University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2
Date: Feb 28, 2024
Group Number: 50

Name	Student Number	CS Alias	E-mail Address
Romina Mahinpei	39405501	g3m4h	mahinpei@student.ubc.ca
Annie Wang	47832274	g4m0w	anniew02@student.ubc.ca
Sohbat Sandhu	79661179	n4c2w	sohbat@student.ubc.ca

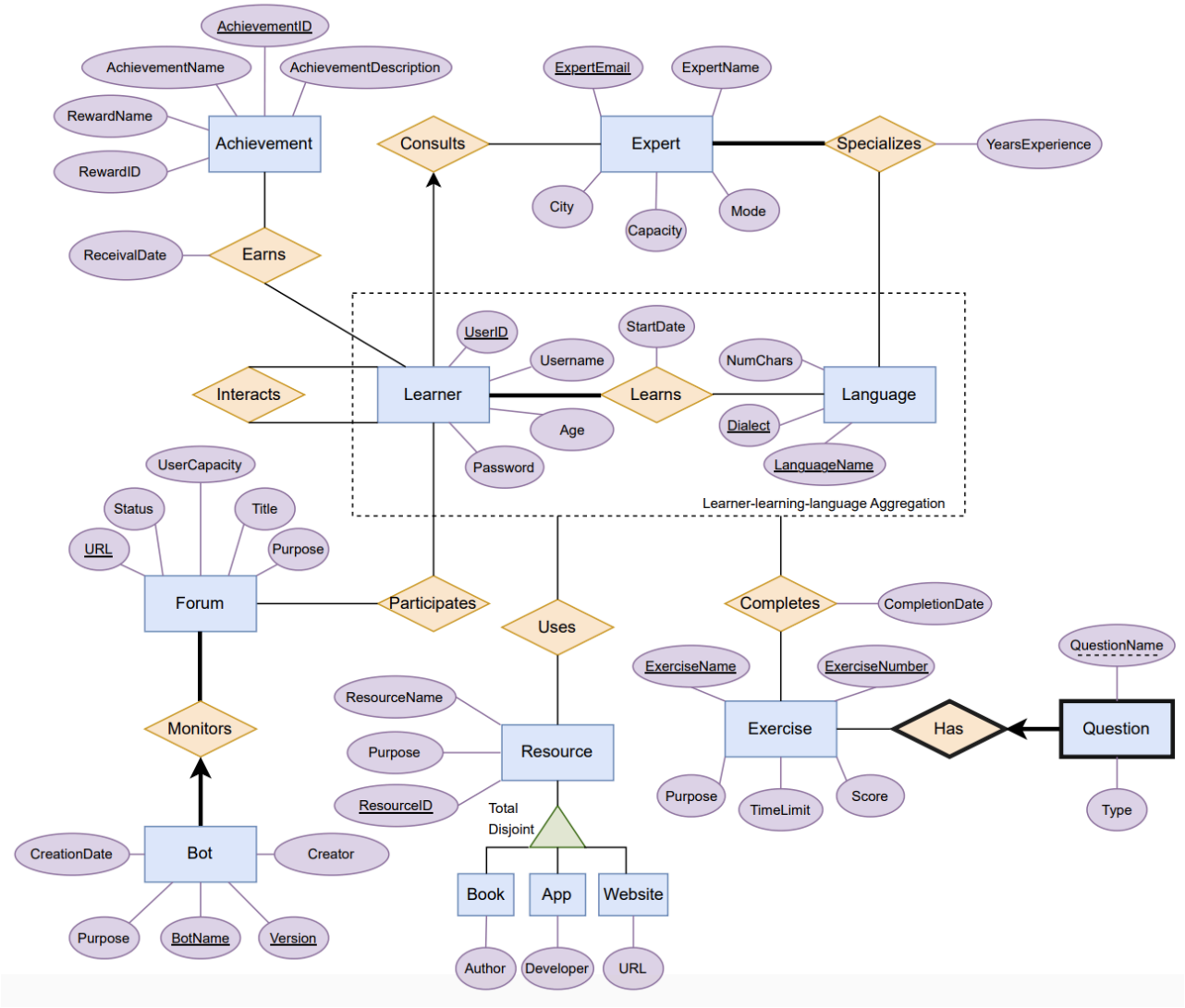
By typing our names and student numbers in the above table, we certify that the work in the attached assignment was performed solely by those whose names and student IDs are included above. (In the case of Project Milestone 0, the main purpose of this page is for you to let us know your e-mail address, and then let us assign you to a TA for your project supervisor.)

In addition, we indicate that we are fully aware of the rules and consequences of plagiarism, as set forth by the Department of Computer Science and the University of British Columbia

Project Summary

Our application aspires to provide an engaging online environment that **allows users to comprehensively learn new languages** with fellow passionate learners while being assisted by various learning resources. Our focus extends beyond mere language acquisition - we strive to facilitate a comprehensive language learning journey. Given this high-level goal, we consider the domain of our application to be **language learning**.

ER Diagram



Many of the entity sets with a Name or ID attribute were renamed to differentiate from other entity sets with the same attribute names. A few additional attributes were added to many entity sets to create a more realistic language-learning application, which also added more interesting functional dependencies to our project.

