Autor: Michał Piątek

Projekt Bazy danych dla gry komputerowej “The Binding of Isaac”

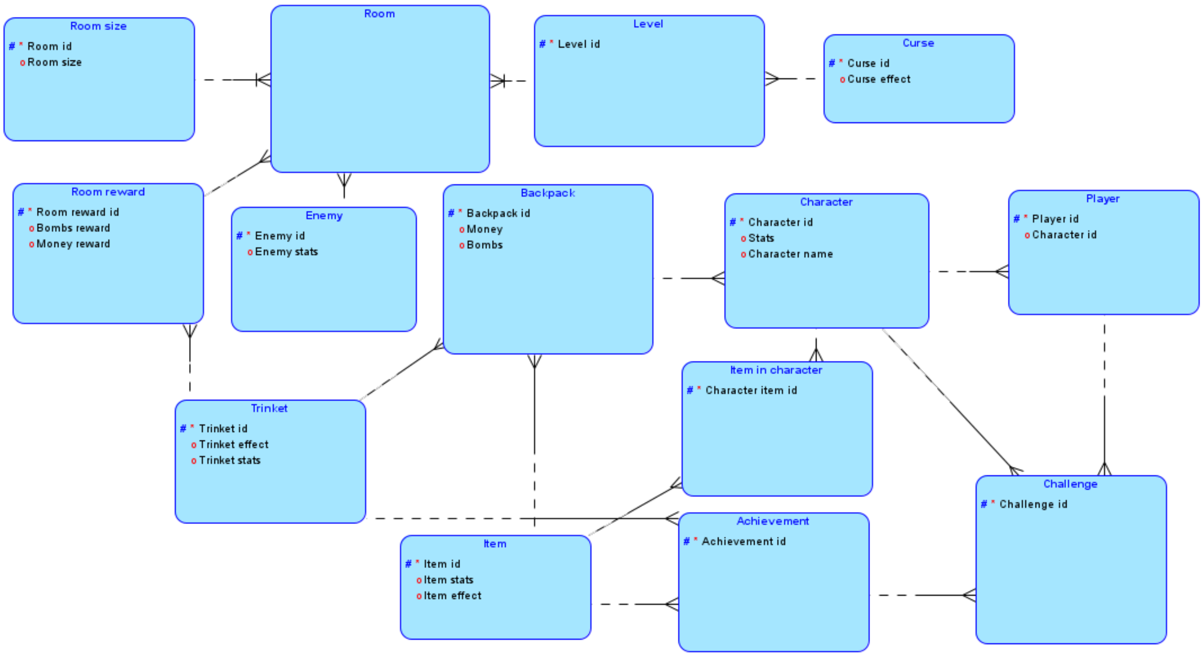
A:

Przygotowana przeze mnie baza danych ma służyć do przechowywania informacji o przedmiotach , postaciach, ozdobach, pokojach, przeciwnikach, osiągnięciach oraz wyzwaniach w grze The Binding of Isaac

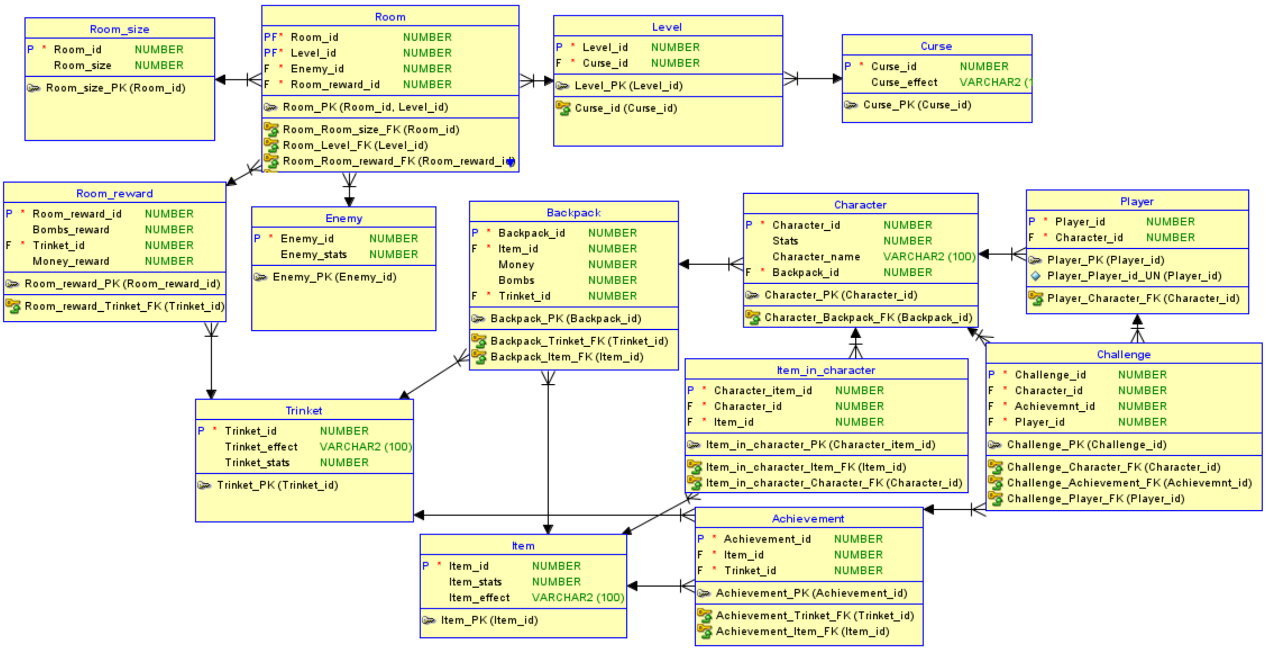
B:

