OXYsquare's Call Database

Anna Poon
Luis Manahan
Group #9

Table of Contents

PHASE 1

1. Fact Finding, Information Gathering, and Conceptual Database Design

This chapter is all about the fact finding, information gathering and conceptual database design of our system. We will go in depth about our corporation and its Consumer Services Department, write about the fact-searching methods used to gather data, and break-down each detail of our conceptual database design.

1.1 Fact-Finding Techniques, Information Gathering, and Conceptual Database Design

OXYsquare is a large imaginary technological company that is a video game entertainment company. This technology company is similar to companies like Sony PlayStation, that produce PlayStation 4 and PlayStation VR to release out into the market. OXYsquare is very popular among gamers-alike and due to its popularity, the Consumer Services department has been overloaded with phone correspondence from consumers ever since. Japan has requested for OXYsquare to quickly lower its Consumer Services department cost down immediately. OXYsquare has reached out to us for a database design to assist with data analysis on these phone calls.

1.1.1 Introduction to Organization: Description on the business; what the business is doing, and its activities.

As many say, the Consumer Services department is the only department that wants to get rid of its work. When the Consumer Services department satisfies all their consumers, it has truly done its job. In order to truly understand where to start, a consumer services analyst must understand where these calls are coming from and why. Therefore, the most important question is: 'why are they calling us?'

In OXYsquare, we aim to improve its Consumer Correspondence Database for maximum efficiency in data analysis. It relies heavily on its data tools to determine why these consumers are calling us and what we can do to fix their problems.

As a brief overview, the OXYsquare Consumer Services Department controls their support call line. Consumers who have issues with a OXYsquare product: the hardware or game will call the company's Consumer Service line for assistance. When the consumer calls he/she

will immediately enter into our Interactive Voice Response Menu to guide them to an agent. OXYsquare keeps record of all calls that come into our IVR. Afterwards, they will be put on hold until an agent answers their call. The agent will then gather information about the issue and create a ticket for the consumer if they do not have an existing ticket.

Consumer Services department is sectioned by major root causes and each Domain Expert is in charge of a specific sector to focus and analyze. As a domain expert, corporate employees need to look at the data of the calls to try and minimize the call volume coming in. If a domain expert can find better ways to assist consumers and deter them from calling us directly through the support line, then the company will save money. In addition, domain experts need to know how our agents are understanding how to handle specific calls.

In our database, each focus area has a unique name, a unique number, tickets that have a unique ticket number, and its own associated call generated IDs. This data will assist our domain data analysts to understand which issues are trending for a particular focus area. Each focus area will be owned by a specific domain expert in our corporate office who will oversee the cost efficiency of that sector. In short, this database will be used to monitor the calls we receive: its length, feedback, and root cause.

In addition, call managers who manage our call centers will use this database to manage our employees. Which employees are doing the best job in handling calls in the shortest time? How many calls does each agent answer? In addition to the agent metrics, a call manager can use our database to measure and forecast call volume. This will allow a call manager to have the proper data analytics to schedule agents accordingly, avoiding understaffing or overstaffing. (Note: OXYsquare gets charged for every call a consumer makes, even if an agent does not answer the phone, because it costs money for the consumer to be on our call menu.)

Overall, our goal is to avoid consumers calling as as much as possible, equip our agents with the knowledge and skills to handle calls as efficiently and quickly as possible, avoid having a consumer stay on hold, and keep consumers away from needing to call our support line again.

This analytical tool provides a complete oversight for all incoming calls and our call performance. In short, this enhanced database system will allow for users to see a high level view of the consumer's account, the call, the journey of the ticket, and lastly the root of the problem. This database will prove its significance due to its ability to categorize, track calls and cost.

1.1.2 Description Fact-Finding Techniques

To build an effective, efficient, and functional database, one must research in great detail the inner workings of the organization or enterprise. With tech companies in Silicon Valley, there are many resources and examples one can use to start their research. As a database systems architect, one must interview the appropriate experts, stakeholders, and users of the project. What is the objective of the database created? What will these people be using the database for? What are the database requirements and the required functionality?

To gather information for this database, we interviewed employees working in Sony PlayStation from the Consumer Services. In addition, we studied in depth their department functionalities around projects and product creation. In a corporation like Sony PlayStation, Consumer Services is broken up into areas of focus: hardware, service, billing, etc. These areas of focus are then assigned to specific domain experts, who are data analysts for that problem area. These analysts are in charge of looking at the data collected at their Consumer Services department and finding cost effective alternatives or solutions. Every year these technological companies have annual goals to improve cost and its efficiency, therefore these database systems are critical for the corporate Consumer Service departments.

The functional description is that every consumer services user should have the ability to use this database to analyze and track call times accordingly. This provides deep oversight for all areas of focus such as: security, account, game inquiries. Users must be able to run reports to track the call handle times, the call costs, and the root problems. Supervisors and managers must be able to monitor and graph their agent's call performance and average call handle time.

1.1.3 Scope of the Conceptual Database

When designing this database, we kept this database in the scope of Consumer Services to hone in the specifics of our data. This database will be fully detailed for optimal query searching and assist our domain experts, managers and executives to see the performance of our Call Centers. We have created this database to be specific and provide strong overview of the correspondence we receive from consumers.

1.1.4 Entity and Relationship Sets Description

Looking at the database requirements, we have created an Entity-Relationship(ER) model to clearly represent how our database system will work to technical and non-technical employees who are interested in a project performance metrics program. We will be going over the specifics of the entity and relationship sets in Section 1.2 below.

1.2 Conceptual Base Design

After all the information about our technology company has been gathered, we used different entities to represent the structure of our diagram. We created both entity sets and relationship sets to define entities in our diagram and how these entities relate to one another. Entities in the most simplest explanation are real world objects like: a video game or a person. These entities in our diagram relate to one another to create the database we desire.

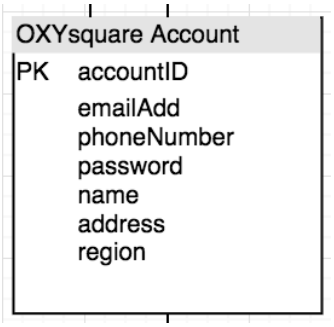
The summary of Section 1.2 is that we will first talk in detail our entities: the attributes and the types they have. After describing in detail our entities, we will talk about the relationships themselves: how they relate, the cardinality and lastly, the participation type they have with one another. By the end of relationship sets section, we will then proceed to show the entire UML diagram for one to completely see the whole picture.

1.2.1 Entity Set Description

Below this section is a detailed description of all the entities we intend to put into our UML diagram. It will include: description, domain type, value range, and other details.
These entities are broken down and described in the tables below:

<u>OXYsquare Account</u> - Strong Entity
Description:

Each gamer playing our hardware devices will be required to sign up for an OXYsquare account. The account tells us the unique account ID associated with the consumer, their email address used to sign up with the account, their phone number, their password, their billing address, and the region they belong to: North America, Europe, etc.



Note: *Customers may have more than one account with OXYsquare, but we are focusing on the calls we receive from each OXYsquare account.*

Candidate Keys: accountID, emailAdd,
Primary Keys: accounttID
Fields to be Indexed: generatedID, keyPointOfConcern, subPointsOfConcern

OXYsquare Account Attributes:

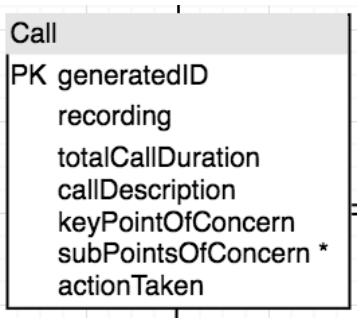
Attribute	accountID	emailAdd	phoneNumber	password
Description	Auto-increment unique identifier for user accounts. The starting characters change across regions.	Email address used to register the account. (Will not accept improper email addresses)	Primary contact number that only takes integers. Special characters will automatically be removed such as '(', ')', '-'	The primary security string that lets the user have access to their account.
Domain/Type	Integer	String	Integer	string
Value-Range	0-MAX_ID	5-254	10-10	6-255
Default Value	MAX_ID + 1	None	None	None
Nullable	No	No	Yes	No
Unique	Yes	Yes	No	No
Single/Multiple	Single	Single	Single	Single
Simple/ Composite	Simple	Simple	Simple	Simple

Attribute	name	address	region
Description	Name of the account holder, (First, Middle, Last)	Address used to ship physical products and send other relevant information.	String used to identify the persons geographic location. This is done by a drop down menu. This is used to change the support or warranty information.
Domain/Type	String (3 Strings)	String(3 Strings), Integer	String
Value-Range	Any	Any, Any, Any, 00000-99999	Any
Default Value	None	None	None
Nullable	No	No	No
Unique	No	No	No
Single/Multiple	Single	Single	Single
Simple/Comp	Composite	Composite	Simple

<u>Call</u> - Strong Entity
Description:

When a consumer with a OXYsquare account calls us they will hear our company's IVR (interactive voice response) which will have them go through a menu to be routed to an agent. For every call we receive, we know the time the consumer stays in our IVR menu and we will automatically be recording each call once an agent receives the call. This call entity was created to have the database sort the call lengths of our consumers and have a brief description of the reason why our consumer is calling and what our agent action our agent took on the call to resolve the issue.

Candidate Keys: generatedID
Primary Keys: generatedID
Fields to be Indexed: generatedID, keyPointOfConcern, subPointsOfConcern, actionTaken



Call Attributes:

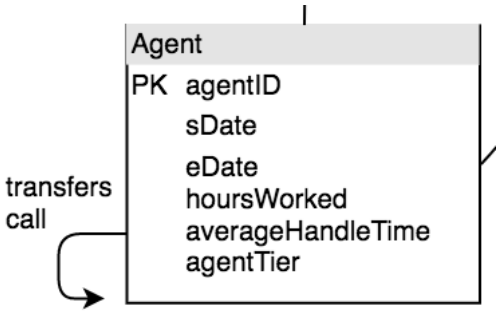
Attribute	generatedID	recording	totalCallDuration	keyPointOfConcern
Description	Auto-increment unique identifier for call. This is to manage the number of calls received.	The recording ID of the call. This used for verifying information, quality assurance, and company liability issues.	The total length of the call, this includes caller wait time.	The main point of concern that the caller raises up. Filled up by Agent.
Domain/Type	Integer	File	Time	String
Value-Range	0-MAX_INT	Any	Any	Any
Default Value	MAX_INT + 1	None	None	None
Nullable	No	No	No	No
Unique	Yes	Yes	No	No
Single/Multiple	Single	Single	Single	Single
Simple/ Composite	Simple	Simple	Simple	Simple

Attribute	subPointsOfConcern	actionTaken
Description	Lesser valuable points, but still relevant on how to solve the problem	A flag used whether a call will be made into a ticket or not. 0 for being no ticket, 1 for creation of ticket.
Domain/Type	String	Boolean
Value-Range	Any	0-1
Default Value	None	0
Nullable	No	No
Unique	No	No

Single/Multiple	Multiple	Single
Simple/Comp	Simple	Single

<u>Agent</u> - Strong Entity
Description:

Every call coming into OXYsquare will have an agent that answers the phone (unless the consumer does not want to stay on the phone and drops the call before an agent answers). This agent will be measured by how fast they handle each call which is "average handle time". In addition, agents will have tiers of expertise, so if our lower tiers of agents do not know the answer, they will need to transfer the call to a higher tiered agent. Due to these measurement specifications needed, we have decided to create a agent entity.



Candidate Keys: agentID
Primary Keys: agentID
Fields to be Indexed: agentID, agentTier, afterCallWork

Agent Attributes:

Attribute	agentID	sDate	eDate	hoursWorked
Description	Auto-increment unique identifier for agent. This is used to identify the company's agents.	The date when the agent was hired.	The date when the agent was let go or resigned.	The number of hours the employee has been on a call.
Domain/Type	Integer	Date	Date	Time
Value-Range	0-MAX_INT	Any	Any	Any
Default Value	MAX_INT + 1	None	None	None
Nullable	No	No	Yes	No

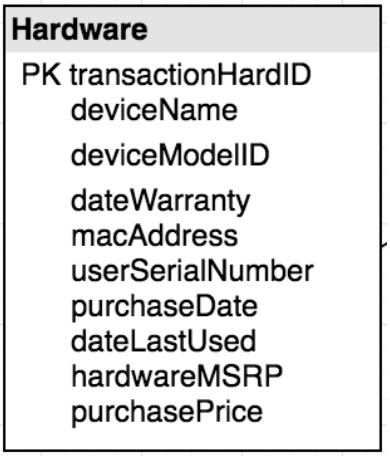
Unique	Yes	No	No	No
Single/Multiple	Single	Single	Single	Single
Simple/ Composite	Simple	Simple	Simple	Simple

Attribute	averageHandleTime	agentTier
Description	The average time it takes for agents to handle a call.	The level of the agent, 1 being the lowest which handles basic tickets. 2 more complicated… 3 being the highest which could be for more technical calls.
Domain/Type	Time	Integer
Value-Range	Any	1-3
Default Value	None	1
Nullable	No	No
Unique	No	No
Single/Multiple	Single	Single
Simple/Comp	Simple	Single

<u>Hardware</u> - Strong Entity

Description:

Every call will either be related to the company's hardware or game we produce. Do to this, hardware entity is needed to keep record of the hardware related to the call, the specific serial number of the device that the consumer is using and many other attribute information. Doing this will allow for us to know how many calls we receive for specific modeled devices, a history of all the issues the consumer experiences for a particular console overtime, etc.



Candidate Keys: transactonID
Primary Keys: transactionID, macAddress, userSerialNumber
Fields to be Indexed: transactionID, deviceName, deviceModelID

Hardware Attributes:

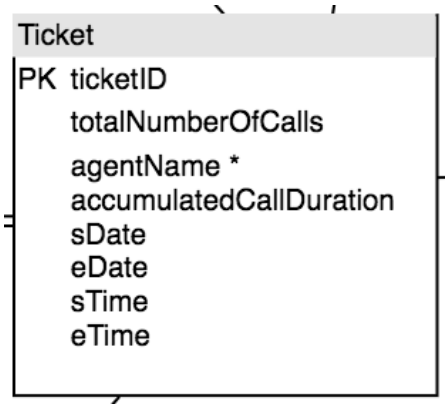
Attribute	transactionID	deviceName	deviceModelID	dateWarranty
Description	Auto-increment unique identifier for hardware. This is used to identify the hardware the user has.	The name of the device or hardware.	The sequence of integers that determines the hardwares name, revision number, model, release date.	The end date of the warranty for the device.
Domain/Type	Integer	String	Integer	Date
Value-Range	0-MAX_INT	Any	0-99999	Any
Default Value	MAX_INT + 1	None	None	None
Nullable	No	No	No	No
Unique	Yes	No	No	No
Single/Multiple	Single	Single	Single	Single
Simple/ Composite	Simple	Simple	Simple	Simple

Attribute	macAddress	userSerial Number	purchaseDate	dateLastUsed	purchasePrice	hardware MSRP
Description	The unique sequence of characters and integers to signify the networking module of a hardware. This is used for security	The serial number of the hardware. This is unique because each device sold gets its own serial number to track.	The purchase date of the hardware in question.	The date when the device was last used.	The price the consumer purchased the hardware for	The manu-facturer's standard price for the game

	purposes.					
Domain/Type	String	String	Date	Date	Int	Int
Value-Range	0 - 281,474,976,710,656	Any	Any	Any	Any	Any
Default Value	None	None	None	None	None	None
Nullable	No	No	No	No	No	No
Unique	Yes	Yes	No	No	No	No
Single/Multiple	Single	Single	Single	Single	Single	Single
Simple/Comp	Simple	Single	Simple	Simple	Simple	Simple

<u>Ticket</u> - Strong Entity
Description:

A ticket is created for each new issue a consumer experiences. These tickets are used to track the total calls we receive for a specific issue, the total call time we needed to resolve the issue with the consumer and with this entity we will be able to relate to other entities to see which root problems, devices, or games we are receiving most calls about, which calls are the hardest calls to complete, etc.



Candidate Keys: ticketID
Primary Keys: ticketID
Fields to be Indexed: ticketID, agentName

Ticket Attributes:

Attribute	ticketID	totalNumberOfCalls	agentName	accumulatedCallDuration

Description	Auto-increment unique identifier for hardware. This is used to identify tickets in question.	The accumulated number of calls the ticket has received. This is to keep track of follow-ups from users.	The list of agents that has worked on the ticket. This is to keep track of responsibility.	The total length of calls for the ticket in question. This is to keep track of any ticket rec
Domain/Type	Integer	int	String	Time
Value-Range	0-MAX_INT	0-100	Any	Any
Default Value	MAX_INT + 1	0	None	None
Nullable	No	No	No	No
Unique	Yes	No	No	No
Single/Multiple	Single	Single	Multiple	Single
Simple/ Composite	Simple	Simple	Simple	Simple

Attribute	sDate	eDate	sTime	eTime
Description	The start date of the ticket.	The date when the ticket has been closed.	The time when the ticket was created.	The time when the ticket is closed.
Domain/Type	Date	Date	Time	Time
Value-Range	Any	Any	Any	Any
Default Value	None	None	None	None
Nullable	No	No	No	No
Unique	No	No	No	No
Single/Multiple	Single	Single	Single	Single
Simple/Comp	Simple	Simple	Simple	Simple

<u>Game</u> - Strong Entity

Description:

As noted before, every call will either have a game or hardware associated with the call itself. Due to this, we will need to create a game entity to be able to query whether a specific game release is causing a spike in the call volume received, whether a certain game is having installation issues on a particular purchase date, etc.

Game	
PK	transactionGameID
	gameName
	softwareVersion
	distributionType
	gameID
	purchaseDate
	purchasePrice
	gameMSRP
	gamePublisher

Candidate Keys: transactionID
Primary Keys: transactionID, gameID
Fields to be Indexed: transactionID, gameName, gameID

Game Attributes

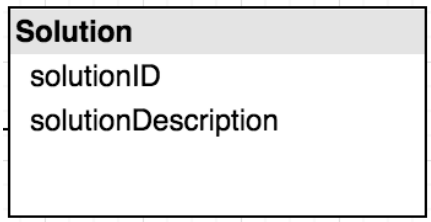
Attribute	transactionID	gameName	softwareVersion	distributionType
Description	Auto-increment unique identifier for software. This is used to identify the game that was purchased by the user.	The name of the game or software.	The sequence of integers that determines the software's revision number. Used to help find bugs in games.	The boolean that will tell if the software or game is physical or digital, 0 being digital and 1 for physical.
Domain/Type	Integer	String	Integer	Boolean
Value-Range	0-MAX_INT	Any	0-99999	0-1
Default Value	MAX_INT + 1	None	None	0
Nullable	No	No	No	No
Unique	Yes	No	No	No
Single/Multiple	Single	Single	Single	Single
Simple/ Composite	Simple	Simple	Simple	Simple

Attribute	gameID	purchaseDate	purchasePrice	gamePublisher	gameMSRP
Description	A uniquely	The date when	The amount the	The name of the	The

	generated number to help distinguish games from one another. It is also used for identifying multiple revisions of a game.	the user purchased the software or game.	user has paid for the software or game.	company that published the game	manufacturer's standard price for the game
Domain/Type	Integer	Date	Float	String	Int
Value-Range	0 to 9999999	Any	Any	Any	Any
Default Value	None	None	None	None	None
Nullable	No	No	No	No	No
Unique	Yes	No	No	No	No
Single/Multiple	Single	Single	Single	Single	Single
Simple/Comp	Simple	Single	Simple	Simple	Simple

<u>Solutions</u> - Strong Entity
Description:

This is a entity that will tell us the solution to the problem associated with the call. When we query this we can see which solutions were found for every problem.



Candidate Keys: solutionID
Primary Keys: solutionID
Fields to be Indexed: solutionId

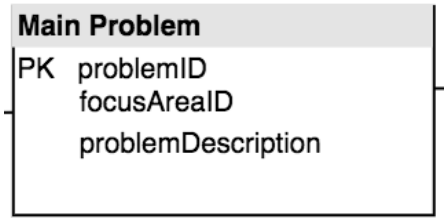
Main Problem Attributes:

Attribute	solutionID	solutionDescription
Description	A uniquely generated number to help distinguish specific solutions from one another.	The description that is based on the solution found. Provides a general description of the solution.
Domain/Type	Integer	String

Value-Range	0 to 9999999	Any
Default Value	None	None
Nullable	No	No
Unique	Yes	No
Single/Multiple	Single	Single
Simple/Comp	Simple	Single

<u>Main Problem</u> - Strong Entity
Description:

This is a entity that is a superclass to record the problems we receive at OXYsquare. This superclass will describe the unique problem ID and the problem description the consumer is experiencing. This problem is related to a subclass of areas of focus: hardware inquiry, game inquiry, service, malicious activity, troubleshooting, billing, and lastly, account issues. We have decided to break this class into subclasses due to the various attributes needed for a specific focus area. If a consumer is calling about a service for a hardware issue, we will require different information to be recorded for querying compared to a consumer calling about their OXYsquare account being hacked into.



Candidate Keys: problemID
Primary Keys: problemID
Fields to be Indexed: problemID, focusAreaID

Main Problem Attributes:

Attribute	problemID	problemDescription	focusAreaID
Description	A uniquely generated number to help distinguish specific problems from one another. This is because a problem could be extremely specific and could be queried on its own.	The description that is based on the ticket and call descriptions. Provides a general description of the problem. This is not a searchable field, just provides information.	The general ID associated with the ticket. This is not unique. Ex. 100 for HardwareInquiry, 110 for Account Issues, etc.
Domain/Type	Integer	String	Integer
Value-Range	0 to 9999999	Any	0-1000
Default Value	None	None	None
Nullable	No	No	No
Unique	Yes	No	No
Single/Multiple	Single	Single	Single
Simple/Comp	Simple	Single	Single

<u>Account Issues</u> - Subclass of Main Problem
Description:

Account Issues is a focus area for OXYsquare account issues that may arise. Some examples of account issues that a consumer will face are:

- An account is banned or suspended due to the company's discretion - violating their terms of service.
- An account is lost due to missing or forgotten password

Adding this to the query will allow for us to know why the consumer is calling us and what time of account issues our consumers are experiencing.

Account Issues
accountType
accountStatus
accountIssue

Candidate Keys: problemID (Inherited)

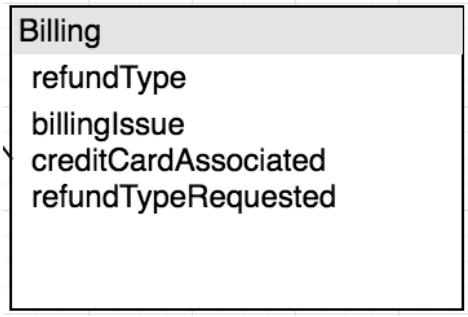
Primary Keys: problemID
Fields to be Indexed: problemID, accountIssue

Account Issues Attributes:

Attribute	accountType	accountStatus	accountIssue
Description	This attribute will tell you if this account is master or subaccount. This attribute can be important because there can be issues with game sharing between two people with multiple sub-accounts	This account status will show you if this account is active, suspended, or banned. This will give you a hint as to why the consumer may be calling in the first place	This is a description area for an agent to identify a specific issue the consumer maybe be experiencing ie. hijacking
Domain/Type	String	String	String
Value-Range	Any	Any	Any
Default Value	None	None	None
Nullable	No	No	No
Unique	No	No	No
Single/Multiple	Single	Single	Single
Simple/Comp	Simple	Single	Single

<u>Billing</u> - Strong Entity (subclass)
Description:

Billing is a focus area for OXYsquare billing issues that come up. Adding this entity will show us what types of refunds requested, how many of the requested refunds were granted, what types of refunds are be getting with specific purchases. Adding this to the query will allow for us to know why the consumer wants a refund, if we gave



them a refund, what types of refunds we are giving the consumer: store credit or credit card.

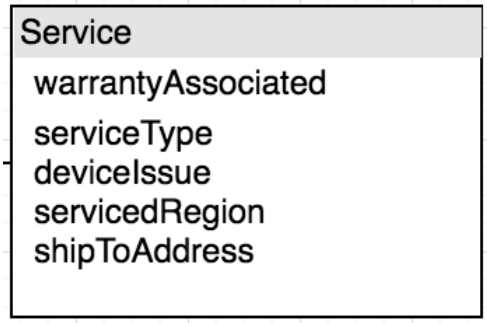
Candidate Keys: problemID (Inherited)
Primary Keys: problemID
Fields to be Indexed: problemID, billingIssue, refundType,

Billing Attributes:

Attribute	refundType	billingIssue	creditCardAssociated	refundTypeRequested
Description	Refund type will identify the type of refund provided to the consumer: store credit or credit card refund or paypal or none	billingIssue describes the reason why a refund is being requested by the consumer: fraud, accidental purchase,	This is the field for the credit card associated with the purchase	Refund type will identify the type of requested by the consumer: store credit or credit card refund or paypal
Domain/Type	String	String	String	String
Value-Range	Any	Any	Any	Any
Default Value	None	None	None	None
Nullable	No	No	No	No
Unique	No	No	No	No
Single/Multiple	Single	Single	Single	Single
Simple/Comp	Simple	Single	Single	Single

<u>Service</u> - Strong Entity (subclass)
Description:

Service is a focus area for OXYsquare service issues that come up. Adding this entity will show us queries like: what types of services we performed: out of warranty, OEM warranty, or extended warranty, what issues for which devices are coming into our facilities, what regions have which services. Adding this to the query will allow for us to know which hardware issues are most prevalent, what types of services are being performed.



Candidate Keys: problemID (Inherited)
Primary Keys: problemID
Fields to be Indexed: problemID, servicedRegion, deviceIssue

Service Attributes:

Attribute	warrantyAssociated	serviceType	deviceIssue
Description	This is the consumer's warranty type associated to the service: out of warranty, OEM warranty, or extended warranty	The type of service provided on the machine ie. HDD replacement, BLOD (blue light of death).	The issue that needs fixing. This will be a dropdown selection identifying the issue for the repair.
Domain/Type	String	String	String
Value-Range	Any	Any	Any
Default Value	None	None	None
Nullable	No	No	No
Unique	No	No	No
Single/Multiple	Single	Single	Single
Simple/Comp	Simple	Single	Single

Attribute	servicedRegion	shipToAddress
Description	The region the repair is being serviced	The address the consumer wants us to ship to.
Domain/Type	String	String
Value-Range	Any	Any
Default Value	None	None
Nullable	No	No
Unique	No	No
Single/Multiple	Single	Single
Simple/Comp	Simple	Single

<u>Troubleshooting</u> - Strong Entity (subclass)
Description:

This troubleshooting entity will help us understand which troubleshooting topics do our agents have the most issues with. Do they have trouble with assisting consumers with internet? Do they need more training on pairing devices together? Do they need more training on network connection issues? This will provide us the best understanding of how we can better train our agents and also provide better troubleshooting guides on our support website.

Candidate Keys: problemID (Inherited)

Primary Keys: problemID
Fields to be Indexed: problemID, errorCode, troubleshootingType,

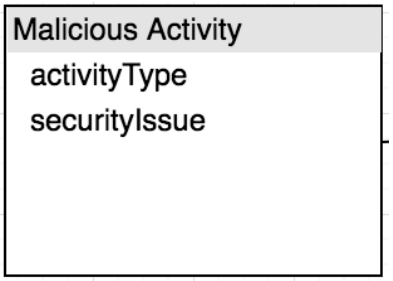
Troubleshooting Attributes:

Attribute	internalPartAssociated	troubleshootingType	errorCode
Description	The internal part of	The type of	The Integer that

	the machine that the agent is troubleshooting.	troubleshooting provided ie. pairing, installation, performance, connection, etc.	shows the error code the user has reported. This classifies the problem and is used to sort the most common errors.
Domain/Type	String	String	Integer
Value-Range	Any	Any	Any
Default Value	None	None	None
Nullable	Yes	No	No
Unique	No	No	No
Single/Multiple	Single	Single	Single
Simple/Comp	Simple	Single	Single

<u>Malicious Activity</u> - Strong Entity (subclass)
Description:

Malicious Activity is an entity that records any malicious activity being reported to us. In this day and age where technology has become a staple of daily life, security is extremely significant to technology companies like OXYsquare. Malicious activity will be an entity where we record issues like an account being hijacked, or fraudulent activities on the account. With this information, a security expert will be able to query the types of malicious activities and range by date and time.



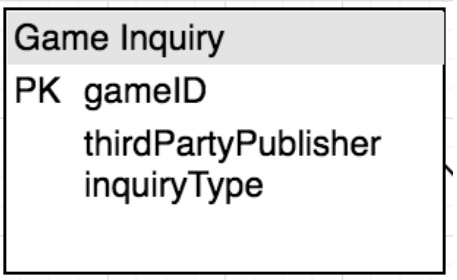
Candidate Keys: problemID (Inherited)
Primary Keys: problemID
Fields to be Indexed: problemID, activityType, securityIssue,

Malicious Activity Attributes:

Attribute	activityType	securityIssue
Description	The activity type is a dropdown of the malicious activity. This mostly includes player reports among other players	A list of who is affected, Game, Company, User, Other Users, etc.
Domain/Type	String	String
Value-Range	Any	Any
Default Value	None	None
Nullable	No	No
Unique	No	No
Single/Multiple	Single	Single
Simple/Comp	Simple	Single

<u>Game Inquiry</u> - Strong Entity (subclass)
Description:

For all general questions that a consumer may have regarding general game specific information like: the release date for Road Fighters and Super Dario Kart. This will show us the number of calls we receive for a particular question for a particular game. If we have enough questions about a particular game, we can then update our website to address specific consumer questions about a game.

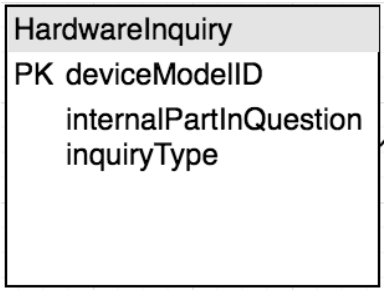


Candidate Keys: problemID (Inherited)
Primary Keys: problemID, gameID
Fields to be Indexed: problemID, gameID

Game Inquiry Attributes:

Attribute	thirdPartyPublisher	gameID	inquiryType
Description	The name of the third party publisher responsible for the game.	A uniquely generated number to help distinguish games from one another. It is also used for identifying multiple revisions of a game. This Id may or not be released yet.	The string that determines whether a question is about "Release Date", "General Information", "Hardware Requirements", etc.
Domain/Type	String	Integer	String
Value-Range	Any	0 to 9999999	Any
Default Value	None	None	None
Nullable	No	No	No
Unique	No	Yes	No
Single/Multiple	Single	Single	Single
Simple/Comp	Simple	Single	Single

<u>Hardware Inquiry</u> - Strong Entity (subclass)
Description:



For all general questions that a consumer may have regarding general hardware specific information like: the specifications of a DS4 or the cost of a OXYsquare VR. This will show us the number of calls we receive for a particular question for a particular device. If we have enough questions, we can then update our website to address these questions.

Candidate Keys: problemID (Inherited)
Primary Keys: problemID, deviceModelID
Fields to be Indexed: problemID, deviceModelID

Hardware Inquiry Attributes:

Attribute	internalPartInQuestion	deviceModelID	inquiryType
Description	The specific part in question by the user. This could be classified as processor, memory, or any other hardware related.	The sequence of integers that determines the hardwares name, revision number, model, release date.	The string that determines whether a question is about "Release Date", "General Information", "Specifications", etc.
Domain/Type	String	String	String
Value-Range	Any	0-99999	Any
Default Value	None	None	None
Nullable	No	No	No
Unique	No	No	No
Single/Multiple	Single	Single	Single
Simple/Comp	Simple	Simple	Single

1.2.2 Relationship Set Diagram

Relationships show how entities relate to one another. Below we will show you how each entity relate to one another and why they are connected. These relationships will piece together our database design to further exemplify the full potential of our database and why we have connected it together in such a fashion.

We will also talk about cardinality, which is the domain and its cardinality ratio between the entities it's related to. Furthermore, we will be discussing participation of each side of entities, ie. partial and total participation. Total participation means that the first entity totally participate with the second entity. If there is no total participation, then we can say that it is a partial participation.

Overall, these relationship set explanations will provide detail for each relation, go into detail of any constraints, the participation of each type, and of course the cardinality of each.

Oxysquare Account **initiates** Call

Description: This relationship will link a Oxysquare Account to Calls to identify how many calls we have received for a specific account at Oxysquare. This will allow for us to query the calls we receive from specific accounts.

Cardinality: This relationship is a **1 to Many** relationship because a consumer can call us as many times as they want. They can call once or many times to inquire about our different products and games. They can also never call.

Participation Constraint: A call cannot exist without an account. The calls we receive from our consumers are all calls associated with a OXYsquare account, thus this is a total participation. A consumer's account can have multiple calls associated, or a consumer can be very tech savvy and never call us, so this is a partial participation.

Call is **answered by** Agent

Description: Each consumer who calls in will wait for an agent to pick up the phone. This relationship between Call entity and the Agent entity will allow us to query details between the relationships of agent and call like how long a particular call took and who answered the call. Along with this, we can query how many times an agent had to transfer the call to another higher tiered agent because they did not have the knowledge or expertise to complete the call. Doing these queries will allow for us to know in depth how knowledgeable our agents are at each tier and how long is call handle time is for each agent. Additionally, we can query to see how many calls are not associated with an agent at all and how long the consumer decided to wait in line for a call. Doing this will allow for us to gauge the amount of agents we many need per time of day or day of week.

Cardinality: This relationship is a **1 to Many** relationship because a call can be answered by zero, one, or many agents if transferred.

Participation Constraint: An agent must be associated with a call which makes this total participation, but a call may or may not have an agent answering the call (in the case of a

consumer dropping the call before an agent answers). This means that call to agent is a partial participation.

Call **references** Ticket

Description: When a consumer calls, our agents will always make a ticket for them. This ticket will allow for us to know how many calls are coming in for a specific ticket and if this consumer is likely to spam our call center. In addition it will show us what types of calls are causing more follow-up calls, which is extremely important for Consumer Services. At OXYsquare, we strive to make sure we can assist the consumer as fast as possible so if we see that we are getting 3, 4, 5 follow-up calls for a particular type of problem we know there is an issue we need to address with the domain experts.

Cardinality: This is a **1 to Many** relationship, since a ticket can contain 1 to many calls. A call can either be associated to a ticket or no ticket at all (when a consumer calls and is too hasty to wait for an agent to answer the phone).

Participation Constraint: Tickets are only made when agents answer the phone and create the ticket or the consumer references an existing ticket. This means that ticket is a total participation. Call entity relationship to Ticket entity is not total participation because a call may not always be answered by agent so the call never gets a ticket created. This also allows for us to be able to query calls not referenced by any ticket, showing us how many calls were never linked to a ticket. If our company sees that many calls are not being linked to a particular ticket, we will know that the agents are understaffed and our consumers were waiting on the phone for too long at a particular time of day. This will significantly assist us with call forecasting so we know how much people we need to staff on a particular time or day of week.

Agent **creates** Ticket

Description: When an agent answers a call, the agent will create a ticket if this is the first time a consumer has called us about a particular issue. If it is the first time we are hearing about the problem a consumer is experiencing, we will have the agent create a ticket and reference the call to the ticket. If this is a follow-up call, an agent will just reference the call to the ticket prior.

Cardinality: This is a **1 to Many** because an agent can create many tickets for consumers in the course of time.

Participation Constraint: An agent has total participation with ticket because all call agents will create tickets for consumers who do not have a ticket. Tickets have total participation with calls because all tickets are created by agents on the phone.

Ticket **addresses** Hardware

Description: A consumer who calls in will always have either a hardware or game associated to the ticket that is created for him. These relationships will allow for us to query what tickets and which calls are linked to a particular hardware product we have so we can address it on a global scale.

Cardinality: This is a **1 to 1** because one ticket will reference a hardware product.

Participation Constraint: A ticket may or may not reference a hardware product because some tickets will not address a hardware product but a game product instead. A hardware system may never have issues so it will not be linked to a ticket.

Ticket **addresses** Game

Description: A consumer who calls in will always have either a hardware or game associated to the ticket that is created for him. These relationships will allow for us to query what tickets and which calls are linked to a particular game product we have so we can address it on a global scale.

Cardinality: This is a **1 to 1** because one ticket will reference a game.

Participation Constraint: A ticket may or may not reference a hardware product because some tickets will not address a game but a hardware product instead. A game on the account may never have issues so it will not be linked to a ticket.

Agent **transfers to** Agent

Description: When an agent does not have the answers and does not know what to do, he/she will transfer to another agent with more expertise. This agent will be a higher tiered agent who in hopes will be able to assist the consumer better.

Cardinality: This is a **1 to 1** relationship because an agent can only transfer the call to another agent.

Participation Constraint: There are no constraints in tiers. A lower tier agent can transfer to a higher tier agent. A higher tier agent may want to transfer to a lower tier agent if there are issues that are easy and only done by lower tiers.

Ticket **describes** Main Problem

Description: A ticket will have a main problem which will have the focus area linked to it. This will allow for us to know what the problem was that the consumer was dealing with and the category of the problem itself. This Main Problem entity is a superclass that links to subclasses: Game Inquiry, Hardware Inquiry, Malicious Activity, Account Issues, Billing, Service, and Troubleshooting. This will allow for us to know the categories of the problems coming into our call center.

Cardinality: This is a **1 to 1** relationship because a ticket should only address one problem that the consumer has an issue with.

Participation Constraint: A ticket has total participation with main problem for every ticket should have a main problem and vice versa.

OXYsquare Account **has** Game

Description: OXYsquare Accounts are able to purchase digital or physical games from the store or any third party store. These games are then tied to the account if it is a digital version of the game. If they are physical, then database will still record of the account playing the game.

Cardinality: This is a **1 to Many** relationship because an OXYsquare Account could have many games, but each instance of Game is only assigned to one account. These are not the base games, but instances of them.

Participation Constraint: An OXYsquare Account has partial participation with game and Game has partial participation as well.

OXYsquare Account **has** Hardware

Description: OXYsquare Accounts are required to be able to use specific OXYsquare consoles or devices. The hardware is then tied to the account unless the user deactivates it from their account. It can also be relinquished if the device has not been used for an extended period of time and it has been registered to a new user.

Cardinality: This is a **1 to Many** relationship because an OXYsquare Account could have many games, but each instance of Game is only assigned to one account. These are not the base games, but instances of them.

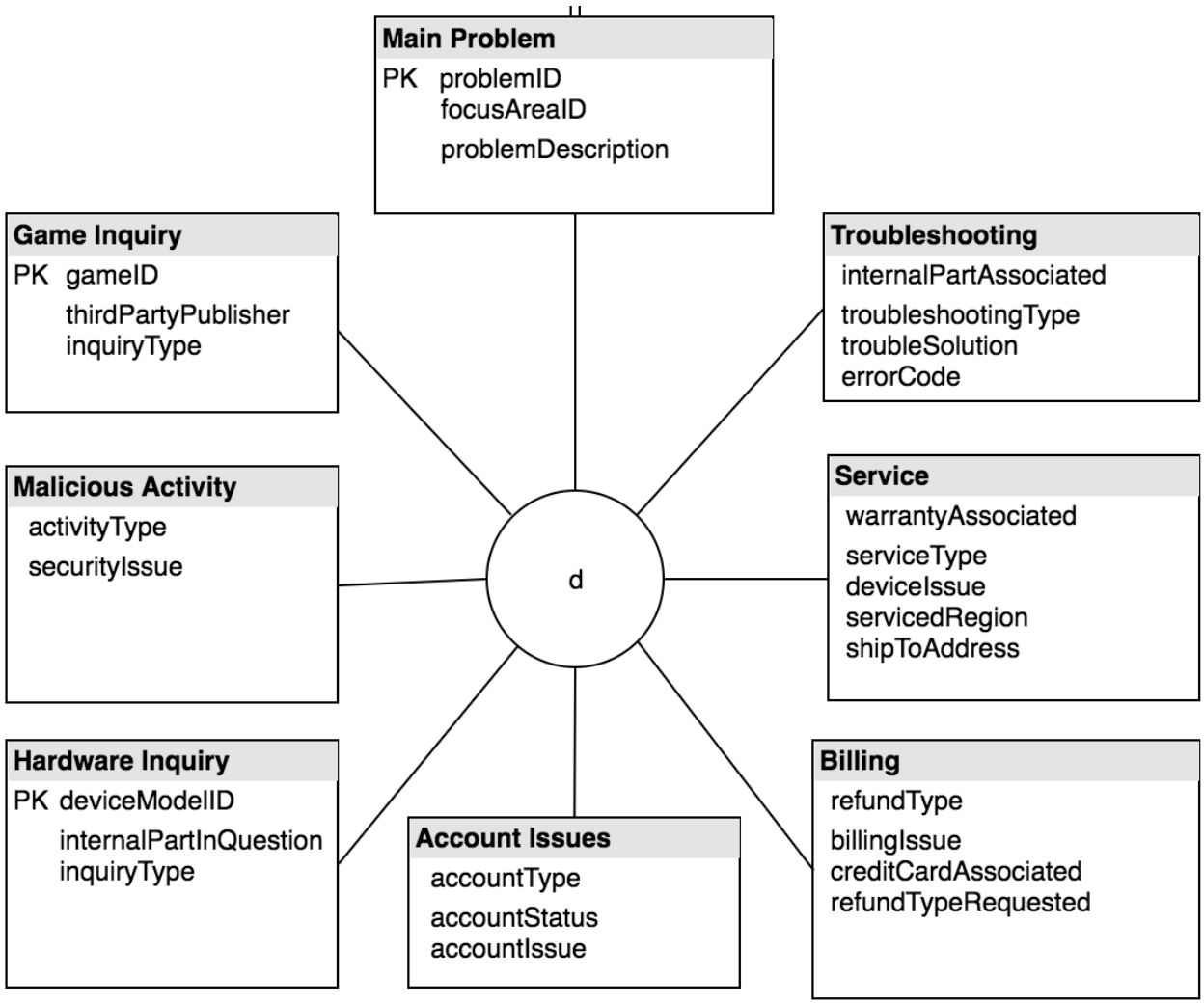
Participation Constraint: An OXYsquare Account has partial participation with game and Game has partial participation as well.

