Data Management and Database Design

P2. Database Specification: Purpose, Business Problem Addressed, and Business Rules

Topic:

Cloud Computing Data Model

(AWS - Amazon Web Services)

Under:

Prof. Simon Wuping Wang

Project Team Members:

Swathi Sharma

Shaurya Agrawal

Sanket Bhosale

Pratiksha Patole

Aman Shah

Database Purpose:

The purpose of the database is to maintain the data generated when a user utilizes the services and resources provided by the Cloud Computing Platform (Amazon Web Services) such as Instances (EC2), Storage (S3), and database (RDS) and charge the user according to the utilization of each resource. Thus, providing the user with pay-per-usage ability. Provide information about usage statistics of EC2, S3 and RDS in every region.

Business Problems Addressed:

- Amazon Web Services (AWS) provides cloud computing platform to different organizations as a pay-as-you-go service.
- The service is free to join, but the user must pay for the utilization of services.
- This database is designed for such cloud computing platforms that permit clients to lease virtual machines to execute code for computer applications and calculations.
- It effectively maintains the cloud assets of the organization like Machine Images, Instances, Storage, RDS and Transactions.
- It generates a detailed bill every month for the user based on the utilization of AWS resources.
- Three resources (of many) offered by AWS are taken into consideration for the design of this database. Users can use EC2, S3 and RDS as stand-alone features.
- This data base facilitates data collection, processing, billing, transactions, and reporting of different business cycles.

Business Rules:

- Each UserInfo has card information to associate payments
- Each user may own zero or more Elastic computer cloud (EC2) instances
- Each EC2 instance has one AMI
- Each EC2 instance is associated with one region
- EC2 instance has a root storage associated with it via tiers
- Each EC2 instance may have additional storage associated
- Every EC2 activity is tracked as per pay-by-hour usage
- Each user may own zero or more Simple Storage Service (S3) buckets
- Each bucket has one or more objects
- Each bucket is associated with a particular region
- Each object is associated with one storage class
- Every S3 bucket/object activity is tracked as per the size of the objects and total storage time
- Each user may own zero or more Relational Database Service (RDS)
- Each RDS(Db) has only one storage.
- Each RDS(Db) has one Engine
- Each RDS(Db) belongs to one DB class
- Each RDS(Db) belongs to one region
- Every RDS(Db) activity is tracked as per pay-by-hour usage
- All the resources have their own activity tracker and sales to track total usage of all resource

Design Requirements (Credit to Prof Wang)

- Use Crow's foot notation
- Specify the Primary key fields in each table by specify "PK" besides the fields.
- Draw a line between the fields of each table to show relationships between each table. This line should be pointed directly to the fields in each table that are used to form the relationship.
- Specify which table is on the one side of the relationship by placing a one next to the field where the line starts.
- Specify which table is on the many sides of the relationship by placing a crow's feet symbol next to the field where the line ends.

Design Decisions:

Entity Name	Why Entity Included	How Entity is Related to other entities
UserInfo	The primary goal of the database is to monitor the usage of cloud resources by the user. Important user information is stored in the UserInfo entity including user IDs, passwords, user contact details and address. This information points to a single sign-in identity and provides complete access to all AWS services and resources in the account.	UserInfo is the main entity of the database that relates to EC2, Bucket and DbInstance entities which store all the information about the AWS resources and these entities provide information about usage of each of the AWS resources (EC2, S3 and RDS) with the help of several associative entities. UserInfo entity also relates to the CardInfo entity through a one-to-one non identifying relationship which helps to store payment information to track bills paid by a particular user.
CardInfo	Another major function of the database is to charge the user every month for to the total resource consumption and utilization. CardInfo entity helps in automatic transaction of payments and provides assurance of payment clearances. User credit card details are securely stored in CardInfo entity including card number, security code, card expiration and details.	CardInfo entity directly relates to UserInfo entity as a crucial factor to map user data and the respective card information exactly. This is a weak relationship as card info can exist without user data . CardInfo and Billing entities are directly related to each other in order to charge the card every month for a bill.
Region	Region is an important parameter to be considered while billing an account. A region is a collection of AWS resources in a geographic area and each region has a RegionName and a unique RegionID. When resources are viewed by users, only those resources are visible that are tied to a specific region and the bills generated are region-specific.	Region entity relates to Ami entity through a one-to-one relationship and helps in gaining insights about how many Ami instances are launched in a particular region. Region entity also directly relates to the Billing entity through a one-to-one relationship which is crucial in deciding the total amount that is billed.
Ami	Ami entity plays a major role in providing pre-configured template information. It captures fine details such as the type of the operating system associated with virtual computing environments, operating system which directly impacts the cost of using EC2 service and creation and termination dates of the instances	Ami entity relates to EC2 and Region entities by one-to-one relationship to provide details of virtual operating systems and other architecture parameter used by an EC2 instance in a certain region.
EC2	EC2 entity is an important one to be considered while a user intends to use AWS EC2 as a resource. It	EC2 entity relates to UserInfo entity through a many-to-one relationship. EC2 as a core entity relates to EC2Class, Storage