1. Baza danych powinna przechowywać informacje o postaciach, przedmiotach, pokojach i innych aspektach gry.
2. Baza danych powinna być w stanie śledzić postęp gracza, przechowując informacje o osiągnięciach i wyzwaniach.
3. Baza danych powinna być w stanie śledzić przedmioty i ozdoby, które gracze mają w swoim plecaku.
4. Baza danych powinna być w stanie przechowywać informacje o klątwach, które mogą być stosowane do postaci.
5. Baza danych powinna być w stanie przechowywać informacje o wrogach w grze.
6. Baza danych powinna być w stanie przechowywać informacje o przedmiotach w grze, w tym ich statystyki i efekty.
7. Baza danych powinna być w stanie przechowywać informacje o przedmiotach, które posiada postać.
8. Baza danych powinna być w stanie przechowywać informacje o poziomach w grze, w tym o klątwach związanych z każdym poziomem.
9. Baza danych powinna być w stanie przechowywać informacje o graczach w grze, w tym o postaciach, z którymi są związani.
10. Baza danych powinna być w stanie przechowywać informacje o pokojach w grze, w tym o nagrodach, które można znaleźć w każdym pokoju.
11. Baza danych powinna być w stanie przechowywać informacje o ozdobach w grze, w tym o ich statystykach i efektach.
12. Baza danych powinna być w stanie przechowywać informacje o rozmiarze pokoju
13. Baza danych powinna być w stanie zapewnić integralność danych przy użyciu ograniczeń, takich jak klucze główne i unikalne klucze.
14. Baza danych powinna być w stanie obsługiwać wiele użytkowników jednocześnie dostęp do danych.

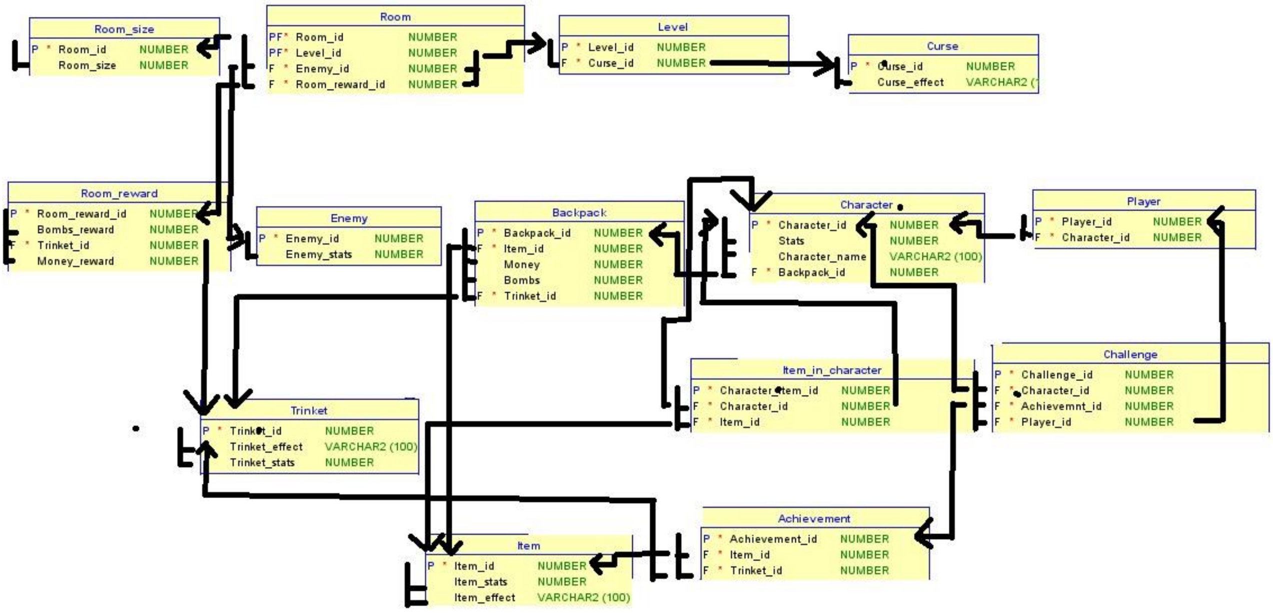
C1 Model logiczny:



