Assignment 7

Database Design

|  |  |
| --- | --- |
| Group # | CP 5 |
| Student Name | Rakesh Mahendranath, Govind Kala Raveendran, Charumathi Satish, Aiswarya Raj, Het Chandubhai Patel, Vignesh Ramasubramanian |
| Course | INFO8686 – Information Technology Business Analysis Capstone Project |
| Section | 5 |
| Professor | Prof. Nilesh Deshmukh |
| Due Date | March 14, 2024 |

Catalyst Driving Business Excellence

Kitchener, Ontario

Generative AI Powered Voice Assistant Customer Service at Rogers

Table of Contents

[Database Design 3](#_Toc161352271)

[Purpose of each of the elements 4](#_Toc161352272)

[Tables 5](#_Toc161352273)

[Primary Keys 8](#_Toc161352274)

[Normalization of tables 9](#_Toc161352275)

[Dependencies of data elements 13](#_Toc161352276)

[References 14](#_Toc161352277)

# Database Design

A screenshot of a computer

Description automatically generated

# Purpose of each of the elements

1. Customers: This table will store customer information like name, contact details, and any relevant preferences or profiles.
2. Tickets: This table would store details of customer service tickets or cases, such as ticket ID, subject, description, status, and other relevant metadata.
3. AgentCallAssignments: This table would map customer service agents to specific customer calls or interactions, enabling tracking and assignment of calls to available agents.
4. Agents: This table will store information about the customer service agents, such as their employee ID, names, and role.
5. ResolutionDetails: This table will store the details of when the customer's issue or ticket was resolved, including a description, solutions provided, and any relevant notes.
6. CallTranscripts: This table would store the transcripts of customer calls or voice interactions with the AI assistant, enabling analysis and training of the conversational AI models.
7. CallIntents: This table would store the various intents that the voice assistant needs to recognize and handle during customer calls.
8. CustomerInteractions: This table would log and store details of customer interactions with the voice assistant, such as interaction ID, Ticket ID, timestamp, and any relevant metadata.
9. TicketCategories: This table would store the different categories or types of customer service tickets or issues, enabling better routing and handling of tickets based on their category.
10. VoiceModels: This table would store the trained voice models, enabling the voice assistant to support multiple languages, accents, and customizable voice personas.
11. InteractionFeedback: This table would store customer feedback and ratings for their interactions with the voice assistant, enabling quality assurance and continuous improvement of the conversational AI models.
12. CallTags: This table would store tags or labels that can be associated with customer calls or interactions, enabling better categorization and analysis of call data.
13. SentimentDetails: This table would store sentiment analysis data extracted from customer interactions, such as sentiment scores, emotions detected, and other relevant metadata. This information can be used to adapt the voice assistant's responses and handle customer emotions more effectively.

# Tables

1. **Customers**

|  |  |  |
| --- | --- | --- |
| **Element Name** | **Value Range** | **Data Type** |
| CustomerID (PK) | Positive integers (no duplicates) | INT |
| FirstName | Text strings | VARCHAR (50) |
| LastName | Text strings | VARCHAR (50) |
| Email | Unique email addresses | VARCHAR (255) |
| PhoneNum | Text strings with appropriate formatting (e.g., +1234567890) | VARCHAR (20) |
| Addr1 | Text strings | VARCHAR (255) |
| Addr2 | Text strings | VARCHAR (255) |
| City | Text strings | VARCHAR (100) |
| State | Text strings (consider using a standardized abbreviation list) | VARCHAR (50) |
| Zip | Text strings with appropriate formatting (e.g., 12345) | VARCHAR (10) |
| Country | Text strings | VARCHAR (100) |
| UserPref | Text strings or a coded system representing preferences | VARCHAR (100) |

