Slowly changing dimensions

Slowly changing dimensions

Till now we have pretended dimensions never change…

… indeed they are rather static usually …

… but surprise… they do change in the real world…

Develop a strategy to handle changes in dimensions…

Slowly changing dimensions

1. *Be proactive: Ask about potential changes*
2. *Business users + IT*
3. *Strategy for each changing attribute*

Kimball introduced SCD in 1995 and distinguished between different types (1, 2, 3, …).

Type 0: Retain Original

Designing SCD1

In SCD type 1, the values are overwritten and no history is maintained, so once the data is updated, there is no way to find out what the previous value was. The new queries will always return the most recent value. Here is an example of an SCD1 table:

CustomerID	Name	City	Email	...
1	Adam	New York	adam@....	...

CustomerID	Name	City	Email	...
1	Adam	New Jersey	adam@....	...

Figure 4.3 – Example of SCD type 1

In this example, the value of the **City** column is changing from **New York** to **New Jersey**. The value just gets overwritten.

Designing SCD2

In SCD2, we maintain a complete history of changes. Every time there is a change, we add a new row with all the details without deleting the previous values. There are multiple ways in which we can accomplish this. Let's take a look at the most common approaches.

Using a flag

In this approach, we use a flag to indicate if a particular value is active or if it is current. Here is an example of this:

SurrogateID	CustomerID	Name	City	isActive	...
1	1	Adam	New York	True	...

SurrogateID	CustomerID	Name	City	isActive	...
1	1	Adam	New York	False	...
2	1	Adam	New Jersey	False	...
3	1	Adam	Miami	True	...

Figure 4.4 – Example of SCD type 2: flag

In the second table, every time there is a change, we add a new row and update the `isActive` column of the previous rows to False. That way, we can easily query the active values by filtering on the `isActive=True` criteria.

Using version numbers

In this approach, we use version numbers to keep track of changes. The row with the highest version is the most current value. Here is an example of this:

SurrogateID	CustomerID	Name	City	Version	...
1	1	Adam	New York	0	...

SurrogateID	CustomerID	Name	City	Version	...
1	1	Adam	New York	0	...
2	1	Adam	New Jersey	1	...
3	1	Adam	Miami	2	...

Figure 4.5 – Example of SCD type 2: version numbers

In the previous example, we need to filter on the **MAX(Version)** column to get the current values.

Using date ranges

In this approach, we use date ranges to show the period a particular record (row) was active, as illustrated in the following example:

SurrogateID	CustomerID	Name	City	StartDate	EndDate
1	1	Adam	New York	01-Jan-2020	NULL

SurrogateID	CustomerID	Name	City	StartDate	EndDate
1	1	Adam	New York	01-Jan-2020	25-Mar-2020
2	1	Adam	New Jersey	25-Mar-2020	01-Dec-2020
3	1	Adam	Miami	01-Dec-2020	NULL

Figure 4.6 – Example of SCD type 2: date ranges

As a variation to the date-range approach, we could also add a flag column to easily identify active or current records. The following example shows this approach:

SurrogateID	CustomerID	Name	City	StartDate	EndDate	isActive
1	1	Adam	New York	01-Jan-2020	25-Mar-2020	False
2	1	Adam	New Jersey	25-Mar-2020	01-Dec-2020	False
3	1	Adam	Miami	01-Dec-2020	NULL	True

Figure 4.7 – Example of SCD type 2: date ranges and flag

Designing SCD3

In SCD3, we maintain only a partial history and not a complete history. Instead of adding additional rows, we add an extra column that stores the previous value, so only one version of historic data will be preserved. As with the SCD2 option, here again, we can choose to add date columns to keep track of modified dates, but we don't need surrogate keys in this case as the identification key of the record doesn't change. Here is an example of this:

CustomerID	Name	City	PrevCity	...
1	Adam	New York	NULL	

CustomerID	Name	City	PrevCity	...
1	Adam	New Jersey	New York	

Figure 4.8 – Example of SCD type 3

In the previous example, we have added a new column called **PrevCity**. Every time the value of **City** changes, we add the previous value to **PrevCity** and update the **City** column with the current city.

