Armazéns de Dados

Departamento de Engenharia Informática (DEI/ISEP)
Paulo Oliveira
pjo@isep.ipp.pt

Adaptado do Original de:
Fátima Rodrigues (DEI/ISEP)

1

Bibliography

- The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming and Delivering Data

  Ralph Kimball, Joe Caserta

  Wiley, 2004

  Chapters 3, 4, 5, and 6

2

Extraction, Transformation, and Loading (ETL)

3

Extraction

4

Data Extraction

- First step in the process of **getting data into the DW** environment
- Means **reading** and **understanding** the **source data** and **copying** the data needed in the DW **into the staging area** for further manipulation
- Often performed by **custom routines** – not recommended because of:
  - High program maintenance
  - No automatically generated metadata
- Increasingly performed by specialized **ETL software**
  - Provides simpler, faster and cheaper development

5

5

Data Extraction

- ETL process needs to **integrate systems having different**:
  - Database management systems
  - Operating systems
  - Hardware
- Necessary to build a **logical data map** that documents the **relationship between original source attributes and final destination attributes**
  - Identify data sources
  - Analyse source systems with a data profiling tool (data quality)
  - Data lineage and business rules
  - Validate calculations and formulas

6

6

Components of the Logical Data Map

- **Logical data map** is presented in a table that includes:
  - Target table, target attribute and table type (dimension or fact)
  - Slowly Changing Dimension (SCD) type per target attribute:
    - Type 1 – overwrite (e.g., customer first name)
    - Type 2 – retain history (e.g., customer city)
    - Type 3 – retain valid alternative values (e.g., customer region)
    - Type 4 – new history table
    - Type 6 – hybrid approach
  - Source database, source table(s) and source attribute(s)
  - Transformation
    - Performed manipulation annotated in SQL or pseudo-code