Listed below are the changes made to each entity and relationship set:

Achievement

- ➔ Replaced Milestone with AchievementID and AchievementDescription (based on our TA's recommendation) where AchievementID is the new primary key
- ➔ Replaced Reward with RewardID and RewardName

Expert

- ➔ Renamed Name to ExpertName
- ➔ Replaced ExpertID with ExpertEmail and ExpertName where ExpertEmail is the new primary key
- ➔ Added attributes Capacity, Mode and City

Learner

- ➔ Renamed Name to Username
- ➔ Added attribute Password to enable an account aspect for the learner

Language

- ➔ Renamed Name and # Chars to LanguageName and NumChars respectively
- ➔ Added attribute Dialect to distinguish between dialects of certain languages.
  - ◆ For example, if a learner wants to learn Chinese, they can specify whether they want to learn Mandarin or Cantonese
- ➔ Removed Region attribute because it doesn't capture the language-learning experience

Forum

- ➔ Renamed Name to Title
- ➔ Added attributes UserCapacity and URL where URL is the new primary key

Bot

- ➔ Renamed Name to BotName
- ➔ Removed Description attribute because it's too similar to the Purpose attribute
- ➔ Added attributes CreationDate, Creator, and Version
- ➔ (BotName, Version) is the new primary key

Resource

- ➔ Renamed Name to ResourceName

App

- ➔ Renamed Publisher to Developer

Exercise

- ➔ Replaced ExerciseID with ExerciseName and ExerciseNumber where (ExerciseName, ExerciseNumber) is the new primary key
- ➔ Added attributes Purpose and TimeLimit

Question

- ➔ Renamed Name to QuestionName

Specializes

- ➔ Renamed # Years of Experience to YearsExperience to eliminate special characters, spaces, and filler words in the attribute name

Relational Schema and Functional Dependencies (FDs)

Note about our Relational Schema: We have underlined our primary keys, **bolded** our foreign keys, and explicitly stated our candidate keys as an additional bullet point. We have also inlined our UNIQUE and NOT NULL constraints within our table definitions and have stated any constraints that we were unable to capture within the relational model.

Note about our Functional Dependencies: We have provided our non-trivial FDs right after each table definition. Furthermore, we have highlighted the FDs implied by our primary and candidate keys in blue, our explicit FDs in green, and our implicit FDs in purple.

1) We first list the table definitions and FDs for the "standalone" entity sets:

Language(LanguageName: VARCHAR(50), Dialect: VARCHAR(50), NumChars: INTEGER)

- ➔ LanguageName, Dialect → NumChars
- ➔ LanguageName → NumChars

Expert(ExpertEmail: VARCHAR(50), ExpertName: VARCHAR(50), Capacity: INTEGER, Mode: VARCHAR(50), City: VARCHAR(50))

- ➔ ExpertEmail → ExpertName, Capacity, Mode, City
- ➔ Mode → Capacity
- ➔ ExpertName, City → Mode
- ➔ ExpertName, City → Capacity

Forum(URL: VARCHAR(100), Status: VARCHAR(50), UserCapacity: INTEGER, Title: VARCHAR(20), Purpose: VARCHAR(50))

- ➔ URL → Status, UserCapacity, Title, Purpose
- ➔ Title → Purpose
- ➔ Purpose → UserCapacity
- ➔ Title → UserCapacity

Exercise(ExerciseName: VARCHAR(50), ExerciseNumber: INTEGER, Purpose: VARCHAR(50), Score: REAL, TimeLimit: TIME)

- ➔ ExerciseName, Exercise Number → Purpose, Score, TimeLimit
- ➔ ExerciseName → Purpose
- ➔ Purpose → TimeLimit
- ➔ ExerciseName → TimeLimit

Achievement(AchievementID: INTEGER, AchievementName: VARCHAR(50) UNIQUE, AchievementDescription: VARCHAR(50), RewardName: VARCHAR(50), RewardID: INTEGER)

- ➔ (AchievementName) is a candidate key and will have a UNIQUE constraint
- ➔ AchievementID → AchievementName, AchievementDescription, RewardName, RewardID
- ➔ AchievementName → AchievementID, AchievementDescription, RewardName, RewardID
- ➔ RewardID → RewardName

➔ RewardName → RewardID

Resource(ResourceID: INTEGER, ResourceName: VARCHAR(50), Purpose: VARCHAR(50))
- ➔ ResourceID → ResourceName, Purpose

Book(**ResourceID**: INTEGER, Author: VARCHAR(50))
- ➔ (ResourceID) is a foreign key referencing Resource
- ➔ ResourceID → Author

App(**ResourceID**: INTEGER, Developer: VARCHAR(50))
- ➔ (ResourceID) is a foreign key referencing Resource
- ➔ ResourceID → Developer

Website(**ResourceID**: INTEGER, URL: VARCHAR(100))
- ➔ (ResourceID) is a foreign key referencing Resource
- ➔ ResourceID → URL

2) We now list the table definitions and FDs for the entity sets that were "merged" with a relationship set:

Learner_Consults(UserID: INTEGER, Username: VARCHAR(50) UNIQUE, Age: INTEGER, Password: VARCHAR(50), **ExpertEmail**: VARCHAR(20))
- ➔ (Username) is a candidate key and will have a UNIQUE constraint
- ➔ (ExpertEmail) is a foreign key referencing Expert
- ➔ UserId → Username, Age, Password, ExpertEmail
- ➔ Username → UserId, Age, Password ExpertEmail

Bot_Monitors(BotName**:** VARCHAR(50), Version: VARCHAR(50), Purpose: VARCHAR(50), Creator: VARCHAR(50), CreationDate: DATE, **URL**: VARCHAR(100) NOT NULL)
- ➔ (URL) is a foreign key referencing Forum
- ➔ Cannot enforce the participation constraint on Forum in the relational model (need to use assertions), but the participation constraint on Bot can be captured!
- ➔ BotName, Version → Purpose, Creator, CreationDate, URL
- ➔ BotName → Purpose
- ➔ BotName, Creator → CreationDate

Question_Has(**ExerciseName**: VARCHAR(50), **ExerciseNumber**: INTEGER, QuestionName: VARCHAR(20), Type: VARCHAR(50))
- ➔ (ExerciseName, ExerciseNumber) is a foreign key referencing Exercise
- ➔ ExerciseName, ExerciseNumber, QuestionName → Type

3) We finally list the table definitions and FDs for the remaining relationship sets:

Participates(**UserID**: INTEGER, **URL**: VARCHAR(100))
- ➔ (UserID) is a foreign key referencing Learner_Consults
- ➔ (URL) is a foreign key referencing Forum
- ➔ Only has trivial FDs

Earns(**UserID**: INTEGER, **AchievementID**: INTEGER, ReceivalDate: DATE)