Designing SCD4

SCD4 was introduced for dimension attributes that change relatively frequently. In type 4, we split the fast-changing attributes of the dimension table into another smaller dimension table and also reference the new dimension table directly from the fact table.

For example, in the following diagram, if we assume that the carpool (also known as *High occupancy vehicles*) pass needs to be purchased every month, we can move that field to a smaller mini-dimension and reference it directly from the fact table:

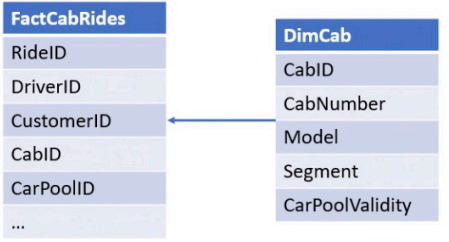


Figure 4.9 – Example of SCD4: before split

We can split the table into a mini **DimCarPool** dimension, as in the following diagram:

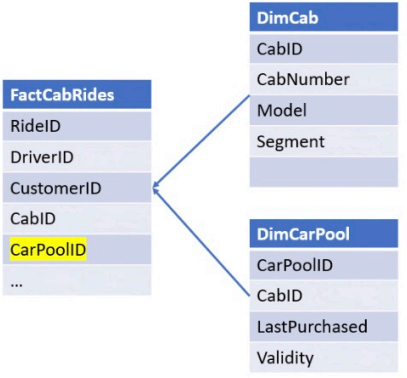


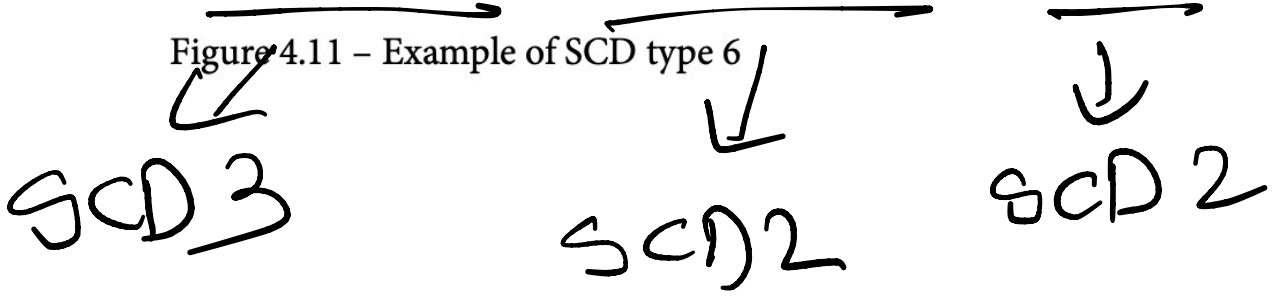
Figure 4.10 – Example of SCD type 4: after split

Designing SCD6

Type 6 is a combination of 1, 2, and 3. In this type, along with the addition of new rows, we also update the latest value in all the rows, as illustrated in the following screenshot:

Surrogate ID	Customer ID	Name	CurrCity	PrevCity	StartDate	EndDate	isActive
1	1	Adam	Miami	NULL	01-Jan-2020	25-Mar-2020	False
2	1	Adam	Miami	New York	25-Mar-2020	01-Dec-2020	False
3	1	Adam	Miami	New Jersey	01-Dec-2020	NULL	True

