University of British Columbia, Vancouver
Department of Computer Science

CPSC 304 Project Cover Page

Milestone #: 2

Date: Oct 14, 2024

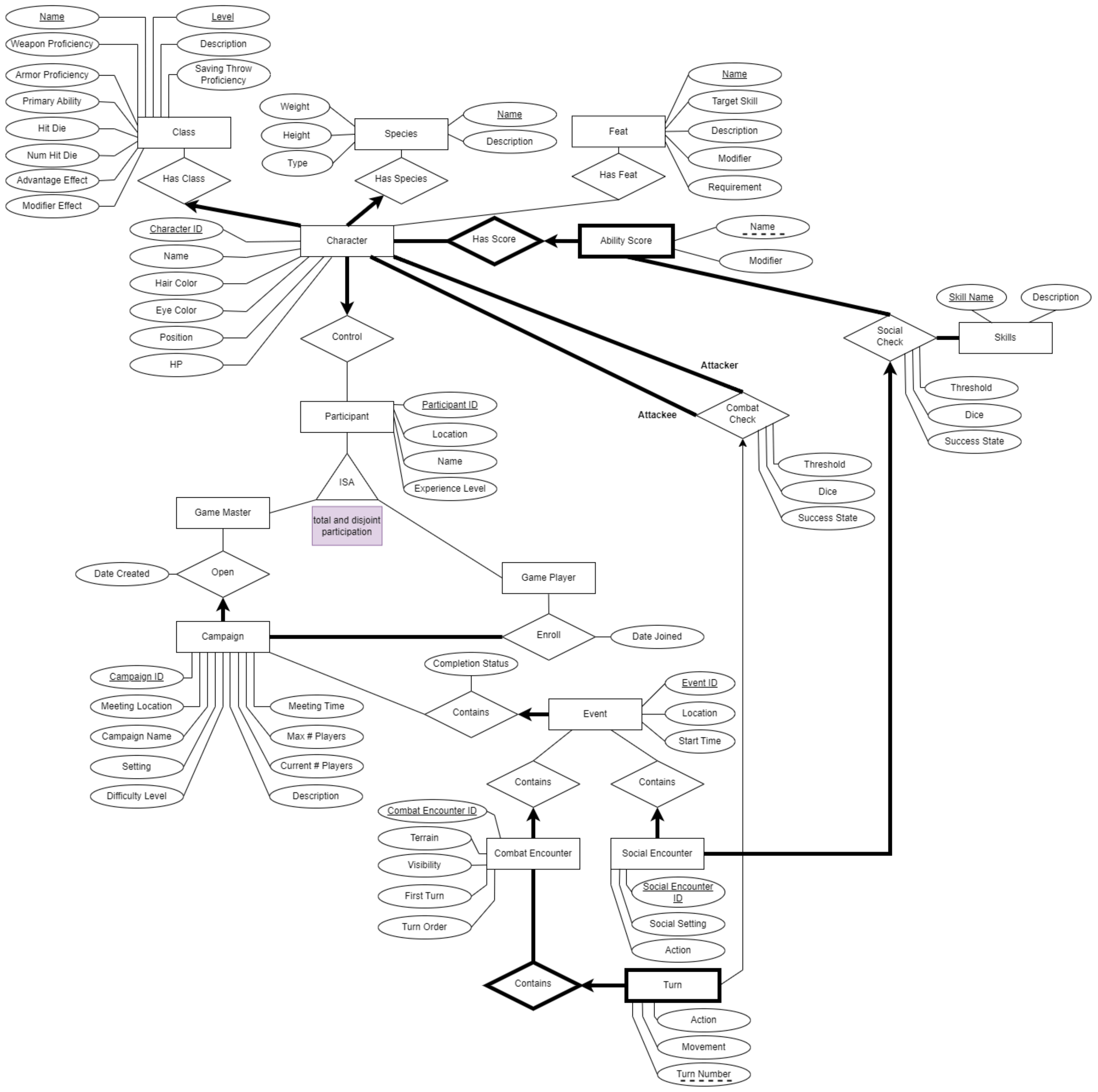
Group Number: 54

Name	Student Number	CS Alias (Userid)	Preferred E-mail Address
Julia Sangster	29688934	y1o9i	juliasangster@hotmail.com
Annie Chung	38565115	r9l7z	aachung@student.ubc.ca
Momin Kashif	13718895	q1d5z	mominkas@student.ubc.ca

Summary

Our application provides a database solution for managing key components of DND campaigns by modeling campaigns, players, characters and events. Players will be able to look up character stats as well as track combat and social encounters. Our application will also enable Game Masters to recall previous events with precision, improving session flow and player immersion.

ER diagram



Response to TA feedback

Ability scores with the same name belonging to two different characters will have different modifiers as modifiers are set per character. Accordingly, we have left Ability Score as a weak entity set that is dependent on Character since each score is tied to a specific character and only exists in relation to that character.

As suggested, we have modified our ER diagram to simplify the Control relationship between Character and Participant. As a result, we have replaced the two Control relationships between the Character subclasses and Participant. This allows Game Players to control multiple characters across campaigns without having to make a new account each time. We can continue to check if a player is a player character controlled by a Game Player or a non-player character controlled by a Game Master.

Other changes

We decided to remove the Characteristics entity and its IsA relationship with the Class, Species and Feat entities as we noticed that Class has dependency on level while Species and Feat do not. You will see later this requires us to normalize Class but not Species or Feat.