1.2.3 Related Entity Set

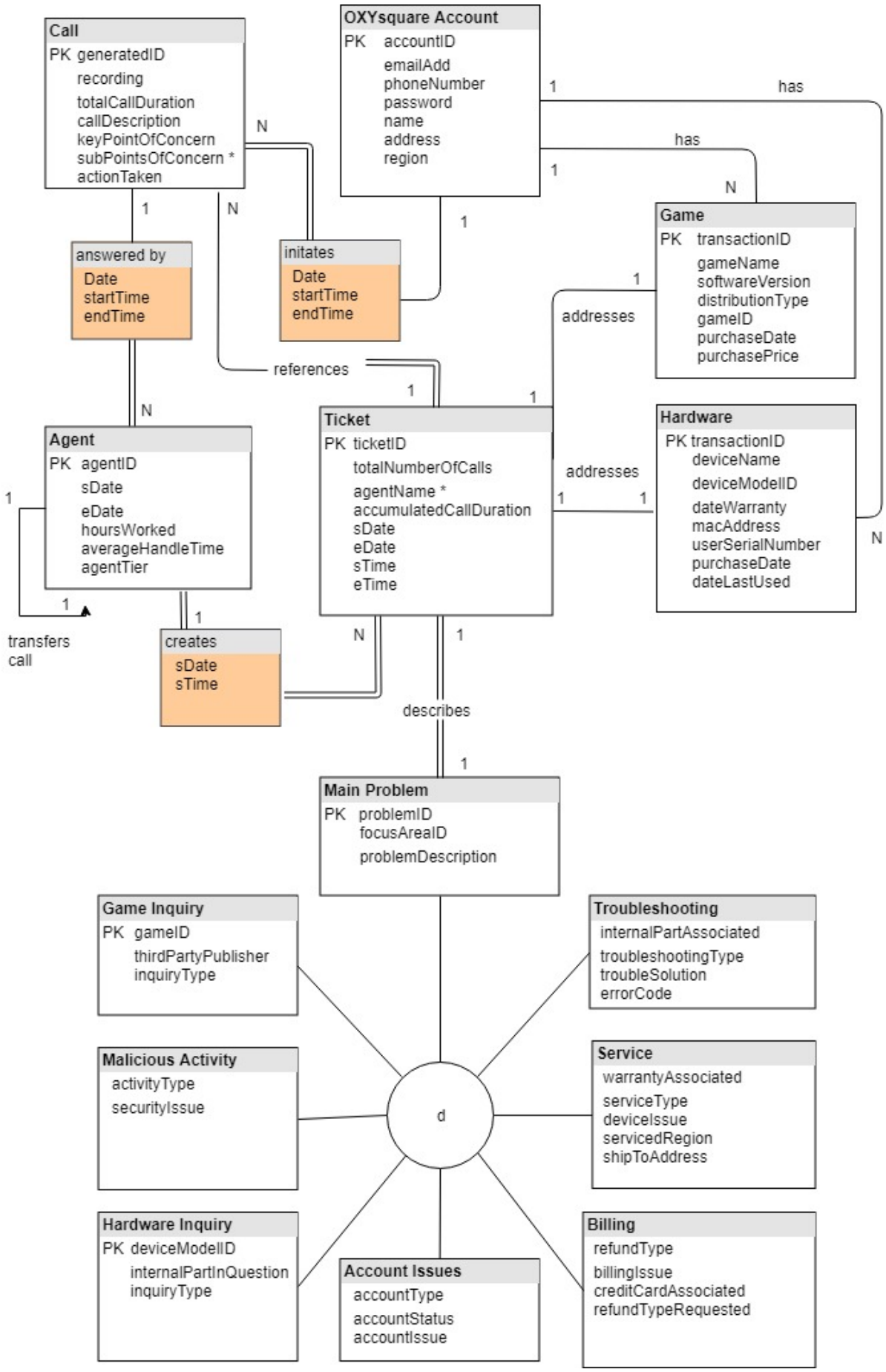
Generalization is the act of grouping together entities who share such similar attributes together that we can create a single entity as a more generalized view. In this scenario, we decided to make a generalized entity of the main problem of each ticket. As each subclass needs a specific problem ID and focus areaID we decided to generalize it into a single entity. This creates a more streamlined problem reporting system so that queries are easily searchable, while still maintaining order when searching for specific focus areas using its focus area ID.

Specialization is the act of separating a large entity into separate entities because they have separate lists of attributes and/or they are related to different entities themselves. In our UML diagram, we decided to have specialization from our Main Problem and have disjoint-subclasses: GameInquiry, Malicious Activity, Hardware Inquiry, Account Issues, Troubleshooting, Service, and Billing. Each individual subclass has enough attributes to justify its existence.

Overall, in our UML diagram, we use generalization and specialization both in its most efficient and logical way to establish condensity and ease of understandability.



1.2.4 UML Diagram


Call
PK generatedID
recording
totalCallDuration
callDescription
keyPointOfConcern
subPointsOfConcern *
actionTaken
OXYsquare Account
PK accountID
emailAdd
phoneNumber
password
name
address
region
has
has
Game
PK transactionID
gameName
softwareVersion
distributionType
gameID
purchaseDate
purchasePrice
answered by
Date
startTime
endTime
initates
Date
startTime
endTime
addresses
references
Agent
PK agentID
sDate
eDate
hoursWorked
averageHandleTime
agentTier
Ticket
PK ticketID
totalNumberOfCalls
agentName *
accumulatedCallDuration
sDate
eDate
sTime
eTime
addresses
Hardware
PK transactionID
deviceName
deviceModelID
dateWarranty
macAddress
userSerialNumber
purchaseDate
dateLastUsed
transfers
call
creates
sDate
sTime
describes
Main Problem
PK problemID
focusAreaID
problemDescription
Game Inquiry
PK gameID
thirdPartyPublisher
inquiryType
Troubleshooting
internalPartAssociated
troubleshootingType
troubleSolution
errorCode
Malicious Activity
activityType
securityIssue
d
Service
warrantyAssociated
serviceType
deviceIssue
servicedRegion
shipToAddress
Hardware Inquiry
PK deviceModelID
internalPartInQuestion
inquiryType
Account Issues
accountType
accountStatus
accountIssue
Billing
refundType
billingIssue
creditCardAssociated
refundTypeRequested

Phase 2: From E-R Model to Relational Model

2 Conceptual Database and Logical Database

Section 2.1 will discuss about the relation between the E-R Model and the Relational model. It will also include why they are being used and what they have to offer.

Section 2.2 will show the OXYsquare conversion from E-R to the Relational model. This includes converting entities, relation types to relations and the constraints for each given relation.

2.1 E-R Model and Relational Model

The Entity Relationship Model and the Relational Models are both design tools used by software architects. Both models serve a different purpose and are better models for different reasons. Entity Relationship Model is more conceptual and easily-understood by non-technical people. Relational model is more logical in design, well translated for the DBMS language. In this section, we will address the details of each model and discuss key differences and similarities.

2.1.1 Description of E-R model and Relational Model

E-R Model History

The early models of databases were called Bachman Diagrams, his model was one of the earliest forms of Data Modeling. Charles Bachman introduced the concept of “Entities” and was considered as the first data structure diagram. Clive Finkelstein and James Martin were also key contributors in the progress of Enterprise Information Architecture.

During the 1970’s an Electrical Engineer named Peter Pin-Shan was inspired by Bachman’s Diagrams. He officially formalized and popularized the E-R Model and Diagram. It was so popular that it became an ANSI standard.

The E-R Model or Entity Relationship Model is a conceptual model that has no logical and expressions in the diagram. Its purpose is to have a high level interpretation of the database being requested. Entities are considered the objects that have specific details and will do all of the actions. Relationships are the actions linking 1 or more entities together. During this part of the of the database design, it is imperative for designers and the clients to have an understanding over how their database operates or functions. As this will lay the groundwork for all future advancements of the database.

Relational Model History

The Relational Model was created by Edgar F. Codd. He created it as a general model for data. Significant researches such as Prototype relational DBMS System R, INGRES, and Peterlee Relational Test Vehicle drived the popularity of the model.

The relational model focuses on the mathematical interpretation of the database. This means that Operations can now be performed on it as it follows conventional standards of discrete mathematics. This is proven by being able to compare specific keys between relations.

The relational model focuses creating a more mathematical perspective where in operations and logical expressions can be made to gather specific results. Its purpose is for creating and showing “tuples” which are called rows or entries. These consist of attributes which are locked in place by columns. This model is essentially the E-R Diagram but with foreign keys and candidate keys present, so that operations can be performed.

2.1.2 Comparison of Two Different Models

There are many different uses for both the Entity Relational(ER) model and the Relational Model. Depending on the phase, one is more useful to use than the other.

For a quick overview of the comparison of two different models, we will first describe in detail the general uses of both models.In the beginning phases of planning out the database design, one should always start with the ER model. The ER model is a great tool developed for universal communication across all fields. The Software Architect who is designing the database system can easy talk to many domain experts and other non-technical workers with clear communication. One does not need a technical degree to understand the Entity Relational Diagram because it removes all the technicalities from it.

The Relational Database Model is also great tool for database systems designers, because it is the only model that has DBMS support for technicians.

2.2 Conversion of Conceptual Database Model to Logical Database Model

Although the E-R model and Relational model may be different by nature, they still share similarities in way that we could convert from one model to the other. We will be describing how to convert Entity types and Relation types and then finally the constraints to make sure

that the database operates efficiently. Entity Relational Model is a conceptual more abstract database design tool to allow for technical people and non-technical people to fully understand the inner-working of the database. The Relational Data Model is as contrast, more mathematical and has more theoretical principles.

2.2.1 Converting Entity Types to Relations

The general guideline from converting the E-R model to the Relational Model is that all entity types from the E-R Model must be converted into a set of relation schemas. In general a relational model is a database of the relations, where each row is a tuple: containing values of a corresponding attribute. A relational schema will have attributes associated like: STUDENT(Name, SSN, Age, GPA, Major). These attributes must be single value or atomic values. These instances are tuples with values inputted for a corresponding attribute ie. 3.50 for 'GPA' attribute. These relation schemas will then have a list of attributes which include the foreign key of those relations the entity has with others.

Converting Strong Entities into Relations

A strong entity E will be converted into a relation schema R, which is notated as R(A_1, A_2...An). The attributes of the relational schema will be the same as the entity's attributes, only they will need to be simple and single attributes. These values must be atomic. When dealing with composite attributes we must break them down to single valued ones. For example, a string of address must be broken into single elements of street, city, state and zip code. The number of attributes associated to a relation is the degree of a relation. Key attributes for strong relation schemas are keys that are unique itself. The primary keys of a relation schema are used to identify the relation itself; there can only be one primary key to identify a tuple. The candidate keys are additional key attributes that are unique to the relation.

Converting Weak Entities into Relations

Weak entities are converted into relations R, just like previously described in strong entities. The schema format is the same as well, where all attributes must be single and simple-value attributes, no composite attributes are allowed. Unique valued attributes, as similarly described, will be candidate bey with a single candidate key assigned as a primary key used to identify the tuple.

Mapping of Simple and Composite Attributes

Mapping simple and composite attributes are different processes. In short, all attributes must be simple attributes. Composite attributes as noted previously, need to be broken down to its simplest, single-valued form to be used as separate attributes individually, whereas the simple attributes in a relation schema are already in its most simplest form as a single attribute.

Mapping of Single and Multi-Value Attributes

The mapping of both single and multi-value attributes are different processes as well. The bottom line is that an attribute in the relation must be single, not multi-valued in relation R. These single attributes of an entity type will just be an attribute of R. When we have a multi-valued attributes will need to to be a new separate relation RA and these attributes of RA will be multi-valued and the primary key of the R relation will be a foreign key. We will combine the multi value attribute with the foreign key to make the primary key of RA.

2.2.2 Converting Relationship Types to Relations

In a relational model, there are only relations, unlike ER Models which have entities. The relational model focuses on the relationships themselves. The relational model will have the concept of the relation so these types are represented through schemas themselves. We will now talk about the different types of relationships and how we can map these types:

- cardinality constraints in one to one, one to many, and many to many relationships
- The different ways of representing “IsA” and “HasA” concepts in specialization and generalizations
- More than 2 entity types
- Recursion relations
- Relationships with other relationship types
- And lastly, union types

Mapping of Relationship Types with a 1:1 Cardinality Constraint

When a relationship type has entity types like A and B with a 1:1 cardinality constraint A and B which will be converted into R_A and R_B, where R_A will have one instance with R_B. The three methods to use are:

1. **Foreign Key Approach:** This is the most useful approach of the three for 1:1 Cardinality Constraints. In the relations R_A and R_B, for the Foreign Key Approach, we will need to add the foreign key in R_A the primary key of R_B, to indicate the relations of between the two relations. The best way is to use total participation in R_A and R_B.

2. **Merged Relation Approach:** This is an alternative to the foreign key approach, the merge relation approach will merge the two relations: R_A and R_B together. The one constraint in this approach is that both relations R_A and R_B must be total participation as well, allowing the tables to have the same number of tuples.
3. **Cross Reference Approach:** For the Cross Reference Approach, there is a new relation R_c where R_C creates the relationship relation itself which will cross-reference the primary keys of both relations: R_A and R_B and all the attributes of the relationship type itself. R's primary key will then one of the foreign keys in both R_A and R_B.

Advantages and Disadvantages: These methods are different and have advantages and disadvantages to each. Below we will further discuss the advantages and disadvantages of both. Furthermore, we will discuss when one will use one approach over the other.

1. **Foreign Key Approach:** This approach as noted above, is the most useful approach of the three for 1:1 Cardinality. This approach is best used for total participation relations so it can avoid extra null values. Foreign key approach is simple and lowers the need for using the join operation during queries.
2. **Merged Relation Approach:** As noted in class, this approach is not useful much at all. This is because if we were to merge two relations into one relation, then why were they separate in the ER model in the first place? It will deem the need for two entities to be separate completely useless. It is unlikely one would need to use this.
3. **Cross Reference Approach:** This approach is best used when the entity types are not total participation. This will require more joins when querying compared to the other approaches.

Mapping of 1:N Relationship Types

When there are entities with a 1:N cardinality, we will need to convert it into relations R_A and R_B where R_A has multiple instances of R_B but R_B will have one instance of R_A thus 1:N. Two methods for this are:

1. **Foreign Key Approach:** This approach is the same approach as previously stated in 1:1 cardinality. There is however one difference: Since R_B is on the N-side, we need to add the primary key R_A into R_B as a foreign key. We do this because tuples on the "N-side" will be related to the "1-side" as a unique relationship.
2. **Cross Reference Approach:** This is the same approach as previously stated in the 1:1 cardinality section. There is however one difference: Since R_B is on the N-side, we need to add the primary key R_A into R_B as a foreign key.

Advantages and Disadvantages:

The advantages and disadvantages are the same as previously described in 1:1 Cardinality. The best method for these approaches is the Foreign Key Approach as previously noted.

Mapping of M:N Relationship Types

When there are entities with a M:N cardinality, we will need to convert it into relations R_A and R_B where R_A is related to multiple instances of R_B and vice versa. The method for this is:

1. **Cross Reference Approach:** This is the only possible method for conversion. A relationship relation R needs to be created for the M:N relationship. It was created to represent that relationship type will contain the primary keys of R_A into R_B as foreign keys. The primary key for R is both keys combined.

Mapping of Superclasses and Subclasses for the "IsA" Relationship

There are different ways for mapping subclasses. The subclasses are the IsA relationship to superclasses. Two main options are: mapping a specialization into a single table or into multiple tables. Attrs(R) is notation for the attributes of the relation R and PK(R) is the notation for primary key R.

1. **Multiple Relations - Superclass and Subclass:** When we make a relation L, it will have Attrs(R) = {k, a_1, a_2 … a_n} and PK(L) = k. Relation R_{super} is created as relation for superclass and R_{sub} is created as relation for subclass. The superclass relation will have the attributes of the superclass entity. The subclass relation will have the attributes of subclass entity and the primary key of the superclass as a foreign key.

2. **Multiple Relations - Subclass only:** Subclass entities have their own relations as well. Relations from these subclass entities will have the attributes of the subclass entities themselves and the superclass attributes as well. If the specialization is overlapped, then the subclass entity can be duplicated in several relations.

3. **Single Relation with one type attribute:** One relation R is made by the union of the attributes of the generalizations and specializations. So using the super class and all the subclasses, we will need to combine all the attributes together fro each.

Advantages and Disadvantages:

These approaches will have their advantages and disadvantages in each. Below will be a breakdown of all the advantages and disadvantages of every method.

1. **Multiple Relations - Superclass and Subclass:** This is a very useful method because it encompasses all types of sub/superclass relationship. This method works for: disjoint "IsA", overlapping "HasA", total and partial participation. As a caveat, however, this method will require a creation of a separate superclass relation, thus needing to use more join operations when querying.

2. **Multiple Relations - Subclass only:** This method will only work for total participation types which will be a disadvantage. The advantage is that it does require less join operations.

3. **Single Relation with one type attribute:** This method will have the least usage for join operations, however, this will have issues with wasted space, because for all attributes where the superclass and entity does not share will show a NULL value. Thus, this is only useful when many of the attributes are similar.

Mapping of Superclasses and Subclasses for the "HasA" Relationship

The "HasA" relation is when the entity types are overlapping subclasses in the superclass entity type. Overlapping means that the subclasses are not disjoint and they are not separated. Two methods that represent these relationships are:

1. **Multiple Relations - Superclass and Subclass:** This is the same as described in the "IsA" superclass/subclass relationship. When we make a relation L, it will have Attrs(R) = {k, a_1, a_2 … a_n} and PK(L) = k. Relation R_{super} is created as relation for superclass and R_{sub} is created as relation for subclass. The superclass relation will have the attributes of the superclass entity. The subclass relation will have the attributes of subclass entity and the primary key of the superclass as a foreign key.

2. **Single Relation with Multiple Type Attributes:** One relation in this method will contain the union of all the attributes from the superclass and all subclasses. In addition it will contain a boolean attribute for every possible subclass showing whether or not the tuple belongs to a subclass.

Advantages and Disadvantages:
These approaches will have their advantages and disadvantages in each. Below will be a breakdown of all the advantages and disadvantages of every method.

1. **Multiple Relations Superclass and Subclass:** This is a very useful method because it encompasses all types of sub/superclass relationship. This method works for: disjoint "IsA", overlapping "HasA", total and partial participation. As a caveat, however, this method will require a creation of a separate superclass relation, thus needing to use more join operations when querying.

2. **Single Relation with Multiple Type Attributes:** This method will require the lowest number of the join operations which will allow for less computational time for querying. However, due to the likelihood that many of these attributes will not be similar, there will be large amounts of NULL values for attributes that do not belong to the relation thus large waste of storage space will be a great disadvantage.

Mapping of Relationships to Other Relationship Types

When using this method, the primary key of the relations will be the foreign key for the one of two methods: Foreign Key Approach or Cross Reference Approach. These methods will be chosen depending on the cardinality of the relationships.

Mapping of Recursive Relationships

When there are relations R that relate to one another, it will be a recursive relationship. These are two methods that will represent this relationship:

1. **Foreign Key Approach:** This approach is the same as described above, however this relation R will contain a foreign key attribute that is the primary key of the same relation of itself.
2. **Cross Reference Approach:** When we use this approach, there is a new relation created: $R_{recursive}$ that is used to represent this recursive relationship described. This will contain two foreign keys that are primary key of R. This combination of the foreign key of R will be the primary key of $R_{recursive}$.

Advantages and Disadvantages:

These approaches will have their advantages and disadvantages in each. Below will be a breakdown of all the advantages and disadvantages of every method.

1. **Foreign Key Approach:** the foreign key approach will have less join operations so it will be computationally optimal however if a relation tuple does not have a particular value for the attribute it will produce NULL values resulting in wasted storage space.
2. **Cross Reference Approach:** This will not have issues with NULL value descriptions however we will need to have more join operations for this approach.

Mapping of Relationships Between More than Two Entity Types
When a relationship has more than two entity types we need to create a new relation R. We must convert all entity types into relations and use their primary keys as the foreign keys in relation R. The combination of the foreign keys of N-side entities will be created as a primary key of relation R. We will also include all simple single-valued attributes.

Mapping of Union Types (Categories)
A union type (categories) will be established when we have specializations that belong to more than one generalization. When a subclass belongs to multiple superclasses, these superclasses will have a surrogate key attribute assigned. This way relation tuples that belong to particular superclass will be identified by the surrogate key associated.

2.2.3 Database Constraints

Database constraints provide order and stability for queries. Database constraints preserve data integrity, assuring that the data stays accurate and meaningful. Using constraints, we will be able to control certain inputs of data to check its accuracy. These constraints are either related to our business rules/protocols or the database schema itself. Constraints are important for methods: deleting, inserting, updating due to the many internal changes these methods will create in our data. The constraints we will discuss in this section will be as follows:

- Domain Constraints
- The Entity Constraint
- Primary Key and Unique Key Constraints
- Referential Constraints
- Check Constraints and Business Rules

Domain Constraints
These constraints will keep the tuple's values within a specified domain in the relation schema. These constraints will have a restricted value for the attributes and subset values. Data types: characters, booleans, fixed-length strings, and variable-length are associated domains for attributes. In addition, date, time, timestamp, and money, or other special data types can also be used as constraints. Using these constraints the DBMS will be able to deny insertions or updates that violate our domain constraints, assisting with data integrity of the system.

The Entity Constraint
This entity constraint is a constraint that will ensure that non-nullable fields are filled in. As an example, all candidate and primary keys must be identified. Due to this, an entity constraint will be applied to ensure that this rule is being followed. Using these constraints the DBMS will be

able to deny insertions or updates that violate our entity constraints, assisting with data integrity of the system.

Primary Key and Unique Key Constraints

Primary keys are used to uniquely identify tuples. Therefore, primary keys are extremely significant in a DBMS. Due to this, we have primary key and unique key constraints. Since these keys are used to identify a specific tuple, our DBMS will be able to control the primary keys and confirm that only unique keys are entered into our DBMS.

Referential Constraints

This constraint will check to see if the foreign key in the relation is referencing an existing tuple from the relation. We do not want to have a foreign key to not be associated to a relation. This is important because this is a constraint that will protect the integrity of our relations, which is the entire focus of a relational database design. When we insert, delete, or update, all these operations will potentially risk data integrity if this constraint is not implemented. We do not want to delete a "Call" tuple and not remove/update all the associated "Ticket" or "Main Problem" related to the call.

1. **Restrict:** Reject the operation of a removal/deletion a tuple or foreign key reference.
2. **Cascade:** Delete all tuples that are associated with deleted tuple
3. **Set Default:** Set foreign key values for other tuples that reference the deleted tuple as NULL

Check Constraints and Business Rules

As noted, these are constraints that describe how we can ensure data integrity within our database. These rules can be either business specific or database specific.

2.3. E-R Model to a Relational/Logical Database

The extensive descriptions and rules that we have covered can finally be implemented in our database. The entity types, relationship types and constraints will be converted into the relational schema. Sample tuples from the relational schema will also be placed after the relations.

2.3.1 Relation Schema

Below are the relation schemas in the database. We will go into detail the attributes, entities, and relationships in the ER model following this format:

Keys	PRIMARY	FOREIGN

Relation Schema: OXYsquare Account

OXYsquare Account (**generatedID, transactionGameID, transactionHardID,** accountID, emailAdd, phoneNumber, password, name, address, region)

Attribute	Domain	Description
generatedID	Integer	Unique Call ID
transactionGameID	Integer	Unique purchase game transaction ID
transactionHardID	Integer	Unique purchase hardware transaction ID
accountID	Integer	Unique Account ID
emailAdd	String	Email Address of the User
phoneNumber	Integer	Phone Number of the User
password	String	Password of the User
name	String	Full Name of the User
address	String	Address of the User
region	String	Region ie. North America

Candidate Keys: transactionHardID, accountID, emailAddr

Primary Key/Entity Integrity Constraint: accountID, it cannot be NULL

Uniqueness Constraint: accountID and emailAdd must be unique, emailAdd and accountID cannot be NULL

Referential Integrity Constraint: generatedID is the foreign key for call, transactionGameID is for foreign key for game, transactionHardID is the foreign key for hardware

Business Constraint: There are none

Derivation from Entity and Relationship Types: Derived from the OXYsquare account entity type.
Represents a 1:N relationship with Call using the foreign key approach since participation of the account is total.
Represents a 1:N relationship with Hardware using the foreign key approach since participation of the account is total.
Represents a 1:N relationship with Game using the foreign key approach since participation of the account is total.

Call

Call (**accountID, agentID,** generatedID, recording, totalCallDuration, callDescription, keyPointOfConcern, subPointsOfConcern, actionTaken)

Attribute	Domain	Description
accountID	Integer	Unique Account ID
agentID	Integer	Unique Agent ID
generatedID	Integer	Unique Call ID
recording	String	String File Name of the Recording file.
totalCallDuration	Integer	The total duration of the call
callDescription	String	A short description of the call
keyPointOfConcern	String	The main point or classification fo the call.
subPointofConcern	String	An array of Keywords detailing the description of the problem
actionTaken	Boolean	0 for a drop and no ticket created, 1 for new call and

		ticket created.

Candidate Keys: accountID, agentID, generatedID, recording

Primary Key/Entity Integrity Constraint: generalID, must be unique cannot be NULL.

Uniqueness Constraint: generalID and recording must be unique

Referential Integrity Constraint: accountID is the foreign key for OXYsquare account, agentID is for foreign key for agent

Business Constraint: None

Derivation from Entity and Relationship Types: Derived from the Call entity type.
Represents a N:1 relationship with OXYsquareaccount using the foreign key approach since participation of the call is total.
Represents a 1:N relationship with agent using the foreign key approach since participation of the call is total.

Ticket

Ticket (**generatedID, agentID, transactionGameID, transactionHardID, problemID,** ticketID, totalNumberOfCalls, agentName, accumulatedCallDuration, sDate, eDate, sTime, eTime)

Attribute	Domain	Description
generatedID	Integer	Unique Call ID
agentID	Integer	Unique Agent ID
transactionGameID	Integer	Unique purchase game transaction ID
transactionHardID	Integer	Unique purchase hardware transaction ID

problemID	Integer	Unique Main Problem ID
ticketID	Integer	Unique Ticket ID
totalNumberOfCalls	Integer	Amount of calls done for the ticket
agentName	String	An array of agent names that handled the ticket.
accumulatedCallDuration	Integer	An integer that adds up all the calls duration to one number.
sDate	Date	Starting Date of the Ticket
eDate	Date	Closing Date of the Ticket
sTime	Time	Starting Time of the Ticket
eTime	Time	Ending Time of the Ticket

Candidate Keys: generatedID, agentID, transactionGameID, transactionHardID, problemID, ticketID

Primary Key/Entity Integrity Constraint: ticketID must be unique, cannot be NULL

Uniqueness Constraint: The ticketID must be unique and cannot be NULL

Referential Integrity Constraint: generatedID is the foreign key for call, agentID is the foreign key for agent, transactionGameID is the foreign key for game, transactionHardID is the foreign key for hardware, problemID is the foreign key for problem,

Business Constraint: eTime must not be before sTime. eDate must not be before sDate. Total number of calls must be greater than 1.

Derivation from Entity and Relationship Types:

Derived from the Ticket entity type.
Represents a 1:N relationship with call using the foreign key approach since participation of the ticket is total.

Represents a 1:N relationship with agent using the foreign key approach since participation of the ticket is total.
Represents a 1:1 relationship with game using the foreign key approach since participation of the ticket is partial.
Represents a 1:1 relationship with hardware using the foreign key approach since participation of the ticket is partial.
Represents a 1:1 relationship with problem using the foreign key approach since participation of the ticket is total.

Game

Game (**accountID, ticketID,** transactionGameID, gameName, softwareVersion, distributionType, gameID, purchaseDate, purchasePrice, gameMSRP, gamePublisher)

Attribute	Domain	Description
accountID	Integer	Unique Account ID
Ticket	Integer	Unique Ticket ID
transactionGameID	Integer	Unique purchase game transaction ID
gameName	String	The name of the game
softwareVersion	Integer	The softwareVersion of the copy of the game.
distributionType	String	Either digital or physical copy of the game.
gameID	Integer	The unique game ID
purchaseDate	Date	The date when the user purchased the game.
purchasePrice	Float	The amount the user paid for the game.
gameMSRP	Float	The standard retail price of the game
gamePublisher	String	The publisher of the game.

Candidate Keys: accountID, ticketID, gameID and transactionGameID are unique and cannot be null

Primary Key/Entity Integrity Constraint: transactionGameID is unique and cannot be NULL

Uniqueness Constraint: gameID and transactionGameID are both unique

Referential Integrity Constraint: accountID is a foreign key for account, ticketID is a foreign key for ticket

Business Constraint: purchasePrice cannot be greater than gameMSRP

Derivation from Entity and Relationship Types:
Derived from the Game entity type.
Represents a N:1 relationship with account using the foreign key approach since participation of the game is total.
Represents a 1:1 relationship with ticket using the foreign key approach since participation of the game is partial.

Hardware

Hardware (**accountID, ticketID,** transactionHardwareID, deviceName, deviceModelID, dateWarranty, macAddress, userSerialNumber, purchaseDate, dateLastUsed, hardwareMSRP, purchasePrice)

Attribute	Domain	Description
accountID	Integer	Unique Account ID
Ticket	Integer	Unique Ticket ID
transactionHardID	Integer	Unique purchase hardware transaction ID
deviceName	String	The name of the hardware
deviceModelID	String	The string identifier of the hardware, this changes due to revisions.

dateWarranty	Date	The ending date of the warranty.
macAddress	String	The unique macc address identifier of the hardware
userSerialNumber	Integer	The unique serial number that is given to every hardware created.
purchaseDate	Date	The date when the hardware was purchased.
hardwareMSRP	Float	The standard retail price of the hardware
purchasePrice	Float	The amount the user paid for the hardware.

Candidate Keys: accountID, ticketID, transactionHardID, macaddress, and userSerialNumber are candidate key cannot be null

Primary Key/Entity Integrity Constraint: transactionHardID are primary keys cannot be null

Uniqueness Constraint: transactionHardID, macaddress, and userSerialNumber must be unique

Referential Integrity Constraint: accountID is the foreign key for account, ticketID is the foreign key ticket

Business Constraint: purchase price cannot be higher than hardware MSRP, the purchase date must be smaller the date last used

Derivation from Entity and Relationship Types:
Derived from the Hardware entity type.
Represents a N:1 relationship with account using the foreign key approach since participation of the hardware is total.
Represents a 1:1 relationship with ticket using the foreign key approach since participation of the hardware is partial.

<u>Agent</u>

Agent (**generatedID, ticketID,** agentID, sDate, eDate, hoursWorked, averageHandleTime, agentTier, agentName)

Attribute	Domain	Description
generatedID	Integer	Unique Call ID
ticketID	Integer	Unique Ticket ID
agentID	Integer	Unique agent number identifier
sDate	Date	Starting date of the agent
eDate	Date	The date of the last day of employent.
hoursWorked	Integer	Number of hours worked pooled from taking calls and handling tickets
averageHandleTime	Integer	Number derived from the average of calls handled by the agent
agentTier	Integer	The level of the agents tier, a tier represents what they could handle. 1,2,3
agentName	String	The name of the agent.

Candidate Keys: generatedID, ticketID, agentID must not be null

Primary Key/Entity Integrity Constraint: agentID cannot be null, must be unique

Uniqueness Constraint: agentID is unique

Referential Integrity Constraint: generatedID is foreign key for Call, ticketID is foreign key for ticket

Business Constraint: eDate must be greater than sDate

Derivation from Entity and Relationship Types:
Derived from the Agent entity type.
Represents a 1:1 relationship with call using the foreign key approach since participation of the agent is total.
Represents a 1:1 relationship with ticket using the foreign key approach since participation of the agent is total.

Main Problem

Main Problem (**ticketID, solutionID,** problemID, focusAreaID, problemDescription)

Attribute	Domain	Description
ticketID	Integer	Unique Ticket ID
solutionID	Integer	Unique solution number ID
problemID	Integer	The unique number to identify and locate the specific problem.
focusAreaID	Integer	The integer to classify different problem types in the database, then correlating them to strings. Such as 0010 = Billing-Credit Card, 00011 = Billing-Cash, 0020 = Account Issues-Password etc.
problemDescription	String	The text field to place a general description of the problem or to note specfiic details.

Candidate Keys: ticketID, solutionID, problemID is unique

Primary Key/Entity Integrity Constraint: problemID, cannot be null

Uniqueness Constraint: problemID cannot be null

Referential Integrity Constraint: ticketID is the foreign key for ticket, solutionID is the foreign key for solution

Business Constraint: none

Derivation from Entity and Relationship Types:
Derived from the Problem entity type.
Represents a 1:1 relationship with solution using the foreign key approach since participation of the problem is total.
Represents a 1:1 relationship with ticket using the foreign key approach since participation of the problem is total.

Solution

Solution (**problemID,** solutionID, solutionDescription)

Attribute	Domain	Description
problemID	Integer	The unique number to identify and locate the specific problem.
solutionID	Integer	Unique solution number ID
solutionDescription	String	The string that contains specific instructions on how to solve the problem.

Candidate Keys: problemID, solutionID, must be unique

Primary Key/Entity Integrity Constraint: solutionID, is not null

Uniqueness Constraint: solutionID must be unique

Referential Integrity Constraint: problemID is foreign key for problem

Business Constraint: None

Derivation from Entity and Relationship Types:
Derived from the Solution entity type.
Represents a 1:1 relationship with solution using the foreign key approach since participation of the problem is total.

Game Inquiry

Game Inquiry (**ticketID, solutionID,** problemID, gameID, thirdPartyPublisher, inquiryType)

Attribute	Domain	Description
ticketID	Integer	Unique Ticket ID
solutionID	Integer	Unique solution number ID
problemID	Integer	The unique number to identify and locate the specific problem.
thirdPartyPublisher	String	The string that contains information of the non OXYsquare game publisher
inquirtyType	String	The string that can labelled as release date, general, price, etc.

Candidate Keys: ticketID, solutionID, problemID is unique

Primary Key/Entity Integrity Constraint: problemID, not nullable

Uniqueness Constraint: problem ID must be unique

Referential Integrity Constraint: ticketID is foreign key for ticket, solutionID is foreign key for solution

Business Constraint: None

Derivation from Entity and Relationship Types:
Derived from the Game Inquiry entity type.
Represents a 1:1 relationship with solution using the foreign key approach since participation of the game inquiry is total.
Represents a 1:1 relationship with ticket using the foreign key approach since participation of the game inquiry is total.

Malicious Activity

Malicious Activity (**ticketID, solutionID,** problemID, activityType, securityIssue)

Attribute	Domain	Description
ticketID	Integer	Unique Ticket ID
solutionID	Integer	Unique solution number ID
problemID	Integer	The unique number to identify and locate the specific problem.
activityType	String	Strings of malicious activities, this can be griefing, rude behavior, and other game related infractions.
securityIssue	String	The string that can be labeled as fraud, hijacking, hacking, etc.

Candidate Keys: ticketID, solutionID, problemID is unique

Primary Key/Entity Integrity Constraint: problemID, not nullable

Uniqueness Constraint: problem ID must be unique

Referential Integrity Constraint: ticketID is foreign key for ticket, solutionID is foreign key for solution

Business Constraint: None

Derivation from Entity and Relationship Types:
Derived from the Malicious Activity entity type.
Represents a 1:1 relationship with solution using the foreign key approach since participation of the malicious activity is total.
Represents a 1:1 relationship with ticket using the foreign key approach since participation of the malicious activity is total.

Hardware Inquiry

Hardware Inquiry (**ticketID, solutionID,** problemID, deviceModelID, internalPartInQuestion, inquiryType)

Attribute	Domain	Description
ticketID	Integer	Unique Ticket ID
solutionID	Integer	Unique solution number ID
problemID	Integer	The unique number to identify and locate the specific problem.
internalPartInQuestion	String	The string that identifies the specific part with the issue.
InquiryType	String	String that identifies the inquiry type such as release date, general information, specficiations, etc.

Candidate Keys: ticketID, solutionID, problemID is unique

Primary Key/Entity Integrity Constraint: problemID, not nullable

Uniqueness Constraint: problem ID must be unique

Referential Integrity Constraint: ticketID is foreign key for ticket, solutionID is foreign key for solution

Business Constraint: None

Derivation from Entity and Relationship Types:
Derived from the Hardware Inquiry entity type.
Represents a 1:1 relationship with solution using the foreign key approach since participation of the Hardware Inquiry is total.
Represents a 1:1 relationship with ticket using the foreign key approach since participation of the Hardware Inquiry is total.

Account Issues

Account Issues (**ticketID, solutionID,** problemID, accountType, accountStatus, accountIssue)

Attribute	Domain	Description
ticketID	Integer	Unique Ticket ID
solutionID	Integer	Unique solution number ID
problemID	Integer	The unique number to identify and locate the specific problem.
accountType	String	The string to determine the account type, master or sub account.
accountStatus	String	String that states the account status such as active, suspended, banned, etc.
accountIssue	String	The general string description for the issue of the account.

Candidate Keys: ticketID, solutionID, problemID is unique

Primary Key/Entity Integrity Constraint: problemID, not nullable

Uniqueness Constraint: problem ID must be unique

Referential Integrity Constraint: ticketID is foreign key for ticket, solutionID is foreign key for solution

Business Constraint: None

Derivation from Entity and Relationship Types:
Derived from the account issues entity type.
Represents a 1:1 relationship with solution using the foreign key approach since participation of the account issues is total.
Represents a 1:1 relationship with ticket using the foreign key approach since participation of the account issues is total.

Billing

Billing (**ticketID, solutionID,** problemID, refundType, billingIssue, creditCardAssociated, refundTypeRequested)

Attribute	Domain	Description
ticketID	Integer	Unique Ticket ID
solutionID	Integer	Unique solution number ID
problemID	Integer	The unique number to identify and locate the specific problem.
refundType	String	The string that determined the refund type such as store credit, credit card, or paypal.
billingIssue	String	The string that describes why

		the refund is being requested by the customer.
creditCardAssociated	Integer	The integer for the credit card in question.
refundTypeRequested	String	The refund type that the user has requested.

Candidate Keys: ticketID, solutionID, problemID is unique

Primary Key/Entity Integrity Constraint: problemID, not nullable

Uniqueness Constraint: problem ID must be unique

Referential Integrity Constraint: ticketID is foreign key for ticket, solutionID is foreign key for solution

Business Constraint: None

Derivation from Entity and Relationship Types:
Derived from the Billing entity type.
Represents a 1:1 relationship with solution using the foreign key approach since participation of the billing is total.
Represents a 1:1 relationship with ticket using the foreign key approach since participation of the billing is total.

Hardware Repair

Hardware Repair (**ticketID, solutionID,** warrantyAssociated, serviceType, deviceIssue, servicedRegion, shipToAddress)

Attribute	Domain	Description
ticketID	Integer	Unique Ticket ID
solutionID	Integer	Unique solution number ID

problemID	Integer	The unique number to identify and locate the specific problem.
warrantyAssociated	String	The warranty type of the device such as OEM warranty, extended warranty, expired warranty.
serviceType	String	The string for the service type such as Hard Drive Replacement, Screen Replacement, etc.
deviceIssue	String	The specific issues with the device.
serviedRegion	String	The region where the device is being serviced
shipToAddress	String	The string for the address requested the user to ship the device to.

Candidate Keys: ticketID, solutionID, problemID is unique

Primary Key/Entity Integrity Constraint: problemID, not nullable

Uniqueness Constraint: problem ID must be unique

Referential Integrity Constraint: ticketID is foreign key for ticket, solutionID is foreign key for solution

Business Constraint: None

Derivation from Entity and Relationship Types:
Derived from the Hardware Repair entity type.
Represents a 1:1 relationship with solution using the foreign key approach since participation of the malicious activity is total.
Represents a 1:1 relationship with ticket using the foreign key approach since participation of the malicious activity is total.

Troubleshooting

Troubleshooting (**ticketID, solutionID,** problemID, internalPartAssociated, troubleshootingType, errorCode)

Attribute	Domain	Description
ticketID	Integer	Unique Ticket ID
solutionID	Integer	Unique solution number ID
problemID	Integer	The unique number to identify and locate the specific problem.
internalPartASsociated	String	The specific internal part in question.
troubleShootingType	String	A string used to classify the troubleshooting query, such as Pairing, installation, performance, etc.
errorCode	Integer	The integer that shows the error code for the hardware or software.

Candidate Keys: ticketID, solutionID, problemID is unique

Primary Key/Entity Integrity Constraint: problemID, not nullable

Uniqueness Constraint: problem ID must be unique

Referential Integrity Constraint: ticketID is foreign key for ticket, solutionID is foreign key for solution

Business Constraint: None

Derivation from Entity and Relationship Types:
Derived from the Troubleshooting entity type.
Represents a 1:1 relationship with solution using the foreign key approach since participation of the Troubleshooting is total.
Represents a 1:1 relationship with ticket using the foreign key approach since participation of the troubleshooting is total.

Creates

Creates (**agentID, ticketID,** sDate, sTime)

Attribute	Domain	Description
agentID	Integer	Unique agent number identifier
ticketID	Integer	Unique Ticket ID
sDate	Date	The date when the ticket was created
sTime	Date	The time when the ticket was created

Candidate Keys: agentID and ticketID

Primary Key/Entity Integrity Constraint: The combined agentID and ticketID is the primary key, must be unique and no null

Referential Integrity Constraint: agentID is foreign key for agent and ticketID is foreign key for ticket

Business Constraint: None

Derivation from Entity and Relationship Types:

Derived from the attributes from the Ticket and Agent. This is a foreign key approach from Ticket and Agent.

Answered By

Answered By (**generatedID, agentID,** date, s_Time, e_Time)

Attribute	Domain	Description
generatedID	Integer	Unique agent number identifier
agentID	Integer	Unique Ticket ID
date	Date	The date when the call was answered by the agent.
s_Time	Time	The time when the call was answered.
e_Time	Time	The time when the call ended.

Candidate Keys: agentID and generatedID

Primary Key/Entity Integrity Constraint: The combined agentID and generatedID is the primary key, must be unique and no null

Referential Integrity Constraint: agentID is foreign key for agent and generatedID is foreign key for call

Business Constraint: sTime must be smaller than eTime

Derivation from Entity and Relationship Types:
Derived from the attributes from the Call and Agent. This is a foreign key approach from Call and Agent.

Initiates

Initiates (**accountID, generatedID,** date, startTime, endTime)

Attribute	Domain	Description
accountID	Integer	Unique Account ID
generatedID	Integer	Unique Call ID
date	Date	The date when the ticket was created
startTime	Time	The time when the call started.
endTime	Time	The time when the call ended.

Candidate Keys: agentID and generatedID

Primary Key/Entity Integrity Constraint: The combined agentID and generatedID is the primary key, must be unique and no null

Referential Integrity Constraint: agentID is foreign key for agent and generatedID is foreign key for call

Business Constraint: startTime must be smaller than endTime

Derivation from Entity and Relationship Types:
Derived from the attributes from the Call and Agent. This is a foreign key approach from Call and Agent.

2.3.2 Sample Data of Relation

These are sample generated data to better understand the tuples.