Figure 4.11 – Example of SCD type 6

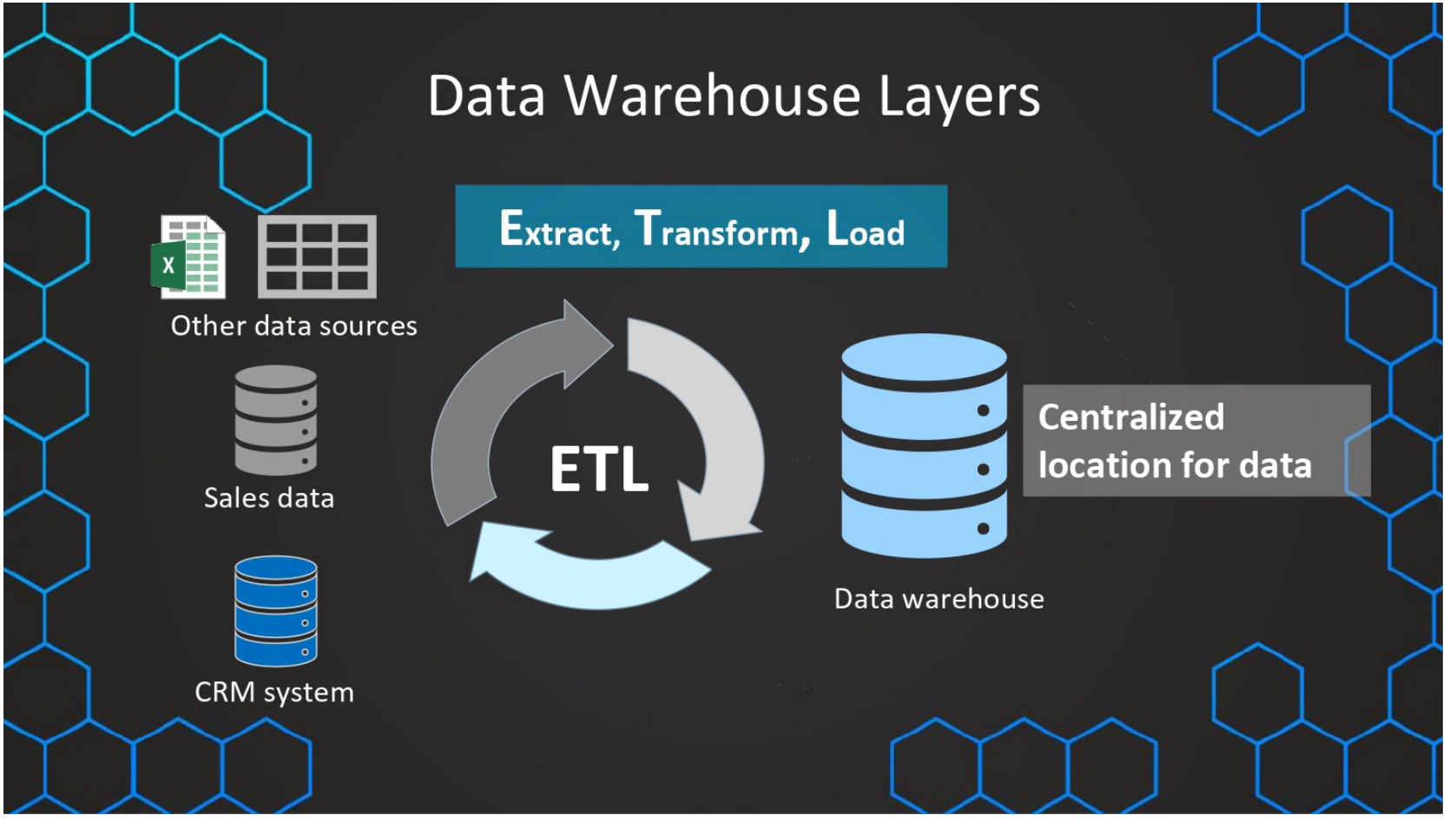


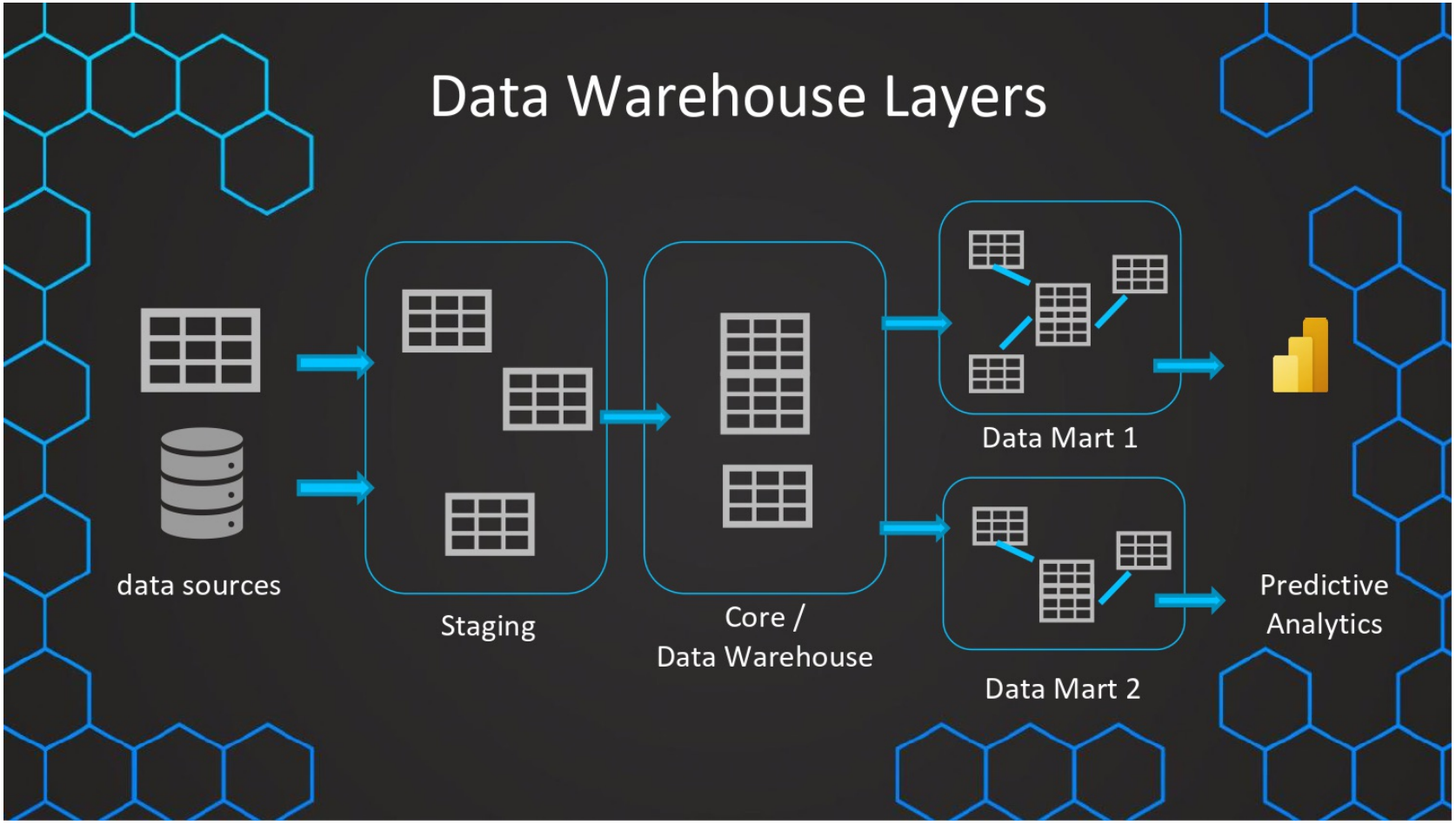
What is an ETL?

What is an ETL?

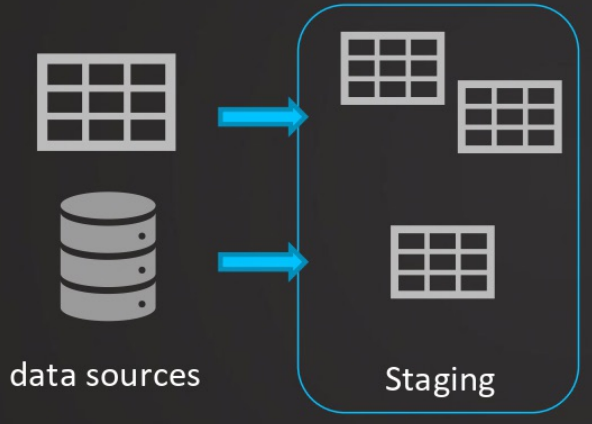
- ✓ How to design dimensional model
- ✓ How to bring data from source to DWH