- ➔ (UserID) is a foreign key referencing Learner_Consults
- ➔ (AchievementID) is a foreign key referencing Achievement
- ➔ UserID, AchievementID → ReceivalDate

Interacts(**LearnerID1**: INTEGER, **LearnerID2**: INTEGER)

- ➔ (LearnerID1) is a foreign key referencing Learner_Consults(UserID)
- ➔ (LearnerID2) is a foreign key referencing Learner_Consults(UserID)
- ➔ Only has trivial FDs

Specializes(**ExpertEmail**: VARCHAR(50), **LanguageName**: VARCHAR(50), **Dialect**: VARCHAR(20), YearsExperience: INTEGER)

- ➔ (ExpertEmail) is a foreign key referencing Expert
- ➔ (LanguageName, Dialect) is a foreign key referencing Language
- ➔ Cannot enforce the participation constraint on Expert in the relational model (need to use assertions)
- ➔ ExpertEmail, LanguageName, Dialect → YearsExperience

Learns(**UserID**: VARCHAR(50), **LanguageName**: VARCHAR(50), **Dialect**: VARCHAR(50), StartDate: DATE)

- ➔ (UserID) is a foreign key referencing Learner_Consults
- ➔ (LanguageName, Dialect) is a foreign key referencing Language
- ➔ Cannot enforce the participation constraint on Learner in the relational model (need to use assertions)
- ➔ UserID, LanguageName, Dialect → StartDate

Uses(**UserID**: VARCHAR(50), **LanguageName**: VARCHAR(50), **Dialect**: VARCHAR(50), **ResourceID**: VARCHAR(50))

- ➔ (UserID, LanguageName, Dialect) is a foreign key referencing Learns
- ➔ (ResourceID) is a foreign key referencing Resource
- ➔ Only has trivial FDs

Completes(**UserID**: VARCHAR(50), **LanguageName**: VARCHAR(50), **Dialect**: VARCHAR(50), **ExerciseName**: VARCHAR(50), **ExerciseNumber**: INTEGER, CompletionDate: DATE)

- ➔ (UserID, LanguageName, Dialect) is a foreign key referencing Learns
- ➔ (ExerciseName, ExerciseNumber) is a foreign key referencing Exercise
- ➔ UserID, LanguageName, Dialect, ExerciseName, ExerciseNumber → CompletionDate

Normalization and SQL CREATE Statements

Note on our Normalization: We chose to normalize all of our relations to be in BCNF. For each relation, we've highlighted the final normalized relations in red during our derivation process.

Note on our SQL Statements: We have included our CREATE statements right after deriving our normalized relations. Since Oracle (our choice of DB) does not support ON UPDATE CASCADE, we have omitted the ON UPDATE clause from our CREATE statements (although we acknowledge that in an ideal situation, we would have included ON UPDATE CASCADE).

1) We first check the relations specified in step 1 of the previous section:

Language(LanguageName, Dialect, NumChars)

- ➔ Language is not in BCNF because LanguageName from LanguageName → NumChars is not a superkey
  - ◆ {LanguageName}$^+$ = {LanguageName, NumChars}
- ➔ Decompose to get Language1(LanguageName, NumChars) and Language2(**LanguageName**, Dialect)
- ➔ Final normalized relations:
  - ◆ Language1(LanguageName, NumChars)
  - ◆ Language2(**LanguageName**, Dialect)
    - (LanguageName) is a foreign key referencing Language1

```
CREATE TABLE Language1(
     LanguageName VARCHAR(50) PRIMARY KEY,
     NumChars     INTEGER
);

CREATE TABLE Language2(
     LanguageName VARCHAR(50),
     Dialect      VARCHAR(50),
     PRIMARY KEY (LanguageName, Dialect),
     FOREIGN KEY (LanguageName) REFERENCES Language1
          ON DELETE CASCADE
);
```

Expert(ExpertEmail, ExpertName, Capacity, Mode, City)

- ➔ Expert is not in BCNF because neither Mode from Mode → Capacity nor (ExpertName, City) from ExpertName, City → Mode, Capacity are superkeys
  - ◆ {Mode}$^+$ = {Mode, Capacity}
  - ◆ {ExpertName, City}$^+$ = {ExpertName, City, Mode, Capacity}
- ➔ Decompose to get Expert1(Mode, Capacity) and Expert2(ExpertEmail, ExpertName, **Mode**, City)
- ➔ Expert2 is not in BCNF because (ExpertName, City) from ExpertName, City → Mode is not a superkey
  - ◆ {ExpertName, City}$^+$ = {ExpertName, City, Mode}

- ➔ Decompose to get Expert3(ExpertName, City, **Mode**) and Expert4(ExpertEmail, **ExpertName**, **City**)
- ➔ Final normalized relations:
  - ◆ Expert1(Mode, Capacity)
  - ◆ Expert3(ExpertName, City, **Mode**)
    - (Mode) is a foreign key referencing Expert1
  - ◆ Expert4(ExpertEmail, **ExpertName**, **City**)
    - (ExpertName, City) is a foreign key referencing Expert3

```
CREATE TABLE Expert1(
     Mode       VARCHAR(50) PRIMARY KEY,
     Capacity   INTEGER
);

CREATE TABLE Expert3(
     ExpertName VARCHAR(50),
     City       VARCHAR(50),
     Mode       VARCHAR(50),
     PRIMARY KEY (ExpertName, City),
     FOREIGN KEY (Mode) REFERENCES Expert1
          ON DELETE CASCADE
);

CREATE TABLE Expert4(
     ExpertEmail VARCHAR(50) PRIMARY KEY,
     ExpertName  VARCHAR(50),
     City        VARCHAR(50),
     FOREIGN KEY (ExpertName, City) REFERENCES Expert3
          ON DELETE CASCADE
);
```

Forum(URL, Status, UserCapacity, Title, Purpose)