We also removed many attributes from Character because they are fully determined by either Class or Species. We also removed the ability to multi-class for simplicity and to conform to single-species restriction already decided.

Schema

Note to TA: Primary keys are PK, candidate keys are CK, and foreign keys are FK.
By definition, all PKs are CKs and all PK attributes cannot be null, so these are not included in the schemas.
We do not have any CKs that are not PKs.

```
Class(
  name: varchar,
  level: integer,
  description: varchar,
  weapon_proficiency: varchar           NOT NULL,
  armor_proficiency: varchar            NOT NULL,
  saving_throw_proficiency: varchar     NOT NULL,
  primary_ability: varchar              NOT NULL,
  hit_die: varchar                      NOT NULL,
  advantage_effect: integer             NOT NULL,
  modifier_effect: integer              NOT NULL,
  num_hit_die: integer                  NOT NULL,
  PK(name, level)
)
```

```
Species(
   name: varchar                     PK,
   description: varchar,
   weight: varchar                   NOT NULL,
   height: varchar                   NOT NULL,
   type: varchar,                    NOT NULL,
)

Feat(
   name: varchar,
   target_skill: varchar,            NOT NULL
   description: varchar,
   modifier: integer                 NOT NULL,
   requirement: varchar,
   PK(name)
)

Character(
   character_id: integer             PK,
   name: varchar                     NOT NULL,
   hair_color: varchar,
   eye_color: varchar,
   level: integer                    NOT NULL,
   position: varchar,
   hp: integer                       NOT NULL,
   class_name:  varchar              NOT NULL,
   species_name: varchar             NOT NULL,
   participant_id: integer           NOT NULL,
   FK(class_name, level)             REFERENCES Class(name, level),
   FK(species_name)                  REFERENCES Species(name),
   FK(participant_id)                REFERENCES Participant(participant_id)
)

Has_Feat(
   character_id: integer,
   feat_name: varchar,
   PK(character_id, feat_name),
   FK(character_id)                  REFERENCES Character(character_id),
   FK(feat_name)                     REFERENCES Feat(feat_name)
)
```

```
Ablity_Score(
   character_id: integer,
   name: varchar,
   modifier: integer              NOT NULL,
   PK(character_id, name),
   FK(character_id)               REFERENCES Character(character_id)
)

Participant(
   participant_id: integer        PK,
   location: varchar,
   name: varchar                  NOT NULL,
   experience level: integer
)

Game_Player(
   game_player_id: integer        PK,
   FK(game_player_id)             REFERENCES Participant(participant_id)
)

Game_Master(
   game_master_id: integer        PK,
   FK(game_master_id)             REFERENCES Participant(participant_id)
)
```

Note to TA: Game_Player and Game_Master are needed for their unique relationships to Campaign and Enrol. The attributes game_master_id and game_player_id are restricted to be valid participant_id values.

```
Campaign(
   campaign_id: integer           PK,
   campaign_name: varchar         NOT NULL,
   meeting_location: varchar,
   meeting_time: time,
   setting: varchar,
   difficulty_level: varchar,
   max_num_players: integer       NOT NULL,
   current_num_players: integer NOT NULL,
   description: varchar,
   date_created: date             NOT NULL,
   game_master_id: integer        NOT NULL,
   FK(game_master_id)             REFERENCES Game_Master(game_master_id)
)
```

```
Enrol(
   game_player_id: integer,
   campaign_id: integer,
   date_joined: date                 NOT NULL,
   PK(game_player_id, campaign_id),
   FK(game_player_id)                REFERENCES Game_Player(game_player_id),
   FK(campaign_id)                   REFERENCES Campaign(campaign_id)
)

Event(
   event_id: integer                 PK,
   location: varchar                 NOT NULL,
   start_time: time                  NOT NULL,
   completion_status: varchar        NOT NULL,
   campaign_id: integer              NOT NULL,
   FK(campaign_id)                   REFERENCES Campaign(campaign_id)
)

Combat_Encounter(
   combat_encounter_id: integer PK,
   terrain: varchar,
   visibility: varchar,
   first_turn: varchar               NOT NULL,
   turn_order: varchar               NOT NULL,
   event_id: integer                 NOT NULL,
   FK(event_id)                      REFERENCES Event(event_id)
)

Social_Encounter(
   social_encounter_id: integer      PK,
   social_setting: varchar,
   action: varchar                   NOT NULL,
   event_id: integer                 NOT NULL,
   FK(event_id)                      REFERENCES Event(event_id)
)

Skill(
   name: varchar                     PK,
   description: varchar
)
```

Social_Check(
```
   character_id: integer                  NOT NULL,
   ability_score_name: name               NOT NULL,
   skill_name: varchar                    NOT NULL,
   social_encounter_id: integer           PK,
   threshold: integer,
   dice: varchar,
   success_state: Boolean                 NOT NULL,
   FK(character_id, ability_score_name)  REFERENCES Ability_Score(character_id, name),
   FK(skill_name)                         REFERENCES Skill(name)
)
```

Turn(
```
   combat_encounter_id: integer,
   turn_number: integer,
   movement: integer,
   action: varchar                        NOT NULL,
   PK(combat_encounter_id, turn_number),
   FK(combat_encounter_id)                REFERENCES Combat_Encounter(combat_encounter_id)
)
```

