**Database Systems Project Part II -**

**Logical Schema Optimization and Unstructured Data Collection**

Ming Gao (mg7572), Wes Wang (lw2534)

# **1. Data Lake Introduction**

Below, we list all the entity tables and all the middle tables supporting our many-to-many relationships. In entity tables, we clearly state the primary keys, foreign keys and the entity they are referencing, and we also note if the primary key is generated by us manually or exists in the original schema itself. In many-to-many middle tables, we clearly state the foreign keys, the entity they are referencing, their name (roles) in the middle table if different to the referenced attribute. We also list all other attributes hence to show they are independent and there’s no dependency towards other attributes in the same table, other than primary keys.

**\*Note\*** notations like “A (B → C)” indicates that the Attribute A in the current table is referencing the Attribute C in Table B.

## **1.1 Entity Tables**

Account

|  |  |
| --- | --- |
| **Keys** | |
| Primary key | Account\_id (generated) |
| Foreign key | CompanyCode (CompanyCode → CompanyCode) |
| **Other Attributes** | |
| AccountName, AccountName2, Address1, Address2, City, State, ZIP, Photo | |

AcctAdmin

|  |  |
| --- | --- |
| **Keys** | |
| Primary key | AcctAdmin\_id (generated) |
| **Other Attributes** | |
| AdminFirstName, AdminLastName, AdminMiddleInitial, AdminSuffix, Address1, Address2, City,State, ZIP | |

BillingAccount

|  |  |
| --- | --- |
| **Keys** | |
| Primary key | BillingAccount\_id (generated) |
| **Other Attributes** | |
| BAcctName, BAcctName2, Address1, Address2, City, State, ZIP | |

AccountLegacyAlias

|  |  |
| --- | --- |
| **Keys** | |
| Primary key | AliasID |
| Foreign key | Account\_id (Account → Account\_id) |
| Foreign key | CompanyCode (CompanyCode → CompanyCode) |
| **Other Attributes** | |
| AliasName, AliasSource, Address1, Address2, City, State, ZIP | |

CompanyCode

|  |  |
| --- | --- |
| **Keys** | |
| Primary key | CompanyCode |
| **Other Attributes** | |
| LegacyCompanyNo, CompanyName | |

Contract

|  |  |
| --- | --- |
| **Keys** | |
| Primary key | ContractNumber |
| Foreign key | CompanyCode (CompanyCode → CompanyCode) |
| **Other Attributes** | |
| ActivityStatus, ActivityStatusDate, CoverageType, BillingMethod, Duration, ExpirationDate | |

ContractBenefit

|  |  |
| --- | --- |
| **Keys** | |
| Primary key | ContractBenefit\_id (generated) |
| Foreign key | ContractNumber (Contract → ContractNumber) |
| **Other Attributes** | |
| PolicyCountContribution | |

ContractPremium

|  |  |
| --- | --- |
| **Keys** | |
| Primary key | PremiumCode |
| Foreign key | ContractBenefit\_id (ContractBenefit → ContractBenefit\_id) |
| **Other Attributes** | |
| AnnualizedPremium, ProcessDate, AppSignDate | |

ManagerContract

|  |  |
| --- | --- |
| **Keys** | |
| Primary key | SitCode |
| Foreign key | Associate\_id (Associate → Associate\_id) |
| **Other Attributes** | |
| IssueDate, ContractType, ContractSignDate, ContractProcessDatem, EndDate, StateCoordinators\_EndDate | |

Associate

|  |  |
| --- | --- |
| **Keys** | |
| Primary key | Associate\_id (generated) |
| **Other Attributes** | |
| AssocFirstName, AssocLastName, AssocMiddleInitial, AssocSuffix, AssocDOB, TenureDate | |

Customer

|  |  |
| --- | --- |
| **Keys** | |
| Primary key | Customer\_id (generated) |
| **Other Attributes** | |
| CusFirstName, CusLastName, CusMiddleInitial, CusDOB, Gender, SSN\_TIN | |

## **1.2** **Many-to-Many Relationship Middle Tables**

Account-Account

|  |  |
| --- | --- |
| **Keys** | |
| Foreign Key 1 | Master\_id (Account → Account\_id) |
| Foreign Key 2 | Member\_id (Account → Account\_id) |

Acct\_AcctAdmin

|  |  |
| --- | --- |
| **Keys** | |
| Foreign Key 1 | Account\_id (Account → Account\_id) |
| Foreign Key 2 | AcctAdmin\_id (AcctAdmin → AcctAdmin\_id ) |

Account\_BillingAccount

|  |  |
| --- | --- |
| **Keys** | |
| Foreign Key 1 | Account\_id (Account → Account\_id) |
| Foreign Key 2 | BillingAccount\_id (BillingAccount → BillingAccount\_id ) |
| **Other Attributes** | |
| RelationshipType, StartDate, BillingFrequency, SpecificationCode, AccountType | |

Account\_Member

|  |  |
| --- | --- |
| **Keys** | |
| Foreign Key 1 | Customer\_id (Customer → Customer\_id ) |
| Foreign Key 2 | Account\_id (Account → Account\_id) |
| **Other Attributes** | |
| StartDate | |

