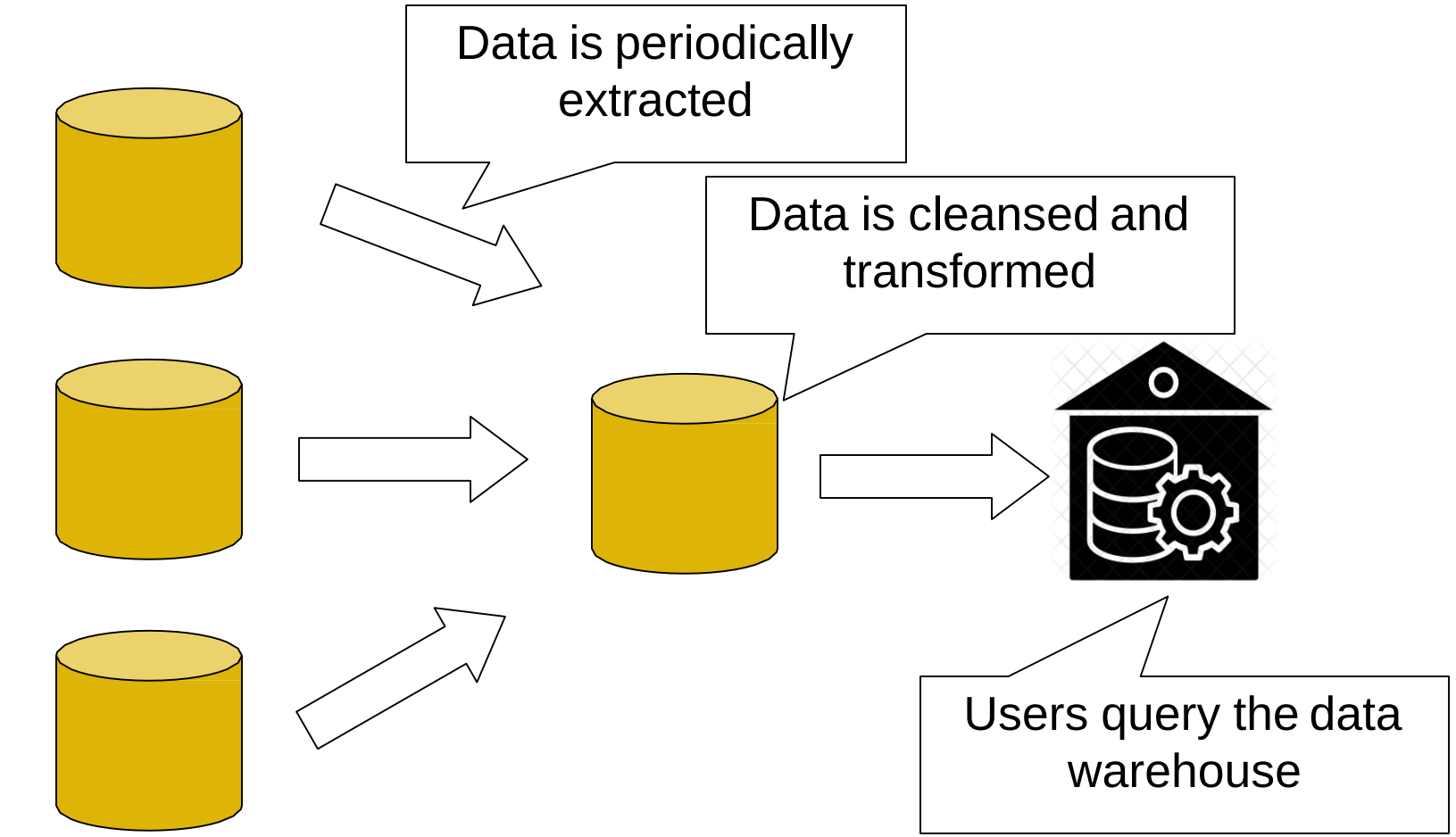


Extract and Transform in ETL

- Data transfer or ETL is an important component of the BI and data warehousing system.
  - **Extract step** should be able to pick data from all these kinds of data sources
    - data might come from different sources such as an **Excel file**, a relational database, or a web site.
  - **Transformation step** should be careful to ensure this data is consistent.
    - build keys; apply data-cleansing rules on the dataset.
  - Fetch the data in an incremental method
    - data in the source system might be removed or replaced by newer data records,
    - some data rows may appear only once in the data source and be replaced with new data.

Slowly Changing Dimension (**SCD**) and incremental load are challenging situations that ETL design need to take into account.

Loading Data to the Data Warehouse



ETL Major Steps

- **Determine the data you need and don't need**
- Determine all the data sources, both internal and external
- **Establish comprehensive data extraction, cleansing rules and transformations rules**
- Plan for aggregated tables
- Organise data staging area and test tools
- **Write procedure for all data loads**
- **Loading dimension tables and fact tables.**

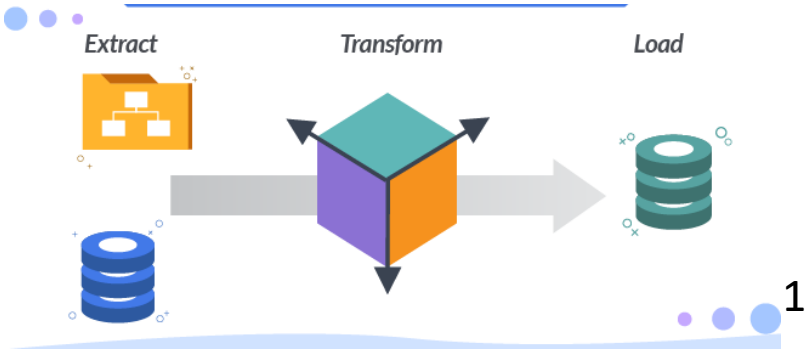
Tools for ETL

Examples of ETL tools:

- MS SQL Server Integration Services (SSIS),
- IBM Infosphere DataStage,
- SAS ETL Studio,
- Oracle Warehouse Builder, Data Integrator,
- Business Objects Data Integrator,
- Pentaho Data Integration.