Agent

generatedID	ticketID	agentID	sDate	eDate	hoursWorked	averageHandleTIme	agentTier	agentName
C47682	T7639573	A37	10/1/2017	NULL	934	14.0807939	2	Alfy Bonnor
C47732	T7639667	A43	1/13/2018	NULL	1162	10.6168358	2	Odille Buttfield
C47792	T7639727	A50	10/24/2017	NULL	791	13.6552252	3	Tamra Willeman
C47878	T7639814	A58	7/7/2017	NULL	1180	11.4398811	1	Thorvald Henric
C47976	T7639883	A64	2/21/2018	NULL	1984	14.8261288	2	Miller Cousens
C48051	T7639978	A70	7/13/2017	NULL	1006	13.4859603	1	Robin Brimman
C48121	T7640028	A78	2/17/2018	NULL	1728	14.5754188	2	Juan Merwe
C48191	T7640119	A86	2/2/2017	NULL	1547	11.5494483	1	Karlan Drogan
C48287	T7640203	A93	1/24/2018	NULL	1280	12.1219216	1	Chevalier Abbe
C48351	T7640257	A101	2/9/2017	NULL	784	14.3691203	1	Gonzalo Beller
C48405	T7640336	A107	4/10/2017	NULL	1423	13.2141611	2	Claudian Barlie
C48464	T7640431	A108	9/26/2017	NULL	783	12.5228549	2	Karissa Iacobo
C48533	T7640530	A114	2/18/2018	NULL	1711	10.9622598	2	Clarence Blampy

C48599	T7640625	A123	2/5/2017	NULL	906	13.8327836	3	Padriac McFadden
C48670	T7640721	A129	10/6/2017	NULL	878	13.9239484	2	Tobey Vickarman
C48747	T7640777	A130	8/1/2017	NULL	1260	13.7111873	3	Rocky Kunkler
C48808	T7640875	A138	12/17/2017	NULL	1484	10.6119832	2	Merci Tinner
C48898	T7640945	A145	12/13/2017	NULL	1388	11.3766322	3	Dewain Gabbitis
C48983	T7641004	A154	1/15/2017	NULL	945	13.9573538	3	Delbert Wadham
C49079	T7641081	A163	3/2/2018	NULL	1339	10.2485282	2	Ty Wilson

OXYsquare Account									
generatedID	**transactionGameID**	**transactionHardID**	**accountID**	**emailAdd**	**phoneNumber**	**password**	**name**	**address**	**region**
C80710	**G59299**	**H83273**	**O80736**	**aabels0@cbsnews.com**	**6638409109**	***TU910H***	**Andras Abels**	**1179 Northland Place Sacramento CA 95894**	**North America**
C80776	**G59399**	**H83339**	**O80829**	**cmaskell1@mediafire.com**	**3706540863**	***ET624O***	**Chilton Maskell**	**158 Muir Lane Port Saint Lucie FL 34985**	**North America**
C80864	**G59481**	**H83393**	**O80895**	**sbarabich2@icio.us**	**6869663127**	***MG829U***	**Simon Barabich**	**682 Arizona Pass San Diego CA 92110**	**North America**

C80955	G59565	H83482	O80987	joxberry3@microsoft.com	4679421609	*CT803V*	Jesse Oxberry	5 Alpine Avenue Cleveland OH 44185	North America
C81029	G59641	H83538	O81076	myannikov4@opensource.org	1524293083	*NJ780K*	Minnnie Yannikov	8137 Fairfield Park Pueblo CO 81005	North America
C81087	G59706	H83609	O81143	dbrandolini5@springer.com	5288843725	*MT859R*	Dareen Brandolini	38 Dixon Terrace Aurora CO 80044	North America
C81168	G59793	H83664	O81225	kbinner6@apache.org	5487854712	*IX460Y*	Karine Binner	83 Monterey Parkway Lubbock TX 79415	North America
C81250	G59864	H83718	O81325	vweedon7@paginegialle.it	2420042511	*EZ592L*	Valeda Weedon	0 1st Circle Baltimore MD 21275	North America
C81307	G59939	H83797	O81408	lcombe8@t.co	9024712527	*AV702E*	Liza Combe	0 Mccormick Road Colorado Springs CO 80905	North America
C81361	G59992	H83884	O81461	tbrodeau9@quantcast.com	3059215232	*HB110O*	Town Brodeau	7961 Evergreen Parkway Washington DC 20029	North America
C81432	G60043	H83969	O81558	bconneaua@state.tx.us	8851556911	*PJ987M*	Balduin Conneau	7 Waxwing Hill Nashville TN 37210	North America
C81497	G60094	H84058	O81648	afairhallb@acquirethisname.com	1192499491	*ZH560V*	Antin Fairhall	4 Dovetail Plaza Alhambra CA 91841	North America

C81564	G60145	H84116	O81712	nsticklerc@biglobe.ne.jp	2861636855	*YF241B*	Nealon Stickler	110 Northfield Parkway San Diego CA 92145	North America
C81646	G60214	H84198	O81778	troosond@spiegel.de	9166180380	*MA899B*	Tallia Rooson	46 Kim Place Port Washington NY 11054	North America
C81696	G60313	H84275	O81833	pparcellse@google.ca	8005668454	*PR606B*	Patrizio Parcells	460 Mccormick Way Saint Louis MO 63136	North America
C81785	G60406	H84368	O81926	dfaierf@timesonline.co.uk	8517714669	*WM640F*	Dare Faier	290 Sunfield Street Naperville IL 60567	North America
C81844	G60488	H84418	O82021	ynewloveg@cafepress.com	8989593075	*BI391M*	Yuma Newlove	14 Columbus Crossing Stockton CA 95205	North America
C81930	G60540	H84498	O82108	cbernardonh@washingtonpost.com		*AU470T*	Constantino Bernardon	318 Kingsford Lane Terre Haute IN 47812	North America
C82027	G60627	H84564	O82188	mwoofi@bbc.co.uk		*UV464X*	Marielle Woof	47 Donald Park Tampa FL 33647	North America
C82114	G60691	H84636	O82276	mlambornj@biglobe.ne.jp		*PT780Z*	Merill Lamborn	5 Muir Terrace Englewood CO 80150	North America

Call

accountID	agentID	generatedID	recording	totalCallDuration	callDescription	keyPointOfConcern	subPointsOfConcern	actonTaken
O90386	A30949	C10071	https://-47628.mp3	80	The caller was complaning about their program.	Pairing	Pairing	0
O90449	A31041	C10169	https://-47722.mp3	74	There was an issue with our servers.	Installation	Power	1
O90518	A31123	C10242	https://-47807.mp3	26	Someone called about the price of an item.	Installation	Connection	0
O90575	A31215	C10319	https://-47877.mp3	115	Password Reset call was made.	Connection	Game	1
O90647	A31295	C10407	https://-47928.mp3	7	There was an issue with our servers.	Installation	Settings	1
O90698	A31392	C10459	https://-48028.mp3	8	There was an issue with our servers.	Installation	Pairing	0
O90796	A31445	C10548	https://-48097.mp3	27	Misdial	Installation	Installation	1
O90853	A31518	C10641	https://-48186.mp3	64	Caller wanted more information	Connection	Power	0
O90931	A31574	C10733	https://-48240.mp3	61	There was an issue with our servers.	Game	Installation	1

O90995	A3166 5	C10800	https://- 48310.mp3	99	The caller was complaning about their program.	Settings	Connection	1
O91061	A3175 4	C10881	https://- 48392.mp3	117	The caller was complaning about their program.	Connection	Power	0
O91124	A3181 1	C10937	https://- 48442.mp3	120	The caller was complaning about their program.	Settings	Game	0
O91177	A3186 7	C11031	https://- 48498.mp3	21	Caller wanted more information	Game	Performance	1
O91246	A3193 5	C11084	https://- 48576.mp3	88	Caller wanted more information	Performance	Installation	0
O91313	A3202 9	C11169	https://- 48670.mp3	29	Caller wanted more information	Installation	Pairing	0
O91377	A3211 9	C11256	https://- 48731.mp3	65	They wanted to change settings.	Connection	Settings	0
O91443	A3221 4	C11320	https://- 48804.mp3	113	There was an issue with our servers.	Pairing	Pairing	1
O91533	A3230 7	C11405	https://- 48895.mp3	18	Someone called about the price of an item.	Pairing	Game	0

O91608	A32376	C11460	https://-48994.mp3	93	They wanted to change settings.	Pairing	Settings	0
O91672	A32447	C11548	https://-49052.mp3	69	They wanted to change settings.	Settings	Connection	1

Ticket												
generatedID	agentID	transactionGameID	transactionHardID	problemID	ticketID	totalNumberOfCalls	agentName	accumulatedCallDuration	sDate	eDate	sTime	eTime
C2908	A50512	G40003	H47907	P91124	T69289	1	Celestia Aspall	46	12/8/2017	1/20/2018	12:50 PM	3:22 PM
C2972	A50581	G40103	H47982	P91215	T69357	2	Shadow Van Hove	110	1/31/2018	2/5/2018	8:48 AM	9:33 AM
C3047	A50666	G40190	H48045	P91301	T69433	4	Niel Episcopio	48	2/19/2018	3/23/2018	11:26 AM	10:19 AM
C3122	A50737	G40245	H48122	P91359	T69531	1	Lira Tuxsell	86	5/31/2017	8/27/2017	12:21 PM	12:50 PM
C3197	A50802	G40325	H48187	P91409	T69590	2	Thibaud Hallward	202	11/14/2017	1/28/2018	3:27 PM	2:49 PM
C3259	A50893	G40381	H48284	P91461	T69682	3	Therine Boulden	231	4/17/2017	4/22/2017	8:23 AM	3:05 PM
C3340	A50966	G40439	H48338	P91552	T69764	1	Humberto Whiteley	17	4/4/2017	4/22/2017	9:54 AM	4:39 PM
C3400	A51022	G40507	H48418	P91641	T69833	3	Selig Winspur	111	2/8/2017	4/27/2017	10:42 AM	12:37 PM
C3482	A51116	G40588	H48504	P91741	T69923	3	Britney Crinion	162	11/17/2017	1/13/2018	10:52 AM	9:06 AM

C3534	A51194	G40665	H48560	P91839	T70017	4	Cecil Polglase	368	10/5/2017	12/16/2017	1:54 PM	2:26 PM
C3622	A51249	G40727	H48620	P91890	T70070	1	Arlene Kilmister	36	12/28/2017	2/21/2018	1:02 PM	4:40 PM
C3679	A51310	G40807	H48682	P91970	T70168	5	Donal Tremblet	465	10/25/2017	1/6/2018	9:44 AM	10:55 AM
C3765	A51388	G40894	H48782	P92063	T70237	1	Elianora Casiroli	55	3/25/2017	6/21/2017	1:32 PM	2:18 PM
C3840	A51487	G40967	H48878	P92135	T70308	2	Rosemonde Gribbins	44	4/16/2017	5/30/2017	3:06 PM	3:10 PM
C3912	A51571	G41037	H48932	P92207	T70389	1	Zed Tidder	15	3/8/2017	4/4/2017	8:50 AM	9:01 AM
C4006	A51622	G41137	H48998	P92265	T70441	5	Lorenza Quickfall	450	7/29/2017	9/1/2017	11:17 AM	8:32 AM
C4085	A51704	G41233	H49080	P92326	T70529	5	Christoper Peat	185	6/4/2017	7/1/2017	8:51 AM	1:49 PM
C4141	A51774	G41289	H49148	P92390	T70611	5	Sander Cayle	570	9/26/2017	11/23/2017	12:16 PM	12:09 PM
C4203	A51827	G41370	H49206	P92446	T70697	5	Shandie Mitrikhin	85	11/22/2017	2/9/2018	2:19 PM	4:49 PM
C4255	A51884	G41441	H49289	P92515	T70793	1	Rosene Loker	67	5/9/2017	7/28/2017	12:20 PM	12:49 PM

Game										
accountID	ticketID	transactionGameID	gameName	softwareVersion	distributionType	gameID	purchaseDate	purchasePrice	gameMSRP	gamePublisher

O93015	T6419	G95552	Need for Fast 10	1.73267022	Digital	GA59291	6/26/2014	34.34	80	Triangle Enix
O93092	T6490	G95605	Good Souls 2	1.42773483	Physical	GA59351	2/29/2016	24.55	100	OXYsquare
O93162	T6572	G95672	Rainbow Dash	1.87390726	Physical	GA59406	1/5/2013	51.63	100	Dintenda
O93234	T6666	G95724	Super Darias World	1.45039009	Physical	GA59467	8/20/2013	4.68	100	Megasoft
O93289	T6725	G95800	Road Fighters	1.34236749	Digital	GA59543	2/11/2017	18.68	70	Megasoft
O93364	T6795	G95882	Baby Princess Story	1.64987926	Physical	GA59597	5/9/2015	25.54	30	Megasoft
O93446	T6886	G95967	Toy Novel 3	1.90355454	Physical	GA59659	4/24/2013	16.03	90	Dintenda
O93523	T6964	G96066	Grand Theft Trucks 4	1.67670082	Digital	GA59727	9/5/2014	81.28	90	Dintenda
O93577	T7052	G96127	IEEE Wrestle Mania	1.52675487	Digital	GA59796	8/31/2016	37.24	40	Activl
O93669	T7122	G96225	North Park	1.10742463	Physical	GA59896	10/9/2017	18.64	30	3k Passive
O93722	T7178	G96320	Resident Good 20	1.48640027	Physical	GA59964	7/23/2016	51.41	80	OXYsquare
O93802	T7278	G96419	Call of Honor 500	1.22757401	Digital	GA60059	9/22/2016	20.33	60	OXYsquare
O93874	T7369	G96469	Friendlyfield 2	1.12509021	Physical	GA60154	12/15/2016	9.91	40	OXYsquare

O93963	T7437	G96548	Living or Dying	1.96720907	Digital	GA60219	7/29/2017	13.92	80	Megasoft
O94054	T7506	G96628	Blue Alive Redemption	1.42915463	Physical	GA60269	7/10/2017	4.21	40	Triangle Enix
O94123	T7556	G96711	Left to Live 2	1.32039701	Digital	GA60322	4/10/2013	26	90	Triangle Enix
O94184	T7645	G96782	God of Peace 2	1.97998067	Physical	GA60400	8/14/2015	56.26	70	OXYsquare
O94246	T7709	G96856	Devil May Smile 4	1.25013604	Physical	GA60453	2/7/2015	11.32	50	OXYsquare
O94346	T7781	G96950	Cowardly Default	1.39350085	Digital	GA60510	10/1/2015	73.98	90	3k Passive
O94429	T7835	G97029	Cold Emblem: Dawn of Lucina	1.35863276	Physical	GA60597	10/3/2013	26.38	70	Activl

Hardware							
deviceModelID	dateWarranty	macAddress	userSerialNumber	purchaseDate	dateLastUsed	hardwareMSRP	purchasePrice
I9	9/15/2018	00E7920022CF	U91505	9/15/2016	10/8/2017	290	88.57
E9	7/28/2017	00634A002D7D	U91585	7/28/2015	3/19/2017	300	217.72
L3	9/7/2017	0052A200710F	U91661	9/7/2015	11/27/2017	210	110.81
B5	4/7/2018	00739000EC0F	U91727	4/7/2017	6/2/2017	330	10.04
G7	10/22/2019	000DE1006427	U91824	10/22/2017	7/1/2017	50	33.53

G6	7/29/2015	00BB2D00A093	U91905	7/29/2014	5/31/2017	130	11.19
K2	7/31/2016	00EB6B001001	U91970	7/31/2013	1/24/2018	370	169.34
I4	10/15/2017	0019780023FC	U92035	10/15/2014	9/17/2017	150	131.53
D7	12/15/2019	008599002228	U92090	12/15/2016	1/17/2018	260	2.58
J5	3/31/2017	007AA5007F04	U92190	3/31/2016	11/2/2017	290	132.44
C6	7/12/2019	0072E200C5AB	U92270	7/12/2017	2/6/2018	80	34.4
G2	6/16/2020	006A9300CF12	U92361	6/16/2017	6/19/2017	70	53.94
O3	3/7/2017	007A5A006197	U92445	3/7/2016	3/3/2018	200	55.53
L6	1/20/2019	0063B700F4EF	U92505	1/20/2017	7/15/2017	200	58.68
N2	5/4/2017	00B63900CA94	U92561	5/4/2016	11/7/2017	330	216.02
B8	9/24/2015	00F95B001B53	U92621	9/24/2013	12/4/2017	100	11.84
K10	6/24/2014	0034CC0016D4	U92694	6/24/2013	4/1/2017	130	65.65
B9	9/29/2020	004C1D00498D	U92767	9/29/2017	11/10/2017	30	8.57
C10	12/19/2016	0027FE0059BB	U92855	12/19/2015	12/9/2017	200	52.86
B5	6/9/2017	005243005DFF	U92926	6/9/2015	6/15/2017	340	181.22

Main Problem				
ticketID	solutionID	problemID	focuseAreaId	problemDescription

T19953	S78624	P8375	TROUBL49	The device is not working
T20003	S78717	P8475	MALACT42	The controller is not working
T20080	S78804	P8557	TROUBL77	The device is not powering on
T20135	S78865	P8637	BILLIN34	The game is not loading
T20205	S78956	P8730	TROUBL31	Unauthorized purchases.
T20270	S79032	P8826	HARDIN40	Cant Log In
T20329	S79124	P8919	HARDIN40	Forgot Password
T20429	S79175	P9005	MALACT57	Forgot Username
T20488	S79243	P9080	MALACT32	Incorrect Charge
T20563	S79317	P9166	BILLIN39	Suspicious Activity
T20662	S79380	P9255	HARREP32	Hacked account
T20752	S79469	P9323	HARDIN40	Insufficient Funds
T20826	S79561	P9381	HARREP95	Game not loading
T20916	S79657	P9474	MALACT16	Hardware is slow
T20977	S79749	P9554	HARREP56	Account is locked
T21044	S79834	P9646	GAMINQ24	Unregistered Device

T21133	S79884	P9744	BILLIN81	Refund Request
T21214	S79940	P9801	BILLIN37	Cant log in on specific days.
T21282	S80001	P9896	HARDIN73	Game is not responding
T21380	S80097	P9972	GAMINQ99	Headhpones are not working.

Solution		
problemID	solutionID	solutionDescription
P23409	S34192	Remove foreign object.
P23480	S34264	Power cycle the device
P23559	S34363	Reconfigure your wireless settings
P23640	S34451	Check the audio player controls. Many audio and video players will have their own separate audio controls. Make sure the sound is turned on and that the volume is turned up in the player.
P23704	S34503	Change your TV settings

P23785	S34560	**Replace the batteries**
P23848	S34637	**Move the wireless router.**
P23927	S34731	**Check the cables. Make sure external speakers are plugged in, turned on, and connected to the correct audio port or a USB port. If your computer has color-coded ports, the audio output port will usually be green.**
P24012	S34781	**Use an ethernet cable.**
P24068	S34868	**Keep water away from the device.**
P24163	S34961	**If it is plugged into an outlet, make sure it is a working outlet. To check your outlet, you can plug in another electrical device, such as a lamp.**
P24261	S35042	**Make sure you're the powe cable is connected.**
P24359	S35135	**Delete your old save data.**

P24436	S35200	Check the volume level. Click the audio button in the top-right or bottom-right corner of the screen to make sure the sound is turned on and that the volume is up.
P24486	S35265	Import your settings.
P24563	S35333	Follow steps 1 to 5 of the debugging manual
P24623	S35422	Use the forgot your password tool.
P24685	S35495	Change your game settings
P24745	S35555	Restart the game
P24834	S35605	Connect headphones to the computer to find out if you can hear sound through the headphones.

Game Inquiry			
ticketID	solutionID	gameID	thirdPartyPublisher
T64749	S2632	GA31944	Triangle Enix

T64830	S2693	GA32036	IDHardware
T64925	S2753	GA32120	Dintenda
T64988	S2830	GA32173	Megasoft
T65061	S2908	GA32225	Megasoft
T65130	S2994	GA32313	Megasoft
T65185	S3069	GA32368	Dintenda
T65268	S3154	GA32423	Dintenda
T65318	S3245	GA32481	ActivI
T65404	S3332	GA32546	3k Passive
T65455	S3388	GA32635	IDHardware
T65530	S3467	GA32692	IDHardware
T65620	S3544	GA32784	IDHardware
T65708	S3614	GA32841	Megasoft
T65758	S3699	GA32932	Triangle Enix
T65812	S3784	GA33015	Triangle Enix
T65892	S3875	GA33110	IDHardware
T65971	S3939	GA33186	IDHardware
T66071	S4018	GA33242	3k Passive

T66165	S4112	GA33323	Activl

Malicious Activity			
ticketID	solutionID	activityType	securityIssue
T39970	S8960	Abuse	User
T40043	S9034	Griefing	Other User
T40112	S9087	Denial of Service	Other User
T40180	S9138	Hijacking	Other User
T40253	S9206	Hacking	Other User
T40333	S9286	Fraud	Game
T40409	S9344	Hijacking	Other User
T40463	S9396	Hijacking	User
T40527	S9472	Denial of Service	User
T40602	S9538	Griefing	User
T40684	S9633	Griefing	Game
T40739	S9704	Fraud	Game
T40821	S9754	Abuse	Other User

T40893	S9809	Stealing	User
T40979	S9859	Fraud	Game
T41078	S9917	Griefing	Other User
T41165	S9970	Abuse	Game
T41263	S10060	Abuse	Company
T41317	S10113	Denial of Service	Other User
T41405	S10164	Abuse	Game

Hardware Inquiry				
ticketID	solutionID	deviceModelID	internalPartInQuestion	inquiryType
T94394	S46222	I4	Controller	Features
T94458	S46288	N8	GPU	Release Date
T94514	S46362	C2	Controller	Release Date
T94609	S46443	C4	Controller	Ports
T94686	S46541	F7	Controller	Features
T94758	S46640	O9	Controller	Features
T94808	S46691	B6	I/O	Revision

T94908	S46753	L9	I/O	General Information
T94968	S46842	E3	I/O	Revision
T95052	S46907	P4	I/O	Ports
T95116	S46964	N4	Controller	General Information
T95205	S47035	N2	Display	Specifications
T95257	S47101	O4	Display	Revision
T95355	S47190	J9	I/O	Compatibility
T95435	S47266	D2	Controller	Ports
T95485	S47318	D10	Controller	General Information
T95548	S47377	G6	Display	Compatibility
T95633	S47465	O10	Controller	Ports
T95705	S47564	M8	Display	Specifications
T95779	S47625	A6	Display	Features

Account Issues				
ticketID	solutionID	accountType	accountStatus	accountIssue
T45335	S73105	Master	Suspended	Change Email Address

T45425	S73167	Master	Flagged	Hacking
T45498	S73222	Sub Account	Banned	Change Email Address
T45550	S73292	Sub Account	Banned	Hacking
T45610	S73376	Master	Banned	Abuse
T45666	S73451	Sub Account	Flagged	Abuse
T45745	S73515	Master	Suspended	Change Email Address
T45804	S73605	Sub Account	Banned	Inactive
T45870	S73691	Master	Banned	Change Email Address
T45970	S73787	Sub Account	Flagged	Abuse
T46020	S73840	Sub Account	Banned	Inactive
T46081	S73899	Master	Banned	Inactive
T46167	S73989	Master	Flagged	Inactive
T46232	S74042	Sub Account	Flagged	Hacking
T46292	S74096	Master	Flagged	Change Email Address
T46381	S74155	Sub Account	Normal	Hacking

T46476	S74237	Sub Account	Banned	Change Email Address
T46545	S74308	Master	Normal	Abuse
T46634	S74390	Master	Suspended	Password
T46690	S74441	Master	Flagged	Abuse

Billing					
ticketID	solutionID	refundType	billingIssue	creditCardAssocaited	refundTypeRequested
T75096	S41040	Credit Card	Accidental Purchase	7275670747294470	Store Credit
T75147	S41121	PayPal	Fraud	2031650940922660	Credit Card
T75245	S41195	None	Fraud	2787140107507510	Store Credit
T75315	S41267	None	Accidental Purchase	8831074160361630	Credit Card
T75386	S41317	None	Fraud	1316159189785760	Store Credit
T75440	S41410	None	Insuffecient Funds	9343841084544420	PayPal
T75499	S41497	PayPal	Insuffecient Funds	1799835141957680	Store Credit
T75584	S41559	None	Fraud	1321075912281380	Store Credit
T75667	S41643	PayPal	Fraud	8526531304577810	Store Credit

T75729	S41724	Store Credit	Insuffecient Funds	5815627769961170	Credit Card
T75819	S41811	PayPal	Insuffecient Funds	5390042817899090	PayPal
T75876	S41884	None	Insuffecient Funds	4819297130812210	Store Credit
T75967	S41968	None	Insuffecient Funds	9212105170095910	Store Credit
T76024	S42020	PayPal	Accidental Purchase	1100539939946470	Store Credit
T76098	S42089	None	Insuffecient Funds	4041480431697750	Store Credit
T76195	S42169	Store Credit	Accidental Purchase	1412486474925880	PayPal
T76250	S42233	Credit Card	Fraud	1268053444698540	PayPal
T76320	S42306	PayPal	Fraud	4722744610494580	PayPal
T76389	S42381	PayPal	Fraud	8124656703259180	Credit Card
T76489	S42480	Store Credit	Insuffecient Funds	4592683209429320	Credit Card

Hardware Repair						
ticketID	solutionID	warrantyAssociated	serviceType	deviceIssue	servicedRegion	shipToAddress

T69177	S46472	OEM Warranty	Refurbished	Disk Drive Failure	North America	54 Quincy Avenue Lancaster PA 17605
T69263	S46557	Extended Warranty	Replacement	Power Supply Failure	Middle East	56796 Paget Circle Albany NY 12222
T69330	S46653	Out of Warranty	New	Main Board	Europe	44381 Dawn Park Columbus OH 43284
T69400	S46708	Extended Warranty	Refurbished	Speaker Failure	North America	864 Heffernan Pass Evanston IL 60208
T69476	S46771	OEM Warranty	Replacement	Main Board	South America	85 Raven Hill Portland OR 97255
T69541	S46828	Extended Warranty	Refurbished	Case	Middle East	320 Pearson Way Metairie LA 70033
T69637	S46919	OEM Warranty	Replacement	Power Supply Failure	Asia	88 Dakota Point Toledo OH 43656
T69718	S47003	OEM Warranty	New	Overheating	North America	575 Warrior Park Schaumburg IL 60193
T69803	S47078	OEM Warranty	Replacement	Monitor Failure	Europe	10908 Hoepker Point Lubbock TX 79410

T69882	S47171	Extended Warranty	Refurbished	Main Board	South America	044 Troy Circle Long Beach CA 90805
T69969	S47256	Out of Warranty	Refurbished	Monitor Failure	North America	029 Comanche Alley Kansas City MO 64187
T70043	S47325	OEM Warranty	Refurbished	Overheating	Austrailia	996 Declaration Hill Peoria IL 61635
T70120	S47425	OEM Warranty	Replacement	Case	Middle East	50 Walton Terrace Anderson SC 29625
T70170	S47505	OEM Warranty	Replacement	Overheating	Asia	695 Gerald Road Burbank CA 91520
T70261	S47567	OEM Warranty	Replacement	Speaker Failure	North America	0123 Hollow Ridge Terrace Washington DC 20010
T70329	S47628	OEM Warranty	Replacement	Main Board	Asia	68 Farwell Alley Atlanta GA 30351
T70421	S47720	Out of Warranty	Refurbished	Main Board	North America	49 Morningstar Place Milwaukee WI 53210

T70475	S47778	OEM Warranty	Replacement	Main Board	South America	7 Twin Pines Center Providence RI 2905
T70564	S47867	Out of Warranty	New	Case	Austrailia	7525 Granby Avenue Sacramento CA 94280
T70638	S47941	OEM Warranty	Replacement	Speaker Failure	North America	08598 Lotheville Crossing Amarillo TX 79118

Troubleshooting				
ticketID	solutionID	internalPartAssociated	troubleshootingType	errorCode
T58576	S75741	Disk Drive	Power	EC-79825
T58626	S75820	Hard Drive	Pairing	EC-79875
T58717	S75887	Indicator Lights	Settings	EC-79958
T58771	S75978	Buttons	Power	EC-80024
T58822	S76048	Hard Drive	Game	EC-80111
T58905	S76119	Battery	Connection	EC-80197
T58966	S76186	USB	Connection	EC-80247
T59055	S76255	Hard Drive	Game	EC-80326
T59111	S76312	Disk Drive	Connection	EC-80408

T59206	S76369	Indicator Lights	Power	EC-80465
T59256	S76454	Disk Drive	Game	EC-80546
T59325	S76521	USB	Performance	EC-80614
T59408	S76594	Hard Drive	Pairing	EC-80672
T59502	S76694	USB	Connection	EC-80725
T59589	S76783	Battery	Connection	EC-80794
T59689	S76854	Buttons	Settings	EC-80850
T59740	S76908	Hard Drive	Pairing	EC-80908
T59802	S76981	Hard Drive	Game	EC-80976
T59900	S77057	Buttons	Settings	EC-81049
T59999	S77133	Battery	Power	EC-81133

Creates			
agentID	ticketID	sDate	sTime
A24485	T3216	9/18/2014	12:32 PM
A24539	T3280	7/20/2017	3:04 PM
A24638	T3351	3/20/2017	2:43 PM
A24713	T3427	10/26/2017	1:18 PM

A24764	T3519	10/7/2014	4:30 PM
A24831	T3595	2/22/2016	2:47 PM
A24925	T3652	6/1/2017	12:14 PM
A25002	T3747	12/14/2014	10:16 AM
A25083	T3839	7/8/2014	11:59 AM
A25181	T3939	10/26/2014	1:58 PM
A25269	T3993	4/26/2014	9:42 AM
A25368	T4073	3/18/2016	11:00 AM
A25460	T4123	1/12/2015	12:25 PM
A25529	T4190	11/23/2015	11:03 AM
A25593	T4260	5/30/2016	3:41 PM
A25649	T4352	2/1/2017	3:03 PM
A25702	T4433	1/14/2018	4:08 PM
A25780	T4487	3/14/2014	8:19 AM
A25836	T4540	2/27/2016	3:32 PM
A25914	T4629	2/18/2017	11:20 AM
A25997	T4713	7/19/2013	8:49 AM
A26073	T4802	3/16/2016	3:48 PM

A26166	T4876	12/28/2017	11:39 AM
A26239	T4949	1/5/2015	4:51 PM
A26325	T5025	12/14/2015	1:55 PM
A26408	T5078	4/12/2016	10:37 AM
A26488	T5149	9/9/2016	1:16 PM
A26557	T5243	1/25/2015	11:54 AM
A26646	T5337	10/1/2013	8:22 AM
A26713	T5399	5/30/2014	1:57 PM
A26801	T5470	11/12/2017	11:41 AM
A26869	T5522	12/30/2016	4:36 PM
A26941	T5574	11/14/2016	4:40 PM
A27033	T5666	8/25/2014	9:20 AM
A27132	T5760	2/10/2015	4:58 PM
A27217	T5820	9/29/2013	12:04 PM
A27287	T5920	3/11/2014	11:31 AM
A27364	T5996	12/17/2016	12:18 PM
A27430	T6082	5/16/2014	8:03 AM
A27523	T6155	10/3/2016	11:00 AM

A27586	T6206	3/14/2017	8:01 AM
A27655	T6265	5/30/2015	10:42 AM
A27737	T6320	1/30/2013	8:07 AM
A27798	T6390	2/4/2013	1:23 PM
A27879	T6489	8/10/2014	10:34 AM
A27937	T6585	12/21/2013	9:30 AM
A28032	T6654	10/28/2016	10:26 AM
A28116	T6751	1/13/2017	10:03 AM
A28196	T6832	1/22/2017	2:36 PM
A28261	T6904	2/19/2015	10:55 AM
A28337	T6972	9/2/2014	4:04 PM
A28415	T7070	5/22/2014	4:40 PM
A28468	T7139	9/6/2016	4:39 PM
A28532	T7225	8/7/2017	8:36 AM
A28601	T7291	1/9/2013	2:29 PM
A28698	T7349	8/29/2013	1:34 PM
A28766	T7434	11/20/2013	8:02 AM
A28851	T7510	1/21/2015	4:01 PM

A28922	T7609	11/21/2016	1:37 PM
A29016	T7669	5/4/2014	10:10 AM
A29076	T7749	10/2/2017	1:11 PM
A29149	T7827	8/11/2013	10:11 AM
A29204	T7899	1/19/2015	4:43 PM
A29280	T7966	7/11/2017	12:45 PM
A29368	T8055	12/5/2016	12:29 PM
A29460	T8112	6/18/2015	3:55 PM
A29517	T8191	11/11/2017	1:18 PM
A29581	T8267	11/29/2016	8:36 AM
A29678	T8323	1/1/2015	12:01 PM
A29762	T8420	10/27/2016	9:24 AM
A29821	T8478	9/12/2016	3:24 PM
A29871	T8557	8/31/2015	1:12 PM
A29941	T8610	9/13/2016	2:03 PM
A30026	T8665	2/18/2017	9:42 AM
A30095	T8719	4/29/2014	8:42 AM
A30183	T8770	4/25/2013	11:48 AM

A30263	T8866	10/2/2014	10:12 AM
A30355	T8962	10/13/2017	8:33 AM
A30407	T9042	1/25/2018	10:11 AM
A30494	T9110	10/13/2014	2:11 PM
A30568	T9178	1/16/2018	1:57 PM
A30635	T9236	1/11/2018	11:25 AM
A30720	T9303	10/12/2014	12:56 PM
A30800	T9379	8/3/2014	2:08 PM
A30898	T9456	6/2/2014	4:09 PM
A30956	T9518	9/22/2016	4:34 PM
A31056	T9569	5/30/2014	3:03 PM
A31147	T9628	2/11/2018	2:53 PM
A31209	T9727	5/18/2013	11:11 AM
A31268	T9780	8/26/2013	9:26 AM
A31355	T9840	12/26/2013	2:48 PM
A31436	T9925	8/20/2013	8:50 AM
A31491	T10000	1/31/2014	9:51 AM
A31542	T10068	3/4/2018	2:09 PM

A31608	T10164	10/20/2013	3:29 PM
A31661	T10250	7/20/2017	4:22 PM
A31740	T10336	6/29/2013	9:03 AM
A31827	T10404	11/17/2014	2:27 PM
A31909	T10467	6/10/2015	10:16 AM
A31966	T10518	10/16/2013	2:55 PM

Answered By				
generatedID	agentID	date	s_Time	e_Time
C76076	A12666	9/29/2013	4:36 PM	5:02 PM
C76144	A12733	9/16/2014	2:10 PM	2:46 PM
C76239	A12817	12/12/2016	3:18 PM	4:46 PM
C76291	A12914	11/15/2017	11:11 AM	12:32 PM
C76365	A12997	11/28/2017	2:25 PM	3:52 PM
C76429	A13072	5/17/2015	2:56 PM	3:13 PM
C76515	A13139	8/4/2016	12:02 PM	1:56 PM
C76583	A13192	4/14/2017	9:26 AM	9:59 AM
C76682	A13247	10/13/2013	11:25 AM	11:54 AM

C76774	A13333	1/5/2018	2:02 PM	2:29 PM
C76845	A13391	12/30/2016	4:32 PM	5:40 PM
C76939	A13442	10/27/2013	11:38 AM	12:55 PM
C77002	A13494	1/4/2018	9:39 AM	10:05 AM
C77059	A13571	12/5/2014	2:46 PM	3:20 PM
C77127	A13666	12/12/2016	10:15 AM	11:59 AM
C77197	A13738	1/13/2014	2:14 PM	3:20 PM
C77260	A13809	2/24/2013	2:29 PM	2:47 PM
C77340	A13894	7/2/2014	2:13 PM	3:27 PM
C77397	A13964	7/14/2015	11:52 AM	1:09 PM
C77473	A14064	2/15/2015	4:07 PM	4:14 PM
C77533	A14117	12/7/2016	1:14 PM	2:05 PM
C77630	A14191	1/4/2015	8:00 AM	9:45 AM
C77715	A14265	7/12/2015	8:33 AM	10:17 AM
C77792	A14354	3/24/2016	2:12 PM	3:10 PM
C77857	A14420	5/21/2015	12:00 PM	12:42 PM
C77935	A14488	10/17/2016	2:33 PM	4:30 PM
C78021	A14554	9/19/2014	9:24 AM	10:27 AM

C78090	A14651	10/22/2015	12:21 PM	1:21 PM
C78170	A14720	1/24/2013	8:32 AM	9:35 AM
C78236	A14814	2/14/2017	1:36 PM	2:54 PM
C78290	A14907	9/1/2016	1:45 PM	2:23 PM
C78388	A14958	8/23/2016	10:26 AM	11:01 AM
C78447	A15044	6/18/2015	4:00 PM	5:26 PM
C78532	A15120	2/9/2015	8:48 AM	9:31 AM
C78596	A15211	11/23/2014	9:17 AM	10:24 AM
C78665	A15291	8/1/2016	2:03 PM	3:33 PM
C78723	A15391	10/22/2015	2:00 PM	3:34 PM
C78789	A15470	6/1/2017	11:09 AM	11:37 AM
C78875	A15564	9/1/2017	2:45 PM	3:00 PM
C78975	A15638	9/18/2017	2:43 PM	4:21 PM
C79025	A15691	6/2/2014	1:33 PM	2:13 PM
C79094	A15787	7/23/2015	1:27 PM	3:08 PM
C79182	A15866	4/10/2017	4:33 PM	5:26 PM
C79233	A15919	12/29/2015	8:29 AM	10:10 AM
C79309	A16007	4/1/2013	9:42 AM	9:52 AM

C79387	A16073	4/21/2016	9:18 AM	11:02 AM
C79483	A16170	5/15/2017	1:18 PM	3:15 PM
C79537	A16268	5/29/2013	4:01 PM	4:29 PM
C79595	A16348	1/25/2014	11:24 AM	1:04 PM
C79667	A16440	9/11/2015	4:27 PM	5:19 PM
C79747	A16513	12/22/2016	2:23 PM	4:15 PM
C79834	A16605	10/11/2016	3:13 PM	4:33 PM
C79892	A16670	1/16/2015	9:06 AM	9:12 AM
C79987	A16721	3/17/2013	3:07 PM	4:17 PM
C80070	A16785	12/2/2013	1:04 PM	1:05 PM
C80142	A16858	2/3/2014	8:45 AM	9:38 AM
C80222	A16954	6/16/2015	10:00 AM	11:08 AM
C80312	A17045	3/10/2016	9:23 AM	9:40 AM
C80399	A17101	3/5/2016	3:46 PM	4:44 PM
C80472	A17194	4/22/2017	9:05 AM	9:28 AM
C80543	A17294	6/6/2013	12:58 PM	2:02 PM
C80594	A17369	7/25/2017	8:16 AM	8:21 AM
C80683	A17457	1/26/2018	1:37 PM	2:21 PM

C80750	A17517	4/25/2014	12:37 PM	1:58 PM
C80801	A17585	12/27/2013	2:20 PM	2:30 PM
C80866	A17671	11/1/2013	2:56 PM	3:45 PM
C80927	A17767	2/20/2013	10:05 AM	11:46 AM
C81009	A17866	1/11/2017	3:50 PM	4:58 PM
C81106	A17931	8/19/2017	12:25 PM	2:17 PM
C81174	A18028	4/11/2017	1:51 PM	2:48 PM
C81248	A18090	6/23/2015	11:04 AM	11:05 AM
C81335	A18173	4/5/2014	2:34 PM	3:15 PM
C81403	A18246	9/30/2013	3:17 PM	4:34 PM
C81458	A18343	5/2/2013	9:28 AM	9:43 AM
C81528	A18438	3/17/2016	4:14 PM	5:09 PM
C81583	A18496	11/19/2013	8:29 AM	9:13 AM
C81633	A18571	1/13/2016	11:43 AM	12:23 PM
C81710	A18654	8/5/2014	11:01 AM	12:41 PM
C81808	A18748	7/2/2016	10:27 AM	11:08 AM
C81900	A18845	2/21/2018	3:33 PM	4:05 PM
C81995	A18924	10/9/2015	2:02 PM	3:38 PM

C82066	A19022	2/21/2016	4:41 PM	6:04 PM
C82153	A19121	8/10/2013	3:54 PM	5:39 PM
C82247	A19191	6/10/2015	3:51 PM	5:43 PM
C82297	A19267	6/28/2017	11:50 AM	12:31 PM
C82375	A19352	6/18/2014	11:13 AM	12:11 PM
C82459	A19434	10/8/2017	3:10 PM	3:43 PM
C82546	A19522	10/16/2015	10:47 AM	11:56 AM
C82615	A19605	10/20/2016	11:29 AM	1:18 PM
C82697	A19659	11/2/2015	12:03 PM	12:18 PM
C82776	A19758	7/30/2014	10:26 AM	12:23 PM
C82872	A19836	4/19/2016	10:53 AM	12:24 PM
C82965	A19908	9/15/2014	8:57 AM	10:50 AM
C83065	A19987	7/25/2013	12:38 PM	1:37 PM
C83141	A20079	1/18/2017	12:18 PM	1:47 PM
C83232	A20136	12/23/2015	12:11 PM	12:26 PM
C83287	A20197	11/5/2017	11:53 AM	1:49 PM
C83387	A20263	12/13/2016	11:23 AM	12:05 PM
C83449	A20326	8/14/2015	2:08 PM	4:04 PM

C83532	A20420	6/25/2016	3:58 PM	5:57 PM

Initiates				
accountID	generatedID	date	startTime	endTime
A41726	C31299	5/28/2014	12:49 PM	1:39 PM
A41820	C31397	1/5/2017	8:33 AM	9:58 AM
A41872	C31453	2/17/2015	12:49 PM	2:25 PM
A41932	C31539	11/21/2017	4:09 PM	5:32 PM
A42004	C31605	9/22/2015	2:02 PM	2:43 PM
A42094	C31684	7/27/2014	2:58 PM	3:46 PM
A42177	C31783	4/28/2014	11:56 AM	1:39 PM
A42275	C31834	6/1/2016	10:40 AM	11:34 AM
A42366	C31920	12/23/2015	12:34 PM	2:15 PM
A42451	C32006	12/13/2014	4:00 PM	4:35 PM
A42549	C32081	11/9/2015	12:23 PM	12:54 PM
A42621	C32170	10/16/2013	8:04 AM	9:09 AM
A42693	C32241	5/28/2015	9:29 AM	10:57 AM
A42792	C32316	2/5/2013	12:31 PM	12:36 PM

A42875	C32405	9/18/2016	2:51 PM	4:44 PM
A42934	C32465	1/14/2017	12:28 PM	2:06 PM
A43024	C32552	6/12/2016	9:27 AM	10:24 AM
A43116	C32606	12/14/2017	11:09 AM	11:23 AM
A43174	C32703	3/10/2018	4:18 PM	5:49 PM
A43225	C32802	3/27/2014	8:35 AM	8:38 AM
A43286	C32880	1/4/2014	10:28 AM	10:48 AM
A43358	C32967	10/27/2014	1:15 PM	1:51 PM
A43427	C33041	9/10/2015	4:05 PM	5:45 PM
A43506	C33132	6/24/2016	9:22 AM	10:21 AM
A43588	C33221	3/13/2013	10:27 AM	12:09 PM
A43665	C33297	8/10/2015	12:30 PM	12:35 PM
A43748	C33371	5/23/2013	2:27 PM	3:06 PM
A43834	C33459	8/30/2013	8:36 AM	10:26 AM
A43927	C33548	5/19/2015	4:23 PM	5:34 PM
A43991	C33637	3/28/2015	12:50 PM	2:01 PM
A44050	C33702	4/23/2016	12:14 PM	12:55 PM
A44122	C33775	7/27/2017	8:06 AM	8:26 AM

A44199	C33870	1/2/2017	3:24 PM	3:25 PM
A44297	C33924	5/30/2014	12:33 PM	1:01 PM
A44392	C34012	11/30/2015	3:00 PM	3:30 PM
A44452	C34079	12/26/2017	9:59 AM	11:20 AM
A44523	C34140	2/5/2017	8:14 AM	9:08 AM
A44577	C34236	2/25/2016	9:09 AM	9:40 AM
A44663	C34324	11/2/2015	11:00 AM	12:42 PM
A44757	C34411	7/17/2017	3:29 PM	3:32 PM
A44856	C34484	12/25/2013	11:28 AM	11:40 AM
A44944	C34564	8/10/2013	2:25 PM	3:32 PM
A45004	C34630	2/9/2014	12:12 PM	2:09 PM
A45094	C34692	3/6/2013	12:28 PM	1:04 PM
A45163	C34792	4/6/2015	3:55 PM	4:09 PM
A45262	C34885	10/10/2013	8:08 AM	10:03 AM
A45313	C34975	10/11/2014	9:46 AM	10:17 AM
A45377	C35027	12/14/2015	1:00 PM	2:26 PM
A45457	C35103	3/21/2015	10:11 AM	10:52 AM
A45537	C35184	3/4/2013	10:26 AM	11:21 AM

A45604	C35261	2/14/2014	10:49 AM	12:06 PM
A45656	C35360	6/19/2017	4:08 PM	6:07 PM
A45744	C35439	1/6/2013	3:13 PM	3:56 PM
A45798	C35520	11/12/2013	11:18 AM	11:40 AM
A45882	C35590	5/15/2014	4:03 PM	5:40 PM
A45947	C35663	3/20/2014	12:48 PM	1:30 PM
A46006	C35759	7/4/2015	1:15 PM	1:28 PM
A46076	C35810	1/23/2018	3:01 PM	3:55 PM
A46145	C35895	12/21/2014	10:00 AM	10:10 AM
A46228	C35983	6/10/2014	8:02 AM	8:32 AM
A46290	C36036	1/21/2014	2:02 PM	2:23 PM
A46363	C36130	11/7/2013	1:23 PM	2:34 PM
A46438	C36183	10/12/2014	10:58 AM	12:56 PM
A46514	C36242	5/25/2016	11:36 AM	1:32 PM
A46569	C36307	6/21/2017	8:16 AM	8:41 AM
A46640	C36387	3/7/2017	4:46 PM	5:18 PM
A46704	C36454	2/24/2018	2:32 PM	3:37 PM
A46762	C36554	8/11/2013	12:16 PM	2:00 PM

A46825	C36646	1/11/2015	12:03 PM	12:34 PM
A46890	C36710	5/22/2017	4:56 PM	6:24 PM
A46984	C36765	11/1/2014	9:59 AM	11:15 AM
A47062	C36846	6/7/2013	8:51 AM	10:03 AM
A47123	C36934	12/15/2013	2:00 PM	2:41 PM
A47175	C37006	12/4/2016	3:38 PM	5:26 PM
A47230	C37082	12/26/2017	1:13 PM	2:30 PM
A47294	C37167	10/25/2013	4:06 PM	5:37 PM
A47344	C37260	6/28/2016	10:46 AM	11:28 AM
A47425	C37356	3/6/2017	4:24 PM	5:59 PM
A47489	C37416	7/13/2016	8:34 AM	9:32 AM
A47564	C37501	5/13/2016	11:33 AM	12:41 PM
A47644	C37596	9/28/2017	9:22 AM	10:59 AM
A47718	C37671	6/13/2017	1:57 PM	3:01 PM
A47790	C37760	8/12/2017	10:45 AM	12:39 PM
A47854	C37823	9/18/2017	9:48 AM	10:03 AM
A47919	C37888	8/6/2016	10:33 AM	12:06 PM
A48015	C37958	8/19/2014	9:49 AM	10:37 AM

A48084	C38054	1/13/2016	12:31 PM	1:32 PM
A48141	C38126	10/6/2016	12:11 PM	1:38 PM
A48195	C38212	12/31/2016	4:21 PM	4:26 PM
A48293	C38266	3/17/2017	4:59 PM	6:31 PM
A48392	C38327	4/30/2017	11:10 AM	12:22 PM
A48445	C38407	7/3/2014	8:05 AM	9:31 AM
A48531	C38469	1/14/2013	2:45 PM	3:37 PM
A48615	C38559	8/13/2013	8:20 AM	10:16 AM
A48704	C38621	10/24/2015	3:44 PM	5:22 PM
A48777	C38707	2/27/2016	2:52 PM	4:12 PM
A48849	C38787	2/23/2016	4:39 PM	4:56 PM
A48907	C38864	8/1/2015	2:06 PM	3:17 PM
A48982	C38933	7/21/2016	10:14 AM	10:56 AM
A49055	C39003	2/13/2014	4:09 PM	6:02 PM

2.4 Sample Queries

We have established the constraints and the relations for the actual implementation of the database. However we have not discussed how to get specific data from requests. This can range from simple queries such as getting the list of all Accounts

σ (Account)

To complicated queries such as getting all tickets that have the cheapest games. The query above is an example of relational algebra. There are two other ways to query and they are Tuple Relational Calculus and Domain Relational Calculus. All three of these methods are simply the logical means of getting the data. They are uniform logical expressions that can be converted to real programming query languages such as postgreSQL, mySQL, and Oracle.