Combat_Check(
```
   attacker_character_id: integer         NOT NULL,
   attackee_character_id: integer         NOT NULL,
   combat_id: integer,
   turn_number: integer,
   threshold: varchar,
   dice: varchar,
   success_state: Boolean                 NOT NULL,
   PK(combat_id, turn_number),
   FK(combat_id, turn_number)             REFERENCES Turn(combat_encounter_id, turn_number),
   FK(attacker_character_id)              REFERENCES Character(character_id),
   FK(attackee_character_id)              REFERENCES Character(character_id),
)
```

Functional dependencies

Note to TA: We have shown our non-PK functional dependencies in **red.**

CLASS
(Name, Level) ⟶ (Description, Advantage Effect, Modifier Effect, Weapon Proficiency, Armor Proficiency, Saving Throw Proficiency, Primary Ability, Hit Die, Num Hit Die)
(Name) ⟶ (Description, Weapon Proficiency, Armor Proficiency, Saving Throw Proficiency, Primary Ability, Hit Die)
(Level) ⟶ (Modifier Effect, Advantage Effect, Num Hit Die)

SPECIES
(Name) ⟶ (Description, Weight, Height, Type)

FEAT
(Name) ⟶ (Description, Advantage Effect, Modifier Effect, Requirement)

CHARACTER
(Character ID) ⟶(Name, Hair Color, Eye Color, Level, Position, HP, Class Name, Species Name, Participant ID)

HAS FEAT
(Character ID, Feat Name, Feat Level) ⟶ (Level Earned)

ABILITY SCORE
(Character ID, Name) ⟶ (Modifier)

PARTICIPANT
(Participant ID) ⟶ (Location, Name, Experience Level)

GAME PLAYER
(Participant ID) ⟶ (Location, Name, Experience Level)

GAME MASTER
(Participant ID) ⟶ (Location, Name, Experience Level)

CAMPAIGN
(Campaign ID) ⟶ (Campaign Name, Meeting Location, Meeting Time, Setting, Difficulty Level, Max Num Players, Current Num Players, Description, Date Created, Game Master ID)

ENROL
(Participant ID, Campaign ID) ⟶ (Date Joined)

EVENT
(Event ID) ⟶ (Location, Start Time, Completion Status, Campaign ID)

COMBAT ENCOUNTER
(Combat Encounter ID) ⟶ (Terrain, Visibility, First Turn, Turn Order, Event ID)

SOCIAL ENCOUNTER
(Social Encounter ID) ⟶ (Social Setting, Action, Event ID)

SKILL
(Name) ⟶ (Description)

SOCIAL CHECKS
(Social Encounter ID) ⟶ (Character ID, Ability Score Name, Skill Name, Threshold, Dice, Success State)

TURN
(Combat Encounter ID, Turn Number) ⟶ (Action, Movement)

COMBAT CHECKS
(Combat Encounter ID, Turn Number) ⟶ (Attacker Character ID, Attackee Character ID, Threshold, Dice, Success State)

Normalization

We normalized each of our tables to be in BCNF. All tables except Class were already in normal form as their PKs fully determine all attributes of each relationship.

This is the original Class table:

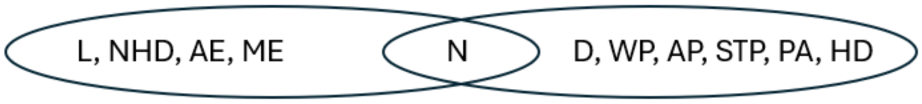
```
Class(
   name: varchar,
   level: integer,
   description: varchar,
   weapon_proficiency: varchar          NOT NULL,
   armor_proficiency: varchar           NOT NULL,
   saving_throw_proficiency: varchar    NOT NULL,
   primary_ability: varchar             NOT NULL,
   hit_die: varchar                     NOT NULL,
   advantage_effect: integer            NOT NULL,
   modifier_effect: integer             NOT NULL,
   num_hit_die: varchar                 NOT NULL,
   PK(name, level)
)
```

We will shorten this to: C(N, L, D, WP, AP, STP, PA, HD, NHD, AE, ME).

Class has the following functional dependencies (FDs):

$N, L \rightarrow D, WP, AP, STP, PA, HD, NHD, AE, ME$
$N \rightarrow D, WP, AP, STP, PA, HD$
$L \rightarrow NHD, AE, ME$

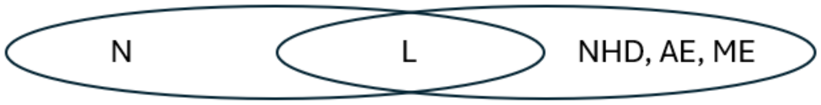
N → D, WP, AP, STP, PA, HD violates BCNF in Class because N is not a superkey of the relation (superkey is N, L), so we decompose:



R1(N, D, WP, AP, STP, PA, HD)
R2(N, L, NHD, AE, ME)

R1 is in BCNF because N is a superkey of the relation and functionally determines all attributes. R2 is not in BCNF as L → NHD, AE, ME violates BCNF requirements (L is not a superkey of R2; superkey is still N ,L).

Decompose R2:



R3(N, L)
R4(L, NHD, AE, ME)

R3 and R4 are in BCNF because N, L is the superkey of R3, and L is the superkey of R4.

