**Data Warehouse Concepts**

**Difference between OLAP and OLTP**

**Online Analytical Processing and Online Transactional Processing:** Are the data processing systems that help you store and analyze business data. You can collect and store data from multiple sources—such as websites, applications, smart meters, and internal systems. OLAP **combines and groups** the data so you can analyze it from different points of view. Conversely, OLTP **stores and updates transactional data** reliably and efficiently in high volumes. OLTP databases can be one among several data sources for an OLAP system.

|  |  |  |
| --- | --- | --- |
|  | **Online Analytical Processing** | **Online Transactional Processing** |
| Purpose | OLAP helps you analyze large volumes of data to support decision-making. | OLTP helps you manage and process real-time transactions. |
| Data source | OLAP uses historical and aggregated data from multiple sources. | OLTP uses real-time and transactional data from a single source. |
| Data structure | OLAP uses multidimensional (cubes) or relational databases. | OLTP uses relational databases. |
| Data model | OLAP uses star schema, snowflake schema, or other analytical models. | OLTP uses normalized or denormalized models. |
| Volume of data | OLAP has large storage requirements. Think terabytes (TB) and petabytes (PB). | OLTP has comparatively smaller storage requirements. Think gigabytes (GB). |
| Response time | OLAP has longer response times, typically in seconds or minutes. | OLTP has shorter response times, typically in milliseconds |
| Example applications | OLAP is good for analyzing trends, predicting customer behavior, and identifying profitability. | OLTP is good for processing payments, customer data management, and order processing. |

**How Normalizations differ from OLTP and OLAP?**

**Normalization:** It is a process of organizing data in relational databases by reducing the redundancies, aiming to remove any data anomalies, and improving the data integrity.

**1NF:** **Atomicity –** Remove duplicate columns from table (Only unique columns)

**2NF: No Partial dependency –** Place subset of data in different table and create a relation with primary key

**3NF: No Transitive dependency –** Removing all the columns that are not dependent on primary key

**4NF:** No attribute is dependent on combination on attributes which is not a candidate key

**Boyce NF: Every attribute is a candidate key.**

**Normalized** data structures are typically used in **OLTP** systems, **denormalized** or multidimensional data structures are more common in **OLAP** systems to improve query performance. These concepts are often complementary and serve different needs within an organization.

**Fact Table and Dimension Table Relationship:**

In a data warehouse, the fact table serves as the repository for key performance indicators (KPIs) or measures. It holds summary data regarding business processes or events. Dimension tables, on the other hand, provide descriptive information about the objects or entities associated with the measures stored in the fact table. The relationship between the fact table and dimension table is established through primary and foreign keys.

**Schema Types:**

* **Star Schema:** In a star schema, the fact table is at the center, surrounded by dimension tables. This design simplifies querying and improves performance due to denormalization.
* **Snowflake Schema:** Snowflake schema represents normalized data with micro granularity. Here, the fact table is surrounded by dimension tables, and these dimension tables may further be connected to additional dimension tables.

**Slowly Changing Dimensions (SCD):**

* **SCD Type 1:** In SCD Type 1, no history of data changes is maintained. It's suitable for scenarios where historical data isn't critical, such as with debit card or credit card details.
* **SCD Type 2:** SCD Type 2 maintains all historical data changes using techniques like versioning, flags, and date tracking. This ensures a comprehensive historical record by storing previous and current versions of data.
* **SCD Type 3:** SCD Type 3 maintains current and previous historical data, offering partial history storage compared to Type 2.

**Surrogate Keys and Data Loading:**

Surrogate keys are system-generated keys used to substitute natural primary keys in dimension tables. These keys provide a unique identifier for each record in the table.

Data loading into the data warehouse can be categorized into two phases:

* **Initial Load:** The initial load involves loading data into the warehouse for the first time, populating the tables with the initial set of data.
* **Batch or Incremental Load:** After the initial load, subsequent updates or additions to the data are performed incrementally using batch or incremental loading techniques. This involves updating records based on attributes such as source update date, current date, and end date.