7

7

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						Surrogate key.
EMPLOYEE_DIM	EMPLOYEE_ID	NUMBER	Dimension		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)
EMPLOYEE_DIM	DEPARTMENT_CODE	VARCHAR2(12)	Dimension	2	HR_SYS	DEPARTMENTS	CODE	VARCHAR2(12)	select d.code from employee p, employee_department pd, departments d where p.employee_id= pd.employee_id and pd.department_id = d.department_id
EMPLOYEE_DIM	DEPARTMENT_NAME	VARCHAR2(75)	Dimension	2	HR_SYS	DEPARTMENTS	DESCRIPTION	VARCHAR2(75)	select d.description from employee p, employee_department pd, departments d where p.employee_id= pd.employee_id and pd.department_id = d.department_id
EMPLOYEE_DIM	PART_TIME_FLAG	VARCHAR2(1)	Dimension	1	HR_SYS	EMPLOYEES	PERCENTAGE	VARCHAR2(1)	select decode(sign(percentage-100),-1,'Y','N') from employee
EMPLOYEE_CONTRACT_FACT	EFFECTIVE_DATE_KEY	NUMBER	Fact	N/A	DW_PROD_HR_SYS	DATE_DIM, EMPLOYEE_CONTRACT	DATE_KEY	NUMBER	where employee_contract.eff_date = dw_prod.date_dim.cal_date
EMPLOYEE_CONTRACT_FACT	END_DATE_KEY	NUMBER	Fact	N/A	DW_PROD_HR_SYS	DATE_DIM, EMPLOYEE_CONTRACT	DATE_KEY	NUMBER	where employee_contract.end_date = dw_prod.date_dim.cal_date
EMPLOYEE_CONTRACT_FACT	CURRENCY_KEY	NUMBER	Fact	N/A	DW_PROD_HR_SYS	CURRENCY_DIM, EMPLOYEE_CONTRACT	CURRENCY_KEY	NUMBER	where employee_contract.currency_code = dw_prod.currency_dim.currency_code
EMPLOYEE_CONTRACT_FACT	RATE_TYPE_KEY	NUMBER	Fact	N/A	DW_PROD_HR_SYS	RATE_TYPE_DIM, EMPLOYEE_CONTRACT	RATE_TYPE_KEY	NUMBER	where employee_contract.rate_type_id = dw_prod.rate_type_dim.rate_type_id
EMPLOYEE_CONTRACT_FACT	PROJECT_KEY	NUMBER	Fact	N/A	DW_PROD_HR_SYS	PROJECT_DIM, EMPLOYEE_CONTRACT	PROJECT_KEY	NUMBER	where employee_contrac.project_code = dw_prod.project_dim.project_code
EMPLOYEE_CONTRACT_FACT	EMPLOYEE_ROLE_KEY	NUMBER	Fact	N/A	DW_PROD_HR_SYS	EMPLOYEE_ROLE_DIM, EMPLOYEE_CONTRACT	EMPLOYEE_ROLE_KEY	NUMBER	where employee_contract.employee_role = employee_role_dim.contractor_role_name
EMPLOYEE_CONTRACT_FACT	CONTRACT_TYPE_KEY	NUMBER	Fact	N/A	DW_PROD_HR_SYS	CONTRACT_TYPE_DIM, EMPLOYEE_CONTRACT	CONTRACT_TYPE_KEY, CONTRACT_TYPE_ID	NUMBER	where dw_prod.employee_contract.contract_type = contract_type_dim.contract_type_id
EMPLOYEE_CONTRACT_FACT	CONTRACT_NUMBER	NUMBER	Fact	N/A	DW_PROD_HR_SYS	EMPLOYEE_CONTRACT	CONTRACT_NUMBER	NUMBER	Degenerate Dimension
EMPLOYEE_CONTRACT_FACT	RATE_AMOUNT_LOCAL	NUMBER	Fact	N/A	DW_PROD_HR_SYS	EMPLOYEE_CONTRACT	AMOUNT	NUMBER	sum(amount)
EMPLOYEE_CONTRACT_FACT	RATE_AMOUNT_USD	NUMBER	Fact	N/A	DW_PROD_HR_SYS	EMPLOYEE_CONTRACT	AMOUNT	NUMBER	select ec.amount * avg(cc.conversion_rate) from employee_contract ec, currency_conversion cc where ec.currency = cc.from_currency and cc.effective_Date between ec.effective_date and ec.end_date group by ec.amount

8

Analysis of the Source System

- **ER model** of the system
- **Reverse engineering** by looking at metadata of the source system to understand it
  - Unique identifiers and natural keys
  - Data types
  - Relationships between tables: 1-to-1; 1-to-many; many-to-many
    - Problematic when source database does not have foreign keys defined
    - Discrete relationships (reference tables)

Dealing with Derived Data

- Derive from base facts or accept calculated columns from the source systems?
- If calculations are recreated in the ETL process they must be synchronized with the business rules that define them
  - If the calculation logic changes in the source system, the ETL process will have to be modified and redeployed – **it is necessary to capture the calculation as metadata**

Integrating Data From Different Sources

- It is very important to determine the **system-of-record** – originating source of data
  - In most enterprises, data is stored across many different systems
- When a dimension is populated by several distinct systems, it is important to store:
  - **The source system from which the data comes**
  - **The unique identifier (primary key) from that source system**
- Identifiers should be viewable by end-users to ensure that the **dimension reflects their data**, and they can **tie back to** their operational system

11

11

Two Generic Types of Data Extracts

- **Static extract** is a method of capturing a snapshot of **all the source data** at a point in time
  - Used **to fill the DW initially**
- **Incremental extract** captures **only the changes** that have occurred in the source data since last capture
  - Used for **ongoing DW updates**

12

12

Incremental Extract

- Retrieve **only the records** from the source that were **inserted or modified since last extraction**
- **Audit columns** usually populated by the **front-end application** or via **database triggers** fired automatically as **records are inserted or updated**
  - Create date/time-stamp
  - Last update date/time-stamp
- **Database logs**
  - Only images that are logged after the last data extraction are selected from the log to identify new and changed records