The final three Class relations are as follows:

R1(N, D, WP, AP, STP, PA, HD)	Class_Description
R3(N, L)	Class - maintains uniqueness of combinations of N,L
R4(L, NHD, AE, ME)	Class_Level_Features

Class_Description(
```
  name: varchar                        PK,
  description: varchar,
  weapon_proficiency: varchar          NOT NULL,
  armor_proficiency: varchar           NOT NULL,
  saving_throw_proficiency: varchar    NOT NULL,
  primary_ability: varchar             NOT NULL.
  hit_die: varchar                     NOT NULL
```
)

```
Class_Level_Features(
   level: integer                      PK,
   num_hit_die: integer                NOT NULL,
   advantage_effect: integer,          NOT NULL,
   modifier_effect: integer,           NOT NULL
)
```

```
Class(
   name: varchar,
   level: integer,
   PK(level, name),
   FK(level)                           REFERENCES Class_Level_Features(level),
   FK(name)                            REFERENCES Class_Description(name)
)
```

The other tables already in BCNF are listed below:

```
Species(
   name: varchar                       PK,
   description: varchar,
   weight: varchar                     NOT NULL,
   height: varchar                     NOT NULL,
   type: varchar,                      NOT NULL,
)
```

```
Feat(
   name: varchar                       PK,
   target_skill: varchar               NOT NULL,
   description: varchar,
   modifier: integer                   NOT NULL,
   requirement: varchar,
)
```

```
Character(
   character_id: integer               PK,
   name: varchar                       NOT NULL,
   hair_color: varchar,
   eye_color: varchar,
   level: integer                      NOT NULL,
   position: varchar,
   hp: integer                         NOT NULL,
   class_name:  varchar                NOT NULL,
   species_name: varchar               NOT NULL,
   participant_id: integer             NOT NULL,
   FK(class_name, level)                 REFERENCES Class(name, level),
   FK(species_name)                      REFERENCES Species(name),
   FK(participant_id)                    REFERENCES Participant(participant_id)
)

Has_Feat(
   character_id: integer,
   feat_name: varchar,
   PK(character_id, feat_name),
   FK(character_id)                      REFERENCES Character(character_id),
   FK(feat_name)                         REFERENCES Feat(name)
)

Ablity_Score(
   character_id: integer,
   name: varchar,
   modifier: integer                     NOT NULL,
   PK(character_id, name),
   FK(character_id)                      REFERENCES Character(character_id)
)

Participant(
   participant_id: integer               PK,
   location: varchar,
   name: varchar                         NOT NULL,
   experience level: integer
)

Game_Player(
   game_player_id: integer               PK,
   FK(game_player_id)                    REFERENCES Participant(participant_id)
)

Game_Master(
   game_master_id: integer               PK,
   FK(game_master_id)                    REFERENCES Participant(participant_id)
)

Campaign(
   campaign_id: integer                  PK,
   campaign_name: varchar                NOT NULL,
   meeting_location: varchar,
   meeting_time: time,
   setting: varchar,
   difficulty_level: varchar,
   max_num_players: integer              NOT NULL,
   current_num_players: integer          NOT NULL,
   description: varchar,
   date_created: date                    NOT NULL,
   game_master_id: integer               NOT NULL,
   FK(game_master_id)                    REFERENCES Game_Master(game_master_id)
)

Enrol(
   participant_id: integer,
   campaign_id: integer,
   date_joined: date                     NOT NULL,
   PK(participant_id, campaign_id),
   FK(participant_id)                    REFERENCES Game_Player(participant_id),
   FK(campaign_id)                       REFERENCES Campaign(campaign_id)
)

Event(
   event_id: integer                     PK,
   location: varchar                     NOT NULL,
   start_time: time                      NOT NULL,
   completion_status: varchar            NOT NULL,
   campaign_id: integer                  NOT NULL,
   FK(campaign_id)                       REFERENCES Campaign(campaign_id)
)

Combat_Encounter(
   combat_encounter_id: integer          PK,
   terrain: varchar,
   visibility: varchar,
   first_turn: varchar                   NOT NULL,
   turn_order: varchar                   NOT NULL,
   event_id: integer                     NOT NULL,
   FK(event_id)                          REFERENCES Event(event_id)
)

Social_Encounter(
   social_encounter_id: integer          PK,
   social_setting: varchar,
   action: varchar                       NOT NULL,
   event_id: integer                     NOT NULL,
   FK(event_id)                          REFERENCES Event(event_id)
)
```

```
Skill(
   name: varchar                              PK,
   description: varchar
)

Social_Check(
   social_encounter_id: integer               PK,
   character_id: integer                      NOT NULL,
   ability_score_name: name                   NOT NULL,
   skill_name: varchar                        NOT NULL,
   threshold: integer,
   dice: varchar,
   success_state: Boolean                     NOT NULL,
   FK(character_id, ability_score_name)  REFERENCES Ability_Score(character_id, name),
   FK(skill_name)                             REFERENCES Skill(name)
)

Turn(
   combat_encounter_id: integer,
   turn_number: integer,
   movement: integer,
   action: varchar                            NOT NULL,
   PK(combat_encounter_id, turn_number),
   FK(combat_encounter_id)                    REFERENCES Combat_Encounter(combat_encounter_id)
)

Combat_Check(
   attacker_character_id: integer             NOT NULL,
   attackee_character_id: integer             NOT NULL,
   combat_id: integer,
   turn_number: integer,
   threshold: varchar,
   dice: varchar,
   success_state: Boolean                     NOT NULL,
   PK(combat_id, turn_number),
   FK(combat_id, turn_number)                 REFERENCES Turn(combat_encounter_id, turn_number),
   FK(attacker_character_id)                  REFERENCES Character(character_id),
   FK(attackee_character_id)                  REFERENCES Character(character_id),
)
```