**Factless Fact Table and Types of Facts:**

* **Factless Fact Table:** Factless fact tables capture many-to-many relationships between dimensions without containing any measurable numeric or textual facts. They are useful for tracking events or occurrences where no measures are involved, such as student registrations.
* **Types of Facts:**
  + **Additive:** Additive facts allow for the summation of measures across all dimensions. Examples include quantity sold or dollars sold.
  + **Semi-Additive:** Semi-additive facts allow for the summation of measures across some dimensions but not all. An example is inventory levels.
  + **Non-Additive:** Non-additive facts do not allow for the summation of measures across any dimensions.

**Types of Dimensions:**

* **Conformed Dimension:** Conformed dimensions are dimension tables that are created or built once and used across multiple instances or processes. An example is a date dimension.
* **Junk Dimension:** Junk dimensions are collections of random transaction codes, flags, or text attributes. They help in simplifying and organizing data storage and analysis.
* **Degenerative Dimension:** Degenerative dimensions are characterized by having a foreign key present in the fact table without an associated primary key. They represent a measurable value in the fact table without a corresponding dimension table.

**Data Catalog:**

data catalog is an organized inventory of data assets in the organization. It uses metadata to help organizations manage their data. It also helps data professionals collect, organize, access, and enrich metadata to support data discovery and governance.

**Data Masking/ Data Anonymization:**

Data masking and anonymization are techniques to protect sensitive or confidential data from unauthorized access or disclosure. They are often used when sharing data with external parties, such as customers, partners, or researchers, or when testing or developing applications that use real data.

**PII Data:**

Personally identifiable information (PII) is any information connected to a specific individual that can be used to uncover that individual's identity, such as their social security number, full name, email address or phone number.

**Data Democratization:**

In a business sense, data democratization is the practice of providing data access to everyone in an organization. This means empowering every team member to access and use data for informed decision-making without requiring technical expertise or IT intervention.

**Data Modeling:**

Data modeling is the process of analyzing and defining all the different data types of your business collects and produces, as well as the relationships between those bits of data.

**Dashboard:**

A dashboard is a way of displaying various types of visual data in one place. Usually, a dashboard is intended to convey different, but related information in an easy-to-digest form.

**Data Ecosystem:**

A data ecosystem is the complex environment of co-dependent networks and actors that contribute to data collection, transfer, and use. They can span across sectors – such as healthcare or finance, to inform one another's practices.

**Data Enrichment:**

Data enrichment is the process of incorporating new updates and information into an organization's existing database to improve accuracy and add missing information. Building on your existing data allows for better business decisions and better customer relationships.

**Data Exchange:**

Data exchange refers to the process of transferring data and large files between different organizations and systems, ensuring that the information and its intended meaning remain intact.

**Data Extraction:**

Data extraction is the process of collecting or retrieving disparate types of data from a variety of sources, many of which may be poorly organized or completely unstructured.

**Data governance:**

Data governance is everything you do to ensure data is secure, private, accurate, available, and usable. It includes the actions people must take, the processes they must follow, and the technology that supports them throughout the data life cycle.

**Data Ingestion:**

Data ingestion is the process of importing large, assorted data files from multiple sources into a single, cloud-based storage medium—a data warehouse, data mart or database—where it can be accessed and analyzed.

**Data Joins:**

A data join is when two data sets are combined in a side-by-side manner, therefore at least one column in each data set must be the same.

**Data lineage:**

Data lineage is the process of understanding, recording, and visualizing data as it flows from data sources to consumption. This includes all transformations the data underwent along the way—how the data was transformed, what changed, and why. Data lineage process.

**Data Mesh:**

A data mesh is a decentralized data architecture that organizes data by a specific business domain—for example, marketing, sales, customer service, and more—providing more ownership to the producers of a given dataset.

**Data portability:**

Data portability allows individuals to obtain and reuse their personal data for their own purposes across different services. It allows them to move, copy or transfer personal data easily from one IT environment to another in a safe and secure way, without affecting its usability.

**Data replication:**

Data replication is the process of copying data from one location to another. The technology helps an organization maintain up-to-date copies of its data in the event of a disaster. Replication can take place over a storage area network, local area network or local wide area network as well as to the cloud.