1. **Tickets**

|  |  |  |
| --- | --- | --- |
| **Element Name** | **Value Range** | **Data Type** |
| TicketID (PK) | Positive integers (no duplicates) | INT |
| Title | Text strings | VARCHAR (255) |
| Description | Text strings | TEXT |
| Status | Defined list of status options | VARCHAR (10) |
| Priority | Defined list of priority options | VARCHAR (10) |
| CreatedAt | Date and time format (e.g., DATETIME) | DATETIME |
| UpdatedAt | Date and time format (e.g., DATETIME) | DATETIME |
| CustomerID | Positive integer referencing Customers.CustomerID | INT |
| ResolutionID | Positive integer referencing ResolutionDetails.ResolutionID (can be NULL if not resolved) | INT |
| TicketCategory | Defined list of category options | VARCHAR (100) |

1. **AgentCallAssignments**

|  |  |  |
| --- | --- | --- |
| **Element Name** | **Value Range** | **Data Type** |
| AssignmentID (PK) | Positive integers (no duplicates) | INT |
| TicketID | Positive integer referencing Tickets.TicketID | INT |
| AgentID (FK) | Positive integer referencing Agents.AgentID | INT |
| AssignedTime | Date and time format (e.g., DATETIME) | DATETIME |
| CompletionTime | Date and time format (e.g., DATETIME) | DATETIME |

1. **Agents**

|  |  |  |
| --- | --- | --- |
| Element Name | Value Range | Data Type |
| AgentID (PK) | Positive integers (no duplicates) | INT |
| EmployeeID | Text strings or integers referencing another system | INT |
| Department | Defined list of department options | VARCHAR (50) |
| RoleID | Positive integer referencing a Roles table (if applicable) | INT |

1. **ResolutionDetails**

|  |  |  |
| --- | --- | --- |
| Element Name | Value Range | Data Type |
| ResolutionId (PK) | A positive integer, auto-incrementing | INT |
| ResolutionDate | Date and time (YYYY-MM-DD HH:MM:SS) | DATETIME |
| ResolutionDesc | Textual description | VARCHAR (255) |
| ResolvedBy | Username or employee ID | VARCHAR (50) |
| Comments | Textual information | VARCHAR (255) |

1. **CallTranscripts**

|  |  |  |
| --- | --- | --- |
| Element Name | Value Range | Data Type |
| TranscriptId (PK) | A positive integer, auto-incrementing | INT |
| Transcript | Text data | TEXT |
| InteractionId (FK) | Positive integer referencing a valid InteractionId | INT |
| TimeStamp | Date and time (YYYY-MM-DD HH:MM:SS) | DATETIME |
| ConfidenceLevel | Decimal value between 0.0 and 1.0 (0 = low, 1 = high) | DECIMAL (3,2) |

1. **CallIntents**

|  |  |  |
| --- | --- | --- |
| Element Name | Value Range | Data Type |
| IntentId (PK) | A positive integer, auto-incrementing | INT |
| IntentName | Textual description (e.g., "check\_balance", "report\_outage") | VARCHAR (50) |
| IntentDesc | Textual explanation of the user's intent | VARCHAR (255) |
| ConfidenceLevel | Decimal value between 0.0 and 1.0 (0 = low, 1 = high) | DECIMAL (3,2) |

1. **CallTags**

|  |  |  |
| --- | --- | --- |
| Element Name | Value Range | Data Type |
| TagId (PK) | A positive integer, auto-incrementing | INT |
| TagName | Textual description (e.g., "urgent", "billing issue") | VARCHAR (50) |
| TagDesc | Textual explanation for applying the tag | VARCHAR (255) |

1. **CustomerInteractions**

|  |  |  |
| --- | --- | --- |
| Element Name | Value Range | Data Type |
| InteractionId (PK) | Positive integer, auto-incrementing | INT |
| TicketId (FK) | Positive integer referencing a valid TicketId or NULL if not linked to a ticket | INT |
| IntentId (FK) | Positive integer referencing a valid IntentId | INT |
| VoiceModelld (FK) | Positive integer referencing a valid VoiceModelld | INT |
| InteractionType | Textual description (limited set of options) | VARCHAR (20) |
| StartTime | Date and time (YYYY-MM-DD HH:MM:SS) | DATETIME |
| EndTime | Date and time (YYYY-MM-DD HH:MM:SS) | DATETIME |
| SentimentId (FK) | Positive integer referencing a valid SentimentAnalysisld | INT |
| Comments | Textual information | VARCHAR (255) |

