Rapport Oblig 1 Databaser og Nettverk av Adrian Ihle

Player(Player_Name, Player_ID, state), Inventory(Item_