**Data privacy:**

Data privacy generally means the ability of a person to determine for themselves when, how, and to what extent personal information about them is shared with or communicated to others. This personal information can be one's name, location, contact information, or online or real-world behavior.

**Data consistency:**

Data consistency refers to the state of data in which all copies or instances are the same across all systems and databases. Consistency helps ensure that data is accurate, up-to-date, and coherent across different database systems, applications, and platforms.

**Data Quality:**

Data quality measures how well a dataset meets criteria for accuracy, completeness, validity, consistency, uniqueness, timeliness, and fitness for purpose, and it is critical to all data governance initiatives within an organization.

**Data Silo:**

A data silo is a collection of data held by one group that is not easily or fully accessible by other groups in the same organization. Finance, administration, HR, marketing teams, and other departments need different information to do their work.

**Data Validation:**

Data validation is the process of verifying and validating data that is collected before it is used. Any type of data handling task, whether it is gathering data, analyzing it, or structuring it for presentation, must include data validation to ensure accurate results.

**Data Wrangling:**

Data wrangling is the process of converting raw data into a usable form. It may also be called data munging or data remediation. You'll typically go through the data wrangling process prior to conducting any data analysis to ensure your data is reliable and complete.

**Data stewardship:**

Data stewardship is the collection of practices that ensure an organization's data is accessible, usable, safe, and trusted.

**Database schema:**

A database schema refers to the logical and visual configuration of the entire relational database. The database objects are often grouped and displayed as tables, functions, and relations. A schema describes the organization and storage of data in a database and defines the relationship between various tables.

**EDI** standards:

EDI standards provide a common language for data exchange and establish a framework for data formatting, message structure, and content.

**Observability:**

Observability is the extent you can understand the internal state or condition of a complex system based only on knowledge of its external outputs.

**Streaming Data:**

Streaming data is data that is emitted at high volume in a continuous, incremental manner with the goal of low-latency processing.

**Data Lake:**

A data lake is a centralized repository designed to store, process, and secure large amounts of structured, semi structured, and unstructured data. It can store data in its native format and process any variety of it, ignoring size limits.

**Lakehouse Architecture:**

A data lakehouse is a modern data architecture that creates a single platform by combining the key benefits of data lakes (large repositories of raw data in its original form) and data warehouses (organized sets of structured data).

**Differentiate between Monolith vs Micro-service Architecture:**

| **Aspect** | **Monolithic Architecture** | **Microservices Architecture** |
| --- | --- | --- |
| Structure | Single, large application | Distributed system composed of small, independent services |
| Deployment | Entire application deployed at once | Services deployed independently |
| Scaling | Horizontal scaling of entire application | Vertical scaling of individual services |
| Development | Generally simpler development process | Requires more intricate coordination and management |
| Technology Stack | Single technology stack | Each service can use a different technology stack |
| Flexibility | Limited flexibility in choosing technologies | Offers flexibility in technology choices |
| Maintenance | Centralized maintenance | Decentralized maintenance |
| Resilience | A single point of failure | Fault isolation ensures resilience to failure |
| Communication | In-process communication | Inter-service communication via APIs |
| Dependency Management | Tight coupling between components | Loose coupling between services |
| Scalability | Limited scalability due to the entire application being scaled together | Scalable at the individual service level |
| Complexity | Easier to understand and manage | More complex due to distributed nature |

**AWS services:**

1. **S3 and S3 Glacier:**

Amazon S3 is a durable, secure, simple, and fast storage service, while Amazon S3 Glacier is used for archiving solutions. Use S3 if you need low latency or frequent access to your data. Use S3 Glacier for low storage cost, and you do not require millisecond access to your data.

1. **Redshift, Amazon RDS and DynamoDB:**

**Amazon Redshift is** a fast, fully managed cloud data warehouse that makes it simple and cost-effective to analyze all your data.

**Amazon Relational Database Service (Amazon RDS)** is a collection of managed services that makes it simple to set up, operate, and scale databases in the cloud.

**Amazon DynamoDB** is a serverless, NoSQL, fully managed database service with single-digit millisecond response times at any scale, enabling you to develop and run modern applications while only paying for what you use.