13

13

Static Extract – Worst Case Scenario

- Source system does not notify changes and does not have date/time-stamp on its own inserts/updates
- For small data tables, use a brute force approach for **comparing every incoming attribute with every attribute in the DW** to see if anything changed
  - Preserve yesterday's entire data in the staging area
  - Bring today's entire data in the staging area
  - Perform a comparison
  - **Inefficient but the most reliable**
- For larger tables, use the **Cyclic-Redundancy Checksum** (CRC) approach

14

14

Static Extract – CRC Approach

- **Procedure**
  - Treat the entire incoming record as a string
  - Compute the CRC value of the string
    - Numeric value of about 20 digits
  - Compare the CRC value of new record with the CRC value of existing record
  - If the CRC values match
    - New record is equal to existing record
  - If the CRC values do not match
    - Do a field-by-field comparison to see what has changed
    - Depending on whether the changed field is a type-1, type-2, type-3, type-4 or type-6 change, do the necessary updates
- **Some ETL packages include CRC computation**

15

Transformation

16

Data Cleaning and Conforming

- **Data cleaning** means identifying and correcting errors in data
  - Misspellings
  - Domain violations
  - Missing values
  - Duplicate records
  - Business rules violations
- **Data conforming** means resolving the conflicts between incompatible data sources so that they can be used together
  - Requires an enterprise-wide agreement to use standardized domains and measures

17

17

Data Quality Dimensions

- **Accuracy**
  - Correct and unambiguous
- **Integrity**
  - Protected from deliberate deviations
- **Consistency**
  - Consistently defined and maintained
  - Values use the same format (e.g., Lisbon and not Lisb nor Lx)
- **Validity**
  - Valid data, based on business or industry rules and standards
- **Completeness**
  - Attributes are not null

18

18

What Causes Poor Data Quality?

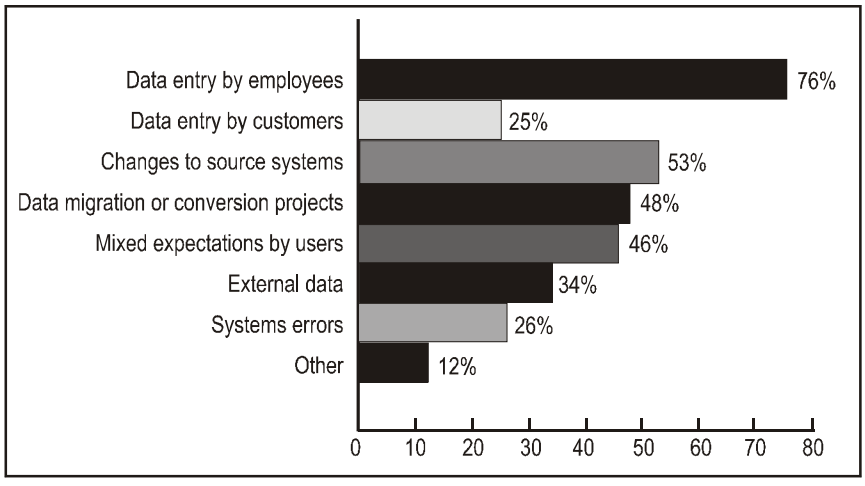
- There are **no standards for data capture**
- **Standards** may exist but **are not enforced** at the point of data capture
- **Inconsistent data entry** occurs (use of nicknames or aliases)
- **Data entry mistakes** happen (character transposition, misspellings, and so on)
- **Integration of data from different systems** with different data quality standards

Data quality problems are perceived as **time-consuming** and **expensive** to fix