C2 Model relacyjny:



C3 Diagram zależności funkcyjnych:



D1 – DDl skrypt:

```
CREATE TABLE achievement (
    achievement_id NUMBER NOT NULL,
    item_id        NUMBER NOT NULL,
    trinket_id     NUMBER NOT NULL
);


ALTER TABLE achievement ADD CONSTRAINT achievement_pk PRIMARY KEY (
achievement_id );


CREATE TABLE backpack (
    backpack_id NUMBER NOT NULL,
    item_id     NUMBER NOT NULL,
    money       NUMBER,
    bombs       NUMBER,
    trinket_id  NUMBER NOT NULL
);

ALTER TABLE backpack ADD CONSTRAINT backpack_pk PRIMARY KEY ( backpack_id );

CREATE TABLE challenge (
   challenge_id  NUMBER NOT NULL,
   character_id  NUMBER NOT NULL,
   achievemnt_id NUMBER NOT NULL,
   player_id     NUMBER NOT NULL
);

ALTER TABLE challenge ADD CONSTRAINT challenge_pk PRIMARY KEY ( challenge_id );

CREATE TABLE character (
   character_id   NUMBER NOT NULL,
   stats          NUMBER,
   character_name VARCHAR2(100),
   backpack_id    NUMBER NOT NULL
);

ALTER TABLE character ADD CONSTRAINT character_pk PRIMARY KEY ( character_id );

CREATE TABLE curse (
   curse_id     NUMBER NOT NULL,
   curse_effect VARCHAR2(100)
);

ALTER TABLE curse ADD CONSTRAINT curse_pk PRIMARY KEY ( curse_id );
```

```
CREATE TABLE enemy (
    enemy_id    NUMBER NOT NULL,
    enemy_stats NUMBER
);

ALTER TABLE enemy ADD CONSTRAINT enemy_pk PRIMARY KEY ( enemy_id );

CREATE TABLE item (
    item_id     NUMBER NOT NULL,
    item_stats  NUMBER,
    item_effect VARCHAR2(100)
);

ALTER TABLE item ADD CONSTRAINT item_pk PRIMARY KEY ( item_id );

CREATE TABLE item_in_character (
    character_item_id NUMBER NOT NULL,
    character_id      NUMBER NOT NULL,
    item_id           NUMBER NOT NULL
);

ALTER TABLE item_in_character ADD CONSTRAINT item_in_character_pk PRIMARY KEY ( character_item_id );

CREATE TABLE "Level" (
    level_id NUMBER NOT NULL,
    curse_id NUMBER NOT NULL
);

ALTER TABLE "Level" ADD CONSTRAINT level_pk PRIMARY KEY ( level_id );
```

```
CREATE TABLE player (
    player_id    NUMBER NOT NULL,
    character_id NUMBER NOT NULL
);

ALTER TABLE player ADD CONSTRAINT player_pk PRIMARY KEY ( player_id );

ALTER TABLE player ADD CONSTRAINT player_player_id_un UNIQUE ( player_id );

CREATE TABLE room (
    room_id        NUMBER NOT NULL,
    level_id       NUMBER NOT NULL,
    enemy_id       NUMBER NOT NULL,
    room_reward_id NUMBER NOT NULL
);

ALTER TABLE room ADD CONSTRAINT room_pk PRIMARY KEY ( room_id,
                                                      level_id );

CREATE TABLE room_reward (
    room_reward_id NUMBER NOT NULL,
    bombs_reward   NUMBER,
    trinket_id     NUMBER NOT NULL,
    money_reward   NUMBER
);

