**AMAZON REDSHIFT**

Presented by: Chi Phuong Diep

***Amazon Redshift***

Our team views Amazon Redshift as a cloud-based data warehouse that allows businesses to store and analyze large volumes of data such as customer purchases, website traffic, and order monitoring swiftly and effectively. It is like having a super-powerful, expandable filing system in the cloud that supports SQL querying and instantly answers complex questions about the companies’ data which makes it user-friendly and efficient. The best thing about this platform is that it is managed by Amazon, which means the companies do not need to worry about technical aspects.

***Project Objectives***

Our goal is to use Amazon Redshift to create a transactional database structure that handles monitor products and customer reviews. We will be demonstrating how the data is transacted using CRUD operations.

***Dataset 1: Amazon Product (Monitors)***

The first dataset we have contains information about monitor products sold by Amazon which includes the following details:

* *Title Description:* The name or description of the monitor model.
* *Brand:* The brand or manufacturer of the monitor.
* *Screen Size:* The size of the monitor screen, typically measured in inches.
* *Resolution:* The display resolution of the monitor, such as FHD 1080p or 4K UHD 2160p.
* *Aspect Ratio:* The aspect ratio of the monitor screen, such as 16:9 or 21:9.
* *Rating:* The customer rating of the monitor, on a scale typically ranging from 1 to 5.
* *Price:* The price of the monitor is in USD.

**\***Source:

https://www.kaggle.com/datasets/durjoychandrapaul/amazon-products-sales-monitor-dataset

***Dataset 2: Amazon Reviews***

The second dataset we have contains information about the reviews of the Amazon Shopping App, which includes the following details. We decided to remove one column from the dataset as it was duplicated (reviewCreatedVersion):

* *Review ID:* The unique ID of each review.
* *Username:* The name of the reviewer.
* *Review:* The content of the review.
* *Rating:* The user rating, on a scale typically ranging from 1 to 5.
* *Like Count:* The count of likes on each review.
* *Date of Review:* The date the review is posted on the app.
* *App Version:* The version of the app.

**\***Source:

<https://www.kaggle.com/datasets/ashishkumarak/amazon-shopping-reviews-daily-updated>

***Preprocessing Data:***

To ensure efficient loading into Amazon Redshift, we performed the following key preprocessing steps:

* *Cleaning Special Characters:* We removed non-standard characters (e.g., Ø, ·) and double quotations (") using a VBA script. This resolved parsing issues and ensured compatibility with loading data into Redshift.
* *Dataset Reduction:* The original Amazon Reviews dataset had over 60,000 reviews, which was reduced to 928 through random sampling. This aligns the review count closely with the 947 monitors in the AmazonMonitor table, allowing some monitors to have no reviews, mimicking a realistic scenario.
* *Text Truncation:* Reviews longer than 256 characters were truncated to fit the VARCHAR(256) limit in Redshift. This avoided data loading errors and standardized the review text length.
* *Data Consistency:* We ensured all monitorID values were sequential from 1 to 947 and matched between the AmazonMonitor and AmazonReview tables. Dates were standardized to YYYY-MM-DD format.

***Product Overview & Transactional Design***

The product is a **relational database** implemented on Amazon Redshift, designed to store data for monitors (AmazonMonitor table) and user reviews (AmazonReview table). The tables are linked through a foreign key (monitorID), forming a one-to-many relationship where one monitor can have multiple reviews. This enforces data integrity and supports efficient querying.

The design is transactional because it adheres to ACID properties:

* *Atomicity:* Each operation (e.g., inserting a review) is a complete transaction, preventing partial updates. When a user submits a review, all associated operations (creating the review record, updating the monitor's average rating, and incrementing the review count) are treated as a single transaction.
* *Consistency:* Data constraints (e.g., primary keys, foreign keys) maintain integrity across tables. If someone tries to post a review for a non-existent monitorID, the database rejects it, maintaining referential integrity.
* *Isolation:* When multiple users submit reviews simultaneously for the same monitor, Redshift's transaction management ensures each review's processing occurs independently to prevent data conflicts.
* *Durability:* Once a review is committed to the database, Redshift's replication ensures the data survives even if a cluster node fails to permanently preserve both monitor and review data.