ETL tools can be used for:

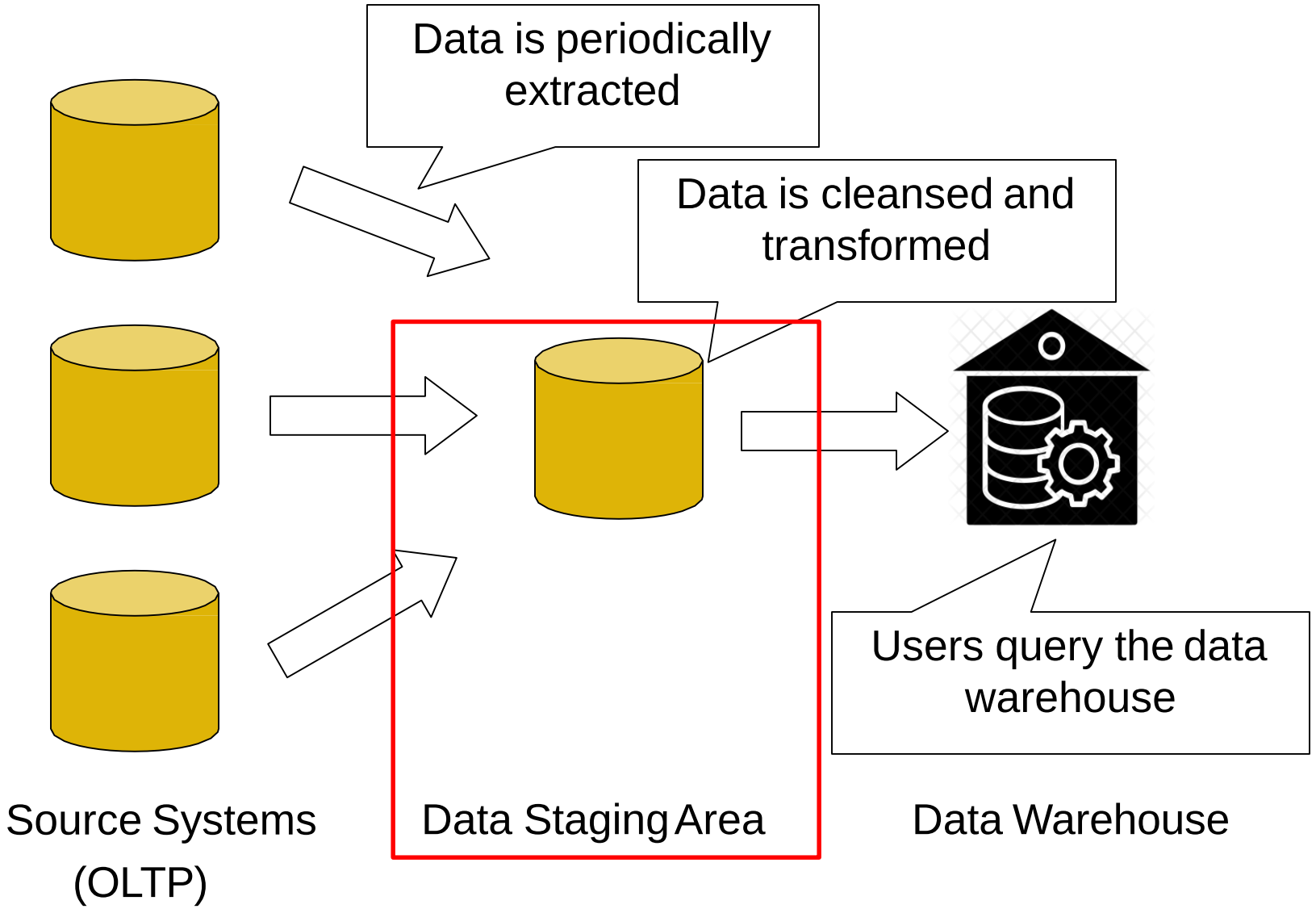
- Data extraction from heterogeneous data sources
- Data transformation, integration, and cleansing
- Data quality analysis and control
- Data loading
- High-speed data transfer
- Data refreshment
- Managing and analysing metadata



Lecture Outline

- ETL Overview
- **Data Staging**
- Data Extraction and Transformation
- Loading Dimension and Fact Tables
- Handling Data Changes

Loading Data to the Data Warehouse



Why Staging

- A staging area ensures that
  - we don't spend any extra activities in the extract step to convert or transform data, and also,
  - data won't be loaded all at once into the memory.

- Extracting data from sources into an integrated database comes with some challenges.
  - For **very big** datasets, fetching data from sources shouldn't be combined with any kind of conversion or transformation.
    - The main reason is that it will reduce the performance and slow down the whole ETL process.
    - Loading a very large dataset into memory will require a high amount of server resources that are not available at all times.