- ➔ Forum is not in BCNF because neither Title from Title → Purpose, UserCapacity nor Purpose from Purpose → UserCapacity are superkeys
  - ◆ $\{Title\}^+ = \{Title, Purpose, UserCapacity\}$
  - ◆ $\{Purpose\}^+ = \{Purpose, UserCapacity\}$
- ➔ Decompose to get Forum1(Title, Purpose) and Forum2(URL, Status, UserCapacity, **Title**)
- ➔ Forum2 is not in BCNF because Title from Title → UserCapacity is not a superkey
  - ◆ $\{Title\}^+ = \{Title, UserCapacity\}$
- ➔ Decompose to get Forum3(**Title**, UserCapacity) and Forum4(URL, Status, **Title**)
- ➔ Final normalized relations:
  - ◆ Forum1(Title, Purpose)
  - ◆ Forum3(**Title**, UserCapacity)
    - (Title) is a foreign key referencing Forum1
  - ◆ Forum4(URL, Status, **Title)**

- (Title) is a foreign key referencing Forum1

```
CREATE TABLE Forum1(
      Title   VARCHAR(50) PRIMARY KEY,
      Purpose VARCHAR(100)
);

CREATE TABLE Forum3(
      Title        VARCHAR(50) PRIMARY KEY,
      UserCapacity INTEGER,
      FOREIGN KEY (Title) REFERENCES Forum1
            ON DELETE CASCADE
);

CREATE TABLE Forum4(
      URL    VARCHAR(100) PRIMARY KEY,
      Status VARCHAR(50),
      Title  VARCHAR(50),
      FOREIGN KEY (Title) REFERENCES Forum1
            ON DELETE CASCADE
);
```

Exercise(ExerciseName, ExerciseNumber, Purpose, Score, TimeLimit)

- ➔ Exercise is not in BCNF because neither ExerciseName from ExerciseName → Purpose, TimeLimit nor Purpose from Purpose → TimeLimit are superkeys
  - ◆ $\{ExerciseName\}^+ = \{ExerciseName, Purpose, TimeLimit\}$
  - ◆ $\{Purpose\}^+ = \{Purpose, TimeLimit\}$
- ➔ Decompose to get Exercise1(ExerciseName, Purpose) and Exercise2(**ExerciseName**, ExerciseNumber**,** Score, TimeLimit)
- ➔ Exercise2 is not in BCNF because ExerciseName from ExerciseName → TimeLimit is not a superkey
  - ◆ $\{ExerciseName\}^+ = \{ExerciseName, TimeLimit\}$
- ➔ Decompose to get Exercise3(**ExerciseName**, TimeLimit) and Exercise4(**ExerciseName**, ExerciseNumber, Score)
- ➔ Final normalized relations:
  - ◆ Exercise1(ExerciseName, Purpose)
  - ◆ Exercise3(**ExerciseName**, TimeLimit)
    - (ExerciseName) is a foreign key referencing Exercise1
  - ◆ Exercise4(**ExerciseName**, ExerciseNumber, Score)
    - (ExerciseName) is a foreign key referencing Exercise1

```
CREATE TABLE Exercise1(
      ExerciseName VARCHAR(50) PRIMARY KEY,
      Purpose      VARCHAR(100)
);
```

```
CREATE TABLE Exercise3(
      ExerciseName VARCHAR(50) PRIMARY KEY,
      TimeLimit    TIME,
      FOREIGN KEY (ExerciseName) REFERENCES Exercise1
            ON DELETE CASCADE
);

CREATE TABLE Exercise4(
      ExerciseName   VARCHAR(50),
      ExerciseNumber INTEGER,
      Score          REAL,
      PRIMARY KEY (ExerciseName, ExerciseNumber),
      FOREIGN KEY (ExerciseName) REFERENCES Exercise1
            ON DELETE CASCADE
);
```

Achievement(AchievementID, AchievementName UNIQUE, AchievementDescription, RewardName, RewardID)

- ➔ Achievement is not in BCNF because neither RewardID from RewardID → RewardName nor RewardName from RewardName → RewardID are superkeys
  - ◆ $\{RewardID\}^+ = \{RewardID, RewardName\}$
  - ◆ $\{RewardName\}^+ = \{RewardName, RewardID\}$
- ➔ Decompose to get Achievement1(RewardID, RewardName UNIQUE) and Achievement2(AchievementID, AchievementName, AchievementDescription, **RewardID**, ReceivalDate)
- ➔ Final normalized relations:
  - ◆ Achievement1(RewardID, RewardName UNIQUE)
    - (RewardName) is a candidate key and has a UNIQUE constraint
  - ◆ Achievement2(AchievementID, AchievementName, AchievementDescription, **Reward ID**, Receival Date)
    - (RewardID) is a foreign key referencing Achievement1

```
CREATE TABLE Achievement1(
      RewardID   INTEGER PRIMARY KEY,
      RewardName VARCHAR(50) UNIQUE
);

CREATE TABLE Achievement2(
      AchievementID              INTEGER PRIMARY KEY,
      AchievementName            VARCHAR(50),
      AchievementDescription VARCHAR(50),
      RewardID               INTEGER,
      ReceivalDate           DATE,
      FOREIGN KEY (RewardID) REFERENCES Achievement1
            ON DELETE CASCADE
);
```

Resource(ResourceID, ResourceName, Purpose)

➔ Already in BCNF

```
CREATE TABLE Resource(
      ResourceID   INTEGER PRIMARY KEY,
      ResourceName VARCHAR(50),
      Purpose      VARCHAR(100)
);
```

Book(**ResourceID**, Author)

➔ Already in BCNF

```
CREATE TABLE Book(
      ResourceID INTEGER PRIMARY KEY,
      Author     VARCHAR(50),
      FOREIGN KEY (ResourceID) REFERENCES Resource
            ON DELETE CASCADE
);
```

App(**ResourceID**, Developer)

➔ Already in BCNF

```
CREATE TABLE App(
      ResourceID INTEGER PRIMARY KEY,
      Developer  VARCHAR(50),
      FOREIGN KEY (ResourceID) REFERENCES Resource
            ON DELETE CASCADE
);
```

Website(**ResourceID**, URL)

➔ Already in BCNF

```
CREATE TABLE Website(
      ResourceID INTEGER PRIMARY KEY,
      URL        VARCHAR(100),
      FOREIGN KEY (ResourceID) REFERENCES Resource
            ON DELETE CASCADE
);
```