1. **EC2 and LightSail:**

**Amazon Elastic Compute Cloud (Amazon EC2)** is a web service that provides secure, resizable compute capacity in the cloud. It is designed to make web-scale cloud computing easier for developers.

A **Lightsail** instance is a virtual private server (VPS) that lives in the AWS Cloud. Use your Lightsail instances to store your data, run your code, and build web-based applications or websites.

1. **Lambda:**

**AWS Lambda** is a serverless compute service that runs your code in response to events and automatically manages the underlying compute resources for you. These events may include changes in state or an update, such as a user placing an item in a shopping cart on an ecommerce website.

1. **Amazon SNS:**

Amazon Simple Notification Service (Amazon SNS) is a web service that makes it easy to set up, operate, and send notifications from the cloud.

1. **Dynamo DB:**

Amazon DynamoDB is a serverless, NoSQL database service that enables you to develop modern applications at any scale. As a serverless database, you only pay for what you use and DynamoDB scales to zero, has no cold starts, no version upgrades, no maintenance windows, no patching, and no downtime maintenance.

1. **CloudWatch and CloudTrail:**

**CloudWatch** enables you to monitor your complete stack (applications, infrastructure, network, and services) and use alarms, logs, and events data to take automated actions and reduce mean time to resolution (MTTR). This frees up important resources and allows you to focus on building applications and business value.

AWS **CloudTrail** is an AWS service that helps you enable operational and risk auditing, governance, and compliance of your AWS account. Actions taken by a user, role, or an AWS service are recorded as events in CloudTrail.

1. **Sage maker:**

Amazon SageMaker helps data scientists and developers to prepare, build, train, and deploy high-quality machine learning (ML) models quickly by bringing together a broad set of capabilities purpose-built for ML. SageMaker supports the leading ML frameworks, toolkits, and programming languages.

With SageMaker, you pay only for what you use. You have two choices for payment: an On-Demand Pricing that offers no minimum fees and no upfront commitments, and the SageMaker Savings Plans that offer a flexible, usage-based pricing model in exchange for a commitment to a consistent amount of usage.

1. **Step Functions:**

Step Functions manages your application's components and logic, so you can write less code and focus on building and updating your application quickly.

**Data Mesh:**

A set of principles for designing modern data architecture.

Breaking down large, centralized data systems into smaller, self-contained units managed by individual teams. Each team focuses on their specific data domain, making it easier to manage, analyze, and innovate with data while promoting collaboration across the organization. So, Data Mesh is like decentralizing and distributing the management of data for better efficiency and agility.

A well implemented data mesh lets you scale a data architecture in two ways. Firstly, the technology scale way, like adding enough computers to make it run fast enough (fundamentally scalable) and secondly, can it be scalable across the organization as a business evolves.

Conceptually about data mesh, it has many genetic influences – Data Marts, Domain Driven Design, Micro Services, Data Streaming

**Principle 1:** Data ownership by domain – **Decentralized**: Instead of one big data team handling everything, let different teams manage their own data. This makes things more organized and easier to handle.

**Domain-oriented Data Ownership**: Letting teams own and understand the data related to their specific area or "domain." This ensures that data is managed by people who know it best.

**Principle 2:** **Data as a product** – Treating data as a valuable product that needs to be well-designed, maintained, and improved over time. This mindset ensures that data is high-quality and useful for everyone.

**Principle 3:** Data available everywhere, serve. **Self-serve Data Platform**: Instead of waiting for someone else to serve them, teams can get the data they need when they need it. This speeds up work and empowers teams to be more independent.

**Principle 4:** Data governed wherever it is. **Federated Computational Governance**: Data Mesh suggests setting up rules and standards for how data is managed and used across different teams. This ensures consistency and fairness, like having the same rules for everyone in a game.