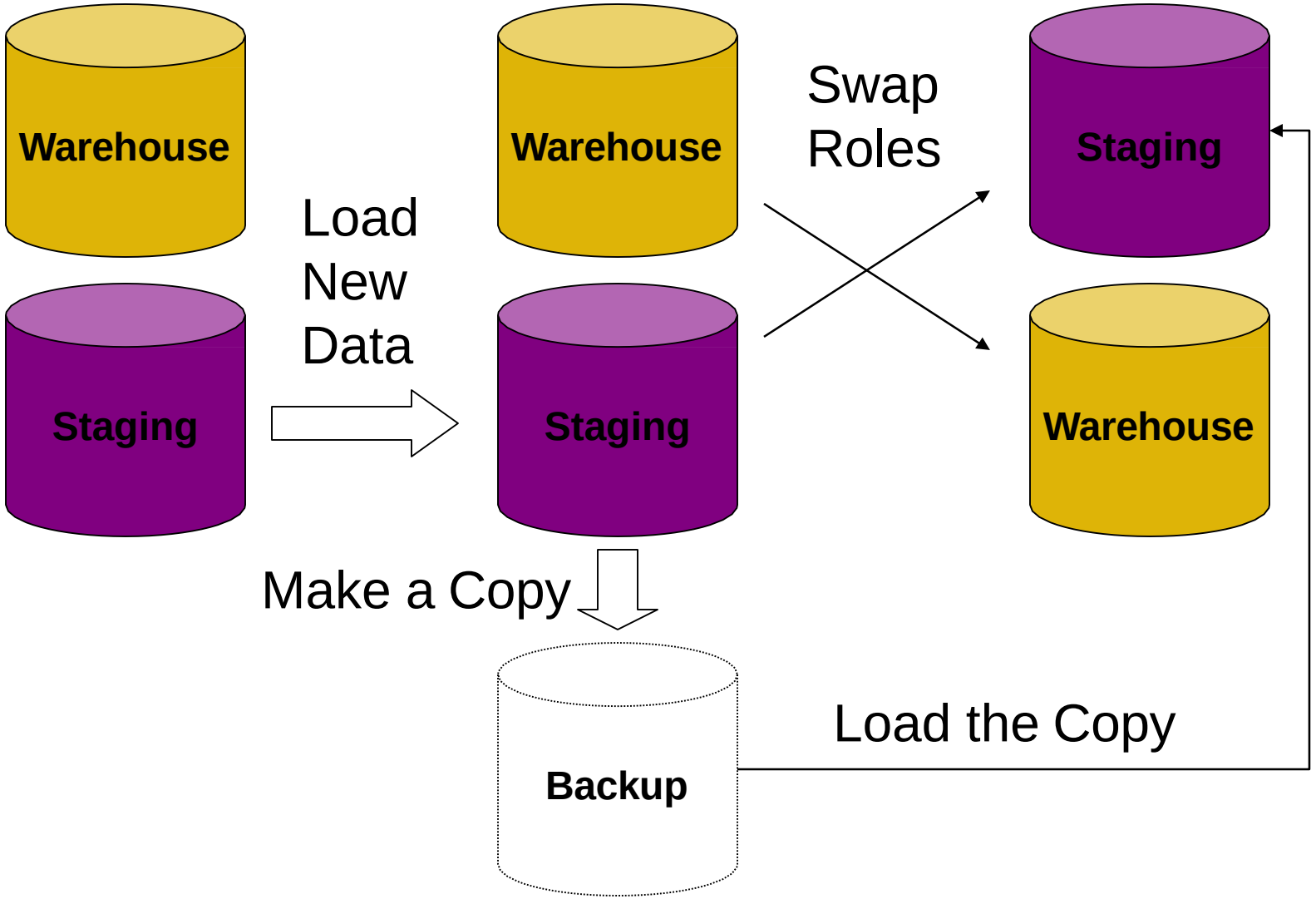
Staging Area vs Data Warehouse

THE UNIVERSITY OF WESTERN AUSTRALIA

- Data warehouse
  - Cleansed, transformed data
  - User-friendly logical design
  - Optimised physical design
    - Indexed data, pre-computed aggregates
- Staging area
  - Intermediate representations of data
  - "Work area" for data transformations
- Same server or different?
  - ➢ Separate staging server and warehouse server
    - Run extraction in parallel with queries
  - ➢ Staging area and warehouse in the same server
    - Less copying of data is required

A data warehouse system might be able to perform very well even without a staging area, and sometimes, the staging area cannot be overlooked.

Alternating Server Approach



ETL vs ELT

- ELT stands for "Extract, Load, and Transform."
- In this process, data gets leveraged via a data warehouse in order to do basic transformations.
- That means there's **no need for data staging**.
- ELT uses cloud-based data warehousing solutions for all different types of data - including structured, unstructured, semi-structured, and even raw data types.

ETL VS ELT

ETL	ELT
Data is transferred to ETL server and transferred back to DB. **High network bandwidth required**	Data remains in the DB except for cross Database loads (e.g. source to target)
Data transformations happen **immediately** after extraction within a staging area. After transformation, the data is loaded into the data warehouse.	Data is extracted, then loaded into the target data system first. Only later is some of the data transformed on an "as-needed" basis for analytical purposes.
Transformations happen within a **staging area** outside the data warehouse.	Transformations happen inside the data system itself, and no staging area is required.
ETL is not normally a solution for **data lakes**. It transforms data for integration with a structured relational data warehouse system.	ELT offers a pipeline for **data lakes** to ingest unstructured data. Then it transforms the data on an as-needed basis for analysis.
ETL works with **cloud-based and onsite data warehouses**. It requires a relational or structured data format.	ELT works with cloud-based data warehousing solutions to support structured, unstructured, semi-structured, and raw data types.

Lecture Outline

- ETL Overview
- Data Staging
- **Data Extraction and Transformation**
- Loading Dimension and Fact Tables
- Handling Data Changes

Data Extraction

Data extraction is a process of fetching **relevant** data from the source systems.

- **Extract step** picks data from different data sources
  - data might come from different sources such as an Excel file, a relational database, or a web site.

What is Logical Data Map?

- The logical data map describes the relationship between the extreme starting points and the extreme ending points of the ETL system usually presented in a table or spreadsheet

Data Extraction: Logical Data Map

Components of Logical Data Map

- The logical data map is usually presented in a table or spreadsheet format and includes the following specific components:
  - 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 or dimension

Data Extraction - Logical Data Map

THE UNIVERSITY OF WESTERN AUSTRALIA

The Logical Data Map example

TARGET						SOURCE					
Mapping Change Date	Target Table	Target Column	Nullable	PK	Data-type, Length	Source Table	Source Column	Data-type, Length	Expression, Transformation	Default Value	Error Types and Handling
Y/M/D	Name of target table	Target table column name	Whether a field can be null	Primary key field for target	Data type & length for target column	Name of source table	Column in source table from which data is extracted	Data type and length for this source column	Decodes, aggregates, conversions, if statements, lookup functions	Value to use in target field when source field is null	Used to document Not null, value if looked up, pk, fk, etc...comments, issues

Data Extraction: Logical Data Map

THE UNIVERSITY OF WESTERN AUSTRALIA

Hints of Logical Data Map:

- **Have a plan -** foundation of the metadata
- **Identify data source candidates -** Identify the likely candidate data sources you believe will support the decisions needed by the business community
- **Analyse source systems with a data-profiling tool** - detected data anomaly must be documented, and best efforts must be made to apply appropriate business rules to rectify data before it is loaded into the data warehouse.