= ETL process


Data Warehouse Layers
Extract, Transform, Load
Other data sources
Sales data
CRM system
ETL
Data warehouse
Centralized location for data


Data Warehouse Layers
data sources
Staging
Core /
Data Warehouse
Data Mart 1
Data Mart 2
Predictive
Analytics

Extracting



- ✓ Data is part of DWH
- ✓ Understanding data
- ✓ From here data is transformed
- ✓ Transient (most commonly)
- ✓ All data copied and then deleted

Extracting types

Initial Load

- ✓ First (real) run
- ✓ All data

Delta Load

- ✓ Subsequent runs
- ✓ Only additional data

Initial Load

- ✓ First initial extraction from source data
- ✓ After discussion with the business users + IT
  - What data is needed
  - When is a good time to load the data (Night? Weekends?)
  - Smaller extractions to test

Same structure

Delta Load

- ✓ Incremental periodic Extraction / Load
- ✓ Delta column for every table

Sales_Date	Name	Amount
2022-06-06	Sunglases TR-7	$25
2022-06-06	Chocolate bar 70% cacao	$3
2022-06-07	Oat meal biscuits	$4
2022-06-07	Chocolate bar 70% cacao	$3
2022-06-08	Oat meal biscuits	$4

- ✓ Transaction date, create_date, etc.

Transform

Main goals

1. **Consolidate** (from multiple systems)

Transaction_ID	Amount	Date
T1	$5030	10/1/2022
T2	$5053	11/1/2022
T3	$654	12/1/2022

Transaction_ID	Amount (in thousands)	Transaction_Date
T14	$5.345	10-1-2022
T15	$7.953	11-1-2022
T16	$9.654	12-1-2022

Making the data compatible & consistent!

Main goals

2. **Reshape** according to business requirements

Month	Januar-2022	February-2022	March-2022	Total
Amount	$5030	$6053	$2455	$13548

Month	Amount
Januar-2022	$5030
February-2022	$6053
March-2022	$2455
Total	$13548

Month	Amount
Januar-2022	$5030
February-2022	$6053
March-2022	$2455

Clean & reshape data

Kinds of transformations

Basic

- Deduplication
- Filtering (rows & columns)
- Cleaning & Mapping (Integration)
- Value Standardization (Integration)
- Key Generation

Advanced

- Joining
- Splitting
- Aggregating
- Deriving new values

Kinds of transformations

Basic

- **Deduplication**

Store 1

product_id	name	category
P521	Almonds 150g	Nuts
P252	Garlic	Fruits & Vegetables
P533	Banana	Fruits & Vegetables
P684	Chocolate	Sweets & Snacks
P755	Spicy Chips	Sweets & Snacks