19

19

Primary Sources of Data Quality Problems



Source: The Data Warehousing Institute, Data Quality and the Bottom Line, 2002

20

20

Data Cleaning

- Source systems contain "**dirty data**" that **must be cleaned**
- ETL software contains **rudimentary data cleaning capabilities**
- **Specialized data cleaning software** is often used
- **Steps in data cleaning**
  - Parsing
  - Correcting
  - Standardizing
  - Matching
  - Consolidating

21

21

Data Cleaning Steps

- **Parsing**
  - Locates and identifies individual data elements in the source attributes and then isolate these data elements in the targets
  - Examples
    - Parsing into first, middle, and last name
    - Parsing into street number and street name
    - Parsing into zip code and city
- **Correcting**
  - Corrects parsed individual values using data algorithms and secondary data sources
  - Example
    - Correct an address by adding a zip code

22

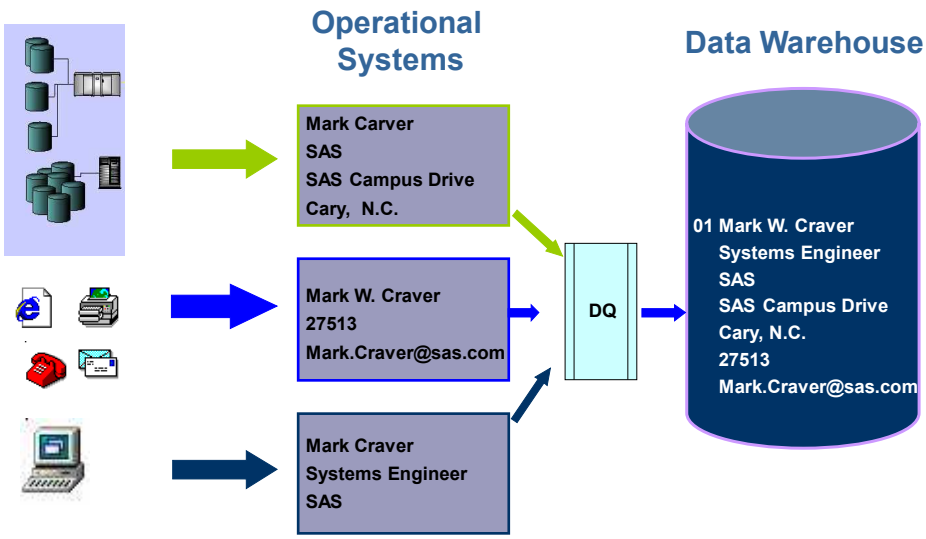
22

Data Cleaning Steps

- **Standardizing**
  - Applies conversion rules to transform data into its preferred and consistent format
  - Example: Replacing an acronym, replacing an abbreviation
- **Matching**
  - Searching and matching records within and across the parsed, corrected and standardized database on predefined detection rules to identify duplicates
  - Example: Identifying similar names and addresses
- **Consolidating**
  - Analyzing and identifying relationships between matched records and **merging them into one representation**

23

Data Cleaning Example



24

Loading

25

Data Loading

- Data is physically **moved to the DW**
- Usually takes place within a "**load window**"
- **Recent trend is near real-time loading of the DW** as they are increasingly used in real-time decisions

26

26

Load of a Dimension

- Creating and assigning **surrogate keys**
- Dealing with the **Slowly Changing Dimension (SCD) attributes**
  - Type 1 (overwrite)
  - Type 2 (partitioning history)
  - Type 3 (alternate reality)
  - Type 4 (history table)
  - Type 6 (hybrid approach)
- Writing the dimension to a physical table with **descriptive attributes**

27

27

Generate Surrogate Keys for Dimensions

- Via **triggers** in the DBMS
  - Read the latest surrogate key, generate the next value, create the record
  - Disadvantage: **performance bottleneck**
- Via the **ETL process** – ETL tool generates the unique numbers
  - Surrogate key counter per dimension
- **Auto-number attribute** in the dimension primary key