ContractingPartyInRole

|  |  |
| --- | --- |
| **Keys** | |
| Foreign Key 1 | Customer\_id (Customer → Customer\_id ) |
| Foreign Key 2 | ContractNumber (Contract → ContractNumber) |

Premium\_MgmtContract

|  |  |
| --- | --- |
| **Keys** | |
| Foreign Key 1 | PremiumCode (ContractPremium → PremiumCode) |
| Foreign Key 2 | SitCode (ManagerContract → SitCode) |
| **Other Attributes** | |
| Amount, CommissionRate | |

Account\_Associate

|  |  |
| --- | --- |
| **Keys** | |
| Foreign Key 1 | Account\_id (Account → Account\_id) |
| Foreign Key 2 | SitCode (ManagerContract → SitCode) |
| **Other Attributes** | |
| StartDate, StopDate | |

Associate-Associate

|  |  |
| --- | --- |
| **Keys** | |
| Foreign Key 1 | Recruiter\_id (Associate → Associate\_id) |
| Foreign Key 2 | Broker\_id (Associate → Associate\_id) |

BenefitForBenefitParty

|  |  |
| --- | --- |
| **Keys** | |
| Foreign Key 1 | Customer\_id (Customer → Customer\_id ) |
| Foreign Key 2 | PremiumCode (ContractPremium → PremiumCode) |

Customer-Customer

|  |  |
| --- | --- |
| **Keys** | |
| Foreign Key 1 | Customer\_id1 (Customer → Customer\_id ) |
| Foreign Key 2 | Customer\_id2 (Customer → Customer\_id ) |

CoordPosition\_ProductionAsset

|  |  |
| --- | --- |
| **Keys** | |
| Foreign Key 1 | PremiumCode (ContractPremium → PremiumCode) |
| Foreign Key 2 | Associate\_id(Associate → Associate\_id) |
| **Other Attributes** | |
| SitCode, IssueDate, ContractType, ContractSignDate, ContractProcessDate, ManagerContractEndDate, ProductionCreditSplitPercentage, EndDate, TerritoryName, TerritoryStartDate, TerritoryEndDate, TerritoryCoordinatorEndDate, Level, LevelName, LevelAbbreviation, StateCoordinatorsEndDate | |

# **2. Fake Data**

We used Mockaroo to create the fake data. We created a table for each of the entities we included in our previous ER diagram, selected the attribute types and domain, and exported them as CSV files with 1000 rows. Then, to simulate the one-to-many and many-to-many relationships, we injected foreign keys into the tables. For instance, suppose table A has a foreign key K referencing the attribute T in table B. We randomly select and inject values in T to each row of K in A, with repetition. By doing this, we ensure that there will be many rows in A where the values of K are corresponding to the same value in T. We do this for both many-to-many and one-to-many relationships, and the only difference is that we need to create additional middle tables for the many-to-many relationships, which we also listed below.

# **3. Optimization for Logical Database Schema**

## **3.1 Normalization**

When creating the logical database schema, we sufficiently considered the potential issues of normalization so we designed and created our logical database schema carefully and comprehensively. Beyond the generated relations via Many-to-Many relationships, we set or added a single primary key for each remaining relation. Hence, for each relation in our logical database schema, we let the primary key(s) determine all the remaining attributes. At the same time, from the previous sections, it’s easy to clearly see that all the non-PK attributes in each relation are independent and don’t have any other functional dependencies.

For the generated relations via Many-to-Many relationships, each of them has composite keys and every part of the key is necessary to determine any non-prime attributes, which indicates the generated relations are all in BCNF. For the original relations, each of them has only one primary key and the primary key determines all other attributes, which also indicates that all the original relations are in BCNF because every determinant in each functional dependency is a superkey (i.e. the primary key).

Therefore, generally speaking, each relation in our logical database schema is in BCNF (i.e. Boyce-Codd Normal Form), which means we don’t need to implement any normalization-related optimization.

## **3.2 Combining Structured and Unstructured Data**

In our data lake, we had an attribute Photo in the relation Account to present the uploaded profile photos of each account. And it was the only unstructured data source in our project. Such a design was to store and manage our structured and unstructured data efficiently.

Amazon S3 is a scalable object storage service provided by Amazon Web Services (AWS). It enables users to store and retrieve any amount of data from anywhere on the web. And it is widely used for backup and recovery, data archives, and data lakes for analytics. Therefore, we would like to use MySQL to store all the structured data and leverage Amazon S3 to store and manage all the unstructured data (i.e. profile photos).

The details of managing our hybrid data based on MySQL and Amazon S3 are as follows:

1. Storing Photos in AWS S3: Upload the photos to AWS S3 and each photo will have a unique URL once uploaded.
2. Designing and Creating MySQL Tables: Follow our logical database schema to create tables in MySQL.
3. Inserting Data into MySQL: Retrieve all photos’ URLs from AWS S3 and store them in the MySQL database.

# **Appendix**

We put all the photo data into Google Drive. The link is as follows:

<https://drive.google.com/drive/folders/1_b91YRIPoG3QZCf7oI5yKAjCmHFoTxki?usp=sharing>