SQL DDL statements

The SQL DDL statements required to create all the tables from item #6. The statements should use the appropriate foreign keys, primary keys, UNIQUE constraints, etc. Unless you know that you will always have exactly x characters for a given character, it is better to use the VARCHAR data type as opposed to a CHAR(Y). For example, UBC courses always use four characters to represent which department offers a course. In that case, you will want to use CHAR(4) for the department attribute in your SQL DDL statement. If you are trying to represent the name of a UBC course, you will want to use VARCHAR as the number of characters in a course name can vary greatly.

Note to TA: Oracle DBMS does not support ON UPDATE CASCADE.
Further, it is recommended to use VARCHAR2 instead of VARCHAR for Oracle databases.
Picked 100 characters for shorter names or descriptions, and 3000 for full descriptions.
Explicitly stating NULL for clarity, but it is unnecessary.

```
CREATE TABLE Class_Description(
    name                        VARCHAR2(100)     PRIMARY KEY,
    description                 VARCHAR2(1000)    NULL,
    primary_ability             VARCHAR2(100)     NOT NULL,
    weapon_proficiency          VARCHAR2(100)     NOT NULL,
    armor_proficiency           VARCHAR2(100)     NOT NULL,
    hit_die                     VARCHAR2(100)     NOT NULL,
    saving_throw_proficiency    VARCHAR2(100)     NOT NULL
);

CREATE TABLE Class_Level_Features(
      level                     INTEGER     PRIMARY KEY,
      num_hit_die               INTEGER     NOT NULL,
      advantage_effect          INTEGER     NOT NULL,
      modifier_effect           INTEGER     NOT NULL
);

CREATE TABLE Class(
      name                      VARCHAR2(100),
      level                     INTEGER,
      PRIMARY KEY (name, level),
      FOREIGN KEY (level)
            REFERENCES Class_Level_Features(level)
            ON DELETE CASCADE,
      FOREIGN KEY (name)
            REFERENCES Class_Description(name)
            ON DELETE CASCADE,
);
```

```
CREATE TABLE Species(
      name                    VARCHAR2(100)     PRIMARY KEY,
      description             VARCHAR2(1000)    NULL,
      weight                  VARCHAR2(100)     NOT NULL,
      height                  VARCHAR2(100)     NOT NULL,
      type                    VARCHAR2(100)     NOT NULL,
);

CREATE TABLE Feat(
      name                    VARCHAR2(100)     PRIMARY KEY,
      target_skill            VARCHAR2(100)     NOT NULL
      description             VARCHAR2(1000)    NULL,
      modifier                INTEGER           NOT NULL,
      requirement             VARCHAR2(1000)    NULL
);

CREATE TABLE Character(
      character_id            INTEGER           PRIMARY KEY,
      name                    VARCHAR2(100)     NOT NULL,
      hair_color              VARCHAR2(100)     NULL,
      eye_color               VARCHAR2(100)     NULL,
      level                   INTEGER           NOT NULL,
      position                VARCHAR2(100)     NULL,
      hp                      INTEGER           NOT NULL,
      class_name              VARCHAR2(100)     NOT NULL,
      species_name            VARCHAR2(100)     NOT NULL,
      participant_id          INTEGER           NOT NULL,
      FOREIGN KEY (class_name, level)
            REFERENCES Class(name, level)
            ON DELETE NO ACTION,
      FOREIGN KEY (species_name)
            REFERENCES Species(name)
            ON DELETE NO ACTION,
      FOREIGN KEY (participant_id)
            REFERENCES Participant(participant_id)
            ON DELETE CASCADE,
);
```

```
CREATE TABLE Has_Feat(
      character_id            INTEGER,
      feat_name               VARCHAR2(100),
      PRIMARY KEY (character_id, feat_name),
      FOREIGN KEY (character_id)
            REFERENCES Character(character_id)
            ON DELETE CASCADE,
      FOREIGN KEY (feat_name)
            REFERENCES Feat(name)
            ON DELETE CASCADE,
);


CREATE TABLE Ability_Score(
      character_id            INTEGER,
      name                    VARCHAR2(100),
      modifier                INTEGER           NOT NULL,
      PRIMARY KEY (character_id, name),
      FOREIGN KEY (character_id)
            REFERENCES Character(character_id)
            ON DELETE CASCADE,
);

CREATE TABLE Participant(
      participant_id          INTEGER           PRIMARY KEY,
      location                VARCHAR2(1000)    NULL,
      name                    VARCHAR2(100)     NOT NULL,
      experience_level        INTEGER           NULL
);

CREATE TABLE Game_Player(
      game_player_id          INTEGER     PRIMARY KEY,
      FOREIGN KEY (game_player_id)