28

28

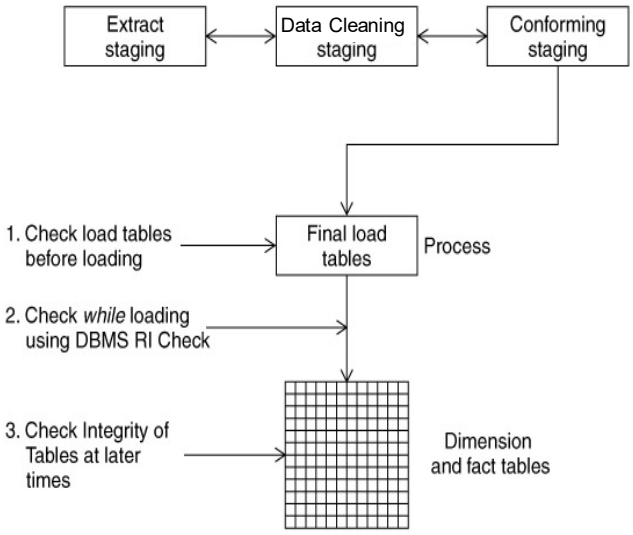
Referential Integrity

- In dimensional modeling **referential integrity means**:
  - Every fact table is filled with legitimate foreign keys
  - No fact table record contains corrupt or unknown foreign key references
- **Two ways to violate referential integrity** in a dimensional schema:
  - Load a fact record with one or more inexistent foreign keys
  - Delete a dimension record whose primary key is being used in the fact table

29

29

Guaranteeing Referential Integrity



30

30

Guaranteeing Referential Integrity

- **Check before loading**
  - Check before adding fact records
  - Check before deleting dimension records
  - **Best approach**
- **Check while loading**
  - DBMS enforces referential integrity
  - **Elegant but typically slow**
    - Some exceptions: Red Brick database system is capable of loading 100 million records in an hour into a fact table where it is checking referential integrity on all the dimensions simultaneously!
- **Check after loading**
  - No referential integrity in the database
  - Periodic checks for invalid foreign keys looking for invalid data
  - **Prohibitively slow**

Check Ref. Integrity After Loading

- **Query example**

```
SELECT ProductKey
FROM FactSales
WHERE ProductKey NOT IN (
                          SELECT ProductKey
                          FROM DimProduct
                        )
```

Loading Surrogate Keys into Fact Table

Option 1:

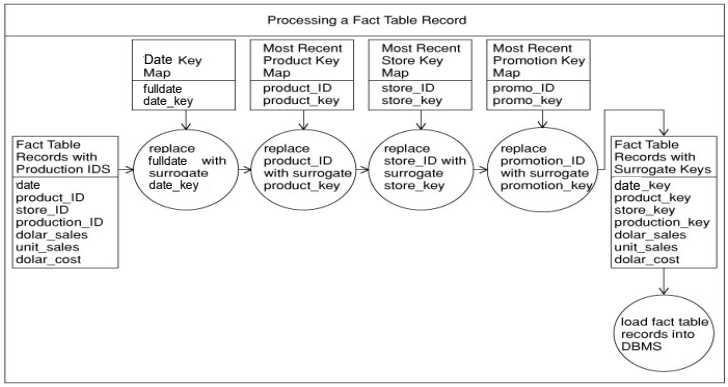
- **Look up the current surrogate key** in each dimension table
- Fetch the record with **the most current surrogate key** for the natural key and use that surrogate key
- **Good option but slower**

Option 2:

- Maintain **a surrogate key lookup table for each dimension**
- Table is updated whenever **a new record is added or when a type-2, type-4 or type-6 update occurs** in an existing dimensional entity
- Dimensions must be updated **before any facts are loaded into the DW** to guarantee referential integrity
- Known as the **surrogate key pipeline method**

33

Surrogate Key Pipeline



- When loading a fact table, the final ETL step **converts the natural keys of the new or updated records into the correct surrogate key** of the dimensions using the **key mapping tables**