2) We now check the relations specified in step 2 of the previous section:

Learner_Consults(UserID, Username UNIQUE, Age, Password, **ExpertEmail**)

➔ Already in BCNF

➔ (Username) is a candidate key and will have a UNIQUE constraint

➔ (ExpertEmail) is a foreign key referencing Expert4

```
CREATE TABLE Learner_Consults(
      UserID      INTEGER PRIMARY KEY,
```

```
      Username    VARCHAR(50) UNIQUE,
      Age         INTEGER,
      Password    VARCHAR(50),
      ExpertEmail VARCHAR(50),
      FOREIGN KEY (ExpertEmail) REFERENCES Expert4
            ON DELETE CASCADE
);
```

Bot_Monitors(BotName, Version, Purpose, Creator, CreationDate, **URL** NOT NULL)

- ➔ Bot_Monitors is not in BCNF because neither BotName from BotName → Purpose nor (BotName, Creator) from BotName, Creator → CreationDate are superkeys
  - ◆ $\{BotName\}^+ = \{BotName, Purpose\}$
  - ◆ $\{BotName, Creator\}^+ = \{BotName, Creator, CreationDate\}$
- ➔ Decompose to get Bot_Monitors1(BotName, Purpose) and Bot_Monitors2(**BotName**, Version, Creator, CreationDate, **URL** NOT NULL)
- ➔ Bot_Monitors2 is not in BCNF because (BotName, Creator) from BotName, Creator → CreationDate is not a superkey
  - ◆ $\{BotName, Creator\}^+ = \{BotName, Creator, CreationDate\}$
- ➔ Decompose to get Bot_Monitors3(**BotName**, Creator, CreationDate) and Bot_Monitors4(**BotName**, Version, **Creator**, **URL** NOT NULL)
- ➔ Final normalized relations:
  - ◆ Bot_Monitors1(BotName, Purpose)
  - ◆ Bot_Monitors3(**BotName**, Creator, CreationDate)
    - (BotName) is a foreign key referencing Bot_Monitors1
  - ◆ Bot_Monitors4(**BotName**, Version, **Creator**, **URL** NOT NULL)
    - (BotName, Creator) is a foreign key referencing Bot_Monitors3
    - (URL) is a foreign key referencing Forum4

```
CREATE TABLE Bot_Monitors1(
      BotName VARCHAR(50) PRIMARY KEY,
      Purpose VARCHAR(100)
);

CREATE TABLE Bot_Monitors3(
      BotName      VARCHAR(50),
      Creator      VARCHAR(50),
      CreationDate DATE,
      PRIMARY KEY (BotName, Creator),
      FOREIGN KEY (BotName) REFERENCES Bot_Monitors1
            ON DELETE CASCADE
);

CREATE TABLE Bot_Monitors4(
      BotName VARCHAR(50),
      Version VARCHAR(50),
      Creator VARCHAR(50),
```

```
      URL     VARCHAR(100) NOT NULL,
      PRIMARY KEY (BotName, Version),
      FOREIGN KEY (BotName, Creator) REFERENCES Bot_Monitors3
            ON DELETE CASCADE,
      FOREIGN KEY (URL) REFERENCES Forum4
            ON DELETE CASCADE
);
```

Question_Has(**ExerciseName**, **ExerciseNumber**, QuestionName, Type)

- ➔ Already in BCNF
- ➔ (ExerciseName, ExerciseNumber) is a foreign key referencing Exercise4

```
CREATE TABLE Question_Has(
      ExerciseName   VARCHAR(50),
      ExerciseNumber INTEGER,
      QuestionName   VARCHAR(50),
      Type           VARCHAR(50),
      PRIMARY KEY (ExerciseName, ExerciseNumber, QuestionName),
      FOREIGN KEY (ExerciseName, ExerciseNumber) REFERENCES Exercise4
            ON DELETE CASCADE
);
```

3) We finally check the relations specified in step 3 of the previous section:

Participates(**UserID**, **URL**)

- ➔ Already in BCNF
- ➔ (UserID) is a foreign key referencing Learner_Consults
- ➔ (URL) is a foreign key referencing Forum4

```
CREATE TABLE Participates(
      UserID INTEGER,
      URL    VARCHAR(100),
      PRIMARY KEY (UserID, URL),
      FOREIGN KEY (UserID) REFERENCES Learner_Consults
            ON DELETE CASCADE,
      FOREIGN KEY (URL) REFERENCES Forum4
            ON DELETE CASCADE
);
```

Earns(**UserID**: INTEGER, **AchievementID**: INTEGER, ReceivalDate: DATE)

- ➔ Already in BCNF
- ➔ (UserID) is a foreign key referencing Learner_Consults
- ➔ (AchievementID) is a foreign key referencing Achievement2

```
CREATE TABLE Earns(
      UserID        INTEGER,
      AchievementID INTEGER,
```

```
      ReceivalDate  DATE,
      PRIMARY KEY (UserID, AchievementID),
      FOREIGN KEY (UserID) REFERENCES Learner_Consults
            ON DELETE CASCADE,
      FOREIGN KEY (AchievementID) REFERENCES Achievement2
            ON DELETE CASCADE
);
```

Interacts(**LearnerID1**, **LearnerID2**)

- ➔ Already in BCNF
- ➔ (LearnerID1) is foreign key referencing Learner_Consults(UserID)
- ➔ (LearnerID2) is foreign key referencing Learner_Consults(UserID)

```
CREATE TABLE Interacts(
      LearnerID1 INTEGER,
      LearnerID2 INTEGER,
      PRIMARY KEY (LearnerID1, LearnerID2),
      FOREIGN KEY (LearnerID1) REFERENCES Learner_Consults(UserID)
            ON DELETE NO ACTION,
      FOREIGN KEY (LearnerID2) REFERENCES Learner_Consults(UserID)
            ON DELETE NO ACTION
);
```

Specializes(**ExpertEmail**, **LanguageName**, **Dialect**, YearsExperience)