2.4.1 Design of Queries

In order to simulate real word queries, we have come up with 10 difficult queries that will truly test the capabilities of our database. These queries will then be answered in Relational Algebra, Tuple Relational Calculus and Domain Relational Calculus.

1. List all tickets that have at least two games with game MSRP with price > $70
2. List Agents that have exactly one ticket with a game that costs > $250.
3. List all American accounts who made at least one call between 8:00AM and 9:30AM PDT.
4. List all the names of games that were tickets with 'Billing Issues' that relate to game publisher X .
5. List the accounts in Asia with second least expensive game from all game publishers.
6. List tickets that have the 2nd most expensive game.
7. List ticket inquiries that has been in each and every region
8. List all tickets that have the cheapest games.
9. List all accounts that have tickets for each game with game publisher: XYZ.
10. List agents that have been in every ticket in the asia region

2.4.2 Relational Algebra Expressions for Queries

Relational algebra is where there are atomic, unary, binary or multiple operations done to form expressions. These are then used for retrieving tuples from the database. It is a procedural language meaning, the other of operations on the expressions are important. Parenthesis play an important role, if they are misused then the results will greatly differ.

1. List all tickets that have at least two games with game MSRP with price > $70

Ticket * π (G1. ticketID (σ(G1.ticketID = G2.ticketID) ^ (G1.MSRP > 70 ^ G2.MSRP > 70) ^ (G1.gameID != G2.gameID) (Game x Game))

2. List Agents that have exactly one ticket with a game that costs > $250.

TG<- σ (g.gameMSRP> 70 (Ticket * Game)
Ticket * π(TG1.ticketID σ(TG1.ticketID = TG2.ticketID ^ TG1.gameID != TG2.gmeID) (TG x TG)))

3. List all American accounts who made at least one call between 8:00AM and 9:30AM PDT.

π a.name (σ a.region = ‘US’ ^ab.startTime >= 8:00AM PDT ^ ab.endTime <= 9:30 AM PDT (Account * Answered By)) * Call

4. List all the names of games that were tickets with ‘Billing Issues’ that relate to game publisher X .

π gameName (σ gamePublisher = ‘X’ (Game * Ticket * Billing))

5. List the accounts in Asia with second least expensive game from all game publishers.
Game01 <- π G1. * (σ G1.gameMSRP > G2.gameMSRP (Game X Game)
σ region = ‘Asia’ (Account * (π (gameId) (game - π g1.* (σ (g1.gameMSRP > g2.gameMSRP (Game01 x Game01))))

6. List tickets that have the 2nd most expensive game.

GameEX <-π g1.* σ g1.gameMSRP < g2.gameMSRP (Game X Game)
Ticket * (π(ticketID) (Game - π g1.* (σ g1.gameMSRP < g2.gameMSRP (GameEX x Game EX))))

7. List ticket inquiries that has been in each and every region

π (t.ticketID) (σ (t.focuseArea = Hardware ((π t.ticketID, a.region (Ticket * Account * Call) ÷ (π a.regon (Ticket * Account * Call))))) * Ticket

8. List all tickets that have the cheapest games.

Ticket * (π gameID (Game - π g1.* (σ g1.gameMSRP > g2.gameMSRP (Game X Game))))

9. List all accounts that have tickets for each game with game publisher: XYZ.

π a.name (σ g.gamepublisher = ‘XYZ’ (π g,gameID, t.ticketID (Account * Call * Ticket *Game) ÷ π g.game (Account * Call * Ticket * Game))) * Account

10. List agents that have been in every ticket in the asia region

ACTA <- (Agent * Call * Ticket * Account)
(π ticketID, agentID (Ticket) ÷ π ticketID (σ region = ‘Asia’ (ACTA))))* Agent

2.4.3 Tuple Relational Calculus Expressions for Queries

Tuple Relational Calculus uses existential and universal quantifiers to generate queries. It does not rely on procedures as it uses a boolean logic to complete the queries. These also rely on variable to signify which relations are being used. If there is a reference of a tuple without it being the output, then it must have an Existential or Universal Qualifier to back it up.

1. List all tickets that have at least two games with game MSRP with price > $70
   {t | ticket(t) ^ ∃(g1)(Game(g1) ^ g1.ticketID = t.ticketID ^ g1.gameMSRP> 70 ^ ∃(g2)(Game(g2)) ^ g2.ticketID = g1.ticketID ^ g2.gameMSRP >70 ^ g1.gameID != g2.gameID) }

2. List Agents that have exactly one ticket with a game that costs > $250.
   {a | Agent(a) ^ (∃t)(∃g) (Ticket(t) ^ Game(g) ^ t.gameID = g.gameID ^ g.gameMSRP > 250 ^ a.ticketID = t.ticketID) ^ (∃t2)(∃g2) (Ticket(t2) ^ Game (g2) ^ t2.gameID = g2.gameID ^ g2.gameMSRP > 250 ^ a.ticketID = t2.ticketID ^ t.ticketID ! = t2.ticketID) }

3. List all American accounts who made at least one call between 8:00AM and 9:30AM PDT
   {a | Account(a) ^ ∃(c) Call(c) ^ ∃(i) (initiates(i)) ^ i.startTime > 8:00 AM PDT ^ i.endTime < 9:30 AM PDT }
4. List all the names of games that were tickets with ‘Billing Issues’ that relate to game publisher X
   {g.name | Game(g) ^ (∃t) (∃b) Ticket(t) ^ Billing(b) ^ g.ticketID = t.ticketID ^ t.ticketID =

b.ticketID ^ g.gamePublisher = ‘X’ }

5. List the accounts in Asia with second least expensive game from all game publishers.
   {a | Account(a) ^ (∃g) (∃g2)(Game(g1)) ^ g.accountID = a.accountID ^ (Game(g2)) ^ g2.accountID = g1.accountID ^ g2.gameMSRP < g.gameMSRP ^ ! (∃g3) (game(g3) ^ g3.gameMSRP < g2.gameMSRP))}

6. List tickets that have the 2nd most expensive game
   { t | Ticket(t) ^ (∃g1)(∃g2) (Game(g1) ^ t.ticketI = g.ticketID ^ Game(g2) ^ g2.gameMSRP > g1.gameMSRP ^ ! (∃g3) (Game(g3) ^ g3.gameMSRP > g2.gameMSRP))}

7. List ticket inquiries that has been in each and every region
   {t | ticket(t) ^ ∀(t) (Ticket(t)) - > (∃c) (Call (c) ^ (∃g) (Game(g)) ^ (a.accountID = c.accountID) ^ (c.ticketId = t.ticketId) ^ (t.gameID = g.gameId) ^ (g.gamePublsher = ‘XYZ’))}

8. List all tickets that have the cheapest games.
   {t | Ticket (t) ^ (∃g) (Game(g) ^ g.ticketID = t.TicketID ^ (∀gx) (Game(gx) - > gx.gameMSRP >= g.gameMSRP)}

9. List all accounts that have tickets for each game with game publisher: XYZ.
   {a | Account (a) ^ (∀t) (Ticket(t)) -> (∃c) Call(c) ^ (∃g) (Game (g)) ^ (a.accountID = c.accountId) ^ (c.ticketID = t.ticketID) ^ (t.gameID = g.gameID) ^ (g.gamePublisher = ‘XYZ’))}

10. List agents that have been in every ticket in the asia region
    {a | Agent(a) ^(∀t) (∃c) (∃acc) (Ticket(t) ^ Account(acc) ^ Call(c) ^ c.generatedID = t.generatedID ^ acc.accountID = c.accountID ^ acc.region = ‘Asia’) -> (∃cr) (Creates (cr) ^ a.tikcetId = cr.ticketID ^ t. ticketID = cr.ticketID))}

2.4.4 Domain Relational Calculus Expressions form Queries

The domain relational calculus is similar to Relational calculus, but the relations that link to each other use variables instead of using attributes to correlate them. Each variable also relates to an attribute of a tuple instead of the entire relation.

1. List all tickets that have at least two games with game MSRP with price > $70
   {<_, _, _, _ , _ , t, _,_,_,_,_,_,_>| Ticket(_,_,_,_,_,t,_,_,_,_,_,_,_) ^ (∃g)

(Game(_,t,g,_,_,_,_,_,_,>70,_) ^ (∃g) (Game(_,t,g,_,_,_,_,_,_,>70,_) }

2. List Agents that have exactly one ticket with a game that costs > $250.
   {<_,t,_,_,_,_,_,_,_>| Agent(_,t,_,_,_,_,_,_,_) ^ (∃g) Ticket(_,_,g,_,_,t,_,_,_,_,_,_,_,)
   ^(Game(_,t,g,_,_,_,_,_,_,>250,_) ^ Ticket(_,_,_,_,_,!t,_,_,_,_,_,_,_,) ^
   (Game(_,t,_,_,_,_,_,_,_,>250,_) }

3. List all American accounts who made at least one call between 8:00AM and 9:30AM PDT.
   { <_,_,_,_,a,_,_,_,_> | Account (_,_,_,_,a,_,_,_,_) ^ (∃t) Ticket(c,_,g,_,_,t,_,_,_,_,_,
   > 8: 00 AM PDT, <9:30 aM PDT,) ^ (∃c) Call (a,_,c,_,_,_,_,_,_) ^ (∃g)
   Game(a,t,g,_,_,_,_,_,_,_,_) }

4. List all the names of games that were tickets with ‘Billing Issues’ that relate to game publisher X .
   { <_,_,g,_,_,_,_,_,_,_,_> | Game(_,_,g,_,_,_,_,_,_,_’X’) ^ (∃p) (∃t)
   (Ticket(_,_,g,_,p,t,_,_,_,_,_,_,_,) ^ Billing (t,_,_,_,_,_)}

5. List the accounts in Asia with second least expensive game from all game publishers.
   { <_,_,_,_,a,_,_,_,_> | Account (_,_,_,_,a,_,_,_,_) ^ (∃g1) (∃g2) (∃g3)
   (∃p1)(Game(a,_,g,_,_,_,_,_,_,<p1,_) ^! (∃p2)(Game(a,_,g,_,_,_,_,_,_,p2,_) ^ (p2 < p1)}
6. List tickets that have the 2nd most expensive game.
   { <_, _, g, _ , _ , _, _,_,_,_,_,_,_>| Ticket(_,_,g,_,_,_,_,_,_,_,_,_,_,) ^
   (∃p1)(Game(_,_,g,_,_,_,_,_,_,>p1,_) ^! (∃p2)(Game(_,_,_,_,_,_,_,_,_,p2,_) ^ (p2 > p1)}

7. List ticket inquiries that has been in each and every region
   {<_, _, _, _ , _ , t, _,_,_,_,_,_,_>| Ticket(_,_,_,_,_,t,_,_,_,_,_,_,_,) ^ (∀a) (∃r)(∃c) (! (Call
   (_,_,c,_,_,t,_,_,_) ^ Account (c,_,_,_a_,_,_,_,_) ^ Ticket(c,_,_,_,_,t,_,_,_,_,_,_,_,)) -> Call
   (a,_,c,_,_,t,_,_,_) }

8. List all tickets that have the cheapest games.
   { <_, _, g, _ , _ , _, _,_,_,_,_,_,_>| Ticket(_,_,g,_,_,_,_,_,_,_,_,_,_,) ^ (∃p1)
   (Game(_,_,g,_,_,_,_,_,_,p1,_) ^ (∀px) (Game(_,_,g,_,_,_,_,_,_,px,_) -> px >= p1))}

9. List all accounts that have tickets for each game with game publisher: XYZ.
   { <_,_,_,_,a,_,_,_,_> | Account (_,_,_,_,a,_,_,_,_) ^ (∀a) (!
   Game(a,t,_,_,_,_,_,_,_,’XYZ’) - > Ticket(_,_,g,_,_,_,_,_,_,_,_,_,_,)}

10. List agents that have been in every ticket in the asia region
    {<_,_a_,_,_,_,_,_,_>| Agent(_,_,a,_,_,_,_,_,_) ^ (∃c) (∀t) (Call (_,_,c,_,_,_,_,_,_) ^ Account (c,_,_,_,_,_,_,_,'Asia') ^ Ticket(c,_,_,_,_,t,_,_,_,_,_,_,_,) V Creates(a, t, _, _)

PHASE 3 PostgreSQL Database Management System

3.1 Normalization of Relations

The purpose of phase 3 use normalization to determine if our database is well-implemented and designed. This is important to test the implementation of the database. This will drive deep into that what is good database implementation and what problems can occur with poor implementation. After we have discussed the fundamentals of database implementation, we will explore the details of our Consumer Services Call Database.

3.1.1 Normalization and Normal Forms

We will start off with the fundamental concepts in normalization and explore the ways to measure normalization though normal forms techniques. Below are the details of all approaches used.

Description of Normalization and Normal Forms

Normalization is important to have the most efficient updating process. The method of the normalization is an act of splitting apart the relation schemas to prevent redundancy. When a relational database is bad, it will create repeated data in its tuples. Normalization can be based on Primary Keys and it is the act of testing a relation schema whether it is certified to be a particular normal form. This is usually done in a top down manner.

It's goal is to minimize redundancy, minimize insertion, deletion, and update anomalies. In a certain relation schema does not meet any of the normal forms, it is segmented until they

fit the norm. However following these norms will not guarantee a good database. Below we will discuss the several methods of normalizations: First Normal Form, Second Normal Form, and lastly Third Normal Form.

First Normal Form

The first normal form of notation (1NF) does not allow multi-valued attributes, composite attributes and combinations. This means first form is only atomic with simple, indivisible values. It notes that the domain of the attribute can only be atomic(simple and indivisible) and that the value of the tuples can only be a single value. First Normal Form allows for relations within relations or relations as attribute values within tuples. Methods for the First Normal Form are:

1. Map the multi-value attribute in the ER-model to relations
2. If there is the same number of values in the multi-valued attribute, a single-value attribute can be added to each value for the multi-value attributes.
3. Lastly, the multi-value attribute can be replace with a new single- value attribute and the others will be put into a separate "duplicate" tuple. Note: this method should be avoided because it produces redundant data.

Overall this First Normal Form can be resolved by creating a nested relation where a tuple can have a relation within it.

Second Normal Form

The Second Normal Form of notation (2NF) is conceptually based on "full functional dependency." This Second Normal Form notes that if X → Y (Y is functionally dependent on X), then the removal of an attribute in X will show that there will no longer have dependency. To expand on that thought, a set of attributes in Y will be dependent on X's set of values that map to the values for Y. This means that the primary key of a specific tuple will show that each primary key will map to a specific tuple itself.

That being said, attributed of each will specifically depend on the primary key value itself. If the primary key is removed then the attributes will no longer hold functional dependency, meaning that nonprime-attributes must fully depend on the entire primary key. Relation schemas must have a single attribute primary key to pass the 2NF test. In general if a relation schema does not follow the Second Normal Form, the solution is to split it further down to smaller schemas. When doing this, we will need to make the new schemas' primary keys as subsets of the primary key of the original relation schema.

Third Normal Form

The Third Normal Form (3NF) is the third test applied towards our database, after first and second normal form. This Third Normal Form is based on the concept of transitive dependency on the primary key itself. In an example of A→ B, the 3NF notes that if there is a set of attributes Z in relation R, it can not be a candidate key and it cannot be a subset of any candidate key in R. This means that both A→ Z and Z → B will stand. In short, it means that there cannot be non-prime attributes depending on other non-prime attributes. In this case, it is similar to the second normal form, but transitivity is not allowed. Each partial key of a primary key may not be duplicated. If a relation does not follow the Third Normal Form, one can broken down into smaller relations where there is functional dependency where a primary key attribute is on the left side of the functional dependency.

Summary of 1NF, 2NF, and 3NF

Normal Form Method	**Test Definition**	**Solution for Normalization**
First Normal Form (1NF)	Relations: - Cannot have multi-valued attributes - Cannot have nested relations	Make a new relation for the multi-values
Second Normal Form (2NF)	Relations with primary keys with multiple attributes: - Attributes (nonkey) can only be functionally dependent on entire primary key (cannot be part)	Break up the relation into smaller relations for the dependent attribute with the partial key
Third Normal Form (3NF)	Relations: - Nonkey attributes should not have transitive dependency on primary key	Break down relation into new relations that will show that nonkey attributes and the other nonkey attributes that are functionally dependent to it.

Boyce-Codd Normal Form

The Boyce-Codd Normal (BCNF) if very similar to the Third Normal Form (3NF) but simpler and stricter. The BCNF relations are also in 3NF but 3NF relations are not necessarily in

BCNF. This means that whenever there is a non-trivial functional dependency where X → A is in R, then the X is a superkey of R as a definition of Boyce-Codd Normal Form.

Anomalies that Result for Poor Normalization

When we follow the normalization forms like BCNF and 3NF, redundant data is preventable. When the relations are not yet normalized and normalization is not satisfied, there will be redundant data that anomalies to the modified data. These can occur when we insert, modify, and delete data. As you can see if the normal forms do not comply with the database, we will have several inconsistencies with the data, violating data integrity.

Insertion Anomalies

This is when redundant information gets updated to several relations that can be natural joined to form one relation. There are other ways of describing this such as needing NULL values on certain attributes in order to insert new relations.

Update Anomalies

This update anomaly type can occur often when there are a set of tuples that join to several other relations. When we modify a tuple, we expect for the other tuples in the relation to have the same update. The attribute values in the single relation will appear in all of tuples so they must be changed in all of these tuples in order for the data to continue its integrity. For instance, if we were the update the attributes of a 'Problem' will need to be updated with its relations such as the 'Ticket' in order to keep all of its parts up to date and correct.

Deletion Anomalies

Deleting from relations that have reliant attributes from its relations to one another, we must ensure that what we delete from one set of tuples will need to reflect from that every single relation to it will be removed. If there is no complete removal of these from the database then there will be deep inconsistencies that in database that reflect incorrect information.

3.1.2 Third Normal or Boyce-Codd Forms for this Database

After learning the properties of normalization and anomalies, we can now present which entities comply with which forms. When looking at these normalizations, we need to analyze

these schemas in our database to satisfy the criteria for Third Normal or the stricter: Boyce-Codd form.

Call

Functional Dependencies:

FD1. Trivial; {**generatedID**} → {recording, totalCallDuration, callDescription, keyPointOfConcern, subPointOfConcern}

FD2. {recording} → {**generatedID**}

Candidate Keys:

generatedID, recording

Normal Form

Checking all the normalization rules, we have concluded that:

- First Normal Form (1NF) is is satisfied, all values are simple with atomic domain.
- Second Normal Form (2NF) is satisfied because the primary key: **generatedID** contains one attribute.
- Third Normal Form (3NF) is satisfied because there are no prime attributes depending on other attributes.
- Boyce-Codd Normal Form (BCNF) is satisfied because there will be no anomalies from modification

Conclusively, this relation has passed the normalization tests in our database system.

OXYsquare Account

Functional Dependencies

FD1. Trivial; {**accountID**} → {emailAddress, phoneNumber, password, name, address, region}

FD2. {emailAddress} → {**accountID**}

Candidate Keys

accountID, emailAddress

Normal Form

Checking all the normalization rules, we have concluded that:

- First Normal Form (1NF) is is satisfied, all values are simple with atomic domain.
- Second Normal Form (2NF) is satisfied because the primary key: **accountID** contains one attribute.
- Third Normal Form (3NF) is satisfied because there are no prime attributes depending on other attributes.

- Boyce-Codd Normal Form (BCNF) is satisfied because there will be no anomalies from modification

Conclusively, this relation has passed the normalization tests in our database system.

Agent

Functional Dependencies

FD1. Trivial; {**accountID**} → {sDate, eDate, agentTier, agentName}

Candidate Keys

agentID

Normal Form

Checking all the normalization rules, we have concluded that:

- First Normal Form (1NF) is is satisfied, all values are simple with atomic domain.
- Second Normal Form (2NF) is satisfied because the primary key: **agentID** contains one attribute.
- Third Normal Form (3NF) is satisfied because there are no prime attributes depending on other attributes.
- Boyce-Codd Normal Form (BCNF) is satisfied because there will be no anomalies from modification

Conclusively, this relation has passed the normalization tests in our database system.

Ticket

Functional Dependencies

FD1. Trivial; {**ticketID**} → {ticketStatus}

Candidate Keys

ticketID

Normal Form

Checking all the normalization rules, we have concluded that:

- First Normal Form (1NF) is is satisfied, all values are simple with atomic domain.
- Second Normal Form (2NF) is satisfied because the primary key: **ticketID** contains one attribute.
- Third Normal Form (3NF) is satisfied because there are no prime attributes depending on other attributes.
- Boyce-Codd Normal Form (BCNF) is satisfied because there will be no anomalies from modification

Conclusively, this relation has passed the normalization tests in our database system.

Game

Functional Dependencies

FD1. Trivial; {**gameID**} → {gameName, gameSoftwareVersion, gamePlatform, gameMSRP, gamePublisher}

FD2. {gameName} → {**gameID**}

Candidate Keys

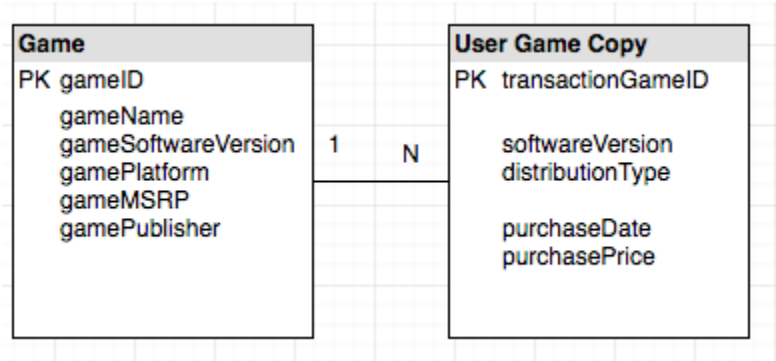
gameID, gameName

Normal Form

Checking all the normalization rules, we have concluded that:

- First Normal Form (1NF) is is satisfied, all values are simple with atomic domain.
- Second Normal Form (2NF) is satisfied because the primary key: **gameID** contains one attribute.
- Third Normal Form (3NF) is satisfied because there are no prime attributes depending on other attributes.
- Boyce-Codd Normal Form (BCNF) is satisfied because there will be no anomalies from modification

Previously, we had a game reflect all attributes for the entailed game. It did not follow 3NF so we needed to reflect the standard normalization. We broke down this relation and created a User Game Copy entity to reflect the needed change. Conclusively, this relation has passed the normalization tests in our database system. Below is the new relation: User Game Copy created for following 3NF:



User Game Copy

Functional Dependencies

FD1. Trivial; {**transactionGameID**} → {distributionType, purchaseDate, purchasePrice}

Candidate Keys
transactionGameID

Normal Form
Checking all the normalization rules, we have concluded that:

- First Normal Form (1NF) is is satisfied, all values are simple with atomic domain.
- Second Normal Form (2NF) is satisfied because the primary key: **transactionGameID** contains one attribute.
- Third Normal Form (3NF) is satisfied because there are no prime attributes depending on other attributes.
- Boyce-Codd Normal Form (BCNF) is satisfied because there will be no anomalies from modification

Conclusively, this relation has passed the normalization tests in our database system.

Hardware

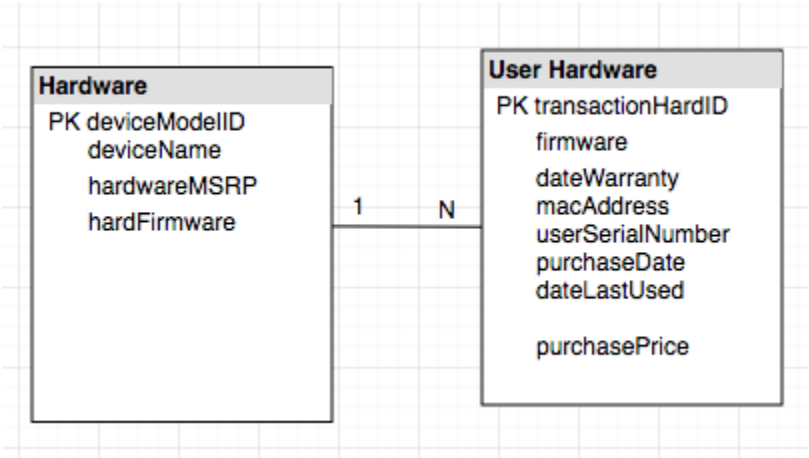
Functional Dependencies
FD1. Trivial; {**deviceModelID**} → {deviceName, hardwareMSRP, hardFirmware}
FD2. {deviceName} → {**deviceModelID**}

Candidate Keys
deviceModelID, deviceName

Normal Form
Checking all the normalization rules, we have concluded that:

- First Normal Form (1NF) is is satisfied, all values are simple with atomic domain.
- Second Normal Form (2NF) is satisfied because the primary key: **deviceModelID** contains one attribute.
- Third Normal Form (3NF) is satisfied because there are no prime attributes depending on other attributes.
- Boyce-Codd Normal Form (BCNF) is satisfied because there will be no anomalies from modification

Previously, we had a hardware reflect all attributes for the entailed hardware. It did not follow 3NF so we needed to reflect the standard normalization. We broke down this relation and created a User Hardware entity to reflect the needed change. Conclusively, this relation has passed the normalization tests in our database system. Below you can see the new split of Hardware to follow 3NF normalization rules:



User Hardware

Functional Dependencies

FD1. Trivial; {**transactionHardID**} → {firmware,dateWarranty ,macAddress, userSerialNumber, purchaseDate, dateLastUsed, purchasePrice}

FD2. {macAddress} → {**transactionHardID**}

FD3. {userSerialNumber} → {**transactionHardID**}

Candidate Keys

transactionHardID, macAddress, and userSerialNumber

Normal Form

Checking all the normalization rules, we have concluded that:

- First Normal Form (1NF) is is satisfied, all values are simple with atomic domain.
- Second Normal Form (2NF) is satisfied because the primary key: **transactionHardID** contains one attribute.
- Third Normal Form (3NF) is satisfied because there are no prime attributes depending on other attributes.
- Boyce-Codd Normal Form (BCNF) is satisfied because there will be no anomalies from modification

Conclusively, this relation has passed the normalization tests in our database system.

Solution

Functional Dependencies

FD1. Trivial; {**solutionID**} → {solutionDescription}

Candidate Keys
solutionID

Normal Form
Checking all the normalization rules, we have concluded that:

- First Normal Form (1NF) is is satisfied, all values are simple with atomic domain.
- Second Normal Form (2NF) is satisfied because the primary key: **solutionID** contains one attribute.
- Third Normal Form (3NF) is satisfied because there are no prime attributes depending on other attributes.
- Boyce-Codd Normal Form (BCNF) is satisfied because there will be no anomalies from modification

Conclusively, this relation has passed the normalization tests in our database system.

Game Inquiry

Functional Dependencies
FD1. Trivial; {**problemID**} → {thirdPartyPublisher, inquiryType}
Candidate Keys
problemID
Normal Form
Checking all the normalization rules, we have concluded that:

- First Normal Form (1NF) is is satisfied, all values are simple with atomic domain.
- Second Normal Form (2NF) is satisfied because the primary key: **problemID** contains one attribute.
- Third Normal Form (3NF) is satisfied because there are no prime attributes depending on other attributes.
- Boyce-Codd Normal Form (BCNF) is satisfied because there will be no anomalies from modification

Conclusively, this relation has passed the normalization tests in our database system.

Malicious Activity

Functional Dependencies
FD1. Trivial; {**problemID**} → {activityType, securityIssue}

Candidate Keys
problemID

Normal Form

Checking all the normalization rules, we have concluded that:

- First Normal Form (1NF) is is satisfied, all values are simple with atomic domain.
- Second Normal Form (2NF) is satisfied because the primary key: **problemID** contains one attribute.
- Third Normal Form (3NF) is satisfied because there are no prime attributes depending on other attributes.
- Boyce-Codd Normal Form (BCNF) is satisfied because there will be no anomalies from modification

Conclusively, this relation has passed the normalization tests in our database system.

Hardware Inquiry

Functional Dependencies

FD1. Trivial; {**problemID**} → {internalPartInQuestion, inquiryType}

Candidate Keys

problemID

Normal Form

Checking all the normalization rules, we have concluded that:

- First Normal Form (1NF) is is satisfied, all values are simple with atomic domain.
- Second Normal Form (2NF) is satisfied because the primary key: **problemID** contains one attribute.
- Third Normal Form (3NF) is satisfied because there are no prime attributes depending on other attributes.
- Boyce-Codd Normal Form (BCNF) is satisfied because there will be no anomalies from modification

Conclusively, this relation has passed the normalization tests in our database system.

Account Issues

Functional Dependencies

FD1. Trivial; {**problemID**} → {accountType, accountStatus, accountIssue}

Candidate Keys

problemID

Normal Form

Checking all the normalization rules, we have concluded that:

- First Normal Form (1NF) is is satisfied, all values are simple with atomic domain.

- Second Normal Form (2NF) is satisfied because the primary key: **problemID** contains one attribute.
- Third Normal Form (3NF) is satisfied because there are no prime attributes depending on other attributes.
- Boyce-Codd Normal Form (BCNF) is satisfied because there will be no anomalies from modification

Conclusively, this relation has passed the normalization tests in our database system.

Billing

Functional Dependencies

FD1. Trivial; {**problemID**} → {refundType, billingIssue, creditCardAssociated, refundTypeRequested}

Candidate Keys

problemID

Normal Form

Checking all the normalization rules, we have concluded that:

- First Normal Form (1NF) is is satisfied, all values are simple with atomic domain.
- Second Normal Form (2NF) is satisfied because the primary key: **problemID** contains one attribute.
- Third Normal Form (3NF) is satisfied because there are no prime attributes depending on other attributes.
- Boyce-Codd Normal Form (BCNF) is satisfied because there will be no anomalies from modification

Conclusively, this relation has passed the normalization tests in our database system.

Hardware Repair

Functional Dependencies

FD1. Trivial; {**problemID**} → {warrantyAssociated, sercviceType, deviceIssue, servicedRegion, shipToAddress}

Candidate Keys

problemID

Normal Form

Checking all the normalization rules, we have concluded that:

- First Normal Form (1NF) is is satisfied, all values are simple with atomic domain.

- Second Normal Form (2NF) is satisfied because the primary key: **problemID** contains one attribute.
- Third Normal Form (3NF) is satisfied because there are no prime attributes depending on other attributes.
- Boyce-Codd Normal Form (BCNF) is satisfied because there will be no anomalies from modification

Conclusively, this relation has passed the normalization tests in our database system.

Troubleshooting

Functional Dependencies

FD1. Trivial; {**problemID**} → {interalPartAssociated, troubleshootingType, errorCode}

Candidate Keys

problemID

Normal Form

Checking all the normalization rules, we have concluded that:

- First Normal Form (1NF) is is satisfied, all values are simple with atomic domain.
- Second Normal Form (2NF) is satisfied because the primary key: **problemID** contains one attribute.
- Third Normal Form (3NF) is satisfied because there are no prime attributes depending on other attributes.
- Boyce-Codd Normal Form (BCNF) is satisfied because there will be no anomalies from modification

Conclusively, this relation has passed the normalization tests in our database system.

Modifies

Functional Dependencies

FD1. Trivial; {**agentID, ticketID**} → {sDate, sTime, agentAction}

Candidate Keys

{agentID, ticketID}

Normal Form

Checking all the normalization rules, we have concluded that:

- First Normal Form (1NF) is is satisfied, all values are simple with atomic domain.
- Second Normal Form (2NF) is satisfied because the primary key contains one attribute.

- Third Normal Form (3NF) is satisfied because there are no prime attributes depending on other attributes.
- Boyce-Codd Normal Form (BCNF) is satisfied because there will be no anomalies from modification

Conclusively, this relation has passed the normalization tests in our database system.

Initiates

Functional Dependencies

FD1. Trivial; {**generatedID, accountID**} → {date, sTime, eTime}

Candidate Keys

{**generatedID, accountID**}

Normal Form

Checking all the normalization rules, we have concluded that:

- First Normal Form (1NF) is is satisfied, all values are simple with atomic domain.
- Second Normal Form (2NF) is satisfied because the primary key contains one attribute.
- Third Normal Form (3NF) is satisfied because there are no prime attributes depending on other attributes.
- Boyce-Codd Normal Form (BCNF) is satisfied because there will be no anomalies from modification

Conclusively, this relation has passed the normalization tests in our database system.

Answered By

Functional Dependencies

FD1. Trivial; {**agentID, callID**} → {aDate, sTime, eTime}

Candidate Keys

{agentID, callID}

Normal Form

Checking all the normalization rules, we have concluded that:

- First Normal Form (1NF) is is satisfied, all values are simple with atomic domain.
- Second Normal Form (2NF) is satisfied because the primary key contains one attribute.
- Third Normal Form (3NF) is satisfied because there are no prime attributes depending on other attributes.
- Boyce-Codd Normal Form (BCNF) is satisfied because there will be no anomalies from modification

Conclusively, this relation has passed the normalization tests in our database system.

3.2 PSQL Main Purpose and Functionality

Now that we have established the necessities of normalization and proven that our database is a working model that abides to the rules for data integrity. As shown, with the model of our database, we will not have any anomalies in our data. Now the next step is to implement the database into action.

PostgreSQL is a object-relational database management system, allowing users to implement the physical implementation process of the database. PostgreSQL allows for DBAs to easily update, create, and maintain a database. It's simple querying functionality allows for users of the database system to easily produce and run reports. PostgreSQL can also allow for users to easily maintain its data through triggers, sequence generators, and stored procedures. Section 3.3 shows an in-depth detailed description of many features of Postgres.

3.3 Schema Objects for PSQL DBMS

Using PSQL, we were able to create many schema objects into our database. Looking at the schema objects that we have in our database we will go through the basic units of everything established such as: tables, views, functions, triggers, serial, schemas, indexes, and lastly, procedures, which entail: functions and triggers.

Tables

Tables are the most basic and main storage for our PSQL database. The data for each tuple of a relation is entered and stored into a table for the relation to connect to. In a table, the columns have columns for attributes such as: for the relation "Account" will have data for its attributes of: accountID, phone number, password, name, address and region.

<u>Syntax from PostgreSQL 9.1 Manual:</u>

CREATE TABLE <**table_name**> ([

{ **column_name data_type** [COLLATE **collation**] [**column_constraint** [...]]

| **table_constraint**

```
        | LIKE parent_table [ like_option ... ] }
        [, ... ]
] )
[ INHERITS ( parent_table [, ... ] ) ]
[ WITH ( storage_parameter [= value] [, ... ] ) | WITH OIDS | WITHOUT OIDS ]
[ ON COMMIT { PRESERVE ROWS | DELETE ROWS | DROP } ]
[ TABLESPACE tablespace ]
```

Written Example In Our Database:

```
CREATE TABLE hardware_copy(
        deviceModelID          INT,
        accountID              INT,
        transactionHardID SERIAL PRIMARY KEY,
        dateWarranty           DATE              NOT NULL,
        firmware               REAL              DEFAULT 1.0,
        macAddress             VARCHAR(12)       NOT NULL     UNIQUE,
        userSerialNumber       INTEGER           NOT NULL     UNIQUE,
        purchaseDate           DATE              NOT NULL,
        dateLastUsed           DATE                      NOT NULL,
        purchasePrice          REAL              NOT NULL,

        CONSTRAINT fk_hardware_account FOREIGN KEY (accountID) REFERENCES
account(accountID),
        CONSTRAINT fk_hard_id FOREIGN KEY (deviceModelID)  REFERENCES
hardware(deviceModelID),
        CONSTRAINT ck_hardware_serial CHECK(userSerialNumber > 0),
        CONSTRAINT ck_date CHECK(dateLastUsed > purchaseDate)
);
```

Some Examples in the Database Implementation:

- Account
- Game
- Ticket
- Call
- Agent
- Problem
- Solution

- Hardware

Views

Views are the results of what you see in a query, that are derived from other tables called the base tables, where we store the physical data, as described above. These views are shown as a virtual table as they do not store any data, but they use the query command written with the select statement to show the data that one queries to see. This way the database administrator can use these queries to view only the selection of the data that they are requesting for. This is very useful when applying to the front end applications from the views instead of actually applying it to our tables of raw data. We never want to do that as it may pose risk to the integrity of our data. When we have information in the base tables changed, views can be recreated to show what we need.

Syntax from PostgreSQL 9.1 Manual:

CREATE VIEW **name** [(**column_name** [, ...])]
  [WITH (**view_option_name** [= **view_option_value**] [, ...])]
  AS **query**

Triggers

This is the CREATE TRIGGER statement which is used to establish any specific automatic actions applied to our database system when there are specific events that can occur. This is needed as an active database with specific events can alter the database system. Trigger is used to monitor the database and implement these actions.

Syntax from PostgreSQL 9.1 Manual:

CREATE [CONSTRAINT] TRIGGER **name** { BEFORE | AFTER | INSTEAD OF } { **event** [OR ...] }
  ON **table**
  [FROM **referenced_table_name**]
  [NOT DEFERRABLE | [DEFERRABLE] { INITIALLY IMMEDIATE | INITIALLY DEFERRED }]
  [FOR [EACH] { ROW | STATEMENT }]
  [WHEN (**condition**)]
  EXECUTE PROCEDURE **function_name (arguments)**

where **event** can be one of:

  INSERT
  UPDATE [OF column_name [, ...]]