1. **SentimentDetails**

|  |  |  |
| --- | --- | --- |
| Element Name | Value Range | Data Type |
| SentimentAnalysisId (PK) | Positive integers, no duplicates | INT |
| SentimentScore | Decimal values between -1.0 (very negative) and 1.0 (very positive), with 0 representing neutral | DECIMAL (5,2) |
| SentimentLabel | Predefined set of strings (e.g., "Positive", "Negative", "Neutral") | VARCHAR (255) |
| AnalysisTimestamp | Date and time values | DATETIME |

1. **TicketCategories**

|  |  |  |
| --- | --- | --- |
| Element Name | Value Range | Data Type |
| TicketCategoryId  (PK) | Positive integers, no duplicates | INT |
| CategoryName | Textual data | VARCHAR (255) |
| CategoryDesc | Textual data, potentially lengthy | TEXT |

1. **InteractionFeedback**

|  |  |  |
| --- | --- | --- |
| Element Name | Value Range | Data Type |
| FeedbackId (PK) | Positive integers, no duplicates | INT |
| InteractionId (FK) | Positive integers, no duplicates | INT |
| Rating | Integers between 1 (lowest) and 5 (highest) | INT |
| Comment | Textual data, maximum 4000 characters | TEXT |
| FeedbackDate | Date values | DATE |

1. **Voice Models Table**

|  |  |  |
| --- | --- | --- |
| **Ele**ment Name | Value Range | Data Type |
| VoiceModelld (PK) | A positive integer, auto-incrementing | INT |
| VoiceModelName | User-defined name | VARCHAR (50) |
| Language | Defined set of language codes (e.g., "en-US", "fr-CA") | VARCHAR (10) |
| Gender | Defined set of options (e.g., "Male", "Female", "Neutral") | VARCHAR (10) |
| Description | Free text entry | VARCHAR (255) |
| Metadata | JSON format | TEXT |

# Primary Keys

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Primary Key Name** | **Datatype** |
| Customers | CustomerID | Integer |
| Tickets | TicketID | Integer |
| AgentCallAssignments | AssignmentID | Integer |
| Agents | AgentID | Integer |
| Voicemodels | VoicemodelID | Integer |
| CallTranscripts | TranscriptID | Integer |
| CallIntents | IntentID | Integer |
| CustomerInteractions | InteractionID | Integer |
| TicketCategories | TicketCategoryID | Integer |
| CallTags | TagID | Integer |
| SentimentDetails | SentimentAnalysisID | Integer |
| InteractionFeedback | FeedbackID | Integer |
| ResolutionDetails | ResolutionID | Integer |

# Normalization of tables

An essential step in database design is normalization, which aims to efficiently organize data, reduce redundancy, and maintain data integrity. We have ensured that the database schema is rationally and appropriately designed using normalization principles, making data management, retrieval, and storage easier. The normalization procedures achieve the third normal form (3NF) in the AI Customer Care Call Platform database schema.

**Normalization**

1. **Identified Entities and Attributes**

* All entities and their respective attributes were identified based on the feature's requirements with Rogers’ Database Schema.
* Each entity represents a distinct object in the system, and attributes describe the characteristics of each entity.

1. **Established Relationships**

* Identified the one-to-one, one-to-many, and many-to-many relationships between entities with the help of different keys.
* The cardinality and how the entities interact with one another determine the establishment of relationships.

1. **Implement normalization rules:**

* Normalization rules are used in this database schema to eliminate data redundancy and maintain data integrity.
* Tables are normalized to the third normal form (3NF), achieving 1NF and 2NF by removing functional, partial, and transitive dependencies.