Data Extraction: Data Discovery Phase

THE UNIVERSITY OF WESTERN AUSTRALIA

Context: The analysis of the source system is usually divided into two major phases:

- The **data discovery phase**
- The **anomaly detection phase**

- Data Discovery Phase **key criterion** for the success of the data warehouse is the **cleanliness** and **cohesiveness** of the data within it.

- Once you understand what the target needs to look like, you need to **identify** and **examine** the data sources.

Physical Data Extraction

- **Online Extraction**
  - Data extracted directly from the source system.
  - May access source tables through an intermediate system.
  - Intermediate system usually similar to the source system.

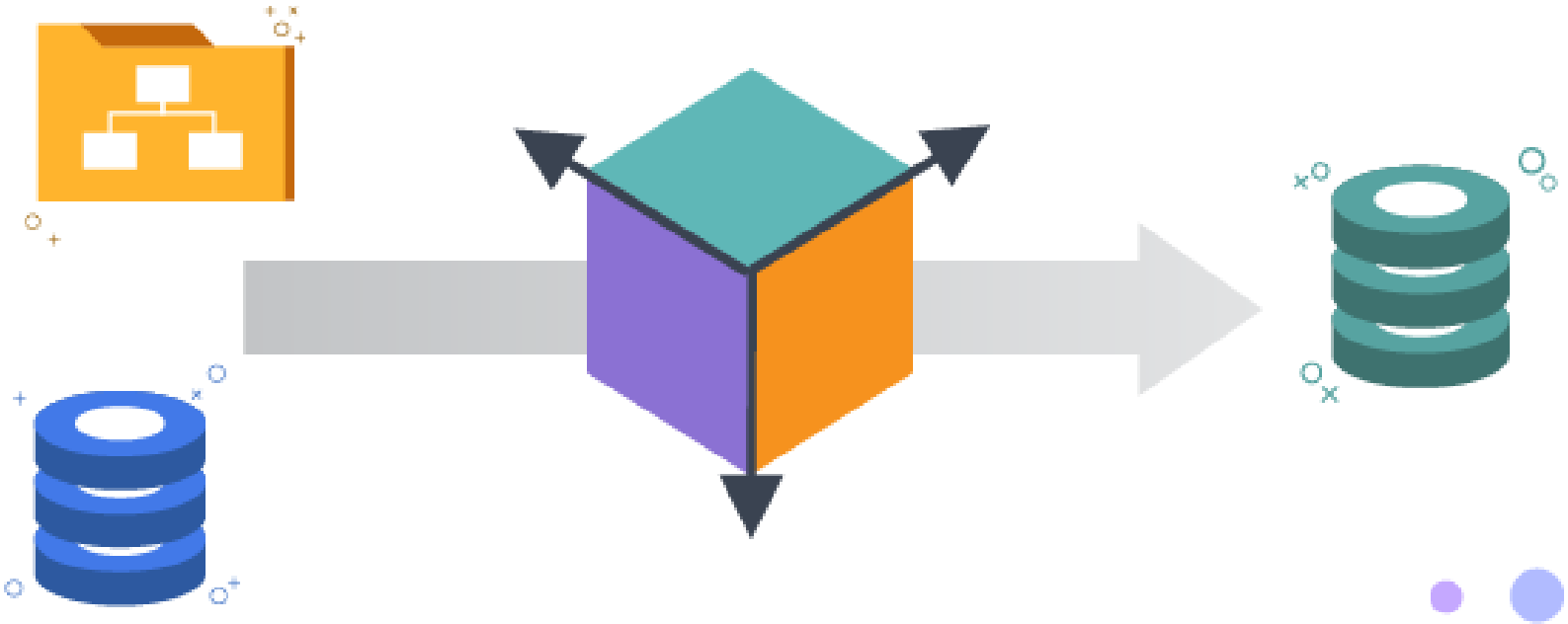
- **Offline Extraction**
  - Data NOT extracted directly from the source system, instead staged explicitly outside the original source system.
  - Data is either already structured or was created by an extraction routine.
  - Some of the prevalent structures are:
    - ✓ Flat files
    - ✓ Dump files
    - ✓ Redo and archive logs
    - ✓ Transportable table-spaces

Logical Data Extraction

- **Full Extraction**
  - The data extracted completely from the source system.
  - No need to keep track of changes.
  - Source data made available as-is with any additional information.

- **Incremental Extraction**
  - Data extracted after a well defined point/event in time.
  - Mechanism used to reflect/record the temporal changes in data (column or table).
  - Sometimes entire tables off-loaded from source system into the Data Warehouse.
  - Can have significant performance impacts on the data warehouse server.

Data Transformation



Data Transformation

- The data after transformation should be
  - Correct
  - Unambiguous
  - Consistent
  - Complete

- Data quality checks are run at 2 places
  - after extraction and
  - after cleaning and confirming