Store 2

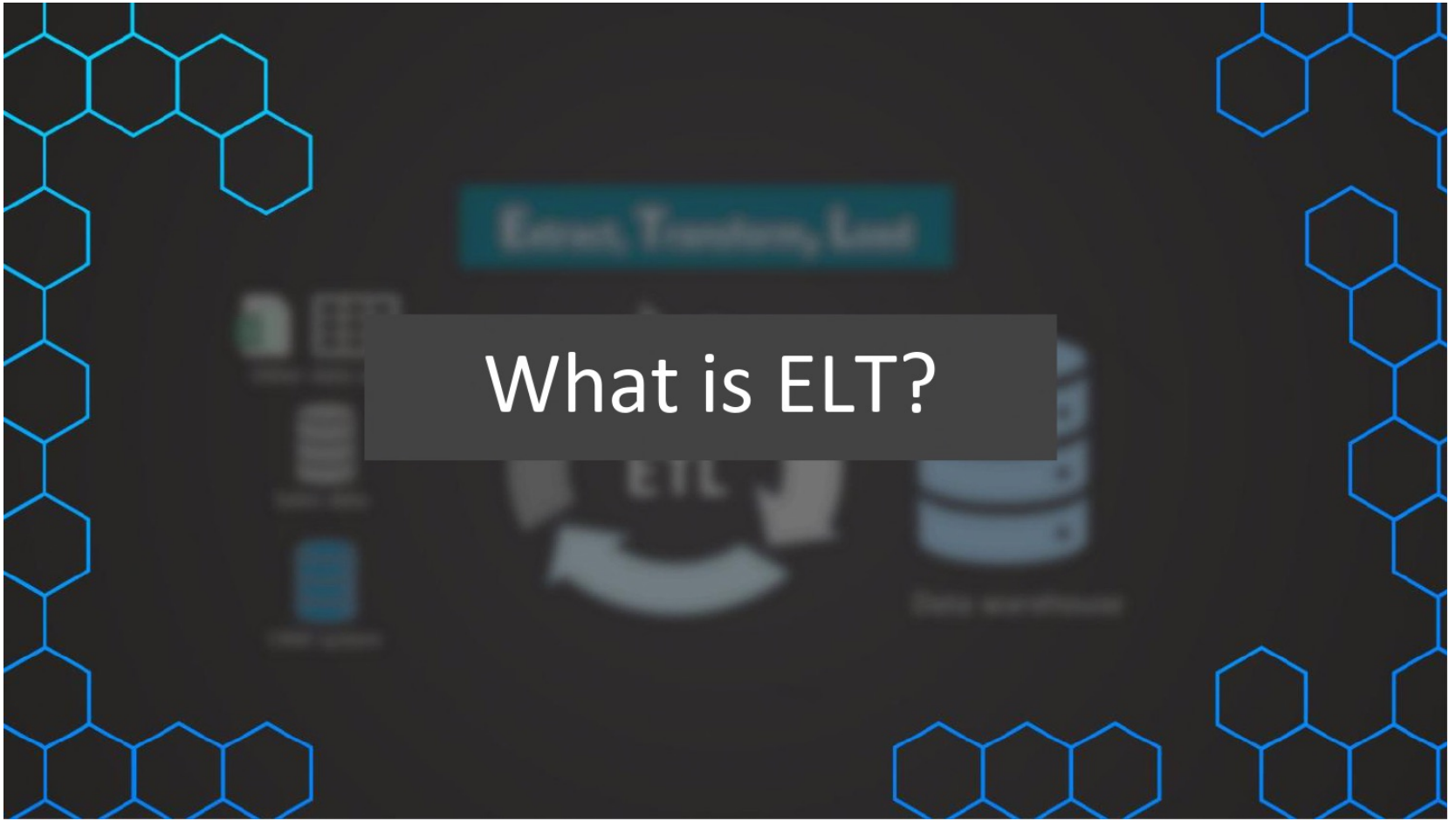
product_id	name	category
P521	Almonds 150g	Nuts
P672	Orange Juice	Drinks
P423	Green Apples	Fruits & Vegetables
P564	Chocolate Cookies	Sweets & Snacks
P755	Spicy Chips	Sweets & Snacks