            REFERENCES Participant(participant_id)
            ON DELETE CASCADE,
);

CREATE TABLE Game_Master(
      game_master_id          INTEGER     PRIMARY KEY,
      FOREIGN KEY (game_master_id)
            REFERENCES Participant(participant_id)
            ON DELETE CASCADE,
);
```

```
CREATE TABLE Campaign(
      campaign_id            INTEGER           PRIMARY KEY,
      campaign_name          VARCHAR2(100)     NOT NULL,
      meeting_location       VARCHAR2(100)     NULL,
      meeting_time           TIME              NULL,
      setting                VARCHAR2(1000)    NULL,
      difficulty_level       VARCHAR2(100)     NULL,
      max_num_players        INTEGER           NOT NULL,
      current_num_players    INTEGER           NOT NULL,
      description            VARCHAR2(1000)    NULL,
      date_created           DATE              NOT NULL,
      game_master_id         INTEGER           NOT NULL,
      FOREIGN KEY (game_master_id)
            REFERENCES Game_Master(game_master_id)
            ON DELETE NO ACTION,
);

CREATE TABLE Enrol(
      game_player_id         INTEGER,
      campaign_id            INTEGER,
      date_joined            DATE              NOT NULL,
      PRIMARY KEY (participant_id, campaign_id)
      FOREIGN KEY (participant_id)
            REFERENCES Game_Player(game_player_id)
            ON DELETE CASCADE,
      FOREIGN KEY (campaign_id)
            REFERENCES Campaign(campaign_id)
            ON DELETE CASCADE,
);

CREATE TABLE Event(
      event_id               INTEGER           PRIMARY KEY,
      location               VARCHAR2(100)     NOT NULL,
      start_time             TIME              NOT NULL,
      completion_status      VARCHAR2(100)     NOT NULL,
      campaign_id            INTEGER           NOT NULL,
      FOREIGN KEY (campaign_id)
            REFERENCES Campaign(campaign_id)
            ON DELETE CASCADE,
);
```

```
CREATE TABLE Combat_Encounter(
      combat_encounter_id     INTEGER           PRIMARY KEY,
      terrain                 VARCHAR2(100)     NULL,
      visibility              VARCHAR2(100)     NULL,
      first_turn              VARCHAR2(100)     NOT NULL,
      turn_order              VARCHAR2(1000)    NOT NULL,
      event_id                INTEGER           NOT NULL,
      FOREIGN KEY (event_id)
            REFERENCES Event(event_id)
            ON DELETE CASCADE,
);

CREATE TABLE Social_Encounter(
      social_encounter_id     INTEGER           PRIMARY KEY,
      social_setting          VARCHAR2(100)     NULL,
      action                  VARCHAR2(100)     NOT NULL,
      event_id                INTEGER           NOT NULL,
      FOREIGN KEY (event_id)
            REFERENCES Event(event_id)
            ON DELETE CASCADE,
);

CREATE TABLE Skill(
      name              VARCHAR2(100)     PRIMARY KEY,
      description       VARCHAR2(1000)    NULL
);

CREATE TABLE Social_Check(
      social_encounter_id     INTEGER           PRIMARY KEY,
      character_id            INTEGER           NOT NULL,
      ability_score_name      VARCHAR2(100)     NOT NULL,
      skill_name              VARCHAR2(100)     NOT NULL,
      threshold               INTEGER           NULL,
      dice                    VARCHAR2(100)     NULL,
      success_state           BOOLEAN           NULL,
      FOREIGN KEY (character_id, ability_score_name)
            REFERENCES Ability_Score(character_id, name)
            ON DELETE CASCADE,
);
```

```
CREATE TABLE Turn(
      combat_encounter_id     INTEGER,
      turn_number             INTEGER,
      movement                INTEGER           NULL,
      action                  VARCHAR2(100)     NOT NULL,
      PRIMARY KEY (combat_enounter_id, turn_number)
      FOREIGN KEY (combat_enounter_id)
            REFERENCES Combat_Encounter(combat_enounter_id)
            ON DELETE CASCADE,
);

CREATE TABLE Combat_Check(
      attacker_character_id   INTEGER           NOT NULL,
      attackee_character_id   INTEGER           NOT NULL,
      combat_id               INTEGER,
      turn_number             INTEGER,
      threshold               INTEGER           NULL,
      dice                    VARCHAR2(100)     NULL,
      success_state           BOOLEAN           NULL,
      PRIMARY KEY (combat_id, turn_number)
      FOREIGN KEY (combat_id, turn_number)
            REFERENCES Turn(combat_encounter_id, turn_number)
            ON DELETE CASCADE,
      FOREIGN KEY (attacker_character_id)
            REFERENCES Character(character_id)
            ON DELETE CASCADE,
      FOREIGN KEY (attackee_character_id)
            REFERENCES Character(character_id)
            ON DELETE CASCADE,
);
```