DELETE
TRUNCATE

Stored Procedures

A stored procedures are modules that enforce the rules of the database system. Stored procedures can enhance modeling power and allow for more complex data types for database users. The can also be used to check constraints more complex than triggers. These stored procedures are useful when:

- there is a database program used by many different applications that can invoke and store data. This will improve software modularity and improve efficiency.
- executing a program at a server. Using a stored procedure can lower the communication cost and reduce data transfer between a client and a host.

Syntax from PostgreSQL Tutorial:

```
CREATE FUNCTION function_name(p1 type, p2 type)
        RETURNS type AS
        BEGIN
         -- logic
        END;
        LANGUAGE language_name;
```

Sequence Generator

A sequence generator will create a table by using a mathematical function to make a series of unique values. When the sequence generator is used it will increment the next number. These sequence generators will generate unique values like primary keys for the schemas.

Syntax from PostgreSQL 9.1 Manual:

CREATE SEQUENCE **name** [INCREMENT [BY] **increment**]
[MINVALUE **minvalue** | NO MINVALUE] [MAXVALUE **maxvalue** | NO MAXVALUE]
[START [WITH] **start**] [CACHE **cache**] [[NO] CYCLE]
[OWNED BY { **table_name.column_name** | NONE }]

Indexes

An Index is used to contain the entries for every value of an indexed column. Creating these indexes will allow for speeding of queries. Indexes are independent from the nuances of the database and can be removed and created at any time. It provides us to access to rows quickly for program efficiency. Different indexes will provide for better queries.

Using PostgreSQL, we have several index types available:

- B-tree
- Hash
- GiST
- SP-GiST
- GIN
- BRIN

Syntax from PostgreSQL 9.1 Manual:

CREATE [UNIQUE] INDEX [CONCURRENTLY] [**name**] ON **table** [USING **method**]
({ **column** | (**expression**) } [COLLATE **collation**] [**opclass**] [ASC | DESC]
[NULLS { FIRST | LAST }] [, ...])
[WITH (**storage_parameter = value** [, ...])]
[TABLESPACE **tablespace**]
[WHERE **predicate**]

3.4 List Relations With PSQL Commands

Account

```
                    Table "public.account"
  Column    |  Type   |                    Modifiers
-------------+---------+-----------------------------------------------------------
 accountid   | integer | not null default nextval('account_accountid_seq'::regclass)
 emailadd    | text    | not null
 phonenumber | bigint  |
 password    | text    | not null
 name        | text    | not null
 address     | text    | not null
 region      | text    | not null
Indexes:
   "account_pkey" PRIMARY KEY, btree (accountid)
   "account_emailadd_key" UNIQUE, btree (emailadd)
Referenced by:
   TABLE "game_copy" CONSTRAINT "fk_account_id" FOREIGN KEY (accountid) REFERENCES
account(accountid)
   TABLE "initiates" CONSTRAINT "fk_account_id" FOREIGN KEY (accountid) REFERENCES
account(accountid)
```

```
   TABLE "ticket" CONSTRAINT "fk_account_ticket" FOREIGN KEY (accountid) REFERENCES
account(accountid)
   TABLE "hardware_copy" CONSTRAINT "fk_hardware_account" FOREIGN KEY (accountid)
REFERENCES account(accountid)


 accountid |          emailadd           | phonenumber |    password     |      name       |  address   |
region
-----------+-----------------------------+-------------+-----------------+-----------------+------------+--------
         1 | anna_poon@yahoo.com         |  5063238000 | lovelybird123   | Anna Poon       |
California | Asia
         2 | tomatomaster123@gmail.com   |  6788923672 | ilovetomatoes   | George Gregory
| Utah       | US
         3 | pork-dumplings548@gmail.com |  9093458923 | password123     | George Orwell
| Spain      | EU
         4 | dfgnsdpowejr@yahoo.com      |  9999999999 | t%634t@44dd=d   | l33tH@cker      |
4Chan      | Moon
         5 | robert-parker12@yahoo.com   |  6820004567 | grimreaper1574  | Robert J. Parker  |
Washington | US
         6 | youtubergod99@hotmail.com   |  7088883456 | 456YrdsdW       | Hitsune Miku      |
Texas      | US
         7 | microsoftxboxwins@gmail.com |  9999999999 | grimreaper1574  | Robert J. Parker
| Washington | US
         8 | DanielClementine05@facebook.com |  6783419021 | ilovetoparty123 | Daniel
Clementine | Beijing    | Africa
(8 rows)
```

Agent

```
                                    Table "public.agent"
     Column      |      Type      |                    Modifiers                    | Storage  | Description
-----------------+----------------+-------------------------------------------------+----------+-------------
 agenttransfersid | integer        |                                                 | plain    |
 agentid          | integer        | not null default nextval('agent_agentid_seq'::regclass) | plain    |
 sdate            | date           | not null                                        | plain    |
 edate            | date           |                                                 | plain    |
 agenttier        | integer        | default 1                                       | plain    |
 agentname        | character(100) | not null                                        | extended |
```

```
Indexes:
   "agent_pkey" PRIMARY KEY, btree (agentid)
Check constraints:
   "ck_agent_dates" CHECK (edate >= sdate OR edate = NULL::date)
Foreign-key constraints:
   "fk_agent_call" FOREIGN KEY (agenttransfersid) REFERENCES agent(agentid)
Referenced by:
   TABLE "agent" CONSTRAINT "fk_agent_call" FOREIGN KEY (agenttransfersid) REFERENCES
agent(agentid)
   TABLE "modifies" CONSTRAINT "fk_agent_id" FOREIGN KEY (agentid) REFERENCES
agent(agentid)
   TABLE "answered_by" CONSTRAINT "fk_agent_id" FOREIGN KEY (agentid) REFERENCES
agent(agentid)
Has OIDs: no


 agenttransfersid | agentid |   sdate    |   edate    | agenttier |
agentname

------------------+---------+------------+------------+-----------+-------------------------------------------------------
----------------------------------------
----
                  |       1 | 2013-03-25 | 2018-03-23 |         1 | Greg Thompson

                  |       2 | 2014-12-25 |            |         2 | Sandra Lee

                  |       3 | 2016-01-15 |            |         3 | Lisa Lu

                  |       4 | 2016-06-25 |            |         1 | Christina Gutierrez

(4 rows)
```

Call

```
                                                    Table "public.call"
     Column       |  Type   |                    Modifiers                    | Storage  | Description
-------------------+---------+-------------------------------------------------------+----------+-------------
 generatedid       | integer | not null default nextval('call_generatedid_seq'::regclass) | plain    |
```

```
 recording        | text    |                                          | extended |
 totalcallduration | integer | not null                                 | plain    |
 calldescription   | text    | not null                                 | extended |
 keypointofconcern | text    | not null                                 | extended |
 subpointofconcern | text    | not null                                 | extended |
Indexes:
   "call_pkey" PRIMARY KEY, btree (generatedid)
   "call_recording_key" UNIQUE, btree (recording)
Referenced by:
   TABLE "answered_by" CONSTRAINT "fk_call_id" FOREIGN KEY (generatedid) REFERENCES
call(generatedid)
   TABLE "initiates" CONSTRAINT "fk_call_id" FOREIGN KEY (generatedid) REFERENCES
call(generatedid)
Has OIDs: no




 generatedid |      recording       | totalcallduration |                          calldescription
| keypointofconcer
n  | subpointofconcern
-------------+----------------------+-------------------+-------------------------------------------------------------------
----------------------+-----------------
---+-------------------
         1 | recording-123456.mp3 |               410 | Consumer is banned, and is requesting an
appeal                              | Malicious Activi
ty | Hacking
         2 | recording-123457.mp3 |               308 | Consumer is banned, follow up
| Malicious Activi
ty | Hacking
         3 | recording-123460.mp3 |               530 | Customer called about Software Issue with PS4
| Troubleshooting
   | PS4
         4 | recording-123458.mp3 |               407 | Fradulent charges, were charged to the
cusomter's credit card                      | Billing
   | Credit Card
         5 | recording-123459.mp3 |               783 | Fradulent charges, follow up
| Billing
```

| Credit Card
6 | recording-123467.mp3 | 444 | The customer was displeased with his Angel Heaven 5 Game purchase and requests a refund. | Billing
| Refund
7 | recording-123468.mp3 | 603 | The customer is seeking warranty services for his console. | Hardware Repair
| PS4
8 | recording-123469.mp3 | 617 | The customer forgot his password
| Account Issues
| Security
9 | recording-123470.mp3 | 471 | The customer is asking the release date for Sanic the Turtle 2 | Game Inquiry
| Release Date
10 | recording-123472.mp3 | 389 | The customer wants to know when the new controller for PS4 is coming out | Hardware Inquiry
| Controller
11 | recording-123478.mp3 | 371 | There is an issue with setting up the PS4 with an HD TV | Troubleshooting
| Hardware
12 | recording-123490.mp3 | 388 | The customer is seeking to change his acount name. | Account Issues
| Name Change
(12 rows)

Game

```
                    Table "public.game"
     Column       |  Type   |                  Modifiers                  | Storage  | Description
---------------------+---------+-----------------------------------------------------+----------+-------------
 gameid              | integer | not null default nextval('game_gameid_seq'::regclass) | plain    |
 gamename            | text    | not null                                | extended |
 gamesoftwareversion | real    | default 1.0                             | plain    |
 gamemsrp            | real    | not null                                | plain    |
 gamepublisher       | text    | not null                                | extended |
 gameplatform        | text    |                                         | extended |
Indexes:
   "game_pkey" PRIMARY KEY, btree (gameid)
Referenced by:
```

```
   TABLE "game_copy" CONSTRAINT "fk_game_id" FOREIGN KEY (gameid) REFERENCES
game(gameid)
Has OIDs: no


 gameid |     gamename     | gamesoftwareversion | gamemsrp | gamepublisher |
gameplatform
--------+------------------+---------------------+----------+---------------+--------------
     1 | Anna's Adventure |                   1 |       80 | Ynos Media    | PS4
     2 | Sanic The Turtle |                   1 |       45 | Capcam        | PS4
     3 | Angel Heaven 5   |                 1.3 |      300 | Circle Enix   | PS VITA
     4 | Judge Dread      |                 3.6 |       10 | Ynos Media    | PS4
(4 rows)
```

Game_Copy

```
                                    Table "public.game_copy"
     Column       |  Type   |                          Modifiers                          | Storage  | Description
------------------+---------+-----------------------------------------------------------------------+----------+-----------
--
 gameid           | integer |                                                             | plain    |
 accountid        | integer |                                                             | plain    |
 transactiongameid | integer | not null default
nextval('game_copy_transactiongameid_seq'::regclass) | plain    |
 softwareversion  | real    | default 1.0                                                 | plain    |
 distributiontype | text    | not null                                                    | extended |
 purchasedate     | date    | not null                                                    | plain    |
 purchaseprice    | real    | not null                                                    | plain    |
Indexes:
   "game_copy_pkey" PRIMARY KEY, btree (transactiongameid)
Check constraints:
   "dist_type" CHECK (distributiontype = ANY (ARRAY['Digital'::text, 'Physical'::text]))
Foreign-key constraints:
   "fk_account_id" FOREIGN KEY (accountid) REFERENCES account(accountid)
   "fk_game_id" FOREIGN KEY (gameid) REFERENCES game(gameid)
Referenced by:
   TABLE "ticket" CONSTRAINT "fk_ticket_game" FOREIGN KEY (transactiongameid) REFERENCES
game_copy(transactiongameid)
Has OIDs: no
```

```
 gameid | accountid | transactiongameid | softwareversion | distributiontype | purchasedate |
purchaseprice
--------+-----------+-------------------+-----------------+------------------+--------------+---------------
     1 |         1 |                 1 |               1 | Physical         | 2016-12-25   |          40.3
     2 |         1 |                 2 |               1 | Digital          | 2015-11-23   |          25.6
     3 |         1 |                 3 |               1 | Digital          | 2015-10-06   |          20.6
     4 |         1 |                 4 |               1 | Digital          | 2016-10-06   |           5.6
     1 |         2 |                 5 |               1 | Physical         | 2016-08-03   |          42.3
     2 |         2 |                 6 |               1 | Physical         | 2016-11-13   |          23.6
     4 |         2 |                 7 |               1 | Digital          | 2015-11-06   |           7.6
     1 |         3 |                 8 |               1 | Physical         | 2015-04-11   |          20.3
     1 |         4 |                 9 |               1 | Physical         | 2016-12-25   |          70.3
     2 |         4 |                10 |               1 | Digital          | 2015-11-23   |          21.6
     3 |         4 |                11 |               1 | Digital          | 2015-10-06   |         280.6
     4 |         4 |                12 |               1 | Digital          | 2016-10-06   |           5.6
     1 |         5 |                13 |               1 | Physical         | 2016-12-25   |          45.3
     4 |         5 |                14 |               1 | Digital          | 2017-01-16   |           0
     1 |         6 |                15 |               1 | Physical         | 2016-03-01   |          56.3
     2 |         6 |                16 |               1 | Digital          | 2015-11-24   |          30.6
     3 |         6 |                17 |               1 | Physical         | 2016-01-06   |           3.6
     4 |         6 |                18 |               1 | Digital          | 2015-05-26   |          12.6
(18 rows)
```

Hardware

```
                                Table "public.hardware"
   Column     |  Type   |                      Modifiers                      | Storage  | Description
--------------+---------+-----------------------------------------------------+----------+-------------
 devicemodelid | integer | not null default nextval('hardware_devicemodelid_seq'::regclass) |
plain    |
 devicename    | text    | not null                                            | extended |
 hardfirmware  | real    | default 1.0                                         | plain    |
 hardwaremsrp  | real    | not null                                            | plain    |
Indexes:
   "hardware_pkey" PRIMARY KEY, btree (devicemodelid)
Referenced by:
```

```
   TABLE "hardware_copy" CONSTRAINT "fk_hard_id" FOREIGN KEY (devicemodelid)
REFERENCES hardware(devicemodelid)
Has OIDs: no

 devicemodelid |      devicename       | hardfirmware | hardwaremsrp
---------------+-----------------------+--------------+--------------
           1 | PS4                   |          3 |         300
           2 | PS Vita               |          2 |         150
           3 | Tripleshock Controller |        1.5 |          40
(3 rows)
```

Hardware_Copy

```
                                    Table "public.hardware_copy"
     Column       |        Type         |                          Modifiers                          | Storage  |
Description
-------------------+----------------------+-----------------------------------------------------------------------------+-----
-----+-------------
 devicemodelid     | integer             |                                                             | plain    |
 accountid         | integer             |                                                             | plain    |
 transactionhardid | integer             | not null default
nextval('hardware_copy_transactionhardid_seq'::regclass) | plain    |
 datewarranty      | date                | not null                                                    | plain    |
 firmware          | real                | default 1.0                                                 | plain    |
 macaddress        | character varying(12) | not null                                                  |
extended |
 userserialnumber  | integer             | not null                                                    | plain    |
 purchasedate      | date                | not null                                                    | plain    |
 datelastused      | date                | not null                                                    | plain    |
 purchaseprice     | real                | not null                                                    | plain    |
Indexes:
   "hardware_copy_pkey" PRIMARY KEY, btree (transactionhardid)
   "hardware_copy_macaddress_key" UNIQUE, btree (macaddress)
   "hardware_copy_userserialnumber_key" UNIQUE, btree (userserialnumber)
Check constraints:
   "ck_date" CHECK (datelastused > purchasedate)
   "ck_hardware_serial" CHECK (userserialnumber > 0)
```

Foreign-key constraints:
"fk_hard_id" FOREIGN KEY (devicemodelid) REFERENCES hardware(devicemodelid)
"fk_hardware_account" FOREIGN KEY (accountid) REFERENCES account(accountid)
Referenced by:
TABLE "ticket" CONSTRAINT "fk_ticket_hardware" FOREIGN KEY (transactionhardid) REFERENCES hardware_copy(transactionhardid)
Has OIDs: no

```
 devicemodelid | accountid | transactionhardid | datewarranty | firmware |  macaddress  |
userserialnumber | purchasedate | datelastused | purchaseprice
---------------+-----------+-------------------+--------------+----------+--------------+------------------+--------------
+--------------+---------------
          1 |         1 |                 1 | 2016-12-21   |        3 | CC612FB1C4F5 |          817936 | 2014-12-
21   | 2018-01-21   |          320
          2 |         1 |                 2 | 2015-12-22   |        2 | 424869925E8A |          577147 | 2014-12-
22   | 2017-06-11   |          140
          3 |         1 |                 3 | 2015-12-21   |      1.5 | 48F3331BCA1A |          123694 | 2014-12-
21   | 2018-02-21   |           30
          1 |         2 |                 4 | 2016-06-21   |        2 | 6AF63BE82907 |          712062 | 2014-06-
21   | 2018-03-21   |          280
          3 |         2 |                 5 | 2015-06-21   |      1.5 | 773E9715BE7A |          729023 | 2014-06-
21   | 2018-01-21   |           20
          2 |         3 |                 6 | 2015-03-12   |      1.8 | 6A3E952CCA31 |          210390 | 2014-
03-12   | 2017-06-12   |          135
          1 |         4 |                 7 | 2017-01-21   |        3 | A5D6F461BE9B |          486724 | 2015-01-
21   | 2018-01-21   |          340
          2 |         4 |                 8 | 2016-01-22   |      1.8 | A0CEF71EDF87 |          669864 | 2015-01-
22   | 2017-06-11   |           40
          3 |         4 |                 9 | 2017-01-21   |      1.4 | A2DD9E44661D |          381495 | 2016-
01-21   | 2018-02-21   |           20
          1 |         5 |                10 | 2016-06-15   |        3 | 22822CA94434 |          499911 | 2014-06-
15   | 2018-04-21   |          180
          3 |         5 |                11 | 2015-06-15   |      1.5 | 05CFAB4CB657 |          585008 | 2014-
06-15   | 2018-04-21   |           25
          1 |         6 |                12 | 2016-10-21   |      2.7 | 35404B79F1A1 |          632612 | 2014-
10-21   | 2018-04-12   |          280
          2 |         6 |                13 | 2015-10-22   |        2 | 8024F9977401 |          935794 | 2014-10-
22   | 2017-04-12   |          120
```

```
          3 |        6 |             14 | 2015-10-21   |      1.4 | 230A34743620 |         936994 | 2014-
10-21   | 2018-04-12   |        27.76
(14 rows)
```

Ticket

```
                                Table "public.ticket"
     Column       |  Type   |                    Modifiers                    | Storage  | Description
------------------+---------+-------------------------------------------------------------+----------+-------------
 accountid        | integer |                                                 | plain    |
 transactiongameid | integer |                                                 | plain    |
 transactionhardid | integer |                                                 | plain    |
 ticketid         | integer | not null default nextval('ticket_ticketid_seq'::regclass) | plain    |
 ticketstatus     | text    |                                                 | extended |
Indexes:
   "ticket_pkey" PRIMARY KEY, btree (ticketid)
Check constraints:
   "ticket_status" CHECK (ticketstatus = ANY (ARRAY['Open'::text, 'Closed'::text, 'Other'::text]))
Foreign-key constraints:
   "fk_account_ticket" FOREIGN KEY (accountid) REFERENCES account(accountid)
   "fk_ticket_game" FOREIGN KEY (transactiongameid) REFERENCES
game_copy(transactiongameid)
   "fk_ticket_hardware" FOREIGN KEY (transactionhardid) REFERENCES
hardware_copy(transactionhardid)
Referenced by:
   TABLE "modifies" CONSTRAINT "fk_ticket_id" FOREIGN KEY (ticketid) REFERENCES
ticket(ticketid)
   TABLE "game_inquiry" CONSTRAINT "fk_ticket_id" FOREIGN KEY (ticketid) REFERENCES
ticket(ticketid)
   TABLE "malicious_activity" CONSTRAINT "fk_ticket_id" FOREIGN KEY (ticketid) REFERENCES
ticket(ticketid)
   TABLE "hardware_inquiry" CONSTRAINT "fk_ticket_id" FOREIGN KEY (ticketid) REFERENCES
ticket(ticketid)
   TABLE "account_issues" CONSTRAINT "fk_ticket_id" FOREIGN KEY (ticketid) REFERENCES
ticket(ticketid)
```

```
   TABLE "billing" CONSTRAINT "fk_ticket_id" FOREIGN KEY (ticketid) REFERENCES
ticket(ticketid)
   TABLE "hardware_repair" CONSTRAINT "fk_ticket_id" FOREIGN KEY (ticketid) REFERENCES
ticket(ticketid)
   TABLE "troubleshooting" CONSTRAINT "fk_ticket_id" FOREIGN KEY (ticketid) REFERENCES
ticket(ticketid)
Has OIDs: no

 select * from ticket;
 accountid | transactiongameid | transactionhardid | ticketid | ticketstatus
-----------+-------------------+-------------------+----------+--------------
         1 |                   |                   |        1 | Open
         2 |                   |                   |        2 | Open
         3 |                   |                   |        3 | Open
         4 |                11 |                   |        4 | Open
         5 |                   |                10 |        5 | Closed
         6 |                   |                   |        6 | Open
         6 |                   |                   |        7 | Open
         3 |                   |                   |        8 | Open
         6 |                   |                12 |        9 | Open
         2 |                   |                   |       10 | Open
(10 rows)
```

Modifies

```
                                Table "public.modifies"
   Column    |         Type         |                    Modifiers                    | Storage  | Description
-------------+----------------------+-------------------------------------------------+----------+-------------
 agentid     | integer              |                                                 | plain    |
 ticketid    | integer              |                                                 | plain    |
 mdate       | date                 | not null default ('now'::text)::date            | plain    |
 mtime       | time without time zone | not null default ('now'::text)::time with time zone | plain
|
 agentaction | text                 |                                                 | extended |
Check constraints:
   "agent_action" CHECK (agentaction = ANY (ARRAY['Create'::text, 'Close'::text, 'Follow
Up'::text, 'Other'::text]))
```

```
Foreign-key constraints:
    "fk_agent_id" FOREIGN KEY (agentid) REFERENCES agent(agentid)
    "fk_ticket_id" FOREIGN KEY (ticketid) REFERENCES ticket(ticketid)
Has OIDs: no


 agentid | ticketid |   mdate    |  mtime   | agentaction
---------+----------+------------+----------+-------------
       1 |        1 | 2017-12-21 | 12:17:30 | Create
       1 |        1 | 2017-03-11 | 08:32:02 | Follow Up
       2 |        2 | 2017-04-20 | 09:02:31 | Create
       3 |        3 | 2017-06-15 | 12:30:31 | Create
       2 |        3 | 2017-12-16 | 08:47:31 | Follow Up
       4 |        4 | 2017-11-02 | 12:25:31 | Create
       2 |        5 | 2017-08-03 | 11:17:31 | Create
       3 |        6 | 2017-04-29 | 03:16:31 | Create
       3 |        7 | 2017-04-12 | 04:16:31 | Create
       2 |        8 | 2017-12-09 | 05:18:31 | Create
       3 |        9 | 2017-10-01 | 12:30:31 | Create
       2 |       10 | 2017-02-04 | 11:26:31 | Create
       4 |        5 | 2017-08-05 | 12:25:31 | Close
(13 rows)
```

Answered_By

```
                          Table "public.answered_by"
   Column    |        Type         |              Modifiers               | Storage | Description
-------------+-----------------------+----------------------------------------+---------+-------------
 generatedid | integer             |                                      | plain   |
 agentid     | integer             |                                      | plain   |
 adate       | date                | default ('now'::text)::date          | plain   |
 astarttime  | time without time zone | default ('now'::text)::time with time zone | plain   |
 aendtime    | time without time zone |                                      | plain   |
Foreign-key constraints:
    "fk_agent_id" FOREIGN KEY (agentid) REFERENCES agent(agentid)
    "fk_call_id" FOREIGN KEY (generatedid) REFERENCES call(generatedid)
Has OIDs: no
```

```
 generatedid | agentid |   adate    | astarttime | aendtime
-------------+---------+------------+------------+----------
           1 |       1 | 2017-12-21 | 12:10:31   | 12:15:31
           2 |       1 | 2017-03-11 | 08:14:12   | 08:30:02
           3 |       2 | 2017-04-20 | 08:06:11   | 08:15:06
           4 |       3 | 2017-06-15 | 12:10:31   | 12:22:31
           5 |       2 | 2017-11-16 | 10:15:31   | 10:30:31
           6 |       4 | 2017-11-02 | 12:05:31   | 12:25:31
           7 |       2 | 2017-08-03 | 10:10:31   | 11:15:31
           8 |       3 | 2017-04-29 | 03:10:31   | 03:13:31
           9 |       3 | 2017-04-12 | 04:10:31   | 04:15:31
          10 |       2 | 2017-12-09 | 05:10:31   | 05:15:31
          11 |       3 | 2017-10-01 | 12:00:31   | 12:03:31
          12 |       2 | 2017-02-04 | 11:10:31   | 11:16:31
(12 rows)
```

Initiates

```
                                Table "public.initiates"
   Column    |         Type         |                    Modifiers                    | Storage | Description
-------------+----------------------+-------------------------------------------------+---------+-------------
 accountid   | integer              |                                                 | plain   |
 generatedid | integer              |                                                 | plain   |
 idate       | date                 | not null default ('now'::text)::date            | plain   |
 istarttime  | time without time zone | not null default ('now'::text)::time with time zone | plain
|
 iendtime    | time without time zone |                                                 | plain   |
Foreign-key constraints:
    "fk_account_id" FOREIGN KEY (accountid) REFERENCES account(accountid)
    "fk_call_id" FOREIGN KEY (generatedid) REFERENCES call(generatedid)
Has OIDs: no

 accountid | generatedid |   idate    | istarttime | iendtime
-----------+-------------+------------+------------+----------
```

```
     1 |        1 | 2017-12-21 | 12:09:40   | 12:15:31
     1 |        2 | 2017-03-11 | 08:10:12   | 08:30:02
     2 |        3 | 2017-04-20 | 08:04:11   | 08:15:06
     3 |        4 | 2017-06-15 | 12:08:31   | 12:22:31
     3 |        5 | 2017-11-16 | 10:13:31   | 10:30:31
     4 |        6 | 2017-11-02 | 12:03:31   | 12:25:31
     5 |        7 | 2017-08-03 | 10:08:31   | 11:15:31
     6 |        8 | 2017-04-29 | 03:05:31   | 03:13:31
     6 |        9 | 2017-04-12 | 04:01:31   | 04:15:31
     3 |       10 | 2017-12-09 | 05:10:31   | 05:15:31
     6 |       11 | 2017-10-01 | 11:57:31   | 12:03:31
     2 |       12 | 2017-02-04 | 11:08:31   | 11:16:31
(12 rows)
```

Solution

```
                                Table "public.solution"
     Column        |  Type   |                      Modifiers                      | Storage  | Description
---------------------+---------+-----------------------------------------------------------+----------+-------------
 solutionid          | integer | not null default nextval('solution_solutionid_seq'::regclass) | plain
|
 solutiondescription | text    | not null                                          | extended |
Indexes:
   "solution_pkey" PRIMARY KEY, btree (solutionid)
Referenced by:
   TABLE "game_inquiry" CONSTRAINT "fk_problem_sol" FOREIGN KEY (solutionid) REFERENCES
solution(solutionid)
   TABLE "malicious_activity" CONSTRAINT "fk_problem_sol" FOREIGN KEY (solutionid)
REFERENCES solution(solutionid)
   TABLE "hardware_inquiry" CONSTRAINT "fk_problem_sol" FOREIGN KEY (solutionid)
REFERENCES solution(solutionid)
   TABLE "account_issues" CONSTRAINT "fk_problem_sol" FOREIGN KEY (solutionid)
REFERENCES solution(solutionid)
   TABLE "billing" CONSTRAINT "fk_problem_sol" FOREIGN KEY (solutionid) REFERENCES
solution(solutionid)
```

```
   TABLE "hardware_repair" CONSTRAINT "fk_problem_sol" FOREIGN KEY (solutionid)
REFERENCES solution(solutionid)
   TABLE "troubleshooting" CONSTRAINT "fk_problem_sol" FOREIGN KEY (solutionid)
REFERENCES solution(solutionid)
Has OIDs: no

 solutionid |                              solutiondescription
------------+-----------------------------------------------------------------------------------------------
          1 | Verify that account in question is banned, if it is, submit an appeal request form.
          2 | Diagnose software issue, by asking questions such as what is working and not working.
          3 | Verify fraudulent charges, then make a request to Financial department to investigate.
          4 | Inform the customer that she can make a refund request if the game is physical.
          5 | Verify warranty date is still in effect, if it is, forward to Hardware Repair
          6 | Ask the relevant questions to verify identtify, if so send request link or provide
instructions.
          7 | Give the customer the relevant date for the game release, if it was publicly
announced.
          8 | Give the customer the relevant date for the hardware release, if it was publicly
announced.
          9 | Diagnose the situation by asking which items are not working, follow through Guide
No. 1482
         10 | Inform the user that this is not possible.
(10 rows)
```

Game_Inquiry

```
                                Table "public.game_inquiry"
     Column        |  Type   |                      Modifiers                       | Storage  | Description
---------------------+---------+-------------------------------------------------------------------+----------+-------------
 solutionid          | integer |                                                      | plain    |
 ticketid            | integer |                                                      | plain    |
 problemid           | integer | not null default nextval('game_inquiry_problemid_seq'::regclass) |
plain    |
 focusareaid         | integer | not null                                             | plain    |
 problemdescription  | text    | not null                                             | extended |
 thirdpartypublisher | text    |                                                      | extended |
```

```
 inquirytype        | text    |                                                  | extended |
Indexes:
    "game_inquiry_pkey" PRIMARY KEY, btree (problemid)
Foreign-key constraints:
    "fk_problem_sol" FOREIGN KEY (solutionid) REFERENCES solution(solutionid)
    "fk_ticket_id" FOREIGN KEY (ticketid) REFERENCES ticket(ticketid)
Has OIDs: no

 solutionid | ticketid | problemid | focusareaid |                  problemdescription                  |
thirdpartypublisher | inquirytype
------------+----------+-----------+-------------+----------------------------------------------------------+-------------
--------+--------------
         7 |        7 |         1 |          75 | Customer is asking the release date for Sanic the Turtle 2 |
Ynos                | Release Date
(1 row)
```

Malicious_Activity

```
                                    Table "public.malicious_activity"
      Column       |  Type   |                          Modifiers                          | Storage  | Description
--------------------+---------+-------------------------------------------------------------------------+----------+---------
----
 solutionid         | integer |                                                             | plain    |
 ticketid           | integer |                                                             | plain    |
 problemid          | integer | not null default
nextval('malicious_activity_problemid_seq'::regclass) | plain    |
 focusareaid        | integer | not null                                                    | plain    |
 problemdescription | text    | not null                                                    | extended |
 activitytype       | text    |                                                             | extended |
 securityissue      | text    |                                                             | extended |
Indexes:
    "malicious_activity_pkey" PRIMARY KEY, btree (problemid)
Foreign-key constraints:
    "fk_problem_sol" FOREIGN KEY (solutionid) REFERENCES solution(solutionid)
    "fk_ticket_id" FOREIGN KEY (ticketid) REFERENCES ticket(ticketid)
Has OIDs: no
```

```
 solutionid | ticketid | problemid | focusareaid |          problemdescription          | activitytype
| securityissue
------------+----------+-----------+-------------+--------------------------------------+--------------+---------------
         1 |        1 |         1 |          79 | The user is banned, because of cheating. | Cheating     |
Banned
(1 row)
```

Hardware_Inquiry

```
                                  Table "public.hardware_inquiry"
       Column        |  Type   |                        Modifiers                        | Storage  | Description
-----------------------+---------+-------------------------------------------------------------------------+----------+-------
------
 solutionid           | integer |                                                         | plain    |
 ticketid             | integer |                                                         | plain    |
 problemid            | integer | not null default
nextval('hardware_inquiry_problemid_seq'::regclass) | plain    |
 focusareaid          | integer | not null                                                | plain    |
 problemdescription   | text    | not null                                                | extended |
 internalpartinquestion | text    |                                                         | extended |
 inquirytype          | text    |                                                         | extended |
Indexes:
   "hardware_inquiry_pkey" PRIMARY KEY, btree (problemid)
Foreign-key constraints:
   "fk_problem_sol" FOREIGN KEY (solutionid) REFERENCES solution(solutionid)
   "fk_ticket_id" FOREIGN KEY (ticketid) REFERENCES ticket(ticketid)
Has OIDs: no


 solutionid | ticketid | problemid | focusareaid |                problemdescription
| internalpartinquestion | inquirytype
------------+----------+-----------+-------------+-------------------------------------------------------------------+----
--------------------+--------------
         8 |        8 |         1 |          56 | The customer is asking the release date for the new PS4
controller. | Controller           | Release Date
(1 row)
```

Account_Issues

```
                                Table "public.account_issues"
     Column       |  Type   |                        Modifiers                        | Storage  | Description
--------------------+---------+-------------------------------------------------------------------+----------+-------------
 solutionid        | integer |                                                          | plain    |
 ticketid          | integer |                                                          | plain    |
 problemid         | integer | not null default nextval('account_issues_problemid_seq'::regclass) |
plain    |
 focusareaid       | integer | not null                                                 | plain    |
 problemdescription | text    | not null                                                 | extended |
 accounttype       | text    |                                                          | extended |
 accountstatus     | text    |                                                          | extended |
 accountissue      | text    |                                                          | extended |
Indexes:
   "account_issues_pkey" PRIMARY KEY, btree (problemid)
Check constraints:
   "account_status" CHECK (accountstatus = ANY (ARRAY['Active'::text, 'Suspended'::text,
'Banned'::text, 'Other'::text]))
   "account_type" CHECK (accounttype = ANY (ARRAY['Master'::text, 'Sub'::text, 'Other'::text]))
Foreign-key constraints:
   "fk_problem_sol" FOREIGN KEY (solutionid) REFERENCES solution(solutionid)
   "fk_ticket_id" FOREIGN KEY (ticketid) REFERENCES ticket(ticketid)
Has OIDs: no

 solutionid | ticketid | problemid | focusareaid |              problemdescription              |
accounttype | accountstatus |   accountissue
------------+----------+-----------+-------------+---------------------------------------------+-------------+----
-----------+-----------------
          6 |        6 |         1 |          10 | The customer forgot his password.                | Master      |
Active       | password failure
         10 |       10 |         2 |          89 | The customer is seeking to change his account name.. |
Master      | Active       | username change
(2 rows)
```

Billing

```
                                Table "public.billing"
      Column        |  Type   |                    Modifiers                    | Storage  | Description
----------------------+---------+-------------------------------------------------------------+----------+-------------
 solutionid          | integer |                                                | plain    |
 ticketid            | integer |                                                | plain    |
 problemid           | integer | not null default nextval('billing_problemid_seq'::regclass) | plain
|
 focusareaid         | integer | not null                                       | plain    |
 problemdescription  | text    | not null                                       | extended |
 refundtype          | text    |                                                | extended |
 billingissue        | text    |                                                | extended |
 creditcardassociated | bigint  |                                               | plain    |
 refundtyperequested  | text    |                                               | extended |
Indexes:
   "billing_pkey" PRIMARY KEY, btree (problemid)
Check constraints:
   "billing_creditcardassociated_check" CHECK (floor(log(abs(creditcardassociated)::double
precision) + 1::double precision) >= 16::double precision AND
floor(log(abs(creditcardassociated)::double precision) + 1::double precision) <= 16::double
precision)
   "refund_type" CHECK (refundtype = ANY (ARRAY['Credit Card'::text, 'PayPal'::text,
'None'::text, 'Store Credit'::text]))
   "refund_type_requested" CHECK (refundtyperequested = ANY (ARRAY['Credit Card'::text,
'PayPal'::text, 'None'::text, 'Store Credit'::text]))
Foreign-key constraints:
   "fk_problem_sol" FOREIGN KEY (solutionid) REFERENCES solution(solutionid)
   "fk_ticket_id" FOREIGN KEY (ticketid) REFERENCES ticket(ticketid)
Has OIDs: no


 solutionid | ticketid | problemid | focusareaid |                problemdescription
| refundtype  | billingissue | creditcardassocia
ted | refundtyperequested
------------+----------+-----------+-------------+-------------------------------------------------------------------+------
-------+--------------+------------------
----+---------------------
```

```
        3 |       3 |        1 |         80 | The account is suspected to have been charged fradulently.
| Credit Card | Fraud       |     3412034201239
920 | Credit Card
        4 |       4 |        2 |         67 | The customer was not happy with the game and is seeking a
refund. | Credit Card | Refund       |     2112067201238
920 | Credit Card
(2 rows)
```

Hardware_Repair

```
                                Table "public.hardware_repair"
     Column       |  Type   |                        Modifiers                        | Storage  | Description
--------------------+---------+-------------------------------------------------------------------------+----------+-------------
-
 solutionid        | integer |                                                         | plain    |
 ticketid          | integer |                                                         | plain    |
 problemid         | integer | not null default nextval('hardware_repair_problemid_seq'::regclass)
| plain    |
 focusareaid       | integer | not null                                                | plain    |
 problemdescription | text    | not null                                                | extended |
 warrantyassociated | text    |                                                         | extended |
 servicetype       | text    |                                                         | extended |
 deviceissue       | text    |                                                         | extended |
 servicedregion    | text    |                                                         | extended |
Indexes:
   "hardware_repair_pkey" PRIMARY KEY, btree (problemid)
Check constraints:
   "servicedregion" CHECK (servicedregion = ANY (ARRAY['North America'::text, 'Africa'::text,
'Asia'::text, 'Australia'::text, 'Europe'::text, 'South America'::text]))
   "servicetype" CHECK (servicetype = ANY (ARRAY['Refurbished'::text, 'Replacement'::text,
'New'::text, 'NULL'::text]))
   "warrantyassociated" CHECK (warrantyassociated = ANY (ARRAY['OEM'::text, 'OOW'::text,
'EW'::text]))
Foreign-key constraints:
   "fk_problem_sol" FOREIGN KEY (solutionid) REFERENCES solution(solutionid)
   "fk_ticket_id" FOREIGN KEY (ticketid) REFERENCES ticket(ticketid)
Has OIDs: no
```

```
 solutionid | ticketid | problemid | focusareaid |            problemdescription              |
warrantyassociated | servicetype |     deviceissue     | service
dregion
------------+----------+-----------+-------------+---------------------------------------------+--------------------+--
-----------+--------------------+--------
--------
         5 |        5 |         1 |          73 | Customer has reported a failure on the product. | OOW
| NULL       | The PS4 wont start. | North A
merica
(1 row)
```

Troubleshooting

```
                                    Table "public.troubleshooting"
       Column        |  Type   |                        Modifiers                        | Storage  | Description
------------------------+---------+-------------------------------------------------------------------+----------+--------
-----
 solutionid           | integer |                                                          | plain    |
 ticketid             | integer |                                                          | plain    |
 problemid            | integer | not null default
nextval('troubleshooting_problemid_seq'::regclass) | plain    |
 focusareaid          | integer | not null                                                 | plain    |
 problemdescription     | text    | not null                                                 | extended |
 internalpartassociated | text    |                                                          | extended |
 troubleshootingtype    | text    |                                                          | extended |
 errorcode            | integer |                                                          | plain    |
Indexes:
   "troubleshooting_pkey" PRIMARY KEY, btree (problemid)
Foreign-key constraints:
   "fk_problem_sol" FOREIGN KEY (solutionid) REFERENCES solution(solutionid)
   "fk_ticket_id" FOREIGN KEY (ticketid) REFERENCES ticket(ticketid)
Has OIDs: no
```

```
 solutionid | ticketid | problemid | focusareaid |                problemdescription                |
internalpartassociated | troubleshootingtype | er
rorcode
------------+----------+-----------+-------------+------------------------------------------------------------+---------
---------------+--------------------+---
--------
         2 |        2 |         1 |          11 | The customer's PS4 is not working.                        | PS4
| Software           |
     1
         9 |        9 |         2 |          72 | The customer is having trouble setting up the PS4 with a
HDTV. | PS4                  | Hardware           |
     3
(2 rows)
```

3.5 Example Queries in PSQL

1. List all accounts that have at least two games with game MSRP with price > $70

```
SELECT a.*
FROM account a INNER JOIN userGameCopy gC1 on t.transactionGameID =
gC1.transactionGameID
INNER JOIN game g1 on g1.gameID = gC1.gameID
INNER JOIN game g2 on g2.transactionGameID <> g1.transactionGameID
INNER JOIN userGameCopy gC2 on g2.gameID = gC2.gameID AND gC2.gameID <> gc1.gameID
WHERE g1.price > 70 AND g2.price > 70;

 accountid |       emailadd        | phonenumber |   password    |    name    | address | region
-----------+-----------------------+-------------+---------------+------------+---------+--------
         4 | dfgnsdpowejr@yahoo.com |  9999999999 | t%634t@44dd=d | l33tH@cker | 4Chan   |
Moon
(1 row)
```

2. List Agents that have exactly one ticket with a game that costs > $250.

```
SELECT a1.agentName, a1.agentID

FROM (agent a1

Natural JOIN modifies m
Natural JOIN ticket t
Natural JOIN game_copy g)

EXCEPT
SELECT a.agentName, a.agentID
FROM (agent a

Natural JOIN modifies m
Natural JOIN ticket t
Natural JOIN game_copy g)

CROSS JOIN

(agent a2

Natural JOIN modifies m2
Natural JOIN ticket t2
Natural JOIN game_copy g2)

WHERE g.purchasePrice > 250 AND g2.purchasePrice > 250 AND t.ticketID != t2.ticketID;
```

```
                    agentname                    | agentid
-----------------------------------------------------------------------------------------------------+---------
 Christina Gutierrez                                                                                 |       4
(1 row)
```

3. List all American accounts who made at least one call between 8:00AM and 9:30AM PDT.

```
SELECT a.accountID, a.emailAdd, a.region
FROM account a INNER JOIN initiates i on a.accountID = i.accountID
INNER JOIN call c on c.generatedID = i.generatedID
WHERE a.region = 'US' AND i.iStartTime > '8:00:00'::time AND i.iEndTime< '9:30:00'::time;
```

```
 accountid |        emailadd         | region
-----------+-------------------------+--------
         2 | tomatomaster123@gmail.com | US
(1 row)
```

4. List all the names of games that were tickets with 'Billing Issues' that relate to game publisher 'CircleEnix' .

```
SELECT distinct gameName, gamepublisher, billingissue
FROM game NATURAL JOIN game_copy
NATURAL JOIN ticket
NATURAL JOIN billing
WHERE gamepublisher = 'Circle Enix';
```

```
   gamename    | gamepublisher | billingissue
---------------+---------------+-------------
 Angel Heaven 5 | Circle Enix   | Refund
(1 row)
```

5. List the accounts in Asia with second least expensive game from all game publishers.

```
SELECT a.accountID, a.region, a.name, g1.gamename, g1.gameMSRP
```

```
FROM account a NATURAL JOIN game g1 NATURAL JOIN game_copy gC1 NATURAL JOIN game
g2 NATURAL JOIN game_copy gC2 NATURAL JOIN game g3 NATURAL JOIN game_copy gC3
WHERE a.region = 'Asia'

EXCEPT
SELECT a.accountID, a.region, a.name, g1.gamename, g1.gameMSRP
FROM account a NATURAL JOIN game g1 NATURAL JOIN game_copy gC1 NATURAL JOIN game
g2 NATURAL JOIN game_copy gC2 NATURAL JOIN game g3 NATURAL JOIN game_copy gC3
WHERE g3.gameMSRP < g1.gameMSRP;
```

```
 accountid | region |   name    |     gamename     | gamemsrp
-----------+--------+-----------+------------------+----------
         1 | Asia   | Anna Poon | Anna's Adventure |       80
         1 | Asia   | Anna Poon | Sanic The Turtle |       45
         1 | Asia   | Anna Poon | Angel Heaven 5   |      300
         1 | Asia   | Anna Poon | Judge Dread      |       10
(4 rows)
```

6. List accounts that have the 2nd most expensive game.

```
SELECT a.accountID, gc.transactionGameID, gc.purchasePrice
FROM game_copy gc
NATURAL JOIN account a
EXCEPT

SELECT a2.accountID, gc3.transactionGameID,  gc3.purchasePrice
FROM (SELECT gc3.*
        FROM game_copy gc3
        NATURAL JOIN account a3
        CROSS JOIN
        game_copy gc4
        WHERE  > gc3.purchasePrice > gc4.purchasePrice)
NATURAL JOIN account a2
```

```
CROSS JOIN
game_copy gc2
WHERE  > gc3.purchasePrice > gc2.purchasePrice;
```

```
 accountid | transactiongameid | purchaseprice
-----------+-------------------+---------------
         4 |                 9 |          70.3
         4 |                10 |          21.6
         4 |                11 |         280.6
         4 |                12 |           5.6
(4 rows)
```

7. List ticket inquiries who are connected by each and all US account

```
SELECT t.ticketID from ticket t
   where not in (select * from account a
      where exists (select * from ticket t2
         where a.accountID = t.accountID AND a.region = 'US'
         )
         and not in(
            select * from account a2
            where a2.accountID = t.accountID
         )
      )
ticketid
-----------
(0 rows)
```

- There are 0 rows, since the constraints were extremely specific.