Data Transformation: Cleaning Data

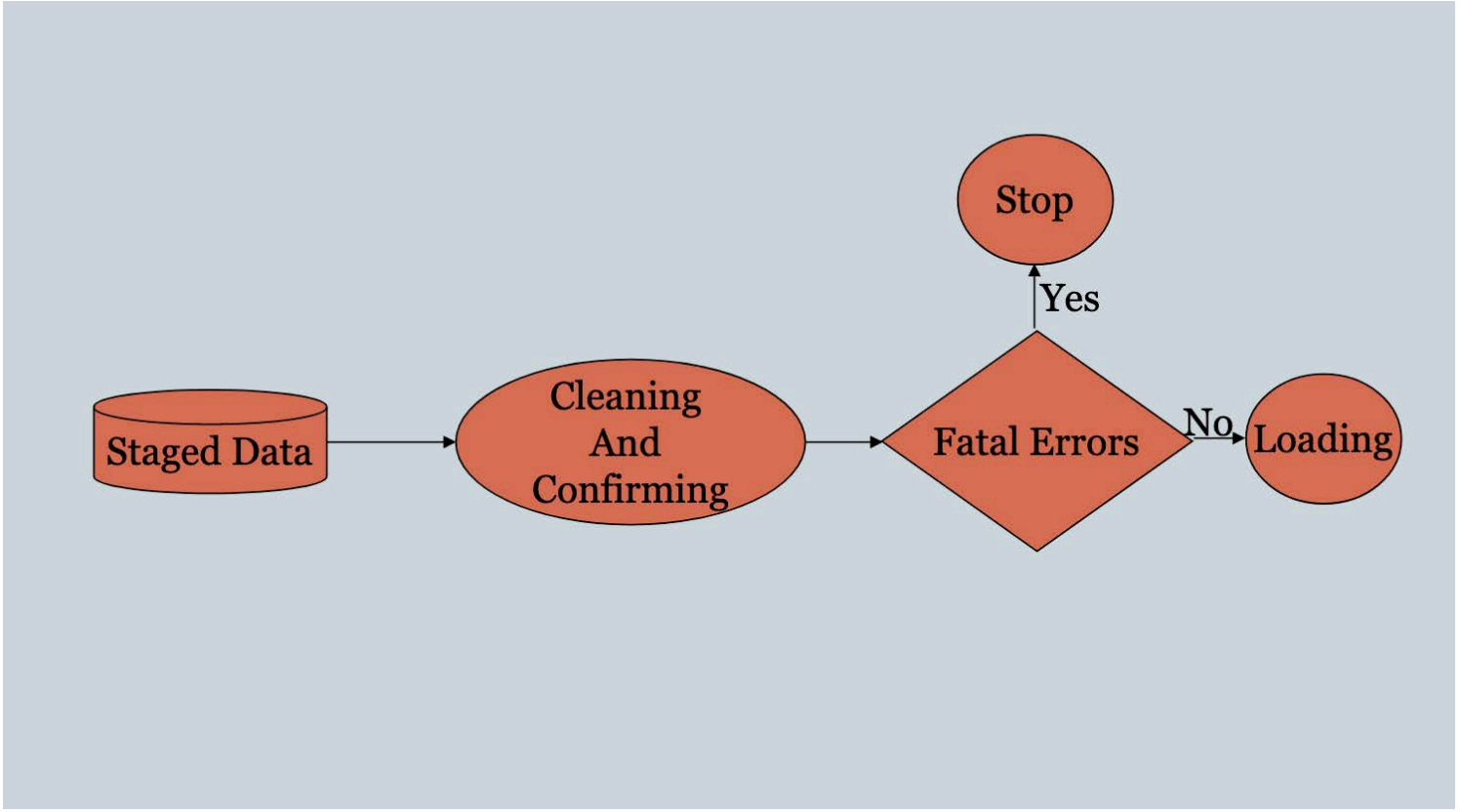
THE UNIVERSITY OF WESTERN AUSTRALIA

- **Anomaly Detection**
  - E.g. a row significantly different from the rest
- **Column Property Enforcement**
  - Null Values in required columns
  - Numeric values that fall outside of expected high and lows
  - Length of an attribute is exceptionally short/long
  - Columns with certain values outside of discrete valid value sets
  - Adherence to a required pattern/member of a set of pattern

Transformation - Confirming

- Confirmation can be done in various ways.
  - Structure Enforcement
  - Tables have proper primary and foreign keys
  - Obey referential integrity
  - Data and rule value enforcement
  - Simple business rules
  - Logical data checks

Transformation



Lecture Outline

- ETL Overview
- Data Staging
- Data Extraction and Transformation
- **Loading Dimension and Fact Tables**
- Handling Data Changes

Data Loads – Loading Dimensions

- The **primary key** is a single field containing meaningless unique integer – Surrogate Keys
- The Data Warehouse owns these keys and never allows any other entity to assign them
- De-normalised 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 that compose the natural key of the dimension

Data Loads – 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, correct natural keys, and final descriptive attributes.

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

- The table is definitely staged, since it is the object to be loaded into the presentation system of the data warehouse.

Surrogate Keys

THE UNIVERSITY OF WESTERN AUSTRALIA

- **Primary keys of dimension tables should be surrogate keys, not natural keys**
  - Natural key: A key that is meaningful to users
  - Surrogate key: A meaningless integer key that is assigned by the data warehouse
  - Avoid using natural keys or codes generated by operational systems.
    - E.g. Account number, UPC code, Social Security Number
  - Syntactic vs. semantic

Benefits of Surrogate Keys

THE UNIVERSITY OF WESTERN AUSTRALIA

- **Data warehouse insulated from changes to operational systems**
- **Easy to integrate data from multiple systems**
  - What if there's a merger or acquisition?
- **Narrow dimension keys (e.g. 8 bytes using natural key, only 4 bytes using surrogate key)**
  - Thinner fact table → Better performance
  - This can actually make a big performance difference.
- **Better handling of exceptional cases**
  - For example, what if the value is unknown or TBD?
  - Using NULL is a poor option
    - Three-valued logic is not intuitive to users
    - Users may get their queries wrong
    - Join performance will suffer
  - Better: Explicit dimension rows for "Unknown", "TBD", "N/A", etc.
- **Avoids tempting query writers to assume implicit semantics**
  - Example: WHERE date_key < '01/01/2020'
  - Will facts with unknown date be included?