- ➔ Already in BCNF
- ➔ (ExpertEmail) is a foreign key referencing Expert4
- ➔ (LanguageName, Dialect) is a foreign key referencing Language2

```
CREATE TABLE Specializes(
      ExpertEmail     VARCHAR(50),
      LanguageName    VARCHAR(50),
      Dialect         VARCHAR(50),
      YearsExperience INTEGER,
      PRIMARY KEY(ExpertEmail, LanguageName, Dialect),
      FOREIGN KEY (ExpertEmail) REFERENCES Expert4
            ON DELETE CASCADE,
      FOREIGN KEY (LanguageName, Dialect) REFERENCES Language2
            ON DELETE CASCADE
);
```

Learns(**UserID**, **LanguageName**, **Dialect**, StartDate)

- ➔ Already in BCNF
- ➔ (UserID) is a foreign key referencing Learner_Consults
- ➔ (LanguageName, Dialect) is a foreign key referencing Language2

```
CREATE TABLE Learns(
```

```
      UserID       INTEGER,
      LanguageName VARCHAR(50),
      Dialect      VARCHAR(50),
      StartDate    DATE,
      PRIMARY KEY (UserID, LanguageName, Dialect),
      FOREIGN KEY (UserID) REFERENCES Learner_Consults
            ON DELETE CASCADE,
      FOREIGN KEY (LanguageName, Dialect) REFERENCES Language2
            ON DELETE CASCADE
);
```

Uses(**UserID**, **LanguageName**, **Dialect**, **ResourceID**)

- ➔ Already in BCNF
- ➔ (UserID) is a foreign key referencing Learner_Consults
- ➔ (LanguageName, Dialect) is a foreign key referencing Language2
- ➔ (ResourceID) is a foreign key referencing Resource

```
CREATE TABLE Uses(
      UserID       INTEGER,
      LanguageName VARCHAR(50),
      Dialect      VARCHAR(50),
      ResourceID   INTEGER,
      PRIMARY KEY (UserID, LanguageName, Dialect),
      FOREIGN KEY (UserID) REFERENCES Learner_Consults
            ON DELETE CASCADE,
      FOREIGN KEY (LanguageName, Dialect) REFERENCES Language2
            ON DELETE CASCADE,
      FOREIGN KEY (ResourceID) REFERENCES Resource
            ON DELETE CASCADE
);
```

Completes(**UserID**, **LanguageName**, **Dialect**, **ExerciseName**, **ExerciseNumber**, CompletionDate)

- ➔ Already in BCNF
- ➔ (UserID) is a foreign key referencing Learner_Consults
- ➔ (LanguageName, Dialect) is a foreign key referencing Language2
- ➔ (ExerciseName, ExerciseNumber) is a foreign key referencing Exercise4

```
CREATE TABLE Completes(
      UserID         INTEGER,
      LanguageName   VARCHAR(50),
      Dialect        VARCHAR(50),
      ExerciseName   VARCHAR(50),
      ExerciseNumber INTEGER,
      CompletionDate DATE,
```

```
      PRIMARY KEY (UserID, LanguageName, Dialect, ExerciseName,
ExerciseNumber),
      FOREIGN KEY (UserID) REFERENCES Learner_Consults
            ON DELETE CASCADE,
      FOREIGN KEY (LanguageName, Dialect) REFERENCES Language2
            ON DELETE CASCADE,
      FOREIGN KEY (ExerciseName, ExerciseNumber) REFERENCES Exercise4
            ON DELETE CASCADE
);
```

SQL INSERT Statements

-- Language1

```
INSERT INTO Language1(LanguageName, NumChars)
VALUES
      ('English', 26),
      ('Spanish', 27),
      ('French', 26),
      ('German', 30),
      ('Chinese', 26),
      ('Korean', 24);
```

-- Language2

```
INSERT INTO Language2(LanguageName, Dialect)
VALUES
      ('English', 'American English'),
      ('English', 'British English'),
      ('Spanish', 'Latin American Spanish'),
      ('Spanish', 'European Spanish'),
      ('French', 'Standard French'),
      ('French', 'Canadian French'),
      ('French', 'Belgian French'),
      ('French', 'African French'),
      ('German', 'Standard German'),
      ('German', 'Swiss German'),
      ('Chinese', 'Mandarin'),
      ('Chinese', 'Sichuanese'),
      ('Chinese', 'Cantonese'),
      ('Korean', 'Gyeonggi dialect');
```

-- Expert1

```
INSERT INTO Expert1(Mode, Capacity)
VALUES
     ('Online In-Province', 20),
     ('In-Person In-Province', 10),
     ('Hybrid In-Province', 15),
     ('Online Out-of-Province', 30),
     ('Hybrid Out-of-Province', 25);
```

-- Expert3

```
INSERT INTO Expert3(ExpertName, City, Mode)
VALUES
     ('Romina M', 'Vancouver', 'Hybrid In-Province'),
     ('Annie W', 'Vancouver', 'In-Person In-Province'),
     ('Sohbat S', 'Vancouver', 'Online In-Province'),
     ('James W', 'Toronto', 'Online Out-of-Province'),
     ('Kayla K', 'Toronto', 'Hybrid Out-of-Province'),
     ('Kate M', 'Edmonton', 'Hybrid Out-of-Province');
```

-- Expert4

```
INSERT INTO Expert4(ExpertEmail, ExpertName, City)
VALUES
     ('romina.m@mail.com', 'Romina M', 'Vancouver'),
     ('annie.w@mail.com', 'Annie W', 'Vancouver'),
     ('sohbat.s@mail.com', 'Sohbat S', 'Vancouver'),
     ('james.w@mail.com', 'James W', 'Toronto'),
     ('kayla.k@mail.com', 'Kayla K', 'Toronto'),
     ('kate.m@mail.com', 'Kate M', 'Edmonton');
```

