**Data Storage**

**Different Types of Pipelines**

**ETL** 🡪 stands for *extract, transform,* and *load* (typically used for data warehouses as the data needs to be transformed before loaded)

* Extract data
* Transform the data ready for storage
* The data is placed into a database, such as SQL or NoSQL

**ELT** 🡪 stands for *extract, load,* and *transform (*typically used for data lakes as data is loaded before it is transformed)

* Data is extracted
* It is loaded into a database
* It is then transformed into a format that can be analysed easily

**Different Types of Processing**

**OLTP 🡪** Online Transaction Processing

**Database transactions** refer to any change to a database:

* Payments received
* Order taken
* Services delivered
* Products moving through inventory

This requires a high degree of **normalisation**

**HEAVY WRITE, LOW READ**

A diagram of a product

Description automatically generated**OLAP** 🡪 Online Analytic Processing

Adds a layer of **abstraction** and **aggregation**

* Semantic data model described meaning of data elements
* Data integrated from multiple sources and aggregated together across multiple dimensions
  + This is known as the OLAP Cube

OLAP is there to gain insight into the database, and optimised to analyse a very large set of data

**HEAVY READ, LOW WRITE**

Important to use the **ACID** acronym to maintain each transaction

* **ATOMICITY** 🡪 each transaction must be treated as a *singular unit*, if part of the transaction fails, it all fails
* **CONSISTENCY** 🡪 each transaction must result in a *valid* database state i.e., primary-foreign key relationships intact
* **ISOLATION** 🡪 concurrent read/write executions should produce the *same result* as sequential executions – transaction that occur together should be able to happen one-by-one to ensure speed
* **DURABILITY** 🡪 transactions should remain committed in the event of system failure (non-volatile memory) – each successful transaction is saved to prevent data loss

Different data warehouses 🡪 amazon redshift, Teradata, oracle, Cloudera, and panoply

**Data Warehouse** 🡪 a storage solution designed for data analysis

* Aggregates data from a variety of data stores
* Supports analysis on the composite data
* Only stores data that answer concrete business questions
* Slices of the warehouse stored as **Data Marts** 🡪 specific to a different business line, e.g., sales, finance sections

Creating Data Warehouses

**Inmon Architecture** 🡪 top-down approach, *normalised data* warehouse is created first, this data is then summarised and distributed out to Data Marts

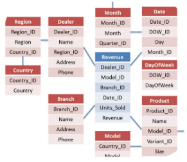
* The data is **non-volatile**; never overwritten or deleted
* The Data Marts are *slices* of the normalised data warehouse

**Kimball Architecture** 🡪 bottom-up approach, separate data marts are created first – create the storage solutions for single departments first

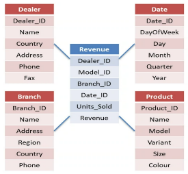
* Dimensional data warehouse combines data from Data Marts
* Quicker and less expressive to *set up,* but harder to maintain

Dimensional Modelling (Kimball Architecture)

|  |  |
| --- | --- |
| **Fact Tables 🡪 events presented as facts** | **Dimension Tables** |
| Presented in the centre of a schema – has several FRs that relate to dimension tables | Typically contains a PK and information related to the specific person, item, or object |
| Represent *events* | Represent *people/items/objects* |
| Contain data corresponding to specific  business practices e.g., sale revenue per month | Contain data corresponding to instances of *objects* e.g., product details, calendar information |
| Contains *measurement data* | Contains *dimensional data* |
| Often *numeric* | Often *text* |
| Easily aggregated | Used for *filtering/labelling* |
| **Grain** – level of *detail* |  |

There are several **schemas** you can used to set this up:

**Star Schema** 🡪 one or more Fact Tables each join with several Dimension Tables using **foreign keys**

* Simple, widely used, effective for *simple queries*, and widely supported by BI tools

**Snowflake Schema** 🡪 like star but the Dimension Tables are **normalised**

* Reduces diskspace, low *redundancy*, harder to perform simple queries
* Each data is only *stored once* but is referred to other tables (normalised)

A diagram of a product

Description automatically generatedA red background with white text

Description automatically generatedA diagram of a company that has been developed

Description automatically generated with medium confidenceA screenshot of a red background

Description automatically generated

**Data Lakes** 🡪 stores structured and unstructured data in its original form

* Processed when analysis is needed, not when stored 🡪 **Schema-on-read** (rather than schema-on-write)
  + Data is stored in its raw form and then schemas are created retroactively
* Newer, made possible with Big Data Technology
* Low-cost storage vs Warehouse 🡪 warehouses must pay an engineer to process the data ready for analysis, this process is a lot easier in data lakes thus cheaper
* Highly scalable
* Highly agile

**Data Lakehouses** 🡪 takes the best aspects of Data Warehouses and Data Lakes, offers some *structure* and *control* including ACID principles

* Flexible and scalable
* Often the architectures can be decoupled
* Governance can be difficult
* Very new technology

A screenshot of a data lake

Description automatically generatedOn the Lakehouse image, it shows the lake on the bottom with some form of schema on top of it

A white rectangular box with black text

Description automatically generatedData Lakehouse technology includes Cloudera, Delta Lake, and Snowflake