	captures every detail of the launched instance(server) such as creation and termination dates, internal memory, active status of the instance and the region where it is launched.	and ActivityTrackerEC2 entities to gain information regarding payment calculation that are associated with an EC2 instance.
EC2Class	It is essential to offer different tier of configuration about EC2 instances. EC2Class entity stores minuscule details about EC2 instances like type of the instance, family of the instance, CPU and RAM utilized by the instance which directly impacts the cost	EC2Class entity relates directly to EC2 entity through a one-to-one relationship.
Storage	Every EC2 instance can have additional block storage associated within itself to store additional data. Storage entity helps in storing this external data and provides details about size of the stored data, volume type and input/output speed internal to the EC2 instance	Storage entity is directly related to EC2 entity through a one-to-one relationship. This is different than every EC2 instance root storage associated internally.
ActivityTrackerEC2	The bill is generated for the user based on the usage of the resource per hour. ActivityTrackerEC2 entity gives complete details regarding the total time spent by the user being active on EC2.	ActivityTrackerEC2 relates to EC2 entity through a many-to-many relationship to draw all the information from EC2 to generate a detailed report of total active time. This in turn helps in deciding the user's total purchase time for EC2 and hence relates directly to TotalSales entity. It is also associated to totalSales in one-to-one relationship
Bucket	Amazon S3 Bucket is the container which is provided as a resource for the user to store all kinds of data. An Amazon S3 bucket name is globally unique. Bucket is an essential entity to track the objects stored.	The S3 Bucket entity relates to Object and UserInfo entities through one-to-many relationships to keep track of the objects being stored in the bucket which directly impacts how the user is billed. Also, the Bucket entity is linked to ActivityTrackerS3 In a one-to-one as the data about objects is accessible from bucket.
Object	As the prime function of the Amazon S3 Bucket is store data, every single file/document is called an object. The various features such as Size, Create Date, Termination Date, Status and URL of the object are stored in the Object entity. Each object can have one storage class which	The Object entity is related to the Bucket entity to keep the track of the objects modified in each individual bucket by every user. Object is related to StorageClass through a one-to-one relationship to establish the tier of storage type.

	impacts the cost	
StorageClass	Amazon S3 offers a range of storage classes designed for different use cases and the pricing differs based on each class. Different storage classes offered include - **S3 Standard**: general-purpose storage of frequently accessed data **S3 Intelligent**-Tiering for data with unknown or changing access patterns **S3 Standard-Infrequent Access & S3 One Zone-Infrequent Access**: long-lived, but less frequently accessed data **Amazon S3 Glacier & Amazon S3 Glacier Deep Archive**: long-term archive and digital preservation.	The kind of Storage Class used by Object is important to track and therefore, StorageClass entity is connected to Object entity through a one-to-one relationship.
ActivityTrackerS3	It is important to gain information about the usage of Buckets, types of StorageClass, Size used by each object in each Bucket, region of the Bucket. ActivityTrackerS3 gathers all such data to generate the total sales.	ActivityTrackerS3 relates to TotalSales by tracking the Storage duration and size of bucket in a one–to-many relationship. Also, to gather the information of the Storage Activity on AWS S3 Bucket, Bucket entity is linked as well. It is also associated to totalSales in one-to-one relationship
DbInstance	RDS is an integral part of AWS when the user wants to use a relational database, and there comes a need to provide database instances. DbInstance entity maintains and tracks various database parameters like DbID, DBport number, DbName, Dbpassword, CreationDate and TerminationDate.	DbInstance is related to the DbStorage through a one-to-one relationship to track the storage size. It is also related to DbClass through one-to-one relationship so that a user can choose from tiers of available classes. It is related to DbEngine through one-to-one relationship. A DbInstance will belong only to one region.
DbClass	When you buy a server, you get CPU, memory, storage, and IOPS, all bundled together. DbClass contains all the parameter values like CPU, RAM, Network, DbFamily, and DbClass information which impact the overall cost	DbClass is connected to the DbInstance through a one-to-one relationship to access and provide information related to the database classes/tiers.
DbStorage	Dbstorage entity provides the total size of the data that a database can store as well as the IOPS (input by output per second) which impacts the price	DbStorage directly relates to DbInstance through a one-to-one relationship to store the compute power and total database size.
Engine	MySQL, MariaDB, PostgreSQL,	Engine directly relates to DbInstance

	Oracle, and Microsoft SQL Server are the available Database engines for the user to choose from. Engine entity provides the user to choose from different vendors which impacts the price	through a one-to-one relationship. Each DbInstance must have one Engine.
ActivityTrackerDb	ActivtyTrackerDb entity helps to give all the information about db region,userinfo, engine related information.	ActivityTrackerDb is a vital entity in terms of billing and Db activity. It stores all the metadata of the activity performed on database and provides calculative inputs. It is associated with Dbinstance in one-to-many relationship. It is also associated to totalSales in one-to-one relationship
TotalSales	The TotalSales entity tracks usage activity of each of the resources (EC2, S3 and RDS), stores it as a successful sale and captures the total bill amount.	TotalSales entity directly relates to ActivityTrackerEC2, ActivityTrackerS3 and ActivityTrackerDb entities in one-to-one relationship to track total usage of these resources and relates to Billing entity (one-to-one relationship) to generate a bill according to the consumption
Billing	One of the main objectives of this database is to generate bills for the user for total resource consumption. The Billing entity provides an atomic decomposition of the bills, so that fine analysis can be carried out for all the utilized resources in a given amount of time. A consolidated final price is produced for the user.	Billing entity directly relates to TotalSales so it can extract all the information to generate and store bills in a one-to-one. It also relates to Transaction entity(one-to-one) which helps in providing necessary information during the payment cycles.
Transaction	Transaction entity maintains payment related data like payment status and payment date and amount. This is an imperative entity in this database which helps in gaining insights on active users and how they are judiciously paying for their usage.	Transaction entity directly relates to Billing and CardInfo entities through one-to-one relationships and captures all the important user related payment data.