Database for a Cloud Computing Platform

Database Specifications: Purpose, Business Problems Addressed and Business Rules

Group 1:
Kruthi Nymisha Kona, Ramya Parameshwar Hebbar,
Vishal Baliga Bantwal, Kirtikumar Waykos

Database Purpose:

This database is built for a cloud computing platform that allows users to rent virtual machines to execute code for computer applications and computations. It serves the purpose of maintaining the cloud resources of the company like MachineImages, Instances, Storage etc. It provides security by encrypting sensitive information like User's Passwords and their Card Details. It automatically creates a bill and executes a transaction once the user stops using an Instance. It is built with the intuition that the users have the ability to configure their own virtual machines and request for additional infrastructure for a specified price. Overall, this database facilitates data collection, processing, billing, transactions and reporting of various processes in this business environment.

Business Problems Addressed:

- Provide a database to monitor and maintain the cloud resources of the company. The resources at disposal would be - Machine Images, Instances, Additional Storage, Security Groups and Server Location
- Provide a database to collect and monitor user data
- Offer encryption and safe storage of user's card details
- Allows users to select Machine Images and Instance of their choice
- Allows users to configure their virtual machines according to their requirements.
- Provide users with the option to save and retrieve customized instances previously used
- Companies can track the total revenue generated in a particular time frame
- Allows automation in generating bills and completing transactions
- Allows companies to monitor their server traffic data
- Allow users and the company to track and analyze resource usage
- Allow users to track the history of their purchase activities

Business Rules:

- Only one account per user
- User can run only one machine [Single Machine Image and Instance] at a time
- For every active session, there will be one bill generated
- Users are charged at the end of each session
- For each bill, payment can be done through only one transaction

Design Requirements (Credit to Professor Simon Wang):

- Use Crow's Foot Notation
- Specify the primary key fields in each table by specifying PK beside the fields
- Draw a line between the fields of each table to show the 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 Decision:

Entity Name	**Why Entity Included**	**How Entity is Related to Other Entities**
UserInfo	*UserInfo* is an entity which stores important information about the user like first name, last name, email, password, phone number, address and ZipCode. This information is vital as one user can have only one unique email and phone number.	Each user's data is stored in the *UserInfo* entity. The primary key UserID maintains a zero-to-many non-identifying relationship with *SaleMaster* to keep a record of the virtual machine selected/configured by the user. It also has a one-to-many non-identifying relationship with the user's CardDetails.
CardDetails	*CardDetails* contains sensitive payment (credit card) information which the user uses to complete the transactions. The card information is encrypted.	*CardDetails* has a one-to-one relationship with *UserInfo* as one card can be traced to only one user. The user on the other hand can have multiple cards. *CardDetails* also maintains a zero-to-many

		non-identifying relationship with *Transactions* to complete the billing process.
MachineImage	*MachineImage* contains a list of Operating systems that the user can select before launching the instance.	*MachineImageId* has a zero-to-many non-identifying relationship with the *SaleMaster.* It also maintains a one-to-one non-identifying relationship with *ServerLocation.*
Instances	Instances are virtual servers that can run applications. The *Instances* entity stores a wide range of information about vCPUs, architecture, cores, storage, and other hardware capabilities that the user can access. It also contains the hourly pricing information for each instance.	*Instance* has a zero-to-many non-identifying relationship with the *SaleMaster* as one type of instance can be used by several users. It also maintains a zero-to-one non-identifying relationship with InstanceConfig as configuration of an instance is optional. (Default configuration is used otherwise)
InstanceConfig	The *InstanceConfig* entity provides the user capability to change the *Network and Subnet* features if need be. Else, the default setting will be set in place.	The *InstanceConfig* has a non-identifying one-to-one relationship with *Instances.*
AdditionalStorage	*AdditionalStorage* will provide additional storage options for the user, each configuration for a particular cost.	The *AdditionalStorage* entity is directly referred to in the *SaleMaster* with a zero-to-many non-identifying relationship as the demand for additional storage space is subject to the user's need.
SecurityGroups	A security group is a set of firewall rules that control the inbound/outbound traffic for a particular instance. If the user doesn’t configure this, the default settings will be set in place.	*SecurityGroups* maintains a zero-to-many non-identifying relationship with the *SaleMaster.*
Tags	A tag consists of a	*Tags* has a one-to-one

	case-sensitive key-value pair. Tags will be applied to a sale.	non-identifying relationship with the *SaleMaster.*
ServerLocation	*ServerLocation* contains a list of the physical servers provided by the company. If not configured, the default server is selected, based on the location.	*ServerLocation* maintains a one-to-many non-identifying relationship with *MachineImage and Instances.*
SaleMaster	The *SaleMaster* is an entity which stores information about every user's configuration of the virtual machine. It is an important base entity, not only because it stores the configured machines, but also assists *BillingInformation* in the billing process. It is critical to know that it contains just the configured machine info, before they are run by the user.	*SaleMaster* holds a direct one-to-one non-identifying relationship with *User, Instances and MachineImage*. It also holds a zero-or-many non-identifying relationship *SecurityGroups, Tags and AdditionalStorage.* It also maintains a one-to-many non-identifying relationship with *BillingInformation and ActivityTracker.*
ActivityTracker	*ActivityTracker* tracks the user's session usage details. For every new session launched by the user, a new activity is recorded and the total usage time of the instance is recorded.	*ActivityTracker* has a one-to-one non-identifying relationship with the *SaleMaster* and *BillingInformation.*
BillingInformation	*BillingInformation* stores the billing details for each session usage of a particular user. With the help of trigger, this table gets automatically populated when there is an activity recorded.	This entity pulls information from the *ActivityTracker and SaleMaster* to calculate the final price for the session and makes it available for the Transaction entity. Thus it has a one-to-one non-identifying relationship with both the *SaleMaster and ActivityTracker.*
Transactions	*Transactions* indicate the total amount charged for a particular session. When the activity is recorded in the ActivityTracker, based on the total bill amount in BillingInformation, the	*Transactions* pulls information from the BillingInformation and fetches the *CardDetails* to successfully process the payment. It holds a one-to-one non-identifying relationship to

	transactions table charges the customers with tax using their *Card details*.	*CardDetails* as well as *BillingInformation*.