***Data Structures***

The database consists of two main tables: AmazonMonitor and AmazonReview, designed to store product and review data.

AmazonMonitor Table:

* *Primary Key:* monitorID (INT, NOT NULL, AUTOMATIC INCREMENT)
* Columns:
  + *monitorID:* Unique identifier for each monitor.
  + *description:* Text description of the monitor (VARCHAR 256).
  + *brand:* Brand name of the monitor (VARCHAR 256).
  + *screensize:* Size of the monitor in inches (REAL).
  + *resolution:* Screen resolution (VARCHAR 256).
  + *aspect\_ratio:* Aspect ratio of the monitor (VARCHAR 256).
  + *rating:* Average user rating (REAL).
  + *price:* Price of the monitor (REAL).

AmazonReview Table:

* *Primary Key:* reviewID (VARCHAR 256, NOT NULL)
* *Foreign Key:* monitorID references AmazonMonitor(monitorID)
* Columns:
  + *reviewID:* Unique identifier for each review.
  + *username:* User who submitted the review (VARCHAR 256).
  + *review\_text:* Content of the review, truncated to 256 characters (VARCHAR 256).
  + *rating:* User rating (INT).
  + *thumbs\_up\_count:* Number of likes on the review (INT).
  + *date\_of\_review:* Date of the review submission (DATE).
  + *appversion:* Version of the app used for submission (VARCHAR 256).
  + *monitorID:* References monitorID in AmazonMonitor, linking the review to a specific monitor.

***ERD Design and Relationship***

A diagram of a software

Description automatically generated

Based on the design, the relationship between the AmazonMonitor and AmazonReview entities is a one-to-many relationship. One monitor can have multiple reviews, but each review is associated with a single monitor.

* ***Cardinality:*** The cardinality of the relationship is one-to-many, meaning one monitor can have zero, one, or many reviews, but each review is associated with only one monitor.
* ***Modality:*** The modality of the relationship is optional-mandatory, meaning that a monitor can exist without any reviews (optional), but each review must be associated with a monitor (mandatory).

Key Explanation:

The AmazonMonitor table serves as the parent entity, uniquely identified by monitorID, and maintains descriptive attributes of each monitor. The AmazonReview table stores individual reviews linked via the foreign key monitorID, ensuring that each review references a valid monitor. This one-to-many relationship is enforced by the primary-foreign key constraint, maintaining referential integrity. The optional-mandatory modality allows monitors to exist without reviews, supporting newly listed products. Meanwhile, each review must always be tied to a specific monitor, upholding the integrity of user feedback data. Additional business logic, such as cascade deletions or aggregate recalculations of monitor ratings, would be considered based on application requirements.

***CRUD Operation***

We write SQL scripts to deliver Create, Read, Update, and Delete (CRUD) operations that would be essential for managing the monitor and review data. We also have provided the transactional context for each operation.

-- **AmazonMonitor Table**

-- **CREATE**

CREATE TABLE AmazonMonitor (

monitorID INTEGER IDENTITY(1, 1) PRIMARY KEY, -- Primary Key

description VARCHAR(255)NOT NULL, -- Description of the monitor

brand VARCHAR(100) NOT NULL, -- Brand name

screen\_size FLOAT NOT NULL, -- Monitor size in inches

resolution VARCHAR(50) NOT NULL, -- Resolution (e.g., 1920x1080)

aspect\_ratio VARCHAR(10) NOT NULL, -- Aspect ratio (e.g., 16:9)

rating FLOAT NOT NULL, -- Average user rating

price FLOAT NOT NULL -- Price of the monitor

);

-- **INSERT**

-- Insert a new monitor into AmazonMonitor

INSERT INTO AmazonMonitor (monitorID, description, brand, screen\_size, resolution, aspect\_ratio, rating, price)

VALUES (1, '27-inch Full HD monitor with IPS display', 'Dell', 27.0, '1920x1080', '16:9', 4.5, 199.99);

-- Insert another monitor into AmazonMonitor

INSERT INTO AmazonMonitor (monitorID, description, brand, screen\_size, resolution, aspect\_ratio, rating, price)

VALUES (2, '32-inch 4K Ultra HD monitor', 'Samsung', 32.0, '3840x2160', '16:9', 4.7, 399.99);