A diagram of data mesh architecture

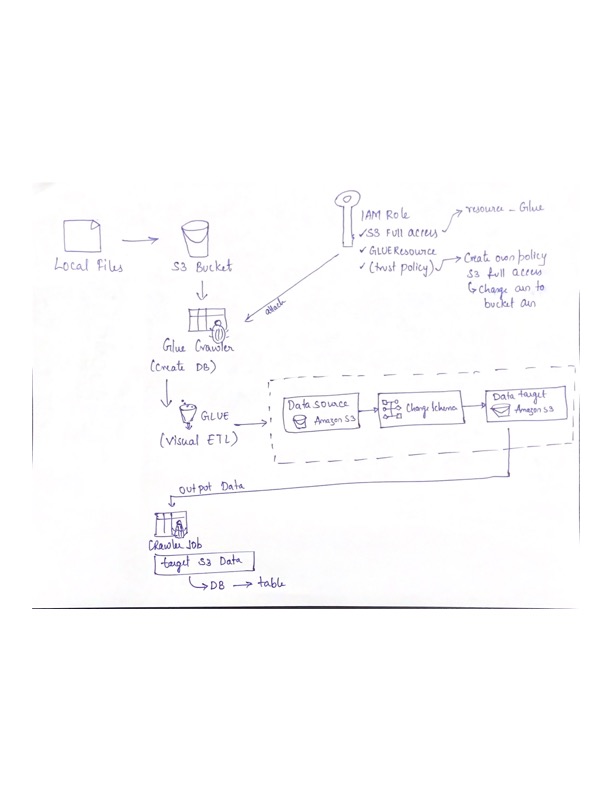
Description automatically generated

1. **Data Product**: A data product is like a package containing all the information needed for analyzing data in a specific area, such as sales or customer behavior. It's managed by a team that understands that aspect of the business well. Think of it as a toolkit tailored for a specific task.
   * *Note*: Data products are like toolkits for analyzing specific types of data. They're created and managed by teams who know that data best.
2. **Data Contract**: A data contract is like an agreement that defines how data should be structured and used between different teams. It includes details like who owns the data, how it can be used, and what format it's in. It ensures everyone is on the same page when sharing and using data.
   * *Note*: Data contracts are like agreements that make sure everyone understands how to use and share data correctly.
3. **Federated Governance**: This is like a group that sets rules and standards for how data is managed across the organization. They make sure everyone follows the same guidelines for things like data quality, security, and privacy.
   * *Note*: Federated governance is like a team that makes sure everyone plays by the same rules when it comes to handling data.
4. **Transformations**: This is the process of preparing data for analysis by cleaning it up and organizing it. It's like getting your ingredients ready before cooking a meal.
   * *Note*: Transformations are like preparing ingredients before cooking. It makes data easier to work with and understand.
5. **Ingesting**: This is about bringing data into the system so it can be analyzed. It's like gathering all the ingredients you need for a recipe before you start cooking.
   * *Note*: Ingesting is like gathering all the ingredients you need before you start cooking. It gets data ready for analysis.
6. **Clean Data**: Clean data is data that's been tidied up and is ready to be analyzed. It's like having a clean kitchen before you start cooking.
   * *Note*: Clean data is like having a clean kitchen before you start cooking. It makes working with data much easier.
7. **Analytics**: Analytics is the process of examining data to find patterns or insights. It's like looking through a recipe book to find the best dish to cook.
   * *Note*: Analytics is like finding the best recipe to cook. It helps you make sense of the data you have.
8. **Data Platform**: This is like the kitchen where all the cooking and preparation happens. It provides the tools and space needed to work with data effectively.
   * *Note*: The data platform is like the kitchen where all the cooking happens. It's where you work with data to analyze and understand it.
9. **Enabling Team**: This team helps other teams understand how to use the data platform and create data products. They're like the chefs who teach others how to cook.
   * *Note*: The enabling team is like the chefs who teach others how to cook. They help teams understand how to work with data effectively.
10. **Mesh**: The mesh is like a network of connections between different teams and their data products. It allows teams to share and use data from other areas of the organization.
    * *Note*: The mesh is like a network that connects different teams and their data products. It allows teams to share and use data from across the organization.

**Data Ingestion Lab**

1. Create a System design flow by reverse engineering the data ingestion process.
2. The submission should include the following:
3. A system design flow-diagram (S3, IAM, GLUE).
4. Explain each step we did in the service and respective terminology as well as concepts we covered.
5. ﻿﻿﻿Understand the JSON to explain the code.