-- Forum1

```
INSERT INTO Forum1(Title, Purpose)
VALUES
     ('General Discussion', 'Discuss various topics'),
     ('Beginner Mandarin Chinese', 'Get help with Beginner
Mandarin'),
     ('French In Canada', 'Moving to Quebec, you will need this'),
     ('Language Learning Book Club', 'Discuss and share thoughts on
books in any language'),
     ('German Speaking Community', 'Get help to learn different
languages with translation from German Speakers');
```

-- Forum3

```
INSERT INTO Forum3(Title, UserCapacity)
VALUES
     ('General Discussion', 1000),
     ('Beginner Mandarin Chinese', 200),
     ('French In Canada', 400),
     ('Language Learning Book Club', 1000),
     ('German Speaking Community', 350);
```

-- Forum4

```
INSERT INTO Forum4(URL, Status, Title)
VALUES
     ('http://example.com/general-discussion', 'Active', 'General
Discussion'),
     ('http://example.com/beginner-mandarin-chinese', 'Active',
'Beginner Mandarin Chinese'),
     ('http://example.com/french-in-canada', 'Inactive', 'French In
Canada'),
     ('http://example.com/language-learning-book-club', 'Active',
'Language Learning Book Club'),
     ('http://example.com/german-speaking-community', 'Active',
('German Speaking Community');
```

-- Exercise1

```
INSERT INTO Exercise1(ExerciseName, Purpose)
VALUES
     ('Active to Passive Voice English', 'Test identification and
conversion in Passive and Active voice in English'),
     ('Parisian Culture', 'Test your knowledge in Parisian
Culture'),
     ('Chinese Vocabulary Quiz', 'Test Chinese vocabulary
knowledge'),
     ('Spanish Grammar Quiz', 'Practice Spanish grammar rules'),
     ('German Pronunciation Workout', 'Practice pronunciations for
common German words');
```

-- Exercise3

```
INSERT INTO Exercise3(ExerciseName, TimeLimit)
VALUES
```

```
     ('Active to Passive Voice English', '00:35:00'),
     ('Parisian Culture', '00:20:00'),
     ('Chinese Vocabulary Quiz', '01:05:00'),
     ('Spanish Grammar Quiz', '00:25:00'),
     ('German Pronunciation Workout', NULL);
```

-- Exercise4

```
INSERT INTO Exercise4(ExerciseName, ExerciseNumber, Score) VALUES
     ('Active to Passive Voice English', 61, 0.905),
     ('Parisian Culture', 62, 0.85),
     ('Chinese Vocabulary Quiz', 63, NULL),
     ('Spanish Grammar Quiz', 64, 0.95),
     ('German Pronunciation Workout', 65, 0.80);
```

-- Achievement1

```
INSERT INTO Achievement1(RewardID, RewardName)
VALUES
     (51, 'Gold Medal'),
     (52, 'Silver Medal'),
     (53, 'Bronze Medal'),
     (54, 'Certificate of Achievement'),
     (55, 'Badge of Honor');
```

-- Achievement2

```
INSERT INTO Achievement2 (AchievementID, AchievementName,
AchievementDescription, RewardID, ReceivalDate)
VALUES
     (41, 'Completionist', 'Complete all exercises', 53,
'2024-02-20'),
     (42, `Master Speaker', 'Complete all pronunciation challenges',
51, '2024-02-20'),
     (43, 'Language Pro', 'Achieve fluency in a language', 55,
'2024-02-20'),
     (44, 'Beginner Spanish Guru', 'Complete Beginner Spanish ', 54,
'2024-02-20'),
     (45, 'Bookworm', 'Read 10 French books', 52, '2024-02-20');
```

-- Resource

```
INSERT INTO Resource(ResourceID, ResourceName, Purpose)
VALUES
     (11, 'English in Use', 'A Self-Study Reference and Practice
```

```
Book for Intermediate Learners of English with Answers'),
     (12, 'Spanish Maestro', 'A Self-Study Reference and Practice
Book for Intermediate Learners of Spanish with Answers'),
     (13, 'Elegance in French', 'A Self-Study Reference and Practice
Book for Intermediate Learners of French with Answers'),
     (14, 'Chinese Stories for Language Learners', 'A treasury of
proverbs and folktales in Chinese and English'),
     (15, 'German for Dummies', 'The fun and easy way to learn the
fascinating language of German with integrated audio clips!'),
     (21, 'Example Bookstore', 'Find Your Favorite and Necessary
books'),
     (22, 'Example Lingo Fitness App', 'Can your tongue pronounce
correctly?'),
     (23, 'Example Language Learning', 'Get help to learn new
language on your own time'),
     (24, 'Example Pronunciation Platform', 'Learn the correct
pronunciations in over 100 languages'),
     (25, 'Example Forum', 'Get access to all online forums'),
     (31, 'Multi-Language Dictionary', 'Get words definitions and
translations in over 20 languages'),
     (32, 'How Good is Your Spanish?', 'Comprehensive Spanish
Language Quizzes'),
     (33, 'Teach A Tongue', 'Learn New languages'),
     (34, 'Contact a LingoXpert', 'Get in touch with language
experts from all over the world'),
     (35, 'Are you speaking Correctly?', 'Learn all the nuances of
the language of your choice');
```

-- Book

```
INSERT INTO Book(ResourceID, Author)
VALUES
     (11, 'John Doe'),
     (12, 'Gabriel Martinez'),
     (13, 'Celestine Dior'),
     (14, 'Vivian Ling'),
     (15, 'Paulina Christensen');
```

-- Website

```
INSERT INTO Website(ResourceID, URL)
VALUES
     (21, 'http://examplebookstore.com'),
```

```
      (22, 'http://examplelingofitnessapp.com'),
      (23, 'http://examplelanguagelearning.com'),
      (24, 'http://examplepronunicationplatform.com'),
      (25, 'http://exampleforum.com');
```

-- App

```
INSERT INTO App(ResourceID, Developer)
VALUES
      (31, 'DD Dev Ops'),
      (32, '3Cent Games'),
      (33, 'Foreign Lingo'),
      (34, 'AllInfo Techs'),
      (35, 'NoSure Info Corps');
```