8. List all accounts that have the cheapest games.

```
SELECT a.accountID, gc.transactionGameID, gc.purchasePrice
```

```
FROM game_copy gc
NATURAL JOIN account a
EXCEPT
SELECT a2.accountID, gc1.transactionGameID,  gc1.purchasePrice
FROM game_copy gc1
NATURAL JOIN account a2
CROSS JOIN
game_copy gc2
WHERE gc1.purchasePrice > gc2.purchasePrice;
```

```
 accountid | transactiongameid | purchaseprice
-----------+-------------------+---------------
         5 |                14 |             0
(1 row)
```

9. List all accounts that have tickets for each game with game publisher: XYZ.

```
SELECT a.accountID
FROM account a
NATURAL JOIN ticket t
NATURAL JOIN game_copy gc
NATURAL JOIN game g
GROUP BY a.accountID, a.name
HAVING COUNT (distinct g.gamePublisher) = (SELECT COUNT (*) FROM  'Ynos')
```

```
 accountid
-----------
(0 rows)
```

- There are 0 rows, since the constraints were extremely specific.

10. List agents that have been in every ticket in the asia region

```
SELECT a.agentID
FROM agent a
```

```
NATURAL JOIN answered_by ab
NATURAL JOIN call c
NATURAL JOIN initiates i
NATURAL JOIN account ac
GROUP BY a.agentID
HAVING COUNT (distinct ac.region) = (SELECT COUNT (*) FROM 'Asia' )

 agentid
-----------
(0 rows)
```

- There are 0 rows, since the constraints were extremely specific.

Additional Queries:

11. List the count of the agents by tier level that are higher than 2.

```
SELECT a.agentTier, count(a.agentTier)
FROM Agent a
GROUP BY agentTier
HAVING agentTier >= 2;

 agenttier | count
-----------+-------
        3 |     1
        2 |     1
(2 rows)
```

12. List all accounts with no game transactions.

```
SELECT *
FROM account
WHERE NOT EXISTS ( SELECT 1
                   FROM game_copy
```

```
WHERE account.accountID = game_copy.accountID);

 accountid |          emailadd          | phonenumber |    password    |      name      | address   |
region
-----------+----------------------------+-------------+----------------+----------------+-----------+--------
        7 | microsoftxboxwins@gmail.com     |  9999999999 | grimreaper1574  | Robert J. Parker
| Washington | US
        8 | DanielClementine05@facebook.com |  6783419021 | ilovetoparty123 | Daniel
Clementine | Beijing    | Africa
(2 rows)
```

3.6 Data Loader

When it comes right down to it, every database needs data to be inserted for it to be useful. There are several ways of doing this such as the normal INSERT method in PSQL which can be done through the terminal or by using pgAdmin and by software or "Front Ends" which are user interfaces to insert, delete, modify, or view relations. There can also be prebuilt functions pre compiled in the DBMS itself.

Insert PSQL Statements

As previously mentioned, the most basic way of inserting data to the database is by the insert method, in PSQL this method is extremely similar to traditional SQL based DBMS. Wherein

The command starts as follows:

```
INSERT INTO <table>  (<column name 1>, <column name 2>, … <column name n>)
VALUES (<expression 1>, <expression 2>, <expression n>);
```

There are also other variations such as,

```
INSERT INTO <table> VALUES (<expression 1>, <expression 2>, <expression n>);
```

Where n can range from 1600 to or more depending on the assigned block size.

As with expressions, queries can be used in place, thus a SELECT statement can be used inside to be placed into another relation.

pgAdmin 4

Although, these commands may seem the most flexible, just using these alone to insert information is impossible especially when new entries need to be filled in at a large rate. pgAdmin 4 is a great tool that provides a GUI to new and experienced database administrators. It enables the user to insert using CSV, BIN, or TEXT files. It is also a streamlined way of performing other commands such as setting triggers, writing scripts, dropping and all the other commands. The tool can also export using the same output file types. However, we still choose to do it by CLI and using .sql files to perform our operations.

Java Data Loader

Java among other popular languages have free publicly available libraries for users to create their own GUI data loading program or a front end. Dr. Huawing Wang has done this using his gradebook program. A loader java class is present there where it takes information from the CSUB website and can parse through and segment the data using delimiters and other specific keywords such as majors. This parsed arrays are then sent to DBConnection. To their respective functions to be inserted into the database. This form of loading speeds up data insertion, however it is limited to a format that needs to be followed.

I

PHASE 4 PostgreSQL Database Management System PL/SQL Components

In our last phase, we focused on the construction of the physical database with PostgreSQL. We mentioned the strengths of the PostgreSQL database and explored their basic components and features. We talked about the very basic functionalities of PostgreSQL, but we need to drive deeper to implement a professional, efficient, and useful database. In this phase, we will explore deeper into the capabilities of PostgreSQL and more advanced functionality, such as creation of data integrity constraints, business constraints, when manipulating our database to ensure correctness.

In this phase, we will drive deeper into implementing a complex, self-sufficient database that will not require continuous data clean-up. In this chapter we will discuss firstly: the purposes of PostgreSQL and highlight the pros and cons, secondly: the features and syntax for PL/pgSQL when implementing these into the database, thirdly: we will provide examples of these implementations to gauge a deeper comprehension of PostgreSQL, and finally: we will do a side by side comparison between SQL and PL/pgSQL and discuss PostgreSQL's overall domination in the industry.

4.1 Postgres PL/pgSQL

Postgres and its extension is as PostgreSQL claims "the most advanced open source relational database," due to its powerful uses over the last 30 years. PL/pgSQL is PostgreSQL's extension of the language SQL, mainly used by industry due to its extensibility and compliance standards. PL/pgSQL is similar to the Oracles PL/SQL as a procedural language for SQL, allowing extensions to load custom subroutines, called stored procedures to be executed. PostgreSQL has features of stored procedures that are precompiled to be executed whenever needed.

PostgreSQL provides the developer many advantages. Due to its support on the internet, PostgreSQL is easy to use and quick to learn. PostgreSQL provides precompiled code for stored procedures so PL/pgSQL will save time with execution and developers will save time when coding functions such as stored procedures. Additionally, due to the store procedures being stored in a database application, there is reusability in the code, thus the common case can be reused over and over again. The complex functions are condensed simpler functions that as a whole can be reused in different parts by different users. Lastly, PostgreSQL uses abstraction to hide all this complexity from the user, assisting the user with easy to learn code and protecting the functionality of code itself. In summary, the advantages and disadvantages are listed below:

The Advantages of PostgreSQL are:

- New and professional procedural language that is Open Source. It is a true open source that is not controlled by enterprise like Oracle
- Good Support for PostgreSQL procedural language with many strong features
- Strong Security:
    - Privileges: users, roles that will allow for you to separate the responsibilities and powers of different users
    - There is full database encryption for great security.
- Replication:
    - There are logical and physical replications
    - Point in Time Recovery
- Performance is strong, which advanced locking mechanisms, tablespaces and partitioned tables to work from
- Cross-platform capabilities

The Disadvantages of PostgreSQL are:

- There are not many disadvantages with this new language except: it is slower than MySQL and others

4.1.1 Program Structure and Control Statements

PL/pgSQL is a procedural language that is a growth of the original procedural language SQL. It uses blocks of code, each block with its own scope and variable, allowing for developers to name or unname certain blocks of code. These named blocks of code can encompass stored procedures and/or functions to be executed when needed. These database procedural language will have conditions and loops which will allow for function calling and error handling through database constraints.

Writing Block Structure

When writing the block in PostgreSQL syntax is as written:

```
[ <<label>> ]
[ DECLARE
      declarations ]
BEGIN
statements;
...
END [ label ];
```

This is the syntax used for PostgreSQL's block. Certain keywords are used such as DECLARE: declaring variables, END: ending the block's definition.

An example of possible written code for a PostgreSQL block is found below:

```
DO $$
<<first_block>>
DECLARE
      count integer := 0;
BEGIN
      count := count + 1;
      RAISE NOTICE 'The counter shows %', count;
END first_block $$;
```

Writing Conditions

When writing the For Loops and Conditions in PostgreSQL the syntax is written:

```
IF condition THEN
      statement;
END IF;
```

This is the syntax used for For Loops. The syntax is a condition is a boolean expression that will reflect as true or false. The statement in the for loop is the command that is to be executed when the condition is met. In this structure, we check the condition and if the condition returns true, we will execute the function, otherwise we will exit.

An example of the IF statement is:

```
if(feedbk >= 5 )
then
     fbLevel = 'Very Satisfactory';
end if;

if(feedbk >= 4 AND feedbk < 5)
then
     fbLevel = 'Satisfactory';
end if;

if(feedbk >= 3 AND feedbk < 4)
then
     fbLevel = 'Average';
end if;

if(feedbk >= 2 AND feedbk < 3)
then
     fbLevel = 'Unsatisfactory';
end if;

if(feedbk >= 1 AND feedbk < 2)
then
     fbLevel = 'Very Unsatisfactory';
end if;
```

Similarly, one can make IF THEN ELSE statements as well which will allow for you to have alternative statements to be executed only if the one condition has not been passed through:

```
IF condition THEN
     statements;
ELSE
     alternative-statements;
END IF;
```

An example of the IF/ELSE statement is:

```
if(feedbk >= 3 )
```

```
      then
            fbLevel = 'Satisfactory';
      else
            fbLevel = 'Not Satisfactory';
      end if;
```

Writing for CASE

There is also another way for one to execute statements like the IF statement called CASE statements which allow you to execute a block if the condition applies.

```
      CASE search-expression
      WHEN expression_1 [, expression_2, ...] THEN
            When-statements
      [ ... ]
      [ELSE
            else-statements ]
      END CASE;
```

An example of the case statement is:

```
CASE WHEN avg(problemarea) >= 3.1
            THEN 'Account Issues'
            WHEN avg(problemarea) >= 2.1
            THEN 'Billing'
            WHEN avg(problemarea) >= 1.1
            THEN 'Repair'
            WHEN avg(problemarea) >= 0
            THEN 'Troubleshooting'
                ELSE ''
            END AS Focus
```

Writing for Loop Statements

```
      <<label>>
      LOOP
```

```
        Statements;
    EXIT [<<label>>] WHEN condition;
    END LOOP;
```

4.1.2 Stored Functions

Postgres is useful in that you will be able to exercise database functionality with defined functions which is what we call stored procedures. These stored procedures will allow for you to make triggers and custom aggregate functions, providing you ability to make complex calculations and controlled structures. Doing so will allow for your to make your functions more effective and much easier.

Postgres will allow for you to use such things as safe languages like SQL, PL/pgSQL, and C in these procedures so you will be able to utilize them for PostgreSQL.

Advantages of using stored procedures are that it can assist with performance of the application, it can be used in different applications and it can simplify the path needed to take between application and database.

Using PL/pgSQL the syntax for the stored procedure is:

```
    CREATE FUNCTION function_name(p1 type, p2 type)
        RETURNS type AS
    BEGIN
        -- logic
    END;
    LANGUAGE language_name;
```

In this syntax we see that CREATE FUNCTION will be the indicator for when a function is to begin and to be defined. The return will show what type the function should return. The beginning and end will indicate to us the beginning and the end of the function we call. lastly , we will note what language the function is written in under the LANGUAGE labels.

An example of the stored function statement is:

```
CREATE OR REPLACE FUNCTION makeCallReport(
    funcCallID INT,
    funcStartDate DATE,
```

```
      funcEndDate DATE
      ) RETURNS VOID  AS
$$

DECLARE /* variables */
      callMins FLOAT;
      feedbk FLOAT;
      fbLevel TEXT;

BEGIN /*calculations */

      callMins = c.totalCallDuration
      from call c natural join initiates i
      where c.generatedid = funcCallID AND i.generatedid =
c.generatedid AND i.idate >= funcStartDate AND i.idate <=
funcEndDate;

      feedbk = c.customerFeedback
      from call c natural join initiates i
      where c.generatedid = funcCallID AND i.generatedid =
c.generatedid AND i.idate >= funcStartDate AND i.idate <=
funcEndDate;

      if(feedbk >= 5 )
      then
            fbLevel = 'Very Satisfactory';
      end if;

      if(feedbk >= 4 AND feedbk < 5)
      then
            fbLevel = 'Satisfactory';
      end if;

      if(feedbk >= 3 AND feedbk < 4)
      then
            fbLevel = 'Average';
      end if;

      if(feedbk >= 2 AND feedbk < 3)
```

```
      then
            fbLevel = 'Unsatisfactory';
      end if;

      if(feedbk >= 1 AND feedbk < 2)
      then
            fbLevel = 'Very Unsatisfactory';
      end if;


      if(callMins > 0)
      then
      insert into consumer_call_report(generatedID, callDurationMins,
customerFeedbackLevel)
      values(funcCallID, callMins/60, fbLevel);
      end if;


EXCEPTION
    WHEN others THEN
    RAISE 'makeCallReport Function Error :Adding call report number %
failed due to [%]', funcCallID,  SQLERRM ;
END;
$$ LANGUAGE plpgsql;
```

4.1.3 Stored Procedures

A stored procedure has similar syntax to the stored function. As seen below we have a stored procedure example:

An example of the stored procedure statement from PostgreSQL manual is:

```
CREATE FUNCTION somefunc() RETURNS integer AS $$
<< outerblock >>
DECLARE
    quantity integer := 30;
BEGIN
```

```
    RAISE NOTICE 'Quantity here is %', quantity;  -- Prints 30
    quantity := 50;
    --
    -- Create a subblock
    --
    DECLARE
        quantity integer := 80;
    BEGIN
        RAISE NOTICE 'Quantity here is %', quantity;  -- Prints 80
        RAISE NOTICE 'Outer quantity here is %', outerblock.quantity;
-- Prints 50
    END;

    RAISE NOTICE 'Quantity here is %', quantity;  -- Prints 50

    RETURN quantity;
END;
$$ LANGUAGE plpgsql;
```

There are differences in a stored procedure and a stored function. Stored procedures and stored functions are ultimately very similar but there are advantages and disadvantages in both. The advantages of a stored function is that we will be able to use them in a expression and return a value from it, whereas stored procedures do not have this functionality. Stored procedures are precompiled so that means that it will be more efficient to run these stored procedures. Stored procedures have advantages where they will be able to return multiple values in different OUT parameters and will be able to return more than one result set.

4.1.4 Packages

In PostgreSQL, there is no such thing as a package, however, the packages can be used through the safe languages as previously mentioned so that means that they can name functions or procedures similarly to SQL or other languages.

Examples given from PostgreSQL manual shows that SQL is:

```
CREATE OR REPLACE PACKAGE BODY acs
AS
  FUNCTION add_user (
    user_id     IN users.user_id%TYPE DEFAULT NULL,
```

```
    object_type     IN acs_objects.object_type%TYPE DEFAULT 'user',
    creation_date   IN acs_objects.creation_date%TYPE DEFAULT
sysdate,
    creation_user   IN acs_objects.creation_user%TYPE DEFAULT NULL,
    creation_ip     IN acs_objects.creation_ip%TYPE DEFAULT NULL,
  ...
  ) RETURN users.user_id%TYPE
  IS
    v_user_id       users.user_id%TYPE;
    v_rel_id        membership_rels.rel_id%TYPE;
  BEGIN
    v_user_id := acs_user.new (user_id, object_type, creation_date,
                creation_user, creation_ip, email, ...
    RETURN v_user_id;
  END;
END acs;
```

Thus similarly, in PostgreSQL manual, following the PostgreSQL syntax shows:

```
CREATE FUNCTION
acs__add_user(INTEGER,INTEGER,VARCHAR,TIMESTAMP,INTEGER,INTEGER,...)
RETURNS INTEGER AS '
DECLARE
    user_id ALIAS FOR $1;
    object_type ALIAS FOR $2;
    creation_date ALIAS FOR $3;
    creation_user ALIAS FOR $4;
    creation_ip ALIAS FOR $5;
    ...
    v_user_id users.user_id%TYPE;
    v_rel_id membership_rels.rel_id%TYPE;
BEGIN
    v_user_id :=
acs_user__new(user_id,object_type,creation_date,creation_user,creatio
n_ip, ...);
    ...

    RETURN v_user_id;
```

```
END;
' LANGUAGE 'plpgsql';
```

4.1.5 Triggers

Triggers are blocks of procedural languages where they are linked to specific tables or views of database. A trigger relies on the event that will be a cause and event for the database. When conditions are met then there will be an action whether it is an: insert, update, delete or truncate type of function. These are great uses for the database because it will maintain a certain maintenance for the database without needing human effort or interaction for maintenance of the database. These triggers can be associated before, after, or instead of. It is all from the design of the trigger function itself.

Using PL/pgSQL the syntax for the trigger procedure is:

```
      CREATE TRIGGER trigger_name {BEFORE | AFTER | INSTEAD OF}
     {event [OR ...]}
            ON table_name
            [FOR [EACH] {ROW | STATEMENT}]
            EXECUTE PROCEDURE trigger_function
```

On line 1, we can see that we declare a trigger through the “CREATE TRIGGER” phase, providing the nature of the trigger whether it is before, after, or instead of on a particular entity. Please note, that instead of is mainly for the insert, update and deleting on views.

An example of the trigger statement is:

```
CREATE or REPLACE closeTicket()
      returns trigger as
      $BODY$
      begin

      if solutiondescription <> NULL then
      insert into ticket values(old.accountid, old.transactiongameid,
old.transactionhardid, old.ticketid, new.status = 'CLOSED');
      end if;
      return new;
```

```
      exception
            when other then
            raise 'closing ticket function of solution id -%-% has
failed due to [%]', solutionid, SQLERRM;


      end;
      $BODY$
      $$
      language plpsql;
```

4.2 PostgreSQL PL/SQL Subprogram Examples

This section is for procedures, functions and triggers. We will provide three examples of the stored procedures that reflect on insert, delete, and average. We will also provide a before update, cascade deletion and instead of.

4.2.1 Stored Procedure: Insert

The insert procedure will insert a ticket into the database. The procedure returns the ticketid. It takes in three parameters a ticket status, a problem description, and a problem area.

```
CREATE OR REPLACE FUNCTION createTicket(
aticketStatus TEXT default 'Open',
aproblemDescription TEXT default 'Not Provided',
aproblemArea INT default null
      ) RETURNS VOID  AS
$$

DECLARE
tempTicketID INT;
   BEGIN
            INSERT INTO ticket(ticketStatus, problemDescription,
            problemArea) VALUES(aticketStatus, aproblemDescription,
            aproblemArea)
            RETURNING ticketID into tempTicketID;
    END;
```

```
$$ LANGUAGE plpgsql;
```

Before Insert:

```
apoon=> select * from ticket;
 transactiongameid | transactionhardid | ticketid | ticketstatus |             problemdescription              | problemarea
-------------------+-------------------+----------+--------------+---------------------------------------------+-------------
                   |                   |        1 | Open         | Game Installation Not Working               |           1
                   |                   |        2 | Open         | Game Installation Not Working               |           1
                   |                   |        3 | Open         | Downloadable Content Not Installing         |           1
                   |                   |        4 | Open         | Missing Trophies in Game                    |           1
                   |                   |        5 | Open         | Game Stuck in Load                          |           1
                   |                   |        6 | Open         | Game Installation Not Working               |           1
                   |                   |        7 | Open         | Firmware Download is Stuck                  |           1
                   |                   |        8 | Open         | Firmware Download is Stuck                  |           1
                   |                   |        9 | Open         | Firmware Download is Stuck                  |           1
                   |                   |       10 | Open         | Game Installation Not Working               |           1
                   |                   |       11 | Open         | Console overheating                         |           2
                   |                   |       12 | Open         | Console overheating                         |           2
                   |                   |       13 | Open         | Console overheating                         |           2
                   |                   |       14 | Open         | Controllers are not pairing                 |           2
                   |                   |       15 | Open         | Controllers are not pairing                 |           2
                   |                   |       16 | Open         | Console has blue rings of death             |           2
                   |                   |       17 | Open         | Console has blue rings of death             |           2
                   |                   |       18 | Open         | Console overheating                         |           2
                   |                   |       19 | Open         | Console port failure                        |           2
                   |                   |       20 | Open         | Console port failure                        |           2
                   |                   |       21 | Open         | Membership renewal, unauthorized            |           3
                   |                   |       22 | Open         | Membership renewal, unauthorized            |           3
                   |                   |       23 | Open         | Household member made unauthorized purchase |           3
                   |                   |       24 | Open         | Household member made unauthorized purchase |           3
                   |                   |       25 | Open         | Household member made unauthorized purchase |           3
                   |                   |       26 | Open         | Account owner made unauthorized purchase    |           3
                   |                   |       27 | Open         | Account owner made unauthorized purchase    |           3
                   |                   |       28 | Open         | Membership renewal, unauthorized            |           3
                   |                   |       29 | Open         | Household member made unauthorized purchase |           3
                   |                   |       30 | Open         | Account owner made unauthorized purchase    |           3
                   |                   |       31 | Open         | Account was stolen                          |           4
                   |                   |       32 | Open         | Password reset needed                       |           4
                   |                   |       33 | Open         | Password reset needed                       |           4
                   |                   |       34 | Open         | Account is Banned                           |           4
                   |                   |       35 | Open         | Invalid email registration                  |           4
                   |                   |       36 | Open         | Account was stolen                          |           4
                   |                   |       37 | Open         | Invalid email registration                  |           4
                   |                   |       38 | Open         | Invalid email registration                  |           1
                   |                   |       39 | Open         | Password reset needed                       |           4
                   |                   |       40 | Open         | Password reset needed                       |           4
                   |                   |       41 | Closed       | Password reset needed                       |           4
                   |                   |       42 | Closed       | Invalid email registration                  |           1
                   |                   |       43 | Open         | Invalid email registration                  |           4
                   |                   |       44 | Open         | Password reset needed                       |           4
                   |                   |       45 | Open         | Account is Banned                           |           4
                   |                   |       46 | Open         | Account is Banned                           |           4
```

After Insert:

```
apoon=> select createTicket('Open', 'Hacked account', 4);
 createticket
--------------

(1 row)

apoon=> select * from ticket;
 transactiongameid | transactionhardid | ticketid | ticketstatus |             problemdescription              | problemarea
-------------------+-------------------+----------+--------------+---------------------------------------------+-------------
                   |                   |        1 | Open         | Game Installation Not Working               |           1
                   |                   |        2 | Open         | Game Installation Not Working               |           1
                   |                   |        3 | Open         | Downloadable Content Not Installing         |           1
                   |                   |        4 | Open         | Missing Trophies in Game                    |           1
                   |                   |        5 | Open         | Game Stuck in Load                          |           1
                   |                   |        6 | Open         | Game Installation Not Working               |           1
                   |                   |        7 | Open         | Firmware Download is Stuck                  |           1
                   |                   |        8 | Open         | Firmware Download is Stuck                  |           1
                   |                   |        9 | Open         | Firmware Download is Stuck                  |           1
                   |                   |       10 | Open         | Game Installation Not Working               |           1
                   |                   |       11 | Open         | Console overheating                         |           2
                   |                   |       12 | Open         | Console overheating                         |           2
                   |                   |       13 | Open         | Console overheating                         |           2
                   |                   |       14 | Open         | Controllers are not pairing                 |           2
                   |                   |       15 | Open         | Controllers are not pairing                 |           2
                   |                   |       16 | Open         | Console has blue rings of death             |           2
                   |                   |       17 | Open         | Console has blue rings of death             |           2
                   |                   |       18 | Open         | Console overheating                         |           2
                   |                   |       19 | Open         | Console port failure                        |           2
                   |                   |       20 | Open         | Console port failure                        |           2
                   |                   |       21 | Open         | Membership renewal, unauthorized            |           3
                   |                   |       22 | Open         | Membership renewal, unauthorized            |           3
                   |                   |       23 | Open         | Household member made unauthorized purchase |           3
                   |                   |       24 | Open         | Household member made unauthorized purchase |           3
                   |                   |       25 | Open         | Household member made unauthorized purchase |           3
                   |                   |       26 | Open         | Account owner made unauthorized purchase    |           3
                   |                   |       27 | Open         | Account owner made unauthorized purchase    |           3
                   |                   |       28 | Open         | Membership renewal, unauthorized            |           3
                   |                   |       29 | Open         | Household member made unauthorized purchase |           3
                   |                   |       30 | Open         | Account owner made unauthorized purchase    |           3
                   |                   |       31 | Open         | Account was stolen                          |           4
                   |                   |       32 | Open         | Password reset needed                       |           4
                   |                   |       33 | Open         | Password reset needed                       |           4
                   |                   |       34 | Open         | Account is Banned                           |           4
                   |                   |       35 | Open         | Invalid email registration                  |           4
                   |                   |       36 | Open         | Account was stolen                          |           4
                   |                   |       37 | Open         | Invalid email registration                  |           4
                   |                   |       38 | Open         | Invalid email registration                  |           1
                   |                   |       39 | Open         | Password reset needed                       |           4
                   |                   |       40 | Open         | Password reset needed                       |           4
                   |                   |       41 | Closed       | Password reset needed                       |           4
                   |                   |       42 | Closed       | Invalid email registration                  |           1
                   |                   |       43 | Open         | Invalid email registration                  |           4
                   |                   |       44 | Open         | Password reset needed                       |           4
                   |                   |       45 | Open         | Account is Banned                           |           4
                   |                   |       46 | Open         | Account is Banned                           |           4
                   |                   |       47 | Open         | Hacked account                              |           4
```

4.2.2 Stored Procedure: Delete

The procedure will delete a ticket in the database. The procedure takes in the ticketid and deletes the specified ticket from the database.

```
CREATE OR REPLACE FUNCTION deleteTicket(

      giventicketid INT

) RETURNS VOID AS
$func$
      DECLARE
```

```
            focusid INT;
      BEGIN
            select problemArea into focusid from ticket where ticketID
            = giventicketid;

            Delete from ticket where ticketID = giventicketid;

            IF focusid = 4 THEN
                  DELETE from account_issues where ticketID =
                  giventicketid;
            ELSIF focusid = 3 THEN
                  DELETE from billing where ticketID = giventicketid;
            ELSIF focusid = 2 THEN
                  DELETE from hardware_repair where ticketID =
                  giventicketid;
            ELSIF focusid = 1 THEN
                  DELETE from troubleshooting where ticketID =
                  giventicketid;
            END IF;

      END;

$func$ LANGUAGE plpgsql;
```

Before Delete:

```
apoon=> select * from account_issues; select * from billing; select * from troubleshooting;
 solutionid | ticketid | datecreated | accountstatus |       accountissue       | solutionprovided
------------+----------+-------------+---------------+--------------------------+------------------
         31 |       31 | 2018-07-03  | Banned        | Hacked by Main Group     | Retrieved Account
         32 |       32 | 2018-11-03  | Active        | Lost Password            | Sent Email
         33 |       33 | 2018-04-01  | Active        | Lost Password            | Sent Email
         34 |       34 | 2018-12-03  | Banned        | Hacker: Do Not Unban     | No Action
         35 |       35 | 2018-11-03  | Active        | Incorrect Email          | Changed Email
         36 |       36 | 2018-12-01  | Banned        | Hacked by Main Group     | Retrieved Account
         37 |       37 | 2018-06-01  | Active        | Incorrect Email          | Changed Email
         38 |       38 | 2018-02-01  | Active        | Incorrect Email          | Changed Email
         39 |       39 | 2018-05-03  | Active        | Lost Password            | Sent Email
         40 |       40 | 2018-12-03  | Active        | Lost Password            | Sent Email
         41 |       41 | 2018-12-03  | Active        | Agent did not send email | Sent Email
         42 |       42 | 2018-06-11  | Active        | Agent did not send email | Changed Email
         43 |       43 | 2018-11-04  | Active        | Case was not worked      | Changed Email
         44 |       44 | 2018-04-03  | Active        | Agent did not sent email | Sent Email
         45 |       45 | 2018-12-27  | Banned        | Hacker: Do Not Unban     | No Action
         46 |       46 | 2018-12-28  | Banned        | Hacker: Do Not Unban     | No Action
(16 rows)

 solutionid | ticketid | datecreated |  refundtype  | refundtyperequested |   solutionprovided
------------+----------+-------------+--------------+---------------------+---------------------
         21 |       21 | 2018-01-30  | Credit Card  | Credit Card         | Refunded to Consumer
         22 |       22 | 2018-03-03  | Credit Card  | Credit Card         | Refunded to Consumer
         23 |       23 | 2018-04-03  | Store Credit | Credit Card         | Refunded to Consumer
         24 |       24 | 2018-08-26  | Store Credit | Credit Card         | Refunded to Consumer
         25 |       25 | 2018-12-03  | None         | Credit Card         | Denied Refund
         26 |       26 | 2018-12-03  | None         | Credit Card         | Denied Refund
         27 |       27 | 2018-12-06  | Store Credit | Store Credit        | Refunded to Consumer
         28 |       28 | 2018-03-03  | None         | Credit Card         | Denied Refund
         29 |       29 | 2018-07-03  | Credit Card  | Credit Card         | Refunded to Consumer
         30 |       30 | 2018-10-03  | Credit Card  | Credit Card         | Refunded to Consumer
(10 rows)

 solutionid | ticketid | datecreated |          solutionprovided          | internalpartassociated |          troubleshootingtype          | errorcode
------------+----------+-------------+------------------------------------+------------------------+---------------------------------------+-----------
          1 |        1 | 2018-12-03  | Free up memory for installation    | HDD                    | HDD Replacement                       |       123
          2 |        2 | 2018-12-03  | Free up memory for installation    | HDD                    | HDD Replacement                       |       123
          3 |        3 | 2018-12-03  | Connected to Internet              | N/A                    | Fixed Internet Connection             |       125
          4 |        4 | 2018-11-23  | Referred to Third Party            | N/A                    | Referred to Third Party               |       126
          5 |        5 | 2018-03-03  | Connected to Internet              | N/A                    | Referred to Internet Service Provider |       125
          6 |        6 | 2018-03-03  | Free up memory for installation    | HDD                    | HDD Replacement                       |       123
          7 |        7 | 2018-02-14  | Used external storage for firmware | N/A                    | Firmware Update                       |       124
          8 |        8 | 2018-12-03  | Used external storage for firmware | N/A                    | Firmware Update                       |       124
          9 |        9 | 2018-01-30  | Used external storage for firmware | N/A                    | Firmware Update                       |       124
         10 |       10 | 2018-02-03  | Free up memory for installation    | HDD                    | HDD Replacement                       |       123
(10 rows)
```

```
apoon=> select * from hardware_repair ; select * from ticket;
 solutionid | ticketid | datecreated |      solutionprovided       | warrantyassociated |      repairnotes
------------+----------+-------------+-----------------------------+--------------------+-----------------------
         11 |       11 | 2018-02-28  | Sent Console Replacement    | Out of Warranty    | Motherboard corrupted
         12 |       12 | 2018-03-03  | Sent Console Replacement    | Out of Warranty    | Motherboard corrupted
         13 |       13 | 2018-02-03  | Sent Console Replacement    | Out of Warranty    | Motherboard corrupted
         14 |       14 | 2018-11-03  | Sent Controller Replacement | In Warranty        | Wires destroyed
         15 |       15 | 2018-10-03  | Sent Controller Replacement | In Warranty        | Water Damage
         16 |       16 | 2018-07-03  | Replaced HDD                | Out of Warranty    | HDD corrupted
         17 |       17 | 2018-05-13  | Replaced HDD                | Out of Warranty    | HDD corrupted
         18 |       18 | 2018-12-03  | Sent Console Replacement    | Out of Warranty    | Motherboard damaged
         19 |       19 | 2018-04-30  | Sent Console Replacement    | Out of Warranty    | Port has been fried
         20 |       20 | 2018-12-03  | Sent Console Replacement    | Out of Warranty    | Port has been fried
(10 rows)

 transactiongameid | transactionhardid | ticketid | ticketstatus |             problemdescription              | problemarea
-------------------+-------------------+----------+--------------+---------------------------------------------+-------------
                   |                   |        1 | Open         | Game Installation Not Working               |           1
                   |                   |        2 | Open         | Game Installation Not Working               |           1
                   |                   |        3 | Open         | Downloadable Content Not Installing         |           1
                   |                   |        4 | Open         | Missing Trophies in Game                    |           1
                   |                   |        5 | Open         | Game Stuck in Load                          |           1
                   |                   |        6 | Open         | Game Installation Not Working               |           1
                   |                   |        7 | Open         | Firmware Download is Stuck                  |           1
                   |                   |        8 | Open         | Firmware Download is Stuck                  |           1
                   |                   |        9 | Open         | Firmware Download is Stuck                  |           1
                   |                   |       10 | Open         | Game Installation Not Working               |           1
                   |                   |       11 | Open         | Console overheating                         |           2
                   |                   |       12 | Open         | Console overheating                         |           2
                   |                   |       13 | Open         | Console overheating                         |           2
                   |                   |       14 | Open         | Controllers are not pairing                 |           2
                   |                   |       15 | Open         | Controllers are not pairing                 |           2
                   |                   |       16 | Open         | Console has blue rings of death             |           2
                   |                   |       17 | Open         | Console has blue rings of death             |           2
                   |                   |       18 | Open         | Console overheating                         |           2
                   |                   |       19 | Open         | Console port failure                        |           2
                   |                   |       20 | Open         | Console port failure                        |           2
                   |                   |       21 | Open         | Membership renewal, unauthorized            |           3
                   |                   |       22 | Open         | Membership renewal, unauthorized            |           3
                   |                   |       23 | Open         | Household member made unauthorized purchase |           3
                   |                   |       24 | Open         | Household member made unauthorized purchase |           3
                   |                   |       25 | Open         | Household member made unauthorized purchase |           3
                   |                   |       26 | Open         | Account owner made unauthorized purchase    |           3
                   |                   |       27 | Open         | Account owner made unauthorized purchase    |           3
                   |                   |       28 | Open         | Membership renewal, unauthorized            |           3
                   |                   |       29 | Open         | Household member made unauthorized purchase |           3
                   |                   |       30 | Open         | Account owner made unauthorized purchase    |           3
                   |                   |       31 | Open         | Account was stolen                          |           4
                   |                   |       32 | Open         | Password reset needed                       |           4
                   |                   |       33 | Open         | Password reset needed                       |           4
                   |                   |       34 | Open         | Account is Banned                           |           4
                   |                   |       35 | Open         | Invalid email registration                  |           4
                   |                   |       36 | Open         | Account was stolen                          |           4
                   |                   |       37 | Open         | Invalid email registration                  |           4
                   |                   |       38 | Open         | Invalid email registration                  |           1
                   |                   |       39 | Open         | Password reset needed                       |           4
                   |                   |       40 | Open         | Password reset needed                       |           4
                   |                   |       41 | Closed       | Password reset needed                       |           4
                   |                   |       42 | Closed       | Invalid email registration                  |           1
                   |                   |       43 | Open         | Invalid email registration                  |           4
                   |                   |       44 | Open         | Password reset needed                       |           4
                   |                   |       45 | Open         | Account is Banned                           |           4
                   |                   |       46 | Open         | Account is Banned                           |           4
```

After Delete:

```
apoon=> select deleteTicket(46);
 deleteticket
--------------

(1 row)
```

```
apoon=> select * from account_issues; select * from billing; select * from troubleshooting;
 solutionid | ticketid | datecreated | accountstatus |        accountissue        | solutionprovided
------------+----------+-------------+---------------+----------------------------+------------------
         31 |       31 | 2018-07-03  | Banned        | Hacked by Main Group       | Retrieved Account
         32 |       32 | 2018-11-03  | Active        | Lost Password              | Sent Email
         33 |       33 | 2018-04-01  | Active        | Lost Password              | Sent Email
         34 |       34 | 2018-12-03  | Banned        | Hacker: Do Not Unban       | No Action
         35 |       35 | 2018-11-03  | Active        | Incorrect Email            | Changed Email
         36 |       36 | 2018-12-01  | Banned        | Hacked by Main Group       | Retrieved Account
         37 |       37 | 2018-06-01  | Active        | Incorrect Email            | Changed Email
         38 |       38 | 2018-02-01  | Active        | Incorrect Email            | Changed Email
         39 |       39 | 2018-05-03  | Active        | Lost Password              | Sent Email
         40 |       40 | 2018-12-03  | Active        | Lost Password              | Sent Email
         41 |       41 | 2018-12-03  | Active        | Agent did not send email   | Sent Email
         42 |       42 | 2018-06-11  | Active        | Agent did not send email   | Changed Email
         43 |       43 | 2018-11-04  | Active        | Case was not worked        | Changed Email
         44 |       44 | 2018-04-03  | Active        | Agent did not sent email   | Sent Email
         45 |       45 | 2018-12-27  | Banned        | Hacker: Do Not Unban       | No Action
(15 rows)
```

transactiongameid	transactionhardid	ticketid	ticketstatus	problemdescription	problemarea
		1	Open	Game Installation Not Working	1
		2	Open	Game Installation Not Working	1
		3	Open	Downloadable Content Not Installing	1
		4	Open	Missing Trophies in Game	1
		5	Open	Game Stuck in Load	1
		6	Open	Game Installation Not Working	1
		7	Open	Firmware Download is Stuck	1
		8	Open	Firmware Download is Stuck	1
		9	Open	Firmware Download is Stuck	1
		10	Open	Game Installation Not Working	1
		11	Open	Console overheating	2
		12	Open	Console overheating	2
		13	Open	Console overheating	2
		14	Open	Controllers are not pairing	2
		15	Open	Controllers are not pairing	2
		16	Open	Console has blue rings of death	2
		17	Open	Console has blue rings of death	2
		18	Open	Console overheating	2
		19	Open	Console port failure	2
		20	Open	Console port failure	2
		21	Open	Membership renewal, unauthorized	3
		22	Open	Membership renewal, unauthorized	3
		23	Open	Household member made unauthorized purchase	3
		24	Open	Household member made unauthorized purchase	3
		25	Open	Household member made unauthorized purchase	3
		26	Open	Account owner made unauthorized purchase	3
		27	Open	Account owner made unauthorized purchase	3
		28	Open	Membership renewal, unauthorized	3
		29	Open	Household member made unauthorized purchase	3
		30	Open	Account owner made unauthorized purchase	3
		31	Open	Account was stolen	4
		32	Open	Password reset needed	4
		33	Open	Passward reset needed	4
		34	Open	Account is Banned	4
		35	Open	Invalid email registration	4
		36	Open	Account was stolen	4
		37	Open	Invalid email registration	4
		38	Open	Invalid email registration	1
		39	Open	Password reset needed	4
		40	Open	Password reset needed	4
		41	Closed	Password reset needed	4
		42	Closed	Invalid email registration	1
		43	Open	Invalid email registration	4
		44	Open	Passward reset needed	4
		45	Open	Account is Banned	4

4.2.3 Stored Procedure: Update

The update procedure will update a ticket in the database. The updateTicket procedure accepts 4 parameters. This procedure updates the ticket status, the problem description, and the problem area for a specific ticketid.

```
CREATE OR REPLACE FUNCTION updateTicket(
aticketid INT,
aticketStatus TEXT default 'Open',
aproblemDescription TEXT default 'Not Provided',
aproblemArea INT default null
     ) RETURNS VOID  AS
$$
   BEGIN
           UPDATE ticket
           SET ticketstatus = aticketStatus, problemdescription =
           aproblemDescription,problemarea = aproblemArea where
           ticketid = aticketid;
    END;
$$ LANGUAGE plpgsql;
```