ETL tools

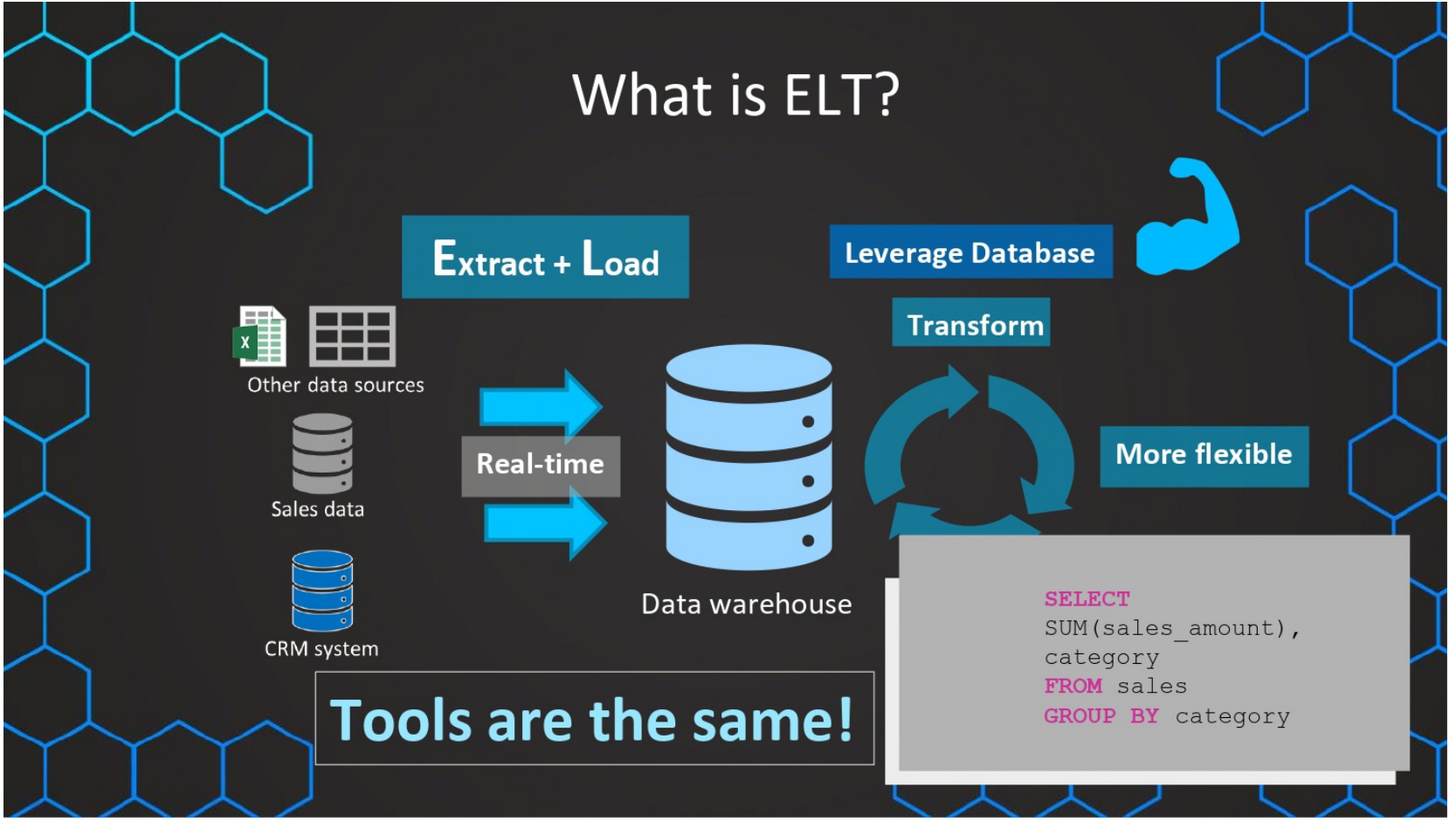
Enterprise	Open-source	Cloud-native	Custom
Commercial	Source code	Cloud technology	Own development
✓ Most mature	✓ Often free	Data already in cloud?	Customized
✓ Graphical interface	✓ Graphical interface	✓ Efficiency	Internal resources
✓ Architecural needs	Support?	Flexibility?	Maintainance?
✓ Support	Ease of use?		Training?

ETL tools

Enterprise	Open-source	Cloud-native
Alteryx	Talend Open Studio	Azure Data Factory
Informatica	Pentaho Data Integration	AWS Glue
Oracle Data Integrator	Hadoop	Google Cloud Data Flow
Microsoft SSIS		Stitch


What is ELT?

What is ELT?


Extract + Load
Leverage Database
Transform
More flexible
Other data sources
Sales data
CRM system
Real-time
Data warehouse
Tools are the same!
SELECT
SUM(sales_amount),
category
FROM sales
GROUP BY category

ETL vs. ELT

ETL

- ✓ More stable with defined transformations
- ✓ More generic use-cases
- ✓ Security

ELT

- ✓ Requires high performance DB
- ✓ More flexible
- ✓ Transformations can be changed quickly
- ✓ Real-time

ETL vs. ELT

ETL

- ✓ Reporting
- ✓ Generic use cases
- ✓ Easy to use

ELT

- ✓ Data Science, ML
- ✓ Real-time requirements
- ✓ Big data