**Design Flow**



**My Understanding:**

1. First, we create a bucket and load the file from our local
2. Create an IAM role – Using AWS Service – we are creating this role for AWS glue service (so we select glue) – then select the permission policies (S3 full access and AWSGlueService) along with this create a policy for S3 full access changing and giving the resources with the bucket name (attach this policy to the role you created)
3. Create a Glue crawler job to infer the schema and create a table for querying in the database using Athena. (create database and attach the IAM role)
4. Once this crawler job is run, we can query the data in Athena.

**For Reverse Engineering:**

1. In Data Catalog – Under Databases – We have Data Catalog tables (from the above crawler job)
2. Select source a data Catalog table.
3. Schema change – We can change the schema (tables name and data types based on requirements)
4. Select target as S3.

A close-up of a sign

Description automatically generated

**Understand the JSON:**

1. **Version**: Specifies the version of the policy language. In this case, it's "2012-10-17", which is the most recent version as of the creation of this policy.
2. **Statement**: Contains one or more individual statements, each of which describes a set of permissions.

{

"Version": "2012-10-17",

"Statement": [

{

"Action": [

* + **Action**: Lists the actions allowed or denied by the policy. These actions include various operations related to Amazon S3, such as getting, putting, listing, and deleting objects, as well as managing bucket settings and configurations.
  + **Resource**: Specifies the AWS resources to which the actions apply. This policy allows actions on specific S3 buckets (arn:aws:s3:::query-result-65d41890, arn:aws:s3:::glue-bucket-65d41890, arn:aws:s3:::query-result-\*/\*, arn:aws:s3:::glue-bucket-\*/\*) and objects within a specific prefix (arn:aws:s3:::globalelectriccar/us/washington/,arn:aws:s3:::globalelectriccar/us/washington/\*).
  + **Effect**: Specifies whether the statement allows or denies the actions specified. In this policy, the effect is "Allow", meaning the actions listed in the statement are permitted.

"Resource": [

"arn:aws:s3:::global-electric-car/us/washington/",

"arn:aws:s3:::global-electric-car/us/washington/\*"

],

"Effect": "Allow"

**Data Transformation Lab:**

1. Create a new ETL flow using what you have learnt in the earlier class by following the provided steps.
2. Copy the data files from Here and upload these files to separate S3 folders in a same S3 Bucket.
3. Read the data from S3 bucket using Extract Node (2 nodes needed to read files from 2 separate folders).
4. Use the Join transform and combine the files from both the nodes using the inner join condition.
5. Use the conditional router transform to route the files into 2
6. output nodes, one node having data less than year 2000 and other data greater than or equal to 2000.
7. Finally add 2 S3 output nodes to create 2 S3 folders and list these two data files in catalog database tables.

**Steps:**

1. Two files (States\_finances\_all.csv and enroll\_states\_sumarry.csv) 🡪 Create a bucket with separate folder for each file. Once the files are uploaded into S3
2. Start preparing glue crawler jobs. Two crawler jobs are created for inferring the schema of two files.
3. Once the crawler jobs are run, we get data catalog tables.
4. Now Create a ETL Glue Job, start by selecting the two source files from the bucket and then start applying transformations (Remember we are selecting data Catalog tables as source here)
5. First the first transformation, we are applying join condition based on primary key. (applying inner join)

Select \* from file1

inner join file2

on file1.primary\_key = file2.primary\_key

1. Once the join condition is applied, we are applying router transformation based on year. All the records with year < 2000 are separated from all the records with year >= 2000.
2. Now, each group from the router condition is kept into separate data target. This data target will be a separate folder which has been create in S3.
3. We can check the target files in output folders of S3 bucket.
4. We can query the data catalog table using Athena.

A screenshot of a computer

Description automatically generated

Additional Transformations tried:

1. Tried to remove the null values in the file. Applied “Remove Null Rows” / “Drop Null Fields” transformations. While it didn’t remove the null values while querying in Athena. Figured out there are multiple ways like using “SQL Query” or “Evaluate Data Quality” transformations, through which we may achieve this.