Surrogate Key Assignment

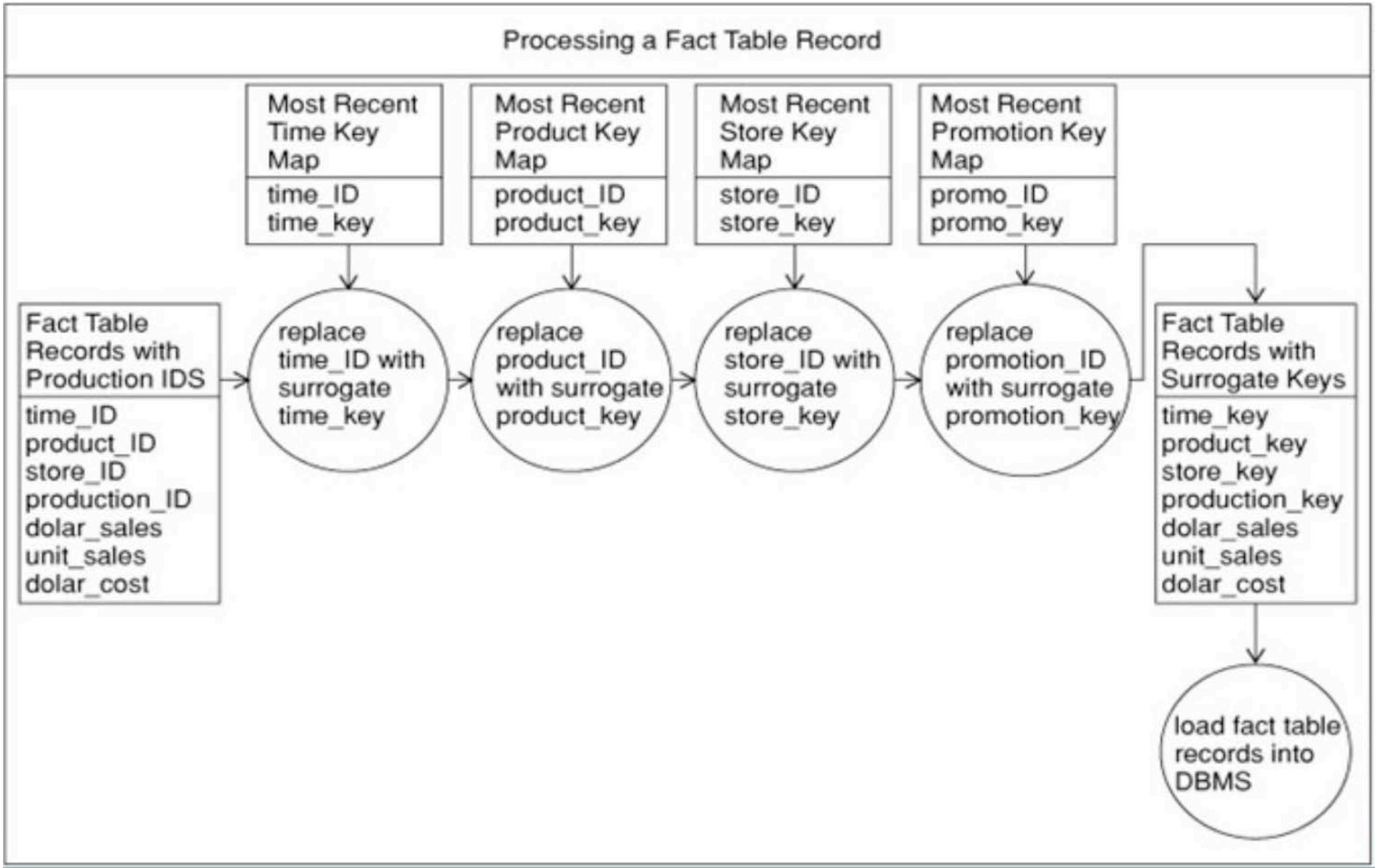
THE UNIVERSITY OF WESTERN AUSTRALIA

- Maintain natural key → surrogate key mapping
  - Separate mapping for each dimension table
  - Can be stored in a relational database table
    - One or more columns for natural key
    - One column for surrogate key
- Need a separate mapping table for each data source
  - Unless data sources already use unified natural key scheme
- Handling multiple dimension rows that preserve history
  - 1st approach: Mapping table contains surrogate key for most current dimension row
  - 2nd approach: Mapping table lists all surrogate keys that were ever used for each natural key
    - Add additional columns to mapping table:
      - Begin_date, End_date, Is_current_flag
    - "Late-arriving" fact rows can use historically correct key
    - Necessary for hybrid slowly changing dimension schemes

Data Loads – 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 modeled as a fact table row.
- If a fact table row exists, it is a measurement

Data Loads – Loading Facts



Fact tables – what to store

- **Additive facts (measures) are easy to work with**
  - Summing the fact value gives meaningful results
  - Additive facts:
    - Quantity sold
    - Total dollar sales
  - Non-additive facts:
    - Averages (average sales price, unit price)
    - Percentages (% discount)
    - Ratios (gross margin)

Month	Quantity Sold
June	12
July	10
August	14
OVERALL	36

Month	Avg. Sales Price
June	$35
July	$28
August	$30
OVERALL	$93 ← Wrong!

Store Additive Quantities

- Store additive quantities in the fact table
- Example:
  - Don't store "unit price"
  - Store "quantity sold" and "total price" instead
- Store additive summaries used for distributive aggregates
  - Numerator and denominator for averages, percentages, ratios
- Big disadvantage of non-additive quantities:
  - Cannot pre-compute aggregates!

Lecture Outline

- ETL Overview
- Data Staging
- Data Extraction and Transformation
- Loading Dimension and Fact Tables
- **Handling Data Changes**

Detecting Changed Rows

THE UNIVERSITY OF WESTERN AUSTRALIA

- Some source systems make things easy
  - All changes timestamped → nothing to do!
    - Usually the case for fact tables
      - Except for Accumulating Snapshot facts
  - For each source system, record latest timestamp covered during previous extraction cycle
- Some source systems just hold snapshot
  - Need to detect new vs. changed vs. unchanged rows
  - New vs. old rows: Use surrogate key mapping table
  - Detecting changed vs. unchanged rows
    - Approach 1: Use a hash function
      - Faster but less reliable
    - Approach 2: Column-by-column comparison
      - Slower but more reliable

Handling Changed Rows

THE UNIVERSITY OF WESTERN AUSTRALIA

- **Using a hash function**
  - Compute a small summary of the data row
  - Store previous hash value in mapping table
  - Compare with hash value of current attribute values
  - If they're equal, assume no change
    - Hash table collisions are possible
  - Cyclic redundancy checksum (aside)
    - Commonly used hash function family
    - No collisions under local changes and byte reorderings
- **Determine which attributes have changed**
  - Requires column-by-column comparison
  - Store untransformed attribute values in mapping table
  - Choose slowly changing dimension approach based on changed attributes

Slowly Changing Dimensions

THE UNIVERSITY OF WESTERN AUSTRALIA

- Compared to fact tables, contents of dimension tables are relatively stable.
  - New sales transactions occur constantly.
  - New products are introduced rarely.
  - New stores are opened very rarely.
- Attribute values for existing dimension rows do occasionally change over time
  - Customer moves to a new address
  - Grouping of stores into districts, regions changes due to corporate re-organisation
- How to handle gradual changes to dimensions?
  - Option 1: Overwrite history
  - Option 2: Preserve history