INSERT statements

```
INSERT INTO Class_Description (name, description, primary_ability,
weapon_proficiency, armor_proficiency, hit_die, saving_throw_proficiency)
VALUES
      (‘Warrior’, ‘A strong fighter skilled in melee combat.’,
      ‘Strength’, ‘Simple Weapons’, ‘Light Armor’, ‘d10’,‘Strength,
      Constitution’),
      (‘Mage’, ‘A master of the arcane arts capable of casting powerful
      spells.’, ‘Intelligence’, ‘Spells’, ‘No Armor’, ‘d6’, ‘Intelligence,
      Wisdom’),
      (‘Rogue’, NULL, ‘Dexterity’, ‘Simple Weapons’, ‘Light Armor’, ‘d8’,
      ‘Dexterity, Intelligence’),
      (‘Cleric’, ‘A divine spellcaster who channels the power of their
      deity.’, ‘Wisdom’, ‘Simple Weapons’, ‘Medium Armor’, ‘d8’, ‘Wisdom,
      Charisma’),
      (‘Ranger’, ‘A skilled tracker and hunter, adept in both ranged and melee
      combat.’, ‘Dexterity’, ‘Martial Weapons’, ‘Medium Armor’, ‘d10’,
      ‘Strength, Dexterity’);

INSERT INTO Class_Level_Features (level, num_hit_die, advantage_effect,
modifier_effect) VALUES
      (1, 1, 2, 2),
      (2, 2, 3, 2),
      (3, 3, 3, 3),
      (4, 4, 4, 3),
      (5, 5, 4, 4);

INSERT INTO Class(name, level) VALUES
      (‘Cleric’, 1),
      (‘Cleric’, 2),
      (‘Cleric’, 3),
      (‘Cleric’, 4),
      (‘Cleric’, 5),
      (‘Wizard’, 1);
```

```
INSERT INTO **Species** (name, description, weight, height, type) VALUES
      (‘Human’, NULL, ‘140-250 lbs’, ‘Average’, ‘Humanoid’),
      ('Elf’, ‘Graceful beings with a natural affinity for magic and nature,
      known for their long lifespans.’, ‘100-145 lbs’, ‘Tall’, ‘Fey’),
      (‘Dwarf’, ‘Stout and hardy, dwarves are known for their strength,
      resilience, and craftsmanship.’, ‘150-200 lbs’, ‘Short’, ’Humanoid’),
      (‘Dragonborn’, ‘Descendants of dragons, dragonborn are proud warriors
      with a strong sense of honor.’, ‘250-350 lbs’, ‘Tall’, ‘Draconic’),
      (‘Halfling’, ‘Small and nimble, halflings are known for their luck and
      ability to stay out of danger.’, ‘40-45 lbs’, ‘Short’, ‘Humanoid’);


INSERT INTO **Feat** (name,  target_skill,  description, modifier, requirement)
VALUES
      (‘Mobile’, ‘DEX’, '’Your speed increases by 10 feet, and difficult
      terrain does not hinder you when you dash.’, 2, NULL),
      (‘Tough’, ‘STR’, ‘Your hit point maximum increases by an amount equal to
      twice your level.’, 2, NULL),
      (‘Sharpshooter’, ‘DEX’, NULL, 2, ‘Proficiency with a ranged weapon’),
      (‘Great Weapon Master’, ‘DEX’, ‘You can wield massive weapons with
      terrifying efficiency, dealing devastating blows.’, 3, ‘Proficiency with
      a heavy weapon’),
      (‘War Caster’, ‘INT’, ‘You have advantage on saving throws to maintain
      concentration on a spell.’, 1, ‘Ability to cast at least one spell’);


INSERT INTO **Character** (character_id, name, hair_color, eye_color, level,
position, hp, class_name, species_name, participant_id) VALUES
      (1, ‘Loathsome Dung Eater, 'Black’, ‘Brown’, 2, “Sally’s Tavern”, 18,
      ‘Paladin’, ‘Dwarf’, 1001),
      (2, ‘Margit the Fell’, NULL, ‘Blue’, 1, “Top of Mount Doom”, 12,
      ‘Fighter’, ‘Human’, 1002),
      (3, ‘General Radhan’, 'Silver', 'Green', 5, “Baldur’s Gate Potion Shop”,
      22, 'Wizard', 'Elf', 1003),
      (4, ‘Zarak Shadowsong’, ‘Brown’, ‘Hazel’, 3, “Ravenloft Salon”, 16,
      ‘Rogue’, ‘Halfling’, 1004),
      (5, ‘Tarnished’, NULL, NULL, 1, “Avernus Concert Hall”, 12, ‘Fighter’,
      'Dragonborn', 1005);
```

```
INSERT INTO Has_Feat (character_id,  feat_name) VALUES
      (1, ‘Sharpshooter’),
      (2, ‘War Caster’),
      (3, ‘Great Weapon Master’),
      (4, ‘Mobile’),
      (5, ‘Tough’);

INSERT INTO Ablity_Score (character_id, name, modifier) VALUES
      (1, ‘Strength’, 2),
      (2, ‘Intelligence’, 3),
      (3, ‘Constitution’, 1),
      (4, ‘Dexterity’, -4),
      (5, ‘Charisma’, 1);

INSERT INTO Participant (participant_id, location, name, experience_level)
VALUES
      (1001, ‘Waterdeep’, ‘Momin Kashif’, 5),
      (1002, ‘Neverwinter’, ‘Julia Sangster’, 6),
      (1003, ‘Baldur’s Gate’ ‘Annie Chung’, 7),
      (1004, ‘Feywild’ ‘Rachel Pottinger’, 4),
      (1005, ‘Icewind Dale’, ‘Jane Doe’ 2);