ALTER TABLE room_reward ADD CONSTRAINT room_reward_pk PRIMARY KEY (
room_reward_id );
```

```
CREATE TABLE room_size (
    room_id   NUMBER NOT NULL,
    room_size NUMBER
);

ALTER TABLE room_size ADD CONSTRAINT room_size_pk PRIMARY KEY ( room_id );

CREATE TABLE trinket (
    trinket_id     NUMBER NOT NULL,
    trinket_effect VARCHAR2(100),
    trinket_stats  NUMBER
);

ALTER TABLE trinket ADD CONSTRAINT trinket_pk PRIMARY KEY ( trinket_id );

ALTER TABLE achievement
    ADD CONSTRAINT achievement_item_fk FOREIGN KEY ( item_id )
        REFERENCES item ( item_id );

ALTER TABLE achievement
    ADD CONSTRAINT achievement_trinket_fk FOREIGN KEY ( trinket_id )
        REFERENCES trinket ( trinket_id );

ALTER TABLE backpack
    ADD CONSTRAINT backpack_item_fk FOREIGN KEY ( item_id )
        REFERENCES item ( item_id );

ALTER TABLE backpack
    ADD CONSTRAINT backpack_trinket_fk FOREIGN KEY ( trinket_id )
```

```
      REFERENCES trinket ( trinket_id );

ALTER TABLE challenge
   ADD CONSTRAINT challenge_achievement_fk FOREIGN KEY ( achievemnt_id )
      REFERENCES achievement ( achievement_id );

ALTER TABLE challenge
   ADD CONSTRAINT challenge_character_fk FOREIGN KEY ( character_id )
      REFERENCES character ( character_id );

ALTER TABLE challenge
   ADD CONSTRAINT challenge_player_fk FOREIGN KEY ( player_id )
      REFERENCES player ( player_id );

ALTER TABLE character
   ADD CONSTRAINT character_backpack_fk FOREIGN KEY ( backpack_id )
      REFERENCES backpack ( backpack_id );

ALTER TABLE "Level"
   ADD CONSTRAINT curse_id FOREIGN KEY ( curse_id )
      REFERENCES curse ( curse_id );

ALTER TABLE item_in_character
   ADD CONSTRAINT item_in_character_character_fk FOREIGN KEY ( character_id )
      REFERENCES character ( character_id );

ALTER TABLE item_in_character
   ADD CONSTRAINT item_in_character_item_fk FOREIGN KEY ( item_id )
      REFERENCES item ( item_id );
```

```
ALTER TABLE player
    ADD CONSTRAINT player_character_fk FOREIGN KEY ( character_id )
        REFERENCES character ( character_id );

ALTER TABLE room
    ADD CONSTRAINT room_enemy_fk FOREIGN KEY ( enemy_id )
        REFERENCES enemy ( enemy_id );

ALTER TABLE room
    ADD CONSTRAINT room_level_fk FOREIGN KEY ( level_id )
        REFERENCES "Level" ( level_id );

ALTER TABLE room_reward
    ADD CONSTRAINT room_reward_trinket_fk FOREIGN KEY ( trinket_id )
        REFERENCES trinket ( trinket_id );

ALTER TABLE room
    ADD CONSTRAINT room_room_reward_fk FOREIGN KEY ( room_reward_id )
        REFERENCES room_reward ( room_reward_id );

ALTER TABLE room
    ADD CONSTRAINT room_room_size_fk FOREIGN KEY ( room_id )
        REFERENCES room_size ( room_id );
```