Overwriting History

- **Simplest option: update the dimension table**
  - "Type 1" slowly changing dimension
- **Example:**
  - Product size incorrectly recorded as "8 oz" instead of "18 oz" due to clerical error
  - Error is detected and fixed in operational system
  - Error should also be corrected in data warehouse
    - Update row in dimension table
    - Update pre-computed aggregates
- **Updating dimension table rewrites history**
  - Brian lived in Perth in 2019
  - Later, Brian moved to Sydney
  - Suppose we update the customer dimension table
  - Query: "Total sales to Perth customers in 2019?"
  - Sales to Brian are incorrectly omitted from the query answer

Preserving History

- Accurate historical reporting is usually important in a data warehouse
- How can we capture changes while preserving history?
- Answer: Create a new dimension row
  - Old fact table rows point to the old row
  - New fact table rows point to the new row
  - “Type 2” slowly changing dimension

Customer Dimension

Cust_key	Name	Sex	City	YOB
457	Brian	Male	Perth	1976
…	…	…	…	…
784	Brian	Male	Syd	1976

⇐ Old dimension row

⇐ New dimension row

Preserving History

THE UNIVERSITY OF WESTERN AUSTRALIA

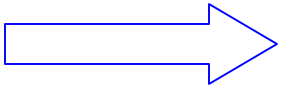
Customer Dimension

Cust_key	Name	Sex	City	YOB
457	Brian	Male	Perth	1976
…	…	…	…	…
784	Brian	Male	Syd	1976

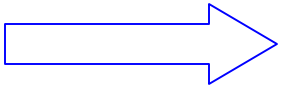
Sales Fact

Cust_key	…	Quantity
…	…	…
457	…	5
…	…	…
784	…	4

Existing fact rows use old dimension row



New fact rows use new dimension row



Overwriting vs. Preserving

Original row in Product dimension:

Product Key	SKU (NK)	Product Description	Department Name
12345	ABC922-Z	IntelliKidz	Education

Updated row in Product dimension:

Product Key	SKU (NK)	Product Description	Department Name
12345	ABC922-Z	IntelliKidz	Strategy

Figure 5-5: SCD type 1 sample rows.

Original row in Product dimension:

Product Key	SKU (NK)	Product Description	Department Name	...	Row Effective Date	Row Expiration Date	Current Row Indicator
12345	ABC922-Z	IntelliKidz	Education	...	2012-01-01	9999-12-31	Current

Rows in Product dimension following department reassignment:

Product Key	SKU (NK)	Product Description	Department Name	...	Row Effective Date	Row Expiration Date	Current Row Indicator
12345	ABC922-Z	IntelliKidz	Education	...	2012-01-01	2013-01-31	Expired
25984	ABC922-Z	IntelliKidz	Strategy	...	2013-02-01	9999-12-31	Current

Figure 5-6: SCD type 2 sample rows.

Pros and Cons

- Type 1: Overwrite existing value
  - \+ Simple to implement
- Type 2: Add a new dimension row
  - \+ Accurate historical reporting
  - \+ Pre-computed aggregates unaffected
  - \- Dimension table grows over time
- Type 2 SCD requires surrogate keys
  - Store mapping from operational key to **most current** surrogate key in data staging area
- To report on Brian's activity over time, constrain on natural key
  - WHERE name = 'Brian'

To overwrite or not?

- Both choices are commonly used
- Easy to “mix and match”
  - Preserve history for some attributes
  - Overwrite history for other attributes

- Questions to ask:
  - Will queries want to use the original attribute value or the new attribute value?
    - In most cases preserving history is desirable
  - Does the performance impact of additional dimension rows outweigh the benefit of preserving history?
    - Some fields like “customer phone number” are not very useful for reporting
    - Adding extra rows to preserve phone number history may not be worth it

Hybrid Solutions

- Suppose we want to be able to report using either old or new values
  - Mostly useful for corporate reorganisations!
  - Example: Sales districts are re-drawn annually
- Solution: Create two dimension columns

- Approach #1: “Previous District” and “Current District”
  - Allows reporting using either the old or the new scheme
  - Whenever district assignments change, all “Current District” values are moved to “Previous District”
  - “Type 3” Slowly Changing Dimension

- Approach #2: “Historical District” and “Current District”
  - Allows reports with the original scheme or the current scheme
  - When district assignment changes, do two things:
    - Create a new dimension row with “Historical District” = old value
    - Overwrite relevant dim. rows to set “Current District” = new value

Example comparison

Original row in Product dimension:

Product Key	SKU (NK)	Product Description	Department Name	...	Row Effective Date	Row Expiration Date	Current Row Indicator
12345	ABC922-Z	IntelliKidz	Education	...	2012-01-01	9999-12-31	Current

Rows in Product dimension following department reassignment:

Product Key	SKU (NK)	Product Description	Department Name	...	Row Effective Date	Row Expiration Date	Current Row Indicator
12345	ABC922-Z	IntelliKidz	Education	...	2012-01-01	2013-01-31	Expired
25984	ABC922-Z	IntelliKidz	Strategy	...	2013-02-01	9999-12-31	Current

Figure 5-6: SCD type 2 sample rows.

Original row in Product dimension:

Product Key	SKU (NK)	Product Description	Department Name
12345	ABC922-Z	IntelliKidz	Education

Updated row in Product dimension:

Product Key	SKU (NK)	Product Description	Department Name
12345	ABC922-Z	IntelliKidz	Strategy

Figure 5-5: SCD type 1 sample rows.

Original row in Product dimension:

Product Key	SKU (NK)	Product Description	Department Name
12345	ABC922-Z	IntelliKidz	Education

Updated row in Product dimension:

Product Key	SKU (NK)	Product Description	Department Name	Prior Department Name
12345	ABC922-Z	IntelliKidz	Strategy	Education

Figure 5-8: SCD type 3 sample rows.