Before Update:

```
apoon=> select * from ticket;
 transactiongameid | transactionhardid | ticketid | ticketstatus |             problemdescription              | problemarea
-------------------+-------------------+----------+--------------+---------------------------------------------+-------------
                   |                   |        1 | Open         | Game Installation Not Working               |           1
                   |                   |        2 | Open         | Game Installation Not Working               |           1
                   |                   |        3 | Open         | Downloadable Content Not Installing         |           1
                   |                   |        4 | Open         | Missing Trophies in Game                    |           1
                   |                   |        5 | Open         | Game Stuck in Load                          |           1
                   |                   |        6 | Open         | Game Installation Not Working               |           1
                   |                   |        7 | Open         | Firmware Download is Stuck                  |           1
                   |                   |        8 | Open         | Firmware Download is Stuck                  |           1
                   |                   |        9 | Open         | Firmware Download is Stuck                  |           1
                   |                   |       10 | Open         | Game Installation Not Working               |           1
                   |                   |       11 | Open         | Console overheating                         |           2
                   |                   |       12 | Open         | Console overheating                         |           2
                   |                   |       13 | Open         | Console overheating                         |           2
                   |                   |       14 | Open         | Controllers are not pairing                 |           2
                   |                   |       15 | Open         | Controllers are not pairing                 |           2
                   |                   |       16 | Open         | Console has blue rings of death             |           2
                   |                   |       17 | Open         | Console has blue rings of death             |           2
                   |                   |       18 | Open         | Console overheating                         |           2
                   |                   |       19 | Open         | Console port failure                        |           2
                   |                   |       20 | Open         | Console port failure                        |           2
                   |                   |       21 | Open         | Membership renewal, unauthorized            |           3
                   |                   |       22 | Open         | Membership renewal, unauthorized            |           3
                   |                   |       23 | Open         | Household member made unauthorized purchase |           3
                   |                   |       24 | Open         | Household member made unauthorized purchase |           3
                   |                   |       25 | Open         | Household member made unauthorized purchase |           3
                   |                   |       26 | Open         | Account owner made unauthorized purchase    |           3
                   |                   |       27 | Open         | Account owner made unauthorized purchase    |           3
                   |                   |       28 | Open         | Membership renewal, unauthorized            |           3
                   |                   |       29 | Open         | Household member made unauthorized purchase |           3
                   |                   |       30 | Open         | Account owner made unauthorized purchase    |           3
                   |                   |       31 | Open         | Account was stolen                          |           4
                   |                   |       32 | Open         | Password reset needed                       |           4
                   |                   |       33 | Open         | Passward reset needed                       |           4
                   |                   |       34 | Open         | Account is Banned                           |           4
                   |                   |       35 | Open         | Invalid email registration                  |           4
                   |                   |       36 | Open         | Account was stolen                          |           4
                   |                   |       37 | Open         | Invalid email registration                  |           4
                   |                   |       38 | Open         | Invalid email registration                  |           1
                   |                   |       39 | Open         | Password reset needed                       |           4
                   |                   |       40 | Open         | Password reset needed                       |           4
                   |                   |       41 | Closed       | Password reset needed                       |           4
                   |                   |       42 | Closed       | Invalid email registration                  |           1
                   |                   |       43 | Open         | Invalid email registration                  |           4
                   |                   |       44 | Open         | Passward reset needed                       |           4
                   |                   |       45 | Open         | Account is Banned                           |           4
                   |                   |       46 | Open         | Account is Banned                           |           4
                   |                   |       47 | Open         | Hacked account                              |           4
                   |                   |       48 | Open         | Renewal                                     |           3
```

After Update:

```
apoon=> select updateTicket(48,'Closed', 'Membership Renewal, unauthorized', 3);
 updateticket
--------------

(1 row)

apoon=> select * from ticket;
 transactiongameid | transactionhardid | ticketid | ticketstatus |             problemdescription              | problemarea
-------------------+-------------------+----------+--------------+---------------------------------------------+-------------
                   |                   |        1 | Open         | Game Installation Not Working               |           1
                   |                   |        2 | Open         | Game Installation Not Working               |           1
                   |                   |        3 | Open         | Downloadable Content Not Installing         |           1
                   |                   |        4 | Open         | Missing Trophies in Game                    |           1
                   |                   |        5 | Open         | Game Stuck in Load                          |           1
                   |                   |        6 | Open         | Game Installation Not Working               |           1
                   |                   |        7 | Open         | Firmware Download is Stuck                  |           1
                   |                   |        8 | Open         | Firmware Download is Stuck                  |           1
                   |                   |        9 | Open         | Firmware Download is Stuck                  |           1
                   |                   |       10 | Open         | Game Installation Not Working               |           1
                   |                   |       11 | Open         | Console overheating                         |           2
                   |                   |       12 | Open         | Console overheating                         |           2
                   |                   |       13 | Open         | Console overheating                         |           2
                   |                   |       14 | Open         | Controllers are not pairing                 |           2
                   |                   |       15 | Open         | Controllers are not pairing                 |           2
                   |                   |       16 | Open         | Console has blue rings of death             |           2
                   |                   |       17 | Open         | Console has blue rings of death             |           2
                   |                   |       18 | Open         | Console overheating                         |           2
                   |                   |       19 | Open         | Console port failure                        |           2
                   |                   |       20 | Open         | Console port failure                        |           2
                   |                   |       21 | Open         | Membership renewal, unauthorized            |           3
                   |                   |       22 | Open         | Membership renewal, unauthorized            |           3
                   |                   |       23 | Open         | Household member made unauthorized purchase |           3
                   |                   |       24 | Open         | Household member made unauthorized purchase |           3
                   |                   |       25 | Open         | Household member made unauthorized purchase |           3
                   |                   |       26 | Open         | Account owner made unauthorized purchase    |           3
                   |                   |       27 | Open         | Account owner made unauthorized purchase    |           3
                   |                   |       28 | Open         | Membership renewal, unauthorized            |           3
                   |                   |       29 | Open         | Household member made unauthorized purchase |           3
                   |                   |       30 | Open         | Account owner made unauthorized purchase    |           3
                   |                   |       31 | Open         | Account was stolen                          |           4
                   |                   |       32 | Open         | Password reset needed                       |           4
                   |                   |       33 | Open         | Passward reset needed                       |           4
                   |                   |       34 | Open         | Account is Banned                           |           4
                   |                   |       35 | Open         | Invalid email registration                  |           4
                   |                   |       36 | Open         | Account was stolen                          |           4
                   |                   |       37 | Open         | Invalid email registration                  |           4
                   |                   |       38 | Open         | Invalid email registration                  |           1
                   |                   |       39 | Open         | Password reset needed                       |           4
                   |                   |       40 | Open         | Password reset needed                       |           4
                   |                   |       41 | Closed       | Password reset needed                       |           4
                   |                   |       42 | Closed       | Invalid email registration                  |           1
                   |                   |       43 | Open         | Invalid email registration                  |           4
                   |                   |       44 | Open         | Passward reset needed                       |           4
                   |                   |       45 | Open         | Account is Banned                           |           4
                   |                   |       46 | Open         | Account is Banned                           |           4
                   |                   |       47 | Open         | Hacked account                              |           4
                   |                   |       48 | Closed       | Membership Renewal, unauthorized            |           3
```

4.2.4 Stored Procedure: Sum and Average

The average and sum procedure will return the average of sum of the specified. Below, the function shows an aggregate of Chats Dropped by the Zip Code. This function will return the details of sum and average for Chats Dropped (Total and Average), Time in Queue, and the Support Costs by Zip Code.

```
CREATE OR REPLACE FUNCTION chatsdroppedZipCodeDetails(
astartdate timestamp,
```

```
aenddate timestamp
      ) RETURNS TABLE (
startdate timestamp,
enddate timestamp,
zipcode text,
totaldropped integer,
averagedropped integer,
queuetime integer,
supportcost integer)
AS
$$
   BEGIN
RETURN QUERY SELECT
      startdate,
      enddate,
      regexp_replace(address, '^.* ', '') as "Zip Code",
      TRUNC(SUM(sessionDropped),3) as "Total Dropped",
      TRUNC(AVG(sessionDropped),3) as "Average Dropped",
      TRUNC(CAST(SUM(EXTRACT(EPOCH FROM (enddate - startdate))) AS
INTEGER), 3) as "Time in Queue",
      TRUNC((CAST(SUM(EXTRACT(EPOCH FROM (enddate - startdate))) AS
INTEGER)/60)*5, 3) as "Total Cost"
      FROM
      account natural join chat
      GROUP BY address, startdate, enddate;
   END;
$$ LANGUAGE plpgsql;
```

```
apoon=> select * from chatsdroppedZipCodeDetails('2018-11-1', '2018-12-12');
     ostartdate       |       oenddate       | zipcode | totaldropped | averagedropped | queuetime | supportcost
----------------------+----------------------+---------+--------------+----------------+-----------+-------------
 2018-11-04 06:46:55 | 2018-11-04 06:56:10 | 92315   |            0 |              0 |       555 |          45
 2018-11-26 13:00:55 | 2018-11-26 13:26:10 | 93311   |            0 |              0 |      1515 |         125
 2018-11-30 08:54:55 | 2018-11-30 08:56:10 | 94602   |            0 |              0 |        75 |           5
 2018-11-29 10:54:55 | 2018-11-29 10:56:10 | 64511   |            0 |              0 |        75 |           5
 2018-11-30 08:54:55 | 2018-11-30 08:56:10 | 64511   |            0 |              0 |       225 |          15
 2018-11-02 06:46:55 | 2018-11-02 06:56:10 | 92315   |            0 |              0 |       555 |          45
 2018-11-30 01:45:55 | 2018-11-30 01:56:10 | 93311   |            0 |              0 |       615 |          50
 2018-11-22 02:34:55 | 2018-11-22 02:56:10 | 92211   |            0 |              0 |      1275 |         105
 2018-11-30 08:54:55 | 2018-11-30 08:56:10 | 93311   |            0 |              0 |        75 |           5
 2018-11-01 08:14:55 | 2018-11-01 08:20:30 | 94602   |            0 |              0 |       335 |          25
 2018-11-30 01:45:55 | 2018-11-30 01:56:10 | 94546   |            0 |              0 |       615 |          50
 2018-11-30 01:45:55 | 2018-11-30 01:56:10 | 92315   |            0 |              0 |       615 |          50
 2018-11-26 09:46:55 | 2018-11-26 09:56:10 | 93311   |            0 |              0 |       555 |          45
 2018-11-26 09:54:55 | 2018-11-26 09:56:10 | 94602   |            0 |              0 |        75 |           5
 2018-11-28 12:54:55 | 2018-11-30 12:56:10 | 92315   |            0 |              0 |    172875 |       14405
 2018-12-01 11:54:55 | 2018-12-01 11:56:10 | 92213   |            0 |              0 |        75 |           5
 2018-11-30 01:45:55 | 2018-11-30 01:56:10 | 93311   |            0 |              0 |       615 |          50
 2018-11-28 11:54:55 | 2018-11-28 11:56:10 | 92213   |            0 |              0 |        75 |           5
 2018-11-30 07:43:55 | 2018-11-30 07:56:10 | 93311   |            0 |              0 |       735 |          60
(19 rows)
```

4.2.5 Trigger: Delete Cascade

The delete cascade procedure cascades down to the base table which is ticket. Deletefocus first deletes from the specified focus table then it deletes from the base table ticket. The related attributes in the ticket and focus tables are focusid and ticketid and by utilizing these attributes to identify which focus table we need to look at.

```
CREATE OR REPLACE FUNCTION deletefocus(
      tid INT
      focusid INT
) RETURNS VOID AS
$func$
      BEGIN
            IF focusid = 4 THEN
                  DELETE from account_issues where ticketID =
              tid;
            ELSIF focusid = 3 THEN
                  DELETE from billing where ticketID = tid;
            ELSIF focusid = 2 THEN
                  DELETE from hardware_repair where ticketID =
              tid;
            ELSIF focusid = 1 THEN
                  DELETE from troubleshooting where ticketID =
              tid;
            END IF;
            Delete from ticket where ticketID = tid;
      END;
$func$ LANGUAGE plpgsql;
```

Before Delete:

```
apoon=> select * from ticket;
 transactiongameid | transactionhardid | ticketid | ticketstatus |             problemdescription              | problemarea
-------------------+-------------------+----------+--------------+---------------------------------------------+-------------
                   |                   |        1 | Open         | Game Installation Not Working               |           1
                   |                   |        2 | Open         | Game Installation Not Working               |           1
                   |                   |        3 | Open         | Downloadable Content Not Installing         |           1
                   |                   |        4 | Open         | Missing Trophies in Game                    |           1
                   |                   |        5 | Open         | Game Stuck in Load                          |           1
                   |                   |        6 | Open         | Game Installation Not Working               |           1
                   |                   |        7 | Open         | Firmware Download is Stuck                  |           1
                   |                   |        8 | Open         | Firmware Download is Stuck                  |           1
                   |                   |        9 | Open         | Firmware Download is Stuck                  |           1
                   |                   |       10 | Open         | Game Installation Not Working               |           1
                   |                   |       11 | Open         | Console overheating                         |           2
                   |                   |       12 | Open         | Console overheating                         |           2
                   |                   |       13 | Open         | Console overheating                         |           2
                   |                   |       14 | Open         | Controllers are not pairing                 |           2
                   |                   |       15 | Open         | Controllers are not pairing                 |           2
                   |                   |       16 | Open         | Console has blue rings of death             |           2
                   |                   |       17 | Open         | Console has blue rings of death             |           2
                   |                   |       18 | Open         | Console overheating                         |           2
                   |                   |       19 | Open         | Console port failure                        |           2
                   |                   |       20 | Open         | Console port failure                        |           2
                   |                   |       21 | Open         | Membership renewal, unauthorized            |           3
                   |                   |       22 | Open         | Membership renewal, unauthorized            |           3
                   |                   |       23 | Open         | Household member made unauthorized purchase |           3
                   |                   |       24 | Open         | Household member made unauthorized purchase |           3
                   |                   |       25 | Open         | Household member made unauthorized purchase |           3
                   |                   |       26 | Open         | Account owner made unauthorized purchase    |           3
                   |                   |       27 | Open         | Account owner made unauthorized purchase    |           3
                   |                   |       28 | Open         | Membership renewal, unauthorized            |           3
                   |                   |       29 | Open         | Household member made unauthorized purchase |           3
                   |                   |       30 | Open         | Account owner made unauthorized purchase    |           3
                   |                   |       31 | Open         | Account was stolen                          |           4
                   |                   |       32 | Open         | Password reset needed                       |           4
                   |                   |       33 | Open         | Password reset needed                       |           4
                   |                   |       34 | Open         | Account is Banned                           |           4
                   |                   |       35 | Open         | Invalid email registration                  |           4
                   |                   |       36 | Open         | Account was stolen                          |           4
                   |                   |       37 | Open         | Invalid email registration                  |           4
                   |                   |       38 | Open         | Invalid email registration                  |           1
                   |                   |       39 | Open         | Password reset needed                       |           4
                   |                   |       40 | Open         | Password reset needed                       |           4
                   |                   |       41 | Closed       | Password reset needed                       |           4
                   |                   |       42 | Closed       | Invalid email registration                  |           1
                   |                   |       43 | Open         | Invalid email registration                  |           4
                   |                   |       44 | Open         | Password reset needed                       |           4
                   |                   |       45 | Open         | Account is Banned                           |           4
                   |                   |       46 | Open         | Account is Banned                           |           4
(46 rows)

apoon=> select * from billing;
 solutionid | ticketid | datecreated |  refundtype  | refundtyperequested |   solutionprovided
------------+----------+-------------+--------------+---------------------+----------------------
         21 |       21 | 2018-01-30  | Credit Card  | Credit Card         | Refunded to Consumer
         22 |       22 | 2018-03-03  | Credit Card  | Credit Card         | Refunded to Consumer
         23 |       23 | 2018-04-03  | Store Credit | Credit Card         | Refunded to Consumer
         24 |       24 | 2018-08-26  | Store Credit | Credit Card         | Refunded to Consumer
         25 |       25 | 2018-12-03  | None         | Credit Card         | Denied Refund
         26 |       26 | 2018-12-03  | None         | Credit Card         | Denied Refund
         27 |       27 | 2018-12-06  | Store Credit | Store Credit        | Refunded to Consumer
         28 |       28 | 2018-03-03  | None         | Credit Card         | Denied Refund
         29 |       29 | 2018-07-03  | Credit Card  | Credit Card         | Refunded to Consumer
```

After Delete:

```
apoon=> select deletefocus(21, 3);
 deletefocus
-------------

(1 row)

apoon=> select * from ticket;
 transactiongameid | transactionhardid | ticketid | ticketstatus |             problemdescription              | problemarea
-------------------+-------------------+----------+--------------+---------------------------------------------+-------------
                   |                   |        1 | Open         | Game Installation Not Working               |           1
                   |                   |        2 | Open         | Game Installation Not Working               |           1
                   |                   |        3 | Open         | Downloadable Content Not Installing         |           1
                   |                   |        4 | Open         | Missing Trophies in Game                    |           1
                   |                   |        5 | Open         | Game Stuck in Load                          |           1
                   |                   |        6 | Open         | Game Installation Not Working               |           1
                   |                   |        7 | Open         | Firmware Download is Stuck                  |           1
                   |                   |        8 | Open         | Firmware Download is Stuck                  |           1
                   |                   |        9 | Open         | Firmware Download is Stuck                  |           1
                   |                   |       10 | Open         | Game Installation Not Working               |           1
                   |                   |       11 | Open         | Console overheating                         |           2
                   |                   |       12 | Open         | Console overheating                         |           2
                   |                   |       13 | Open         | Console overheating                         |           2
                   |                   |       14 | Open         | Controllers are not pairing                 |           2
                   |                   |       15 | Open         | Controllers are not pairing                 |           2
                   |                   |       16 | Open         | Console has blue rings of death             |           2
                   |                   |       17 | Open         | Console has blue rings of death             |           2
                   |                   |       18 | Open         | Console overheating                         |           2
                   |                   |       19 | Open         | Console port failure                        |           2
                   |                   |       20 | Open         | Console port failure                        |           2
                   |                   |       22 | Open         | Membership renewal, unauthorized            |           3
                   |                   |       23 | Open         | Household member made unauthorized purchase |           3
                   |                   |       24 | Open         | Household member made unauthorized purchase |           3
                   |                   |       25 | Open         | Household member made unauthorized purchase |           3
                   |                   |       26 | Open         | Account owner made unauthorized purchase    |           3
                   |                   |       27 | Open         | Account owner made unauthorized purchase    |           3
                   |                   |       28 | Open         | Membership renewal, unauthorized            |           3
                   |                   |       29 | Open         | Household member made unauthorized purchase |           3
                   |                   |       30 | Open         | Account owner made unauthorized purchase    |           3
                   |                   |       31 | Open         | Account was stolen                          |           4
                   |                   |       32 | Open         | Password reset needed                       |           4
                   |                   |       33 | Open         | Passward reset needed                       |           4
                   |                   |       34 | Open         | Account is Banned                           |           4
                   |                   |       35 | Open         | Invalid email registration                  |           4
                   |                   |       36 | Open         | Account was stolen                          |           4
                   |                   |       37 | Open         | Invalid email registration                  |           4
                   |                   |       38 | Open         | Invalid email registration                  |           1
                   |                   |       39 | Open         | Password reset needed                       |           4
                   |                   |       40 | Open         | Password reset needed                       |           4
                   |                   |       41 | Closed       | Password reset needed                       |           4
                   |                   |       42 | Closed       | Invalid email registration                  |           1
                   |                   |       43 | Open         | Invalid email registration                  |           4
                   |                   |       44 | Open         | Passward reset needed                       |           4
                   |                   |       45 | Open         | Account is Banned                           |           4
                   |                   |       46 | Open         | Account is Banned                           |           4
(45 rows)

apoon=> select * from billing;
 solutionid | ticketid | datecreated |  refundtype  | refundtyperequested |   solutionprovided
------------+----------+-------------+--------------+---------------------+----------------------
         22 |       22 | 2018-03-03  | Credit Card  | Credit Card         | Refunded to Consumer
         23 |       23 | 2018-04-03  | Store Credit | Credit Card         | Refunded to Consumer
         24 |       24 | 2018-08-26  | Store Credit | Credit Card         | Refunded to Consumer
```

4.2.6 Trigger: Before Update

The before update procedure will fire before an update procedure. It inserts a ticket with the problem status as closed.

```
CREATE or REPLACE FUNCTION closeTicket()
      returns trigger as
$$
```

```
      begin

      if solutiondescription <> NULL then
      insert into ticket values(old.accountid, old.transactiongameid,
old.transactionhardid, old.ticketid, new.status = 'CLOSED');
      end if;
      return new;

      exception
            when others then
            raise NOTICE 'closing ticket function of solution id -%-% has
failed due to [%]', solutionid, SQLERRM;


      end;
$$ language plpgsql;
```

4.2.7 Trigger: Instead Of

The instead of procedure allows you to skip over a event so for exceptions it will be able to use the trigger to modify the views as appropriate. It will return NULL if nothing is changed and it will return the view row that was modified meaning that everything was altered correctly. The version of PostgreSQL we used did not have the instead of functionality. In later versions PostgreSQL 9.4 had this functionality. An example of the instead of trigger that can be used for our database is as shown below.

```
CREATE OR REPLACE VIEW view_account AS
    SELECT  accountid, emailadd, phonenumber, password, name, address,
region
      FROM account;

CREATE OR REPLACE FUNCTION updatePassword()
      RETURNS trigger AS
$$
BEGIN
      insert into account values(old.accountid, old.emailadd,
old.phonenumber, new.password, old.name, old.address, old.region);
      end if;
      return new;

      exception
```

```
            when other then
            raise 'password cannot be changed: id -%-% has failed due to
[%]', password, SQLERRM;

RETURN NEW;
END;

$$

CREATE TRIGGER view_insert
    INSTEAD OF INSERT ON view_account
    FOR EACH ROW
    EXECUTE PROCEDURE updatePassword();
```

4.3 Postgres PL/pgSQL Comparison to Other Tools (Microsoft SQL, MySQL)

Here we will look in depth regarding the comparisons of different but very similar DBMSs like Microsoft SQL and MySQL. We will talk about how we would be able to implement the same features in these two procedural languages and we will compare them to PostgreSQL in its advantages and disadvantages.

4.3.1 Microsoft SQL Server: T-SQL

T - SQL provides unique functionality allowing one to use object-oriented programming like in PostgreSQL. It will allow for a programmer to use nested try and catch statements for their functions and procedure calls providing more complex functions. It will not need to have casting or changing data types which would make calling functions or procedures that calculate complex calculations much easier. Below we will go over the syntax and examples of T-SQL

When writing the procedure in T-SQL syntax is as written:

```
CREATE [ OR ALTER ] { PROC | PROCEDURE }
     [schema_name.] procedure_name [ ; number ]
     [ { @parameter [ type_schema_name. ] data_type }
          [ VARYING ] [ = default ] [ OUT | OUTPUT | [READONLY] [ [
,...n ]
[ WITH <procedure_option> [ ,...n ] ]
[ FOR REPLICATION ]
AS { [ BEGIN ] sql_statement [;] [ ...n ] [ END ] } [;]
```

```
<procedure_option> ::=
     [ ENCRYPTION ]
     [ RECOMPILE ]
     [ EXECUTE AS Clause ]
```

When writing the function in T-SQL syntax is as written:

```
CREATE FUNCTION [ schema_name. ] function_name
     ( [ { @parameter_name [ AS ][ type_schema_name. ]
parameter_data_type
     [ = default ] [ READONLY ] } [ ,...n ] ]
     )
          RETURNS return_data_type
     [ WITH <function_option> [ ,...n ] ]
     [ AS ]
     BEGIN
          function_body
RETURN scalar_expression
END
[ ; ]
```

When writing the loops in T-SQL syntax is as written:

```
WHILE @iterator < iterator_total
     BEGIN
          {         statements to be executed         }
          SET @iterator = @iterator + 1;
     END;
```

4.3.2 MySQL

MySQL is very similar to PostgreSQL and T-SQL. While MySQL has many similarities to the other DBMSs, it does have many missing features. It offers the basic features such as control structions, but it does not have any for loops, instead one would use while loops. You will not be able to use any packages on the MySQL. MySQL does not comply with SQL standards, such as foreign key references. Other limitations in this is that it is limited to over trigger per action and triggers are not defined on views.

When writing the procedures in MySQL syntax is as written:

```
CREATE
      [DEFINER = { user | CURRENT_USER }]
      PROCEDURE sp_name ([proc_parameter[,...]])
      [characteristic ...] routine_body
```

When writing the functions in MySQL syntax is as written:

```
CREATE
      [DEFINER = { user | CURRENT_USER }]
      FUNCTION sp_name ([func_parameter[,...]])
      RETURNS type
      [characteristic ...] routine_body
```

When writing the loops in MySQL syntax is as written:

```
[begin_label:] WHILE search_condition DO
statement_list
END WHILE [end_label]
```

4.3.3 PostgreSQL

PostgreSQL was the language we decided to focus on for our database. It is a very powerful tool, that has many advantages, providing table inheritances, foreign key references and constraints/checks, nested transactions, etc. It also has safe languages that allow for the programmer to use: such as SQL, C/C++ and Java. Postgres is extremely versatile and powerful for all your database needs.

When writing the Function in PostgreSQL syntax is as written:

```
CREATE FUNCTION function_name(p1 type, p2 type)
           RETURNS type AS
      BEGIN
           -- logic
      END;
```

```
      LANGUAGE language_name;
```

When writing the Loops in PostgreSQL syntax are as written:

```
      <<label>>
      LOOP
            Statements;
      EXIT [<<label>>] WHEN condition;
      END LOOP;
```

An example would be:

```
LOOP
    -- statement for execution
    IF iterator > 0 THEN
        EXIT;  -- exit loop
    END IF;
END LOOP;
```

PHASE 5 Graphical User Interface Implementation

In phase 5, we will go over our implementation of the graphical user interface (GUI) with the database we have designed and created. We will be going through the specifics of each part of the graphical user interface and how we have made it very agent friendly for efficiently and speed when agents are on the call with the consumer. We will talk about the users of our group and their individual needs in the GUI. We will be describe the specifics of what was needed to build certain parts of this software application and the PostgreSQL features used to create our GUI. Lastly, we will be providing a detailed overview of the database implementation and overall, our lessons learned in this class.

5.1 Daily User Activities

When we designed our graphical user interface, we worked hard to really envision what the players in our database will be doing on a day to day basis. We asked ourselves, what can we do to make these daily user activities simpler and more efficient for agents, managers, specialists, and consumers-alike. Each party require a different set of necessity for the best consumer experience in our GUI and it is our job to really look into what makes this is the best GUI possible for the party. In our project, we only implemented the interface for one of the user groups, which is the Agent View. Below is a full description of all the parties and their needs.

5.1.1 Consumer Users

The consumer view is very simple. It will not need too much functionality, but the consumer-view will need their interface to be extremely clear-cut, because consumers can get confused easily. The consumer side must be straightforward and easy to use.

Required Functionalities in the Consumer View (for each associated Account) are:

- View all Hardware and Software associated with Account
- Drop-Down of Possible Issues and Focus Area
- Chat box creation

5.1.2 Agent Users

The agent only creates the tickets for the specialist to work on. They do not solve the tickets themselves. The view is focused on speed and information. An agent may have issues with working on the spot so the graphical user interface must be able to assist the agent and be very speedly as well so that the agent can get on and off a chat as soon as possible. It will also need

report abilities to see how they or their team is doing as opposed to others. This view is extremely performance heavy and and speed heavy.

Required Functionalities in Agent View:

- View Chat Pending in the Queue with Waiting Time
- Creation and Modifying of the Ticket for a Specialist to Finish
- Solutions Guide on Chat
- A Dynamic Chat Window

5.1.3 Specialist Users

The specialists are the individuals who focus on solving the tickets. They are in the background working and solving the tickets while agents are collecting the customer problems to be solved. Their design must be dynamic and easy to pull up the view of solutions and possible solutions. Their design should be able to have all possible solutions while also being speedy as well

Required Functionalities in Specialists View:

- All solutions possible for closing of the Tickets
- Quick selection of closing the tickets
- Simple displays for each focus area

5.1.4 Manager Users

The managers are primarily interested in the cost of support. They are interested in the cost effectiveness. Their design must be detailed and very report oriented in the payroll perspective.

Required Functionalities for the Manager's View

- Reports showing the support cost
- Reports forecasting chat demands
- Simple dynamic report generator that is customizable

5.2 Relations, Views, and Subprograms

In this section, we will be going over parts of the backend of our application to show some of the implementation done on this. Some tables created for the consumer services database are shown below:

```
CREATE TABLE account(
```

```
   accountID      SERIAL    PRIMARY KEY,
   emailAdd       TEXT      NOT NULL UNIQUE,
   phoneNumber              BIGINT,
   password       TEXT      NOT NULL,
   name           TEXT        NOT NULL,
   address          TEXT            NOT NULL,
   region           TEXT            NOT NULL,
   accountType    TEXT,
   accountStatus  TEXT,

   CONSTRAINT account_type CHECK (accountType = ANY(
ARRAY['Master','Sub','Other'])),
   CONSTRAINT account_status CHECK (accountStatus = ANY(
ARRAY['Active','Suspended','Banned','Other']))
);

CREATE TABLE agent(
      agentTransfersID    INT        DEFAULT NULL,
    agentID             SERIAL     PRIMARY KEY,
    agentTier           INTEGER    DEFAULT 1,
    agentName           CHAR(100)  NOT NULL,
    emailAdd            TEXT       NOT NULL UNIQUE,
      password            TEXT         NOT NULL

);

CREATE TABLE ticket(
      transactionGameID     INT,
      transactionHardID     INT,
    ticketID              SERIAL       PRIMARY KEY,
      ticketStatus          TEXT,
      problemDescription    TEXT,
      problemArea           INT,

      CONSTRAINT ticket_status CHECK (ticketStatus = ANY(
ARRAY['Open','Closed','Other'])),
      CONSTRAINT fk_ticket_game FOREIGN KEY (transactionGameID) REFERENCES
game_copy(transactionGameID),
    CONSTRAINT fk_ticket_hardware FOREIGN KEY (transactionHardID)
REFERENCES hardware_copy(transactionHardID)
);
```

In order to make use of Postgres in making the best application for an agent, we will be using PL/pgSQL functionalities provided by PostgreSQL DBMS. These functionalities that have been used are: views and stored procedures as detailed below:

View: Agent and Focus Area Performance

This view was used to show the Agent Performance. For each agent, what is their main speciality, what is their average customer service, the total time they worked and the salary they have made in that date range. The second view was created to group this all by the Speciality (Focus Area) so we can see which specialized agents need the most help: billing, repair, troubleshooting, or account issues.

```
CREATE OR REPLACE VIEW salesAgentReport AS
      SELECT
      astarttime,
      aendtime,
            CASE
             WHEN avg(problemarea) >= 3.1
            THEN 'Account Issues'
             WHEN avg(problemarea) >= 2.1
            THEN 'Billing'
             WHEN avg(problemarea) >= 1.1
            THEN 'Repair'
             WHEN avg(problemarea) >= 0
            THEN 'Troubleshooting'
                  ELSE ''
             END AS Speciality,
      TRUNC(avg(customerfeedback),2) as Customer_Satisfaction,
      SUM(EXTRACT(EPOCH FROM (aendtime - astarttime)))/6 as Time_Worked,
      (SUM(EXTRACT(EPOCH FROM (aendtime - astarttime)))/60)*100 as Salary
      FROM agent natural join answered_by natural join chat natural join
ticket
      GROUP BY problemarea, astarttime, aendtime;

CREATE OR REPLACE VIEW aggSalesAgentReport AS
      select
            astarttime,
            aendtime,
            speciality,
            trunc(avg(customer_satisfaction), 2) as AvgCustSat,
            trunc(cast(sum(time_worked) as integer), 2) as TotalTime,
```

```
            trunc(cast(sum(salary) as integer), 2) as TotalCost
            from salesAgentReport group by speciality,astarttime, aendtime;
```

View: Chats Pending

This view was made to see the chats pending in queue. You will be able to see in ascending order, how long each consumer has been waiting in queue so that the agent can chat with them. This is the view used to see the chats in queue.

```
CREATE OR REPLACE VIEW chatPending AS
      SELECT
      accountID as "account",
      emailadd as "email",
      phonenumber as "number",
      startDate as "initiated",
      EXTRACT(DOW FROM (startdate)) as "day of week",
      EXTRACT(EPOCH FROM (current_timestamp - startdate)) as "seconds
waited",
      to_char((current_timestamp - startdate),'HH24 hrs MI "minutes"
SS "seconds"') as "time waited"

      FROM chat NATURAL JOIN account WHERE enddate IS NULL ORDER BY
startdate ASC;
```

View: Chats Dropped

This view shows chats that were dropped by zip and how much we lost in revenue. The first view is for the tables, it shows the dates where the calls were dropped and also the zip code.

```
CREATE OR REPLACE VIEW chatsDropped AS
      SELECT
      startdate,
      enddate,
      regexp_replace(address, '^.* ', '') as "Zip Code",
      TRUNC(SUM(sessionDropped),3) as "Total Dropped",
      TRUNC(AVG(sessionDropped),3) as "Average Dropped",
      TRUNC(CAST(SUM(EXTRACT(EPOCH FROM (enddate - startdate))) AS
      INTEGER), 3) as "Time in Queue",
      TRUNC((CAST(SUM(EXTRACT(EPOCH FROM (enddate - startdate))) AS
      INTEGER)/60)*5, 3) as "Total Cost"
```

```
      FROM
      account natural join chat
      GROUP BY address, startdate, enddate;
```

View: Chats Dropped by Zip Code

The second view is very similar to the above view. It will show the total chats dropped and its percent by zip code within that date range.

```
CREATE OR REPLACE VIEW chatsDroppedByZipCode AS
       select
       startdate,
       enddate,
       "Zip Code",
       COUNT("Zip Code") as "Total",
       TRUNC(sum("Total Dropped"),3) as "Total Dropped",
       TRUNC(sum("Total Cost"), 2) as "Total Cost",
       TRUNC(sum("Time in Queue"),2)  as "Time in Queue"
       from chatsDropped
       group by "Zip Code", startdate, enddate;
```

View: Customer Satisfaction

This view shows the consumer satisfaction achieved per agent.

```
CREATE OR REPLACE VIEW CustomerSatisfaction AS
      SELECT
      agentname,
      astarttime,
      aendtime,
      TRUNC(avg(customerfeedback),2) as "Customer Satisfaction",
      TRUNC(CAST(SUM(EXTRACT(EPOCH FROM (aendtime - astarttime)))/6
AS INTEGER), 2) as "Time Worked",
      TRUNC(CAST(SUM(((EXTRACT(EPOCH FROM (aendtime -
astarttime)))/60)*15) AS INTEGER),2) as "Salary",
      CASE WHEN avg(problemarea) >= 3.1
            THEN 'Account Issues'
            WHEN avg(problemarea) >= 2.1
            THEN 'Billing'
```

```
            WHEN avg(problemarea) >= 1.1
            THEN 'Repair'
            WHEN avg(problemarea) >= 0
            THEN 'Troubleshooting'
                ELSE ''
            END AS Focus
      FROM agent natural join answered_by natural join chat natural
join ticket
      GROUP BY agentname, astarttime, aendtime;
```

View: Solutions Guide

This is a view I used to see the aggregate amount of solutions from each focus area to assist with agents on the call. This will show all possible solutions for an issue so that agents will be able to see on the call how to guide the chat.

```
CREATE OR REPLACE VIEW accountSolutions AS
      select solutionprovided as "Solution",
      count(solutionprovided) as "Count"
      from account_issues group by solutionprovided;
```

Stored Procedure: Insert Ticket

This is the stored procedure used to insert a ticket. When the agent is on the phone, they will need to have a new ticket created for the consumer so they have a ticket they could refer to. This stored procedure is used to create that ticket.

```
CREATE OR REPLACE FUNCTION createTicket(
aticketStatus TEXT default 'Open',
aproblemDescription TEXT default 'Not Provided',
aproblemArea INT default null

      ) RETURNS VOID  AS
$$

DECLARE
tempTicketID INT;

BEGIN
```

```
INSERT INTO ticket(ticketStatus, problemDescription, problemArea)
VALUES(aticketStatus, aproblemDescription, aproblemArea)
RETURNING ticketID into tempTicketID;
END;
$$ LANGUAGE plpgsql;
```

Stored Procedure: Update Ticket

This procedure was used to update the ticket because agents will need to update tickets if there is a mistake. This update stored procedure allows for the agents to change the tickets on the call

```
CREATE OR REPLACE FUNCTION updateTicket(

aticketid INT,
aticketStatus TEXT default 'Open',
aproblemDescription TEXT default 'Not Provided',
aproblemArea INT default null


     ) RETURNS VOID  AS
$$

   BEGIN

UPDATE ticket
SET ticketstatus = aticketStatus, problemdescription =
aproblemDescription, problemarea = aproblemArea where ticketid =
aticketid;
END;
$$ LANGUAGE plpgsql;
```

5.3 Menus and Displays

Agents are big on the menus and displays because we need to make sure these menus and displays are easy and straightforward while on the chat with consumers. Below are detailed images and explanations of the GUI menus and displays for each.

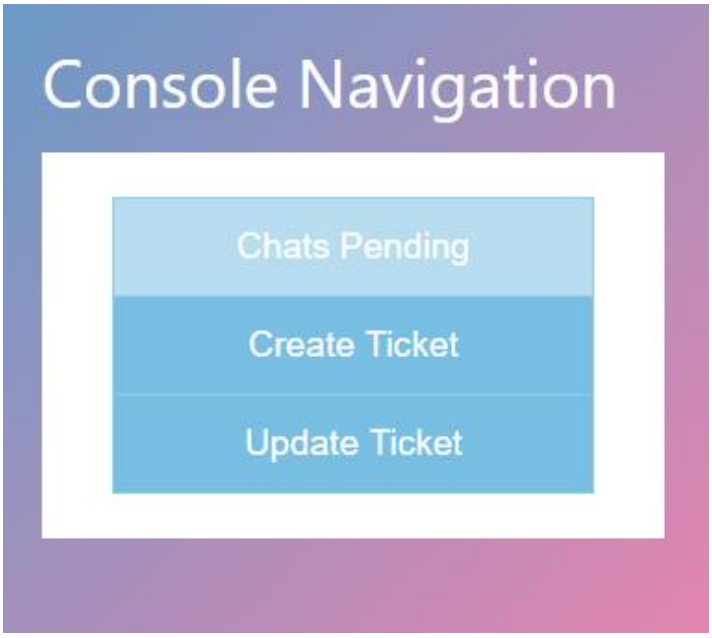
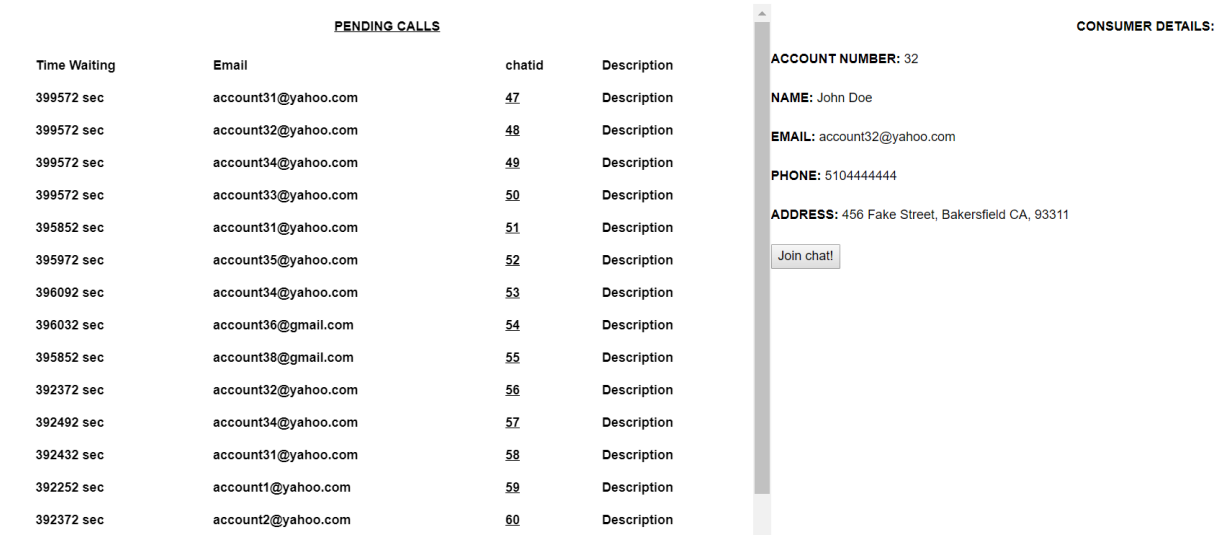


Figure: Console Navigation

Above shown is the console navigation for the Agent View. You will be able to select Three options: to view Chats Pending, to Create a Ticket for the Agent and lastly, to Update the Ticket. Depending on which option the agent chooses, you will receive different functionalities.

Chats Pending View:

Chat Console:



PENDING CALLS

Time Waiting	Email	chatid	Description
399572 sec	account31@yahoo.com	47	Description
399572 sec	account32@yahoo.com	48	Description
399572 sec	account34@yahoo.com	49	Description
399572 sec	account33@yahoo.com	50	Description
395852 sec	account31@yahoo.com	51	Description
395972 sec	account35@yahoo.com	52	Description
396092 sec	account34@yahoo.com	53	Description
396032 sec	account36@gmail.com	54	Description
395852 sec	account38@gmail.com	55	Description
392372 sec	account32@yahoo.com	56	Description
392492 sec	account34@yahoo.com	57	Description
392432 sec	account31@yahoo.com	58	Description
392252 sec	account1@yahoo.com	59	Description
392372 sec	account2@yahoo.com	60	Description

CONSUMER DETAILS:

ACCOUNT NUMBER: 32

NAME: John Doe

EMAIL: account32@yahoo.com

PHONE: 5104444444

ADDRESS: 456 Fake Street, Bakersfield CA, 93311

Join chat!

Figure: Chat Console for Pending

In this Pending Chats View, you will be able to select the consumer that is pending to talk to them. The Pending Chats View is descending to display the longest waiting chat pending. When you click on the chatid you will see the consumer details. When you click [Join Chat] you will be able to chat with the person for support.

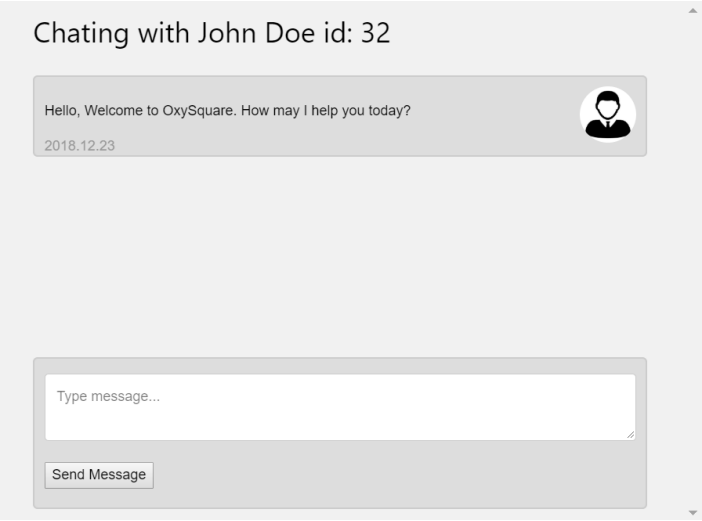


Figure: Chat Window

In this Chat Window, the agent will be able to communicate with the consumer throughout the entire Agent Console: Chats Pending, Create Ticket and Update Ticket.