D3 Przykładowe inserty:

```
INSERT INTO achievement (achievement_id, item_id, trinket_id)
VALUES (1, 1, 2);
INSERT INTO achievement (achievement_id, item_id, trinket_id)
VALUES (2, 2, 1);

INSERT INTO backpack (backpack_id, item_id, money, bombs, trinket_id)
VALUES (1, 2, 100, 5, 3);
INSERT INTO backpack (backpack_id, item_id, money, bombs, trinket_id)
VALUES (2, 1, 50, 6, 4);


INSERT INTO challenge (challenge_id, character_id, achievemnt_id, player_id)
VALUES (1, 1, 1, 1);
INSERT INTO challenge (challenge_id, character_id, achievemnt_id, player_id)
VALUES (2, 2, 2, 2);


INSERT INTO character (character_id, stats, character_name, backpack_id)
VALUES (1, 100, 'John', 2);
INSERT INTO character (character_id, stats, character_name, backpack_id)
VALUES (2, 100, 'Adam', 1);


INSERT INTO curse (curse_id, curse_effect)
VALUES (2, 'Decreases health by 10%');
INSERT INTO curse (curse_id, curse_effect)
VALUES (1, 'Decreases health by 5%');
```

```sql
INSERT INTO enemy (enemy_id, enemy_stats)
VALUES (1, 50);
INSERT INTO enemy (enemy_id, enemy_stats)
VALUES (2, 70);

INSERT INTO item (item_id, item_stats, item_effect)
VALUES (1, 20, 'Increases attack by 10');
INSERT INTO item (item_id, item_stats, item_effect)
VALUES (2, 30, 'Increases attack by 15');

INSERT INTO item_in_character (character_item_id, character_id, item_id)
VALUES (1, 2, 1);
INSERT INTO item_in_character (character_item_id, character_id, item_id)
VALUES (2, 1, 2);

INSERT INTO "Level" (level_id, curse_id)
VALUES (1, 1);
INSERT INTO "Level" (level_id, curse_id)
VALUES (2, 2);

INSERT INTO player (player_id, character_id)
VALUES (1, 1);
INSERT INTO player (player_id, character_id)
VALUES (2, 2);
```

```
INSERT INTO room (room_id, level_id, enemy_id, room_reward_id)
VALUES (1, 1, 1, 1);
INSERT INTO room (room_id, level_id, enemy_id, room_reward_id)
VALUES (2, 2, 2, 2);
```

```
INSERT INTO room_reward (room_reward_id, bombs_reward, trinket_id, money_reward)
VALUES (1, 2, 1, 10);
INSERT INTO room_reward (room_reward_id, bombs_reward, trinket_id, money_reward)
VALUES (2, 1, 2, 20);
```

```
INSERT INTO room_size (room_id, room_size)
VALUES (1, 20);
INSERT INTO room_size (room_id, room_size)
VALUES (1, 10);
```

```
INSERT INTO trinket (trinket_id, trinket_effect, trinket_stats)
VALUES (1, 'Increases health by 5%', 10);
INSERT INTO trinket (trinket_id, trinket_effect, trinket_stats)
VALUES (2, 'Increases health by 10%', 5);
```

D5 Przykładowe widoki:

View player_characters - Ten widok pokazuje kolumny player_id i character_name z tablic player i character, ten widok będzie pokazywał tylko wiersze w których character_id w tablicy player będzie równe character_id w tablicy character.

```
CREATE VIEW player_characters AS SELECT player_id, character_name FROM player JOIN
character ON player.character_id = character.character_id;
```

View trinket_backpacks- Ten widok pokazuje kolumny backpack_id i trinket effect z tablic backpack i trinket, ten widok będzie pokazywał tylko wiersze w których trinket_id w tablicy backpack będzie równe trinket_id w tablicy trinket.

```
CREATE VIEW trinket_backpacks AS SELECT backpack_id, trinket_effect FROM backpack
JOIN trinket ON backpack.trinket_id = trinket.trinket_id;
```

View room_rewards - Ten widok pokaże kolumny room_id, money_reward, i trinket_effect z tabel room, room_reward, i trinket, i będzie pokazywał tylko wiersze, gdzie room_reward_id w tabeli room będzie równe room_reward_id w tabeli room_reward i trinket_id w tabeli room_reward będzie równe trinket_id w tabeli trinket.

```
CREATE VIEW room_rewards AS SELECT room_id, money_reward, trinket_effect FROM
room JOIN room_reward ON room.room_reward_id = room_reward.room_reward_id JOIN
trinket ON room_reward.trinket_id = trinket.trinket_id;
```

View Player_Achievements - This view will combine data from the player, character, challenge, and achievement tables to show the player_id, character_name, and achievement_id for each player.

```
CREATE VIEW Player_Achievements AS SELECT player_id, character_name, achievement_id
FROM player JOIN character ON player.character_id = character.character_id JOIN challenge
ON player.player_id = challenge.player_id JOIN achievement ON challenge.achievement_id
= achievement.achievement_id;
```

Viev Character_Inventory - This view will combine data from the character, backpack, and item tables to show the character_name, item_id, item_stats, and item_effect for the items that each character has in their backpack.

```
CREATE VIEW Character_Inventory AS SELECT character_name, item_id, item_stats,
item_effect FROM character JOIN backpack ON character.backpack_id =
backpack.backpack_id JOIN item ON backpack.item_id = item.item_id;
```