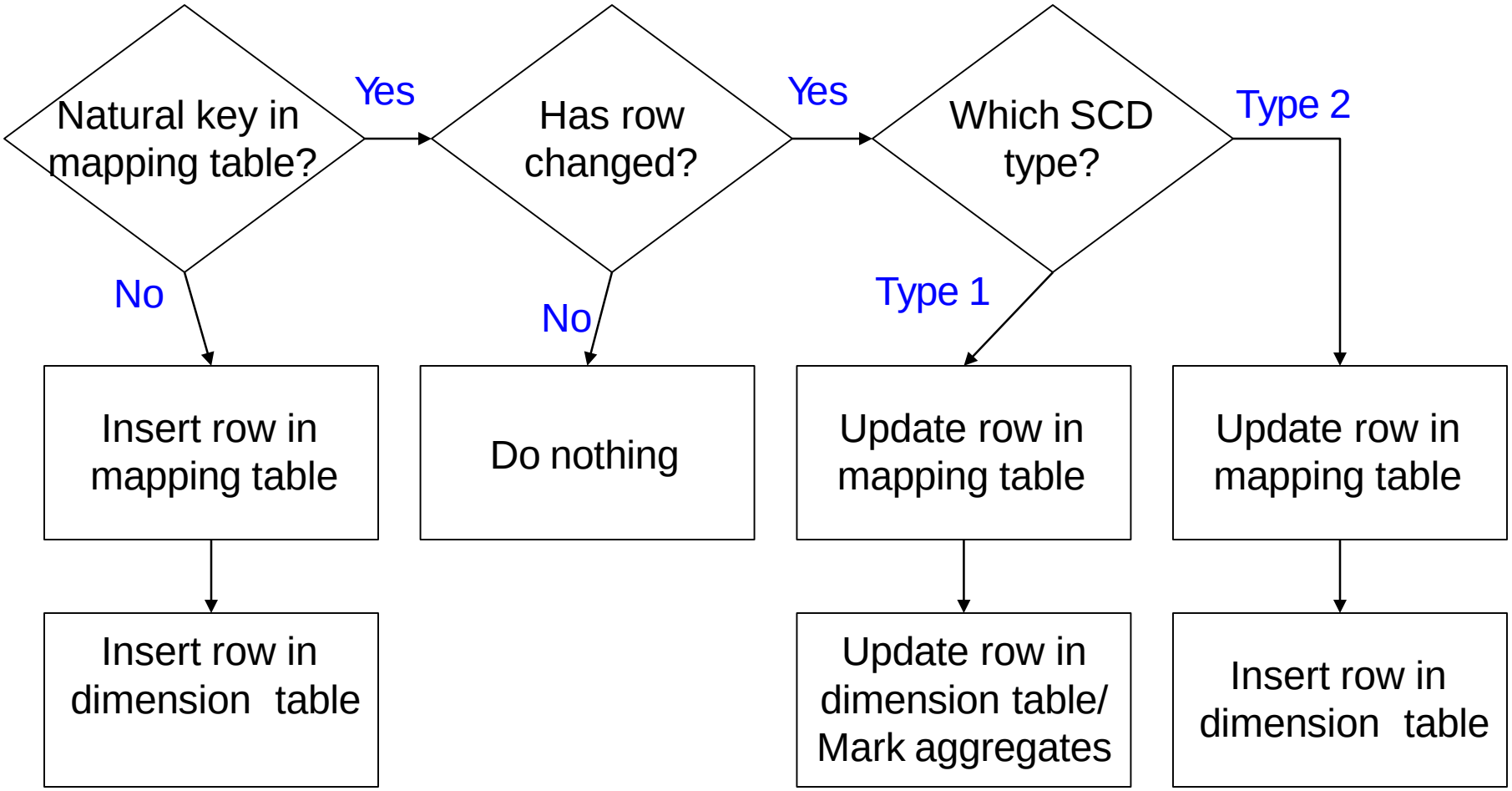
Summary of SCD Techniques

THE UNIVERSITY OF WESTERN AUSTRALIA

SCD Type	Dimension Table Action	Impact on Fact Analysis
Type 0	No change to attribute value.	Facts associated with attribute's original value.
Type 1	Overwrite attribute value.	Facts associated with attribute's current value.
Type 2	Add new dimension row for profile with new attribute value.	Facts associated with attribute value in effect when fact occured.
Type 3	Add new column to preserve attribute's current and prior values.	Facts associated with both current and prior attribute alternative values.
Type 4	Add mini-dimension table containing rapidly changing attributes.	Facts associated with rapidly changing attributes in effect when fact occured.
Type 5	Add type 4 mini-dimension, along with overwritten type 1 mini-dimension key in base dimension.	Facts associated with rapidly changing attributes in effect when fact occurred, plus current rapidly changing attribute values.
Type 6	Add type 1 overwritten attributes to type 2 dimension row, and overwrite all prior dimension rows.	Facts associated with attribute value in effect when fact occurred, plus current values.
Type 7	Add type 2 dimension row with new attribute value, plus view limited to current rows and/or attribute values.	Facts associated with attribute value in effect when fact occurred, plus current values.

Figure 5-17: Slowly changing dimension techniques summary.

Dimension Loading Workflow



Duplicates from multiple sources

- Information about the same logical entity found in multiple source systems
- Combine info. into single dimension row
- Problems:
  - Determine which rows reference same entity
    - Sometimes it's hard to tell!
    - Referred to as the merge/purge problem
  - Resolve conflicts between source systems
    - Matching records with different values for the same field
    - Approach #1: Believe the "more reliable" system
    - Approach #2: Include both values as separate attributes

Merge/Purge

- How can we determine that these are the same person?

FName	LName	Address	City	Zip
B	Babcock	3135 Campus Dr #112	San Mateo	94403
Brian	Bobcock	3135 Compass Drive Apt 112	San Mateo	94403-3205

- Fuzzy matching based on textual similarity

- Transformation rules
- Comparison to known good source
  - NCOA: National Change Of Address database
- Related application: Householding
  - Can we determine when two individuals/accounts belong to the same household?
  - Send one mailing instead of two

References

- Some slides are adapted from
  - Stanford CS345
  - “The Data Warehouse ETL Toolkit” by Ralph Kimball
  - ETL Slides
- Readings
  - ELT on Cloud Data Warehouse