INSERT INTO Game_Player (game_player_id) VALUES
      (15),
      (23),
      (453),
      (78),
      (2718);

INSERT INTO Game_Master (game_master_id) VALUES
      (100),
      (200),
      (1),
      (45),
      (987);
```

```
INSERT INTO Campaign (campaign_id, campaign_name, meeting_location,
meeting_time, setting, difficulty_level, max_num_players, current_num_players,
description, date_created, gm_id) VALUES
      (1, ‘The Lost Mines of Phandelver’, ‘CS Building UBC, ‘18:00:00’, ‘Sword
      Coast’, ‘Easy’, 5, 3, ‘A classic adventure for new players.’,
      ‘2024-01-15’, 100),
      (2, ‘Curse of Strahd’, NULL, ‘19:30:00’, ‘Ravenloft’, ‘Hard’, 6, 4,
      NULL, ‘2024-02-20’, 200),
      (3, ‘Storm King’s Thunder’, ‘AMS Nest’, NULL, ‘Sword Coast’, ‘Medium’,
      7, 5, ‘A quest to unite the realms against a giant threat.’, NULL, 1),
      (4, ‘Tales from the Yawning Portal’, ‘Stanley Park’, ‘20:00:00’, NULL,
      ‘Medium’, 5, 5, ‘A collection of classic D&D adventures.’, ‘2024-04-10’,
      45),
      (5, ‘Descent into Avernus’, ‘Dundas Square’, ‘16:00:00’, ‘Avernus’,
      NULL, 6, 2, NULL, ‘2024-05-15’, 987);

INSERT INTO Enrol (game_player_id, campaign_id,  date_joined) VALUES
      (15,   1, ‘2024-01-15’),
      (23,   2, ‘2024-02-20’),
      (453,  1, ‘2024-04-10’),
      (78,   2, ‘2024-03-05’),
      (2718, 1, ’2024-05-25’);

INSERT INTO Event (event_id, location, start_time, completion_status,
campaign_id) VALUES
      (1, ‘Castle of Shadows’, ‘18:00:00’, ‘Completed’, 1),
      (2, ‘Forest of Whispers’, ‘14:30:00’, ‘In Progress’, 1),
      (3, ‘Mountain Fortress’, ‘20:00:00’, ‘Pending’, 2),
      (4, ‘City of Gold’, ‘19:00:00’, ‘Completed’, 2),
      (5, ‘Desert Ruins’, ‘15:00:00’, ‘In Progress’, 3);



INSERT INTO Combat_Encounter (combat_encounter_id, terrain, visibility,
first_turn, turn_order, event_id) VALUES
      (1, ‘Forest’, ‘Low’, ‘Player 1’, ‘Player 1, Player 2, Monster A’, 101),
      (2, ‘Cave’, ‘Dark’, ‘Monster B’, ‘Monster B, Player 3, Player 1’, 102),
      (3, ‘Open Field’, ‘Clear’, ‘Player 2’, ‘Player 2, Player 3, Monster C’,
      103),
      (4, ‘Dungeon’, ‘Dim’, ‘Player 1’, ‘Player 1, Monster D, Player 2’, 104),
      (5, ‘Ruins’, ‘Foggy’, ‘Player 3’, ‘Player 3, Monster E, Player 1’, 105);
```

```
INSERT INTO Social_Encounter (social_encounter_id, social_setting, action,
event_id) VALUES
      (1, ‘Tavern’, ‘Negotiate with the bartender’, 201),
      (2, ‘Marketplace’, ‘Haggle for prices’, 202),
      (3, ‘Noble’s Ball’, ‘Dance with a noble’, 203),
      (4, ‘Street Corner’, ‘Informally chat with locals’, 204),
      (5, ‘Library’, ‘Research ancient texts’, 205);

INSERT INTO Skill (name, description) VALUES
      (‘Stealth’, ‘The ability to move silently and avoid detection.’),
      (‘Persuasion’, ‘The ability to convince others to agree with your point
      of view.’),
      (‘Athletics’, ‘The ability to perform physical feats, such as climbing
      or jumping.’),
      (‘Arcana’, ‘The ability to understand the mystical arts and magicians’),
      (‘Insight’, ‘The ability to read people and sense their motivations.’);

INSERT INTO Social_Check (character_id, ability_score_name, skill_name,
social_encounter_id, threshold, dice, success_state) VALUES
      (1, ‘Strength’, ‘Persuasion’, 1, 15, ‘1d20’, TRUE),
      (2, ‘Charisma’, ‘Deception’, 2, 12, ‘1d20’, FALSE),
      (1, ‘Intelligence’, ‘Insight’, 3, 10, ‘1d20’, TRUE),
      (3, ‘Wisdom’, ‘Stealth’, 4, 14, ‘2d20’, TRUE),
      (2, ‘Dexterity’, ‘Athletics’, 5, 8, ‘1d20’, FALSE);

INSERT INTO Turn (combat_encounter_id, turn_number, movement, action) VALUES
      (1, 1, 30, ‘Move to the north and attack’),
      (1, 2, 20, ‘Cast a spell’),
      (2, 1, 25, ‘Take cover behind a tree’),
      (2, 2, 15, ‘Charge at the enemy’),
      (3, 1, 0,  ‘Use a ranged attack from a distance’);

INSERT INTO Combat_Check (attacker_character_id, attackee_character_id,
combat_id, turn_number, threshold, dice, success_state) VALUES
      (1, 2, 1, 1, ‘15’, ‘2d8’, TRUE),
      (2, 1, 1, 2, ‘12’, ‘1d20’, FALSE),
      (3, 1, 4, 1, ‘18’, ‘1d10’, TRUE),
      (2, 3, 2, 1, ‘10’, ‘3d20’, TRUE),
      (1, 4, 2, 2, ‘20’, ‘1d8’, FALSE);
```