-- Learner_Consults

```
INSERT INTO Learner_Consults(UserID, UserName, Age, Password,
ExpertEmail)
VALUES
      (1, 'User 1', 21, 'pass1', 'romina.m@mail.com'),
      (2, 'User 2', 22, 'pass2', 'annie.w@mail.com'),
      (3, 'User 3', 25, 'pass3', 'sohbat.s@mail.com'),
      (4, 'User 4', 19, 'pass4', 'romina.m@mail.com'),
      (5, 'User 5', 18, 'pass5', 'annie.w@mail.com'),
      (6, 'User 6', 20, 'pass6', 'sohbat.s@mail.com');
```

-- Bot_Monitors1

```
INSERT INTO Bot_Monitors1(BotName, Purpose)
VALUES
      ('SpamGuardBot', 'Monitor spam messages'),
      ('AutoResponderBot', 'Automatically respond to messages'),
      ('AnalyticsBot', 'Analyze user interactions'),
      ('SecurityBot', 'Monitor security threats'),
      ('FeedbackBot', 'Collect user feedback');
```

-- Bot_Monitors3

```
INSERT INTO Bot_Monitors3(BotName, Creator, CreationDate)
VALUES
      ('SpamGuardBot', 'TechzCorp', '2023-01-15'),
      ('AutoResponderBot', 'TechzCorp', '2023-03-10'),
      ('AnalyticsBot', 'DataInsightsAnswers', '2022-11-20'),
```

```
     ('SecurityBot', 'CyberSafeSols', '2023-05-05'),
     ('FeedbackBot', 'UserInsights4All', '2023-02-28');
```

-- Bot_Monitors4

```
INSERT INTO Bot_Monitors4(BotName, Version, Creator, URL)
VALUES
     ('SpamGuardBot', 'v1.0', 'TechzCorp',
'http://example.com/general-discussion'),
     ('AutoResponderBot', 'v2.0', 'TechzCorp',
'http://example.com/beginner-mandarin-chinese'),
     ('AnalyticsBot', 'v1.5.2', 'DataInsightsAnswers',
'http://example.com/french-in-canada'),
     ('SecurityBot', 'v1.2', 'CyberSafeSols',
'http://example.com/language-learning-book-club'),
     ('FeedbackBot', 'v1.8', 'UserInsights4All',
'http://example.com/german-speaking-community');
```

-- Question_Has

```
INSERT INTO Question_Has(ExerciseName, ExerciseNumber, QuestionName,
Type)
VALUES
     ('Active to Passive Voice English', 61, 'Translate verb
tenses', 'Intermediate'),
     ('Parisian Culture', 62, 'Name Parisian musicians',
'Advanced'),
     ('Parisian Culture', 62, 'Name Parisian actors', 'Advanced'),
     ('Chinese Vocabulary Quiz', 63, 'Daily life vocabulary',
'Beginner'),
     ('Chinese Vocabulary Quiz', 63, 'Work office vocabulary',
'Beginner');
```

-- Participates

```
INSERT INTO Participates(UserID, URL)
VALUES
     (1, 'http://example.com/general-discussion'),
     (1, 'http://example.com/beginner-mandarin-chinese'),
     (2, 'http://example.com/french-in-canada'),
     (3, 'http://example.com/french-in-canada'),
     (4, 'http://example.com/language-learning-book-club');
```

-- Earns

```
INSERT INTO Earns(UserID, AchievementID)
VALUES
      (1, 41),
      (2, 42),
      (2, 43),
      (4, 44),
      (5, 44),
      (5, 45);
```

-- Interacts

```
INSERT INTO Interacts(LearnerID1, LearnerID2)
VALUES
      (1, 2),
      (2, 3),
      (3, 4),
      (3, 5),
      (4, 5);
```

-- Specializes

```
INSERT INTO Specializes(ExpertEmail, LanguageName, Dialect,
YearsExperience)
VALUES
      ('romina.m@mail.com', 'Korean', 'Gyeonggi dialect', 10),
      ('annie.w@mail.com', 'Chinese', 'Mandarin', 10),
      ('sohbat.s@mail.com', 'English', 'American English', 20),
      ('james.w@mail.com', 'French', 'Canadian French', 10),
      ('kayla.k@mail.com', 'Spanish', 'Latin American Spanish', 15),
      ('kate.m@mail.com', 'German', 'Swiss German', 5);
```

-- Learns

```
INSERT INTO Learns(UserID, LanguageName, Dialect, StartDate)
VALUES
      (1, 'Korean', 'Gyeonggi dialect', '2023-09-01'),
      (2, 'Chinese', 'Mandarin', '2023-09-12'),
      (3, 'English', 'American English', '2023-06-13'),
      (4, 'French', 'Canadian French', '2023-12-30'),
      (5, 'Spanish', 'Latin American Spanish', '2023-10-13'),
      (6, 'German', 'Swiss German', '2023-12-04');
```

-- Uses

```
INSERT INTO Uses(UserID, LanguageName, Dialect, ResourceID)
VALUES
      (1, 'Korean', 'Gyeonggi dialect', 35),
```

```
     (2, 'Chinese', 'Mandarin', 14),
     (3, 'English', 'American English', 24),
     (4, 'French', 'Canadian French', 31),
     (5, 'Spanish', 'Latin American Spanish', 31),
     (6, 'German', 'Swiss German', 15);
```

-- Completes

```
INSERT INTO Completes(UserID, LanguageName, Dialect, ExerciseName,
ExerciseNumber, CompletionDate)
VALUES
     (1, 'German', 'Swiss German', 'German Pronunciation Workout',
65, '2023-03-09'),
    (2, 'Chinese', 'Mandarin', 'Chinese Vocabulary Quiz', 63,
'2023-10-10'),
     (3, 'English', 'American English', 'Active to Passive Voice
English', 61, '2023-07-01'),
     (4, 'French', 'Canadian French', 'Parisian Culture', 62,
'2024-01-10'),
     (5, 'Spanish', 'Latin American Spanish', 'Spanish Grammar
Quiz', 64, '2023-12-01');
```