34

Correcting Facts

- Should we change fact data once they are in the DW?
  - **No**, if they **represent business events**
  - **Yes**, if they **represent errors** in the operational system

1. Negate the fact and reload it

- Create an exact duplicate of the fact where all the measurements are negated (minus), so the measures "cancel" each other in summaries
- Reasons:
  - Audit purposes (primary reason)
  - Capturing/measuring erroneous entries is significant to the business (analytical reasons)

2. Update the fact

- Fact is updated using an SQL Update

35

35

Correcting Facts

3. Delete and reload the fact

- Physical delete – record is deleted
- Logical delete – record is tagged "deleted"
  - Use of an additional Boolean column
  - Every query that includes the fact table must apply a constraint on the Boolean column to filter out the logically deleted records

36

36

Late-Arriving Fact Rows

- What to do when **late-arriving data** that should have been loaded into the DW weeks or months ago is received?
- Suppose sales facts that are several months old are received
  - It is necessary to choose the old contemporary dimension rows that apply to those sales
  - If the dimensions are a type 2 SCD, inserting these late arriving facts involves:
    1. For each dimension, find the corresponding dimension row whose date stamp is the **latest date stamp less than or equal to the date of the sales**
    2. Using the surrogate keys found in each of the dimension rows from step 1, **replace the natural keys of the late-arriving fact rows with the surrogate keys**
    3. **Insert the late-arriving fact rows into the fact table**

37

Late-Arriving Fact Rows (example)

CustomerKey	CustomerID	...	EffectiveDate	ExpiredDate	IsCurrent
⋮	⋮	⋮	⋮	⋮	⋮
844	728	⋮	16/11/2018	19/03/2020	No
⋮	⋮	⋮	⋮	⋮	⋮
1924	728	⋮	20/03/2020	08/09/2023	No
⋮	⋮	⋮	⋮	⋮	⋮
3726	728	⋮	09/09/2023	null	Yes
⋮	⋮	⋮	⋮	⋮	⋮

If a sales arrives with the date of 25/07/2023, it will be inserted in the Sales Fact Table with the CustomerKey 1924 (and not with the key of the current record)

38

Late-Arriving Dimension Rows

- Suppose that John Smith's customer dimension row contains a marital attribute that always contained the value 'single'
- There are a number of customer rows for John Smith, because this is a Type 2 SCD, and other attributes like address, zip code and job have changed
- Today we are notified that John Smith married on July 15, 2023
- To add this new information to the DW requires:
  1. **Insert a new row with a new surrogate key**, for John Smith, into the customer dimension, with the new marital status and the *effective date* set to July 15, 2023
  2. Scan forward in the customer dimension table from July 15, 2023 finding any other rows for John Smith, and overwrite the *marital status* to *married*
  3. **Find all fact rows** involving John Smith from July 15, 20123 to the first next change for him in the dimension after July 15, 20123 and **update the customer foreign key in those fact rows to the new surrogate key** created in step 1

39

39

Late-Arriving Dimension Rows (example)

Before Customer Dimension Update:

CustomerKey	CustomerID	...	MaritalStatus	EffectiveDate	ExpiredDate	IsCurrent
844	728		Single	16/11/2018	19/03/2020	No
1924	728		Single	20/03/2020	08/09/2023	No
3726	728		Single	09/09/2023	null	Yes

After Customer Dimension Update:

CustomerKey	CustomerID	...	MaritalStatus	EffectiveDate	ExpiredDate	IsCurrent
844	728		Single	16/11/2018	19/03/2020	No
1924	728		Single	20/03/2020	14/07/2023	No
3726	728		Married	09/09/2023	null	Yes
4341	728		Married	15/07/2023	08/09/2023	No

Sales Fact Table records regarding the period between 15/07/2023 and 09/09/2023 need to be updated with the new customer dimension key 4341

51

40