-- **UPDATE**

-- Update the price of a monitor

UPDATE AmazonMonitor

SET price = 179.99

WHERE monitorID = 1;

-- Update the rating of a monitor

UPDATE AmazonMonitor

SET rating = 4.8

WHERE monitorID = 2;

-- **SELECT**

-- Select all columns for all monitors

SELECT \* FROM AmazonMonitor;

-- Select only the brand, screen size, and price for monitors with a rating of 4.5 or higher

SELECT brand, screen\_size, price

FROM AmazonMonitor

WHERE rating >= 4.5;

-- Select all monitors with a screen size greater than 30 inches

SELECT monitorID, description, screen\_size

FROM AmazonMonitor

WHERE screen\_size > 30;

-- Find monitors with a specific resolution (e.g., '1920x1080')

SELECT monitorID, brand, screen\_size, price

FROM AmazonMonitor

WHERE resolution = '1920x1080';

-- Select monitors in a specific price range (e.g., between 100 and 300)

SELECT monitorID, description, brand, price

FROM AmazonMonitor

WHERE price BETWEEN 100 AND 300;

-- **DELETE**

-- Delete a monitor based on monitorID

DELETE FROM AmazonMonitor

WHERE monitorID = 2;

-- Delete monitors with a rating below 4.0

DELETE FROM AmazonMonitor

WHERE rating < 4.0;

-- **AmazonReview Table**

-- **CREATE**

CREATE TABLE AmazonReview (

reviewID INTEGER PRIMARY KEY NOT NULL, -- Primary Key

monitorID INTEGER, -- Foreign Key referencing AmazonMonitor

username VARCHAR(100) NOT NULL, -- User who submitted the review

review\_text TEXT NOT NULL, -- Content of the review

rating INTEGER CHECK (rating BETWEEN 1 AND 5), -- Rating given by the user (1-5)

thumbs\_up\_count INTEGER, -- Upvotes on the review

date\_of\_review DATE, -- Date of the submitted review

app\_version VARCHAR(20), -- App version of the review

FOREIGN KEY (monitorID) REFERENCES AmazonMonitor(monitorID) -- Foreign Key relationship

);

-- **INSERT**

-- Insert a new review for a monitor

INSERT INTO AmazonReview (reviewID, monitorID, username, review\_text, rating, thumbs\_up\_count, date\_of\_review, app\_version)

VALUES (101, 1, 'user123', 'Great monitor for work and gaming!', 5, 10, '2024-01-15', '1.2.3');

-- Insert another review for a different monitor

INSERT INTO AmazonReview (reviewID, monitorID, username, review\_text, rating, thumbs\_up\_count, date\_of\_review, app\_version)

VALUES (102, 2, 'tech\_guru', 'Excellent 4K display with vibrant colors.', 5, 25, '2024-01-20', '1.3.0');

-- **UPDATE**

-- Update the thumbs\_up\_count for a review

UPDATE AmazonReview

SET thumbs\_up\_count = thumbs\_up\_count + 1

WHERE reviewID = 101;

-- Change the rating for a specific review

UPDATE AmazonReview

SET rating = 4

WHERE reviewID = 102;

-- **SELECT**

-- Select all reviews for a specific monitor

SELECT \* FROM AmazonReview

WHERE monitorID = 1;

-- Select username and review text for all reviews with a rating of 5

SELECT username, review\_text

FROM AmazonReview

WHERE rating = 5;

-- Select all reviews with more than 20 thumbs up

SELECT reviewID, monitorID, username, thumbs\_up\_count

FROM AmazonReview

WHERE thumbs\_up\_count > 20;

-- Find all reviews submitted after January 1, 2024

SELECT reviewID, monitorID, date\_of\_review

FROM AmazonReview

WHERE date\_of\_review > '2024-01-01';

-- Select all reviews for monitors with a specific resolution (joining with AmazonMonitor)

SELECT r.reviewID, r.username, r.review\_text, r.rating

FROM AmazonReview r

JOIN AmazonMonitor m ON r.monitorID = m.monitorID

WHERE m.resolution = '1920x1080';

-- **DELETE**

-- Delete a review by reviewID

DELETE FROM AmazonReview

WHERE reviewID = 102;

-- Delete reviews with a rating of 1

DELETE FROM AmazonReview

WHERE rating = 1;