Create Ticket View:

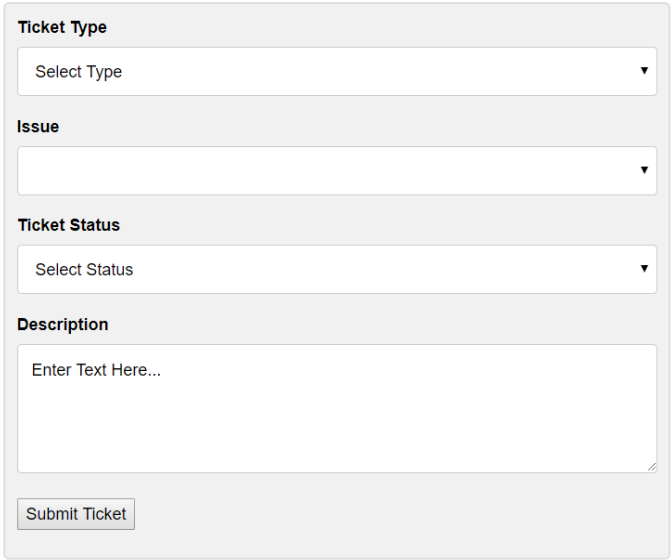
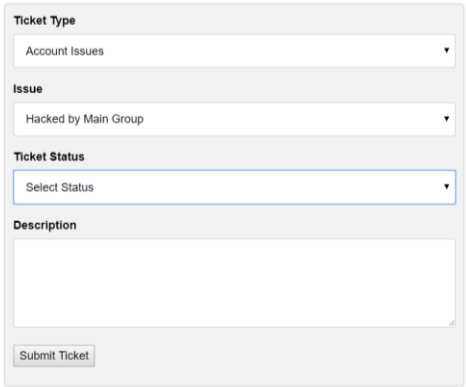
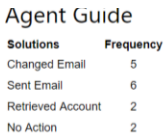


Figure: Create Ticket

When an agent selects “Create Ticket” they will see a form for the dropdown and you will be able to select options: Account Issues, Billing, Repair, Troubleshooting. Upon selecting you are given different displays that assist the agent in guiding the consumer through the call before the agent sends the ticket to a specialist for investigation and solving. In the Agent View, agents will take the ticket and send the ticket appropriately for the specialists to solve the ticket in the back end group. The ticket type will show the problem area, the problem itself, and the ticket status, and the description of the problem. Then the ticket will be forwarded to the specialist. Below you will see the different views for each problem area:





Agent Guide

Solutions	Frequency
Changed Email	5
Sent Email	6
Retrieved Account	2
No Action	2

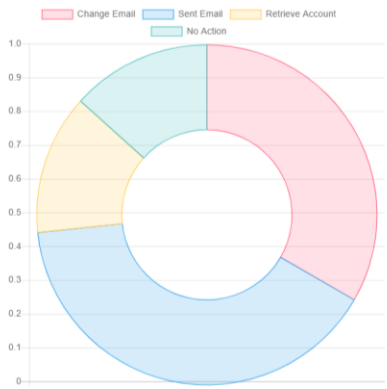


Figure: Account Issues Ticket

Account Issues Ticket contains dropdowns customized specifically for Account Issues such as: Hacked by Main Group, Hacker, Incorrect Email, Lost Password.

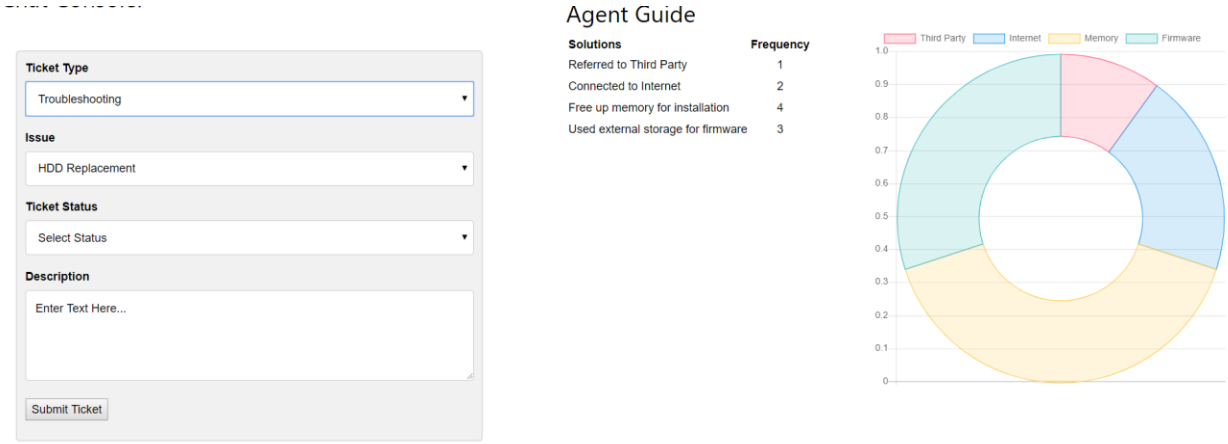


Figure: Troubleshooting Ticket

Troubleshooting Ticket contains dropdowns customized specifically for Troubleshooting Issues such as:HDD Replacement, Referred to Third Party, Referred to ISP, or Firmware Update

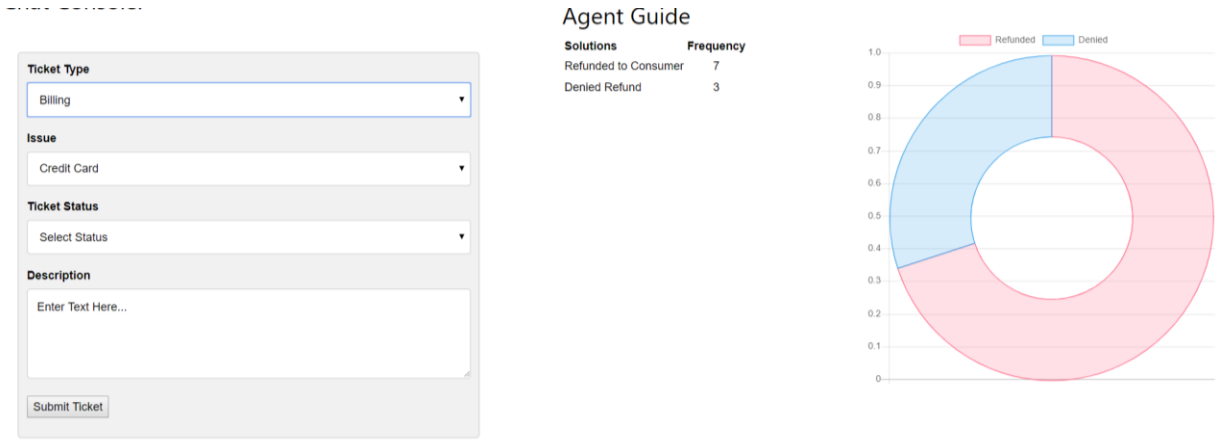


Figure: Billing Ticket

Billing Ticket contains dropdowns customized specifically for Billing Issues such as: Credit Card Refund or Store Credit Refund

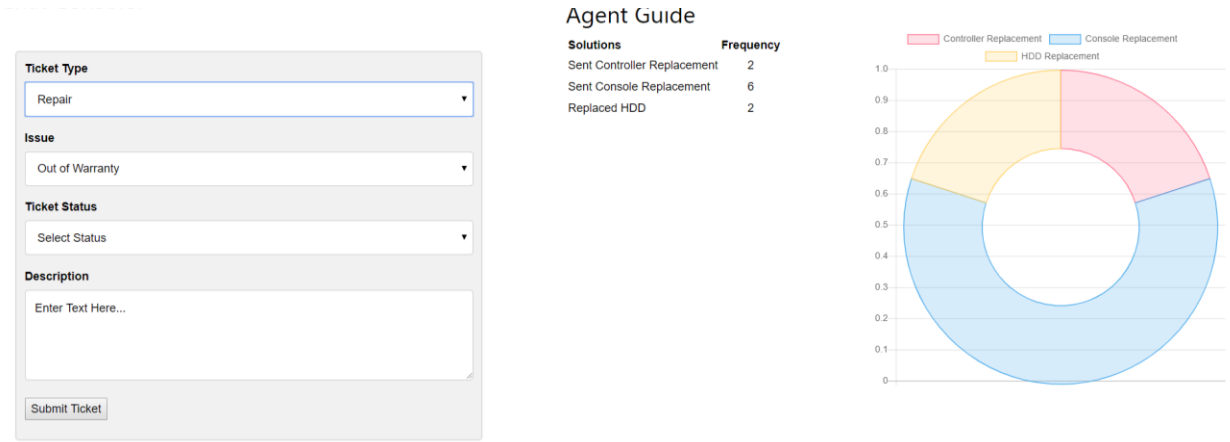


Figure: Repair Ticket

Repair Ticket contains dropdowns customized specifically for Repair Issues such as: Out of Warranty Repair Needed or In Warranty Repair Needed

Update Ticket View:

This view allows for an agent to select a ticket id which will populate the form with the existing information. You will then be able to change any part of the information and update the ticket below. Once submitted, the modified ticket will be shown on the table on the side with the new values.

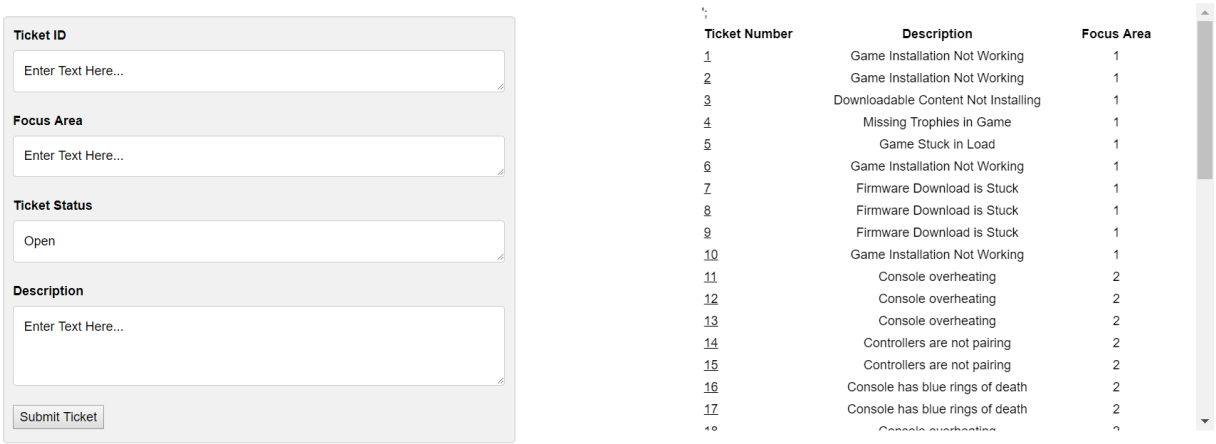


Figure: Update Ticket

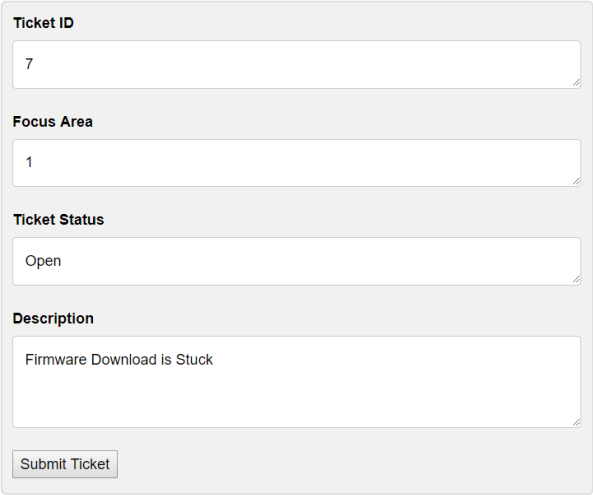


Figure: Click Ticket Number and Data Will Populate

Report Generator View:

This view allows for an agent to pull a report from the database. The reports to be pulled in the database will assist the agent in knowing more about the information. You are able to query by a certain range of dates and select the report from a drop down of the reports you want to pull. In the table area, you will see all the information pulled and you can filter or pull certain reports.

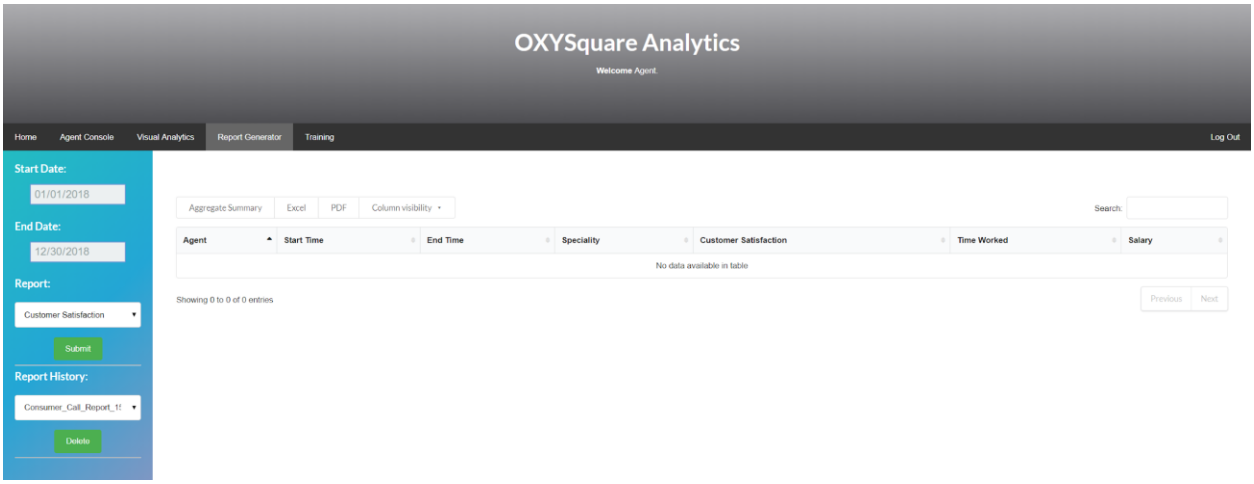


Figure: Agent Report Generator

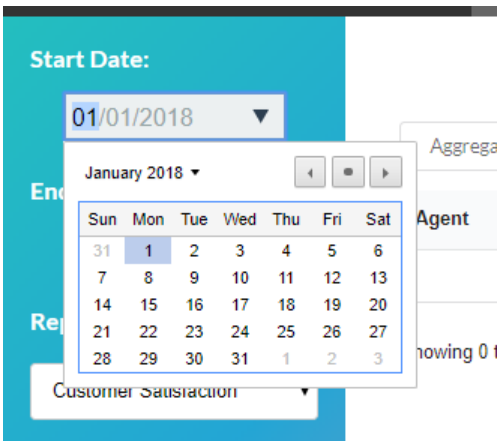
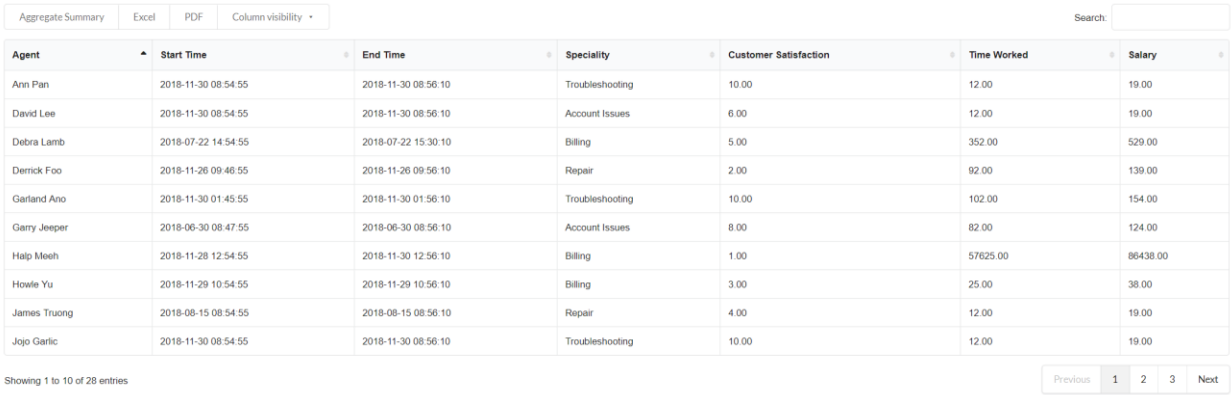


Figure: Dates can be Queried



Aggregate Summary | Excel | PDF | Column visibility

Search:

Agent	Start Time	End Time	Speciality	Customer Satisfaction	Time Worked	Salary
Ann Pan	2018-11-30 08:54:55	2018-11-30 08:56:10	Troubleshooting	10.00	12.00	19.00
David Lee	2018-11-30 08:54:55	2018-11-30 08:56:10	Account Issues	6.00	12.00	19.00
Debra Lamb	2018-07-22 14:54:55	2018-07-22 15:30:10	Billing	5.00	352.00	529.00
Derrick Foo	2018-11-26 09:46:55	2018-11-26 09:56:10	Repair	2.00	92.00	139.00
Garland Ano	2018-11-30 01:45:55	2018-11-30 01:56:10	Troubleshooting	10.00	102.00	154.00
Garry Jeeper	2018-06-30 08:47:55	2018-06-30 08:56:10	Account Issues	8.00	82.00	124.00
Halp Meeh	2018-11-28 12:54:55	2018-11-30 12:56:10	Billing	1.00	57625.00	86438.00
Howie Yu	2018-11-29 10:54:55	2018-11-29 10:56:10	Billing	3.00	25.00	38.00
James Truong	2018-08-15 08:54:55	2018-08-15 08:56:10	Repair	4.00	12.00	19.00
Jojo Garlic	2018-11-30 08:54:55	2018-11-30 08:56:10	Troubleshooting	10.00	12.00	19.00

Showing 1 to 10 of 28 entries

Previous 1 2 3 Next

Figure: Report Tables

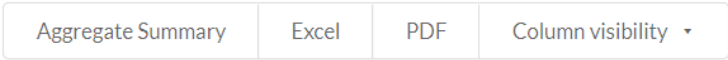


Figure: Options on Tables

In the Agent Report Generator, you are able to query selected dates on different reports. Once you query the database for the report, it shows an aggregate of the information. For Agent Performance we see the aggregate of agents time, average customers service, total time worked and the salary.

If you need the information from the table pulled up in an Excel or PDF, you can always select it to have it in those forms:

OXYSquare Analytics

Agent	Start Time	End Time	Speciality	Customer Satisfaction	Time Worked	Salary
Ann Pan	2018-11-30 08:54:55	2018-11-30 08:56:10	Troubleshooting	10.00	12.00	19.00
Bruce Wayne	2018-01-17 08:32:55	2018-01-17 08:40:10	Billing	6.00	72.00	109.00
Candy Cane	2018-01-10 02:15:55	2018-01-10 02:18:10	Troubleshooting	9.00	22.00	34.00
Candy Crush	2018-04-22 15:24:55	2018-04-22 15:26:10	Account Issues	7.00	12.00	19.00
Couch Potato	2018-01-22 02:34:55	2018-01-22 02:56:10	Troubleshooting	1.00	212.00	319.00
David Lee	2018-11-30 08:54:55	2018-11-30 08:56:10	Account Issues	6.00	12.00	19.00
Davy Rune	2018-05-28 16:54:55	2018-05-28 16:56:10	Account Issues	9.00	12.00	19.00

In the table you will be able to pull a FPDF of a more condensed report by Zip Code or by Focus Area, a Excel file or a PDF of the queried data. The table also allows for you to make any column visible invisible and you will also be able to search your queried result.

OXYSQUARE AGENT SUMMARY REPORT

Call Problem Summary			
Zip Code	Dropped Percentage	Time	Cost
64511	0.333	32930.00	395231.00
92211	0.142	33115.00	397481.00
92213	0.000	245.00	3030.00
92315	0.285	80065.00	960877.00
93311	0.350	230465.00	2765867.00
94546	0.166	33475.00	401786.00
94602	0.400	131500.00	1578154.00

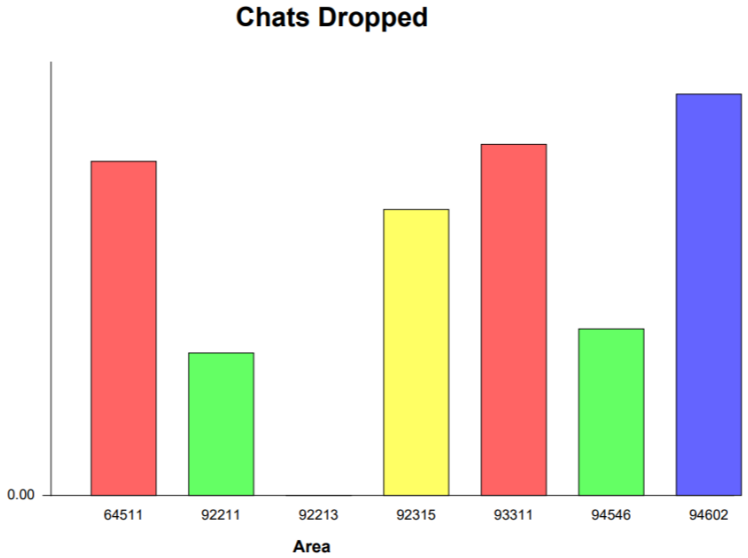


Figure: Report PDF with Table and Graphs Dynamic to Query and Report Type

Upon querying the database for the report, you can open a PDF that will allow you to have a dynamic graph created from the specific query you selected. This will allow you to have a pdf of any data query you make for its report. This report can tell you which zip code where agents work are having too much impact and how many chats have been dropped and how much it has cost the company with the chats.

OXYSQUARE AGENT SUMMARY REPORT

Call Problem Summary			
Specialty	AVG Consumer Satisfaction	Time Worked	Support Cost
Repair	3.0000000000000000	994.00	9983.00
Troubleshooting	7.3333333333333333	706.00	7075.00
Billing	5.5000000000000000	59076.00	590800.00
Account Issues	7.2666666666666667	550.00	5575.00

Customer Satisfaction

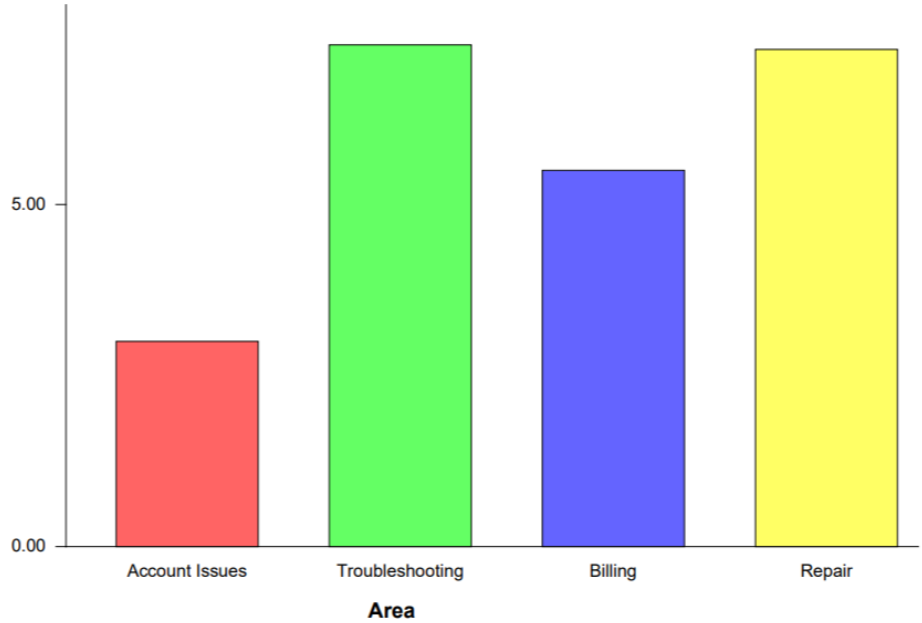


Figure: Report II PDF with Table and Graphs Dynamic to Query and Report Type

The second graph and report tells you which problem area has the lowest customer service satisfaction. Customer Satisfaction is extremely important because the lower the customer service satisfaction achieved the more likely our consumer will call back. This report shows you which area is giving us the most grief and which one is costing the company the most money.

5.4 Description of Code

In this section, I will give an overview of the functionalities of the agent view in software and describe in detail each part as shown.

5.4.1 Database Connection and Interaction

The connection code is a global configuration on our website, written and included on every page of the site under “session.php”. This “session.php” file has the information to connect in the database such as the hostname and the login information.

```
<?php
     require_once('session.php');

?>

<?php
     $dbconn = pg_connect("host=localhost dbname=apoon port=5432
user=apoon password=***********")
     or die('Could not connect: ' . pg_last_error());

     session_start();

     $user_insession = $_SESSION['login_user'];
     $query = pg_query($dbconn, "select emailAdd from agent where
emailAdd = '$user_insession'");

     $row = pg_fetch_array($query, PGSQL_ASSOC);
     $login_sess = $row['emailAdd'];
     if(!isset($_SESSION['login_user'])){
          header("location:index.php");
     }

?>
```

When writing stored procedures and using php we need to embed the php code and use php to query the database. Here is the code for connecting to the database: $dbconn = pg_connect("host=localhost dbname=apoon port=5432 user=apoon password=***********") And we can use pg_query to use our stored procedure:

```
$sql = "SELECT createTicket('$status', '$prob_des', '$prob_area')";

                  $result = pg_query($dbconn, $sql);
```

Stored procedures need parameters passed in for the sql statement. We use pg_query for this function. Passing these parameters into the stored procedure function, we will see how the function executes it.

```
CREATE OR REPLACE FUNCTION createTicket(
aticketStatus TEXT default 'Open',
aproblemDescription TEXT default 'Not Provided',
aproblemArea INT default null,
aaccountID INT

      ) RETURNS VOID  AS
$$

DECLARE
tempTicketID INT;

   BEGIN

            INSERT INTO ticket(ticketStatus, problemDescription,
problemArea) VALUES(aticketStatus, aproblemDescription, aproblemArea)
            RETURNING ticketID into tempTicketID;

END;
$$ LANGUAGE plpgsql;
```

5.4.2 Reports and Report Generator

We mainly used FPDF and Data Tables for making the reports. FPDF is an old tool, a PHP class that allows for you to generate PDF files from only PHP code. Data tables is a JQuery library that has tools for making enhanced tables with functionalities like searching, filtering, column visibility and more. We used Charts.js to make the beautiful graphs.

The reports and report generator uses views that have been created above. We used Data Tables Library to show the Data Tables and use its features to output the data, search through

the data, and extract Excel and PDF file from it. For a more detailed and meaningful report, we allowed for an aggregate summary of what is more in the report by using FPDF to use the aggregated view to make a table and output the customized graph from that specific query and dates.

Below is a part of the code written to display the customized graphs in FPDF through pure PHP:

```
//Graphs

$pdf->SetFont( 'Arial', 'B', 19 );
$chartTitle = "Chats Dropped";
$pdf->Cell( 180, 50, $chartTitle, 0, 0, 'C' );
$chartColours = array(
                  array( 255, 100, 100 ),
                  array( 100, 255, 100 ),
                  array( 100, 100, 255 ),
                  array( 255, 255, 100 ),
                );
$data = array(
          array( 2,  1, 0,  1 ),
          array( 1, 2, 2, 2 ),
          array( 2, 2, 2, 2 ),
          array( 2, 2, 3, 3 ),
        );
$chartXPos = 0;
$chartYPos = 250;
$chartWidth = 200;
$chartHeight = 100;
$chartXLabel = "Area";
$chartYStep = 5;

$xScale = count($rowLabels)/ ($chartWidth -30);


$yScale = ($maxTotal) / $chartHeight;
```

```
// Compute the bar width
$barWidth = /*$xScale-1;*/( 1 / $xScale ) / 1.5;
// Add the axes:
$pdf->SetFont( 'Arial', '', 10 );
// X axis

$pdf->Line( $chartXPos + 30, $chartYPos, $chartXPos +
$chartWidth, $chartYPos );
for ( $i=0; $i < count( $rowLabels ); $i++ ) {
  $pdf->SetXY( $chartXPos + 40 +  $i / $xScale, $chartYPos
);
  $pdf->Cell( $barWidth, 10, $rowLabels[$i], 0, 0, 'C' );
}
// Y axis
$pdf->Line( $chartXPos + 30, $chartYPos, $chartXPos + 30,
$chartYPos - $chartHeight - 8 );

for ( $i=0; $i <= $maxTotal; $i += $chartYStep ) {
  $pdf->SetXY( $chartXPos + 7, $chartYPos - 5 - $i /
$yScale );
  $pdf->Cell( 20, 10, ' ' . number_format( $i ) .'.00' , 0,
0, 'R' );
  $pdf->Line( $chartXPos + 28, $chartYPos - $i / $yScale,
$chartXPos + 30, $chartYPos - $i / $yScale );
}

// Add the axis labels
$pdf->SetFont( 'Arial', 'B', 12 );
$pdf->SetXY( $chartWidth / 2 + 20, $chartYPos + 8 );
$pdf->Cell( -50, 10, $chartXLabel, 0, 0, 'C' );
$pdf->SetXY( $chartXPos + 7, $chartYPos - $chartHeight - 12
);
$pdf->Cell( 20, 10, $chartYLabel, 0, 0, 'R' );
```

```
// Create the bars
$xPos = $chartXPos + 40;
$bar = 0;

foreach ( $array as $dataRow ) {

 $totalSales = 0;
   $totalSales = $dataRow;

  // Create the bar
  $colourIndex = $bar % count( $chartColours );
  $pdf->SetFillColor( $chartColours[$colourIndex][0],
$chartColours[$colourIndex][1],
$chartColours[$colourIndex][2] );
  $pdf->Rect( $xPos, $chartYPos - ( $totalSales / $yScale
), $barWidth, $totalSales / $yScale, 'DF' );
  $xPos += ( 1 / $xScale );
  $bar++;
}
```

5.4.3 Chat Features

These agents are for chat, so we would need the functionality of chat. In our application there is a chat functionality allowing you to send messages to the consumer. This Chat is saved in transcript and you will be able to pull up the history in html format whenever an agent chats with the consumer. This is done with Jquery and Ajax.

Chat Window:

This is the code for the chat window. This is the html code that allows for the user to write messages to the consumer.

```
<button class="collapsible" style="position: absolute; bottom:2%; right:
2%;">Open Chat</button>
      <div id="draggable"  class="content" style = "width: 750px; overflow-
y: scroll;"><div id = "chatwindow" style = "width:auto;">
        <h2 class = "chatperson">Chatting with: </h2>
            <div class="messages_display" id="chatstart"></div>
            <div class="container darker"  onsubmit="submitaction(event);">
                  <form action="" >
                        <textarea placeholder="Type message... " name="msg"
id="msgarea" style="width:100%" required></textarea>
                        <button type = "submit" class="btn">Send
Message</button>
                  </form>
            </div>

</div></div>
```

Chat Functionality:

This below code allows for additional functionality for the chat. This functionality includes: the agent's chat will be saved in history: chatstart() and updatechat(), the agent to send the message with the submit button: submitaction(). When an agent clicks [Join Chat] and opens up the chat, the chatstart() function will fire up and whenever there is a message that is sent through with the submitaction() will fire and pull the fillchatdisplay.php to append to the chat array of messages. The updatechat() is to update the chat on the receiving end for messages from the other party. This way we can parse the information to retrieve the previous messages.

```
function chatstart(e, id, chatid){
      e.preventDefault();
      $( ".chatperson" ).load( "chatbegin.php", {accountid: id, chat:
chatid} );
      $("#chatstart").html("");
      state = 0;
}
function submitaction(e){
      e.preventDefault();
      message = $('#msgarea').val()
      $('#msgarea').val("");
      $.ajax({
         type: "POST",
         url: "sendmessage.php",
         data: {
                        'message': message,
```

```
                        },
         success: function(data){
         },
      });
}

function updateChat(){

      $.ajax({
         type: "POST",
         url: "fillchatdisplay.php",
         data: {
                        'state':state,
                  },
         dataType: "json",
         success: function(data){

                  if (data.text != null) {
                        $("#chatstart").html("");

                        for (var i = 0; i < data.text.length; i++) {
                              $('#chatstart').append(data.text[i]);
                        }
                        document.getElementById('chatstart').scrollTop =
document.getElementById('chatstart').scrollHeight;

                  }
                  state = data.state;
                  setTimeout(updateChat, 1500);
         },
      });
}
```

fillchatdisplay.php

```
<?php
      session_start();

      function getfile($f) {

      if (file_exists($f)) {
            $lines = file($f);
```

```
        }

        return $lines;

    }

    function getlines($fl){
          return count($fl);
    }

    $state = htmlentities(strip_tags($_POST['state']), ENT_QUOTES);
      $file = "./chats/chattranscript";
      $file .= $_SESSION['chatid'];
      $file .= ".txt";


    if ($state == $count) {

        $log['state'] = $state;
        $log['text'] = false;

    } else {

        $text= array();
        $log['state'] = $state + getlines(getfile($file)) - $state;

        foreach (getfile($file) as $line_num => $line) {
            $text[] =  $line = str_replace("\n", "", $line);

            $log['text'] = $text;
        }
    }
    echo json_encode($log);
?>
```

5.4.4 Agent Console Features

The agent console is written with simple HTML and CSS with Chart.js for the Solution Guide. Below is the code that shows the form and the solution guide for each:

Form for Creating Ticket:

This is the form for creating the ticket. We will be using this with Jquery to connect the information with the solution guide.

```
      <form class="modal-content" action="" id="register_form"
method="post">
      <div class="container" style = "width:600px; height: 500px; ">
    <label for="category_dropdown"><b>Ticket Type</b></label>
    <select id="ticket_problem" name ="ticket_drop">
      <option selected="selected" disabled="disabled">Select  Type</option>
      <option value="AI">Account Issues</option>
    <option value="BI">Billing</option>
      <option value="RE">Repair</option>
      <option value="TS">Troubleshooting</option>
      </select>
      <br>
      <label for="problem_dropdown"><b>Issue</b></label>
    <select id="problem_category" name="problem_drop">
      <option selected="selected" disabled="disabled">
    </select>
      </br>
    </select>
      <label for="status_dropdown"><b>Ticket Status</b></label>
    <select id="ticket_status" name ="status_drop">
      <option selected="selected" disabled="disabled">Select
Status</option>
      <option value="Open">Open</option>
    <option value="Closed">Closed</option>
      </select>
      <!--label for="subject"><b>Subject</b></label>
    <input type="text" placeholder="Enter Subject" name="subject" required-
->
      <label for="description"><b>Description</b></label>
      <textarea rows="4" cols="50" name="problem_descr"
form="register_form" onfocus="this.value='';" >Enter Text
Here...</textarea>
```

```
      <button type="submit" class="signupbtn" name="submit_button">Submit
Ticket </button>
    </div>
  </form>
```

Code for Solution Guide:

This code uses chart.js and shows the output of the graphs itself from the database. Jquery will put this all together.

```
<?php

            $ChgE = 0;
            $SentE = 0;
            $RetrA = 0;
            $NA = 0;
            $query7 = "select * from accountSolutions";
            $result7 = pg_query($dbconn, $query7);
            $resultArr7 = pg_fetch_all($result7);
            echo
            '
            <h2 style="position: absolute; top: 90px; left: 50%;">Agent
Guide</h2>

                  <table style="position: absolute; top: 150px; left:
50%;">

                        <tr>
                         <th style ="text-align:left;"> Solutions</th>
                         <th> Frequency</th>
                        </tr>';

      foreach($resultArr7 as $array)
      {
                  echo '
                        <tr>
                              <td>'. $array['Solution'].'</td>
                              <td style="text-align:center;">'.
$array['Count'].'</td>
                        </tr>
                        <tr>
```

```
                        ';
                    switch ($array['Solution']) {
                        case "Changed Email":
                            $ChgE = $array['Count'];
                            break;
                        case "Sent Email":
                            $SentE = $array['Count'];
                            break;
                        case "Retrieved Account":
                            $RetrA = $array['Count'];
                            break;
                        case "No Action":
                            $NA = $array['Count'];
                            break;
                    }
        }
        echo '</table>
        <div class="boximg1" style="height:500px; width:500px; position:
absolute; top: 120px; right: 200px;"><canvas id="barChart1" style="
background-color: #FFF;"></canvas></div></div>
        <script>
        var cpx = document.getElementById("barChart1").getContext(\'2d\');
        var myBarChart1 = new Chart(cpx, {
            type: \'doughnut\',
            data: {
                    labels: ["Change Email", "Sent Email", "Retrieve
Account", "No Action"],
                    datasets: [{
                        label: \'Tickets Called For\',
                        data: [ '.$ChgE.', '.$SentE.', '.$RetrA.','.$NA.'],
                        backgroundColor: [
                            \'rgba(255, 99, 132, 0.2)\',
                            \'rgba(54, 162, 235, 0.2)\',
                            \'rgba(255, 206, 86, 0.2)\',
                            \'rgba(75, 192, 192, 0.2)\',
                            \'rgba(153, 102, 255, 0.2)\',
                            \'rgba(255, 159, 64, 0.2)\'
                        ],
                        borderColor: [
                            \'rgba(255,99,132,1)\',
                            \'rgba(54, 162, 235, 1)\',
                            \'rgba(255, 206, 86, 1)\',
                            \'rgba(75, 192, 192, 1)\',
```

```
                                \'rgba(153, 102, 255, 1)\',
                                \'rgba(255, 159, 64, 1)\'
                        ],
                        borderWidth: 1
                }]
            },
            options: {
                scales: {
                        yAxes: [{
                                ticks: {
                                        beginAtZero:true
                                }
                        }]
                }
            }
        });
        </script>';
        ?>
```

Code for JQuery:

Putting it all together, this JQuery code was used to display certain solutions and different dropdowns for specific choices ie: Account Issues, Billing, Troubleshooting, and Repair.

```
//Scroll Function
window.onscroll = function() {myFunction()};

var navbar = document.getElementById("navbar");
var rep_drop = document.getElementById("repair_drop");
var acct_drop = document.getElementById("account_drop");
var bill_drop = document.getElementById("billing_drop");
var ts_drop = document.getElementById("trouble_drop");

//SELECTION
$('#ticket_problem').on('change', function(){
   console.log($('#ticket_problem').val());
    $('#problem_category').html('');
    if($('#ticket_problem').val()=='AI'){
        $('#problem_category').append('<option value="7">Hacked by Main
Group</option>');
            $('#problem_category').append('<option value="2">Hacker: Do Not
```

```
Unban</option>');
            $('#problem_category').append('<option value="1">Incorrect
Email</option>');
        $('#problem_category').append('<option value="2">Lost
Password</option>');
            $('#problem_category').append('<option value="2">Agent did not
send email</option>');
            $('#problem_category').append('<option value="2">Case was not
worked</option>');

            rep_drop.style.display = "none";
            acct_drop.style.display = "block";
            bill_drop.style.display = "none";
            ts_drop.style.display = "none";
      }
     if($('#ticket_problem').val()=='BI'){
        $('#problem_category').append('<option value="4">Credit
Card</option>');
        $('#problem_category').append('<option value="6">Store
Credit</option>');
            rep_drop.style.display = "none";
            acct_drop.style.display = "none";
            bill_drop.style.display = "block";
            ts_drop.style.display = "none";
    }
      if($('#ticket_problem').val()=='RE'){
        $('#problem_category').append('<option value="5">Out of
Warranty</option>');
        $('#problem_category').append('<option value="3">In
Warranty</option>');
            rep_drop.style.display = "block";
            account_drop.style.display = "none";
            bill_drop.style.display = "none";
            ts_drop.style.display = "none";
    }
            if($('#ticket_problem').val()=='TS'){
        $('#problem_category').append('<option value="5">HDD
Replacement</option>');
        $('#problem_category').append('<option value="3">Fixed Internet
Connection</option>');
            $('#problem_category').append('<option value="3">Referred to
Third Party</option>');
            $('#problem_category').append('<option value="3">Referred to
```

```
Internet Service Provider</option>');
        $('#problem_category').append('<option value="6">Firmware
Update</option>');
            rep_drop.style.display = "none";
            account_drop.style.display = "none";
            bill_drop.style.display = "none";
            ts_drop.style.display = "block";
    }
});
```

5.4.5 Major Features

The major features of the program is that it is able to give dynamic reports for the agent to view such important details like: customer satisfaction, performance, and chats dropped in the reports. They are able to export the data into Excel file for data manipulation, they are able to pull PDFs to see the aggregated results of the analysis.

Also, agents are able to chat and use the agent console to create and update tickets, all the while displaying possible solutions to guide that call and submit to the specialist to complete. This database and its functionality is a great tool for future customer service departments because it can guide agents to finishing the chat faster with dynamic guidance systems.

5.4.6 Learning New Tools

This was made with HTML, CSS, Javascript, JQuery, Ajax, PHP. The libraries used were: FPDF, Data Tables, and Charts.js. I have always been a backend person and I only knew C/C++ and Java programming before so front-end was completely new to me. I did not know as much as I wish I did coming into this class, but I was so happy I learned so much. I feel as though this has really helped me with my coding and design skills. Because I was just using HTML, CSS, Javascript these languages had much support and it was easy to find answers and learn from resources.

5.5 Design and Implementation Process

To note, we would like to briefly go over all the steps we have followed to implement this database application and what we have gathered through this process.

Requirements Collection and Analysis

This phase starts it all. It is all about understanding your business, your needed functionalities, the components and how each component will connect to one another. In this phase we need to really know about our business and the operations and how it is all connected with one another.

We need to ask the Subject Matter Experts and many operation experts about the business to really know about the operations itself.

Conceptual Database Design

This phase requires a designer to know the organization's structure very well to develop a visual representation of the structure through a E-R diagram. In this phase, we must be careful in really looking into all the needs and functionalities for each party to ensure that our design works. We need to make sure we explore all the needs to expose any potential problems early during this phase so we can fix them now, since this is the foundation of our application.

Logical Database Design

This phase requires us to convert this conceptual E-R model to a relational model for software implementation on our application. This is a must when we are starting to implement the software side. Through this conversion we need to make sure that converting is correctly being done the way it should.

Physical Database Implementation

This physical database implementation phase is when we start building the database with DBMS like PostgreSQL with functionalities useful to our application. The database should have many insertions of data so we can visualize any potential problems that arise early on before development. This is also where we will write the stored procedures, views, and triggers to use for the application. Really thinking about the relationships and how the database will be used with our application assisted me greatly in this phase.

Database Application

This is the last phase where we will build a user interface for the parties to connect and interact with the database. This will include all the functionalities needed and discussed since phase 1. The main priority with this phase is really to make this as user-friendly as possible. Once implemented all the users will be able to test the website and make sure that everything is working accordingly.

5.6 Peer Evaluation

Outcome	Anna Poon	Luis Manahan
An ability to analyze a problem, and identify and define the computing requirements and specifications appropriate to its solution.	9	10
An ability to design, implement and evaluate a computer-based system, process, component, or program to meet desired needs, An ability to understand the analysis design, and implementation of a computerized solution to a real-life problem.	9	9
An ability to communicate effectively with a range of audiences. An ability to write a technical document such as a software specification white paper or a user manual.	9	8
An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrations comprehension of the tradeoffs involved in design choices.	9	8