**Steps:**

1. **Identify functional dependencies**:

* Determine all the attributes that depend on the primary key.
* Identify dependencies that are partial or transitive.

1. **Eliminate Patrial dependencies:**

* Ensure that each non-key attribute is fully functionally dependent on the primary key.
* If any attributes are partially dependent on the primary key, transfer them to a separate relation.

1. **Remove Transitive dependencies**:

* Move any transitive dependencies to a new table, adding more tables to the schema. Transitive dependencies occur when a non-key attribute determines another non-key attribute.

Ensure the table is in the third normalization form, confirming that all attributes are fully dependent on the primary key and that no transitive dependencies exist.

**Considering the Tickets table:**

Before normalization:

A screenshot of a computer

Description automatically generated

**Normalization process:**

1. **Identify functional dependencies:**

* TicketID à Title, Description, Status, Priority, CreatedAt, UpdatedAt, CustomerId, ResolutionId, TicketCategoryId
* (Determinant à Dependencies)

1. **Eliminate Patrial dependencies:**

* There is a partial dependency between ResolutionId and ResolutionDate, ResolvedBy, and Comments.
* The attributes ResolutionDate, ResolvedBy, and Comments are functionally dependent on ResolutionId, which is not the primary key.
* Created a new table named ResolutionDetails with ResolutionId as the primary key and attributes ResolutionDate, ResolvedBy, and Comments.

A screenshot of a computer

Description automatically generated

1. **Remove Transitive dependencies:**

* There is a transitive dependency between TicketCategoryId and CategoryName and CategoryDesc.
* TicketCategoryId determines CategoryName and CategoryDesc, and vice versa. This implies that CategoryName and CategoryDesc transitively determine TicketCategoryId.
* Created a new table named TicketCategories with TicketCategoryId as the primary key with attributes CategoryName and CategoryDesc.

A screen shot of a ticket categories

Description automatically generated

**After normalization:**

Three tables have been created.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screen shot of a ticket categories

Description automatically generated

# Dependencies of data elements

1. Each customer interaction is associated with exactly one voice model. However, a voice model may be associated with zero or one interaction.
2. A customer interaction can have either zero or one interaction feedback. Conversely, each interaction feedback is tied to exactly one customer interaction.
3. Every customer interaction contains exactly one sentiment detail, but a sentiment detail may be linked to multiple interactions or none.
4. Each customer interaction is linked to precisely one and only one ticket, while a ticket can be associated with multiple customer interactions or none.
5. Every customer interaction will have one and only one call intent, but a call intent may be connected to multiple customer interactions or none.
6. A customer interaction may have zero or multiple call tags, and vice versa.
7. Each customer interaction will have one and only one call transcript, and vice versa.
8. A customer can have many zero or many tickets and a ticket can be connected to one and only one customer.
9. A ticket can have zero or many agent call assignments while a agent call assignment will be connected to one and only one ticket.
10. A ticket will have one and only one resolution detail while one resolution detail can be connected to zero or many tickets.
11. A ticket can have one and only one ticket category and vice versa.
12. An agent call assignment can be connected to one and only one agent while an agent can have zero or many agent call assignment.

# References

1. (2023, November 6). *Normal Forms in DBMS*. Geeksforgeeks. [https://www.geeksforgeeks.org/normal-forms-in-dbms/](https://mcas-proxyweb.mcas.ms/certificate-checker?login=false&originalUrl=https%3A%2F%2Fwww.geeksforgeeks.org.mcas.ms%2Fnormal-forms-in-dbms%2F%3FMcasTsid%3D15600&McasCSRF=3333181fe886a82420cfaecd236bcac7568239972df778b176e4c6973080688f)
2. Dutton, C. (2018, May). *Database normalization - Microsoft Excel Video Tutorial | LinkedIn Learning, formerly Lynda.com*. LinkedIn. https://www.linkedin.com/learning/excel-business-intelligence-data-modeling-101/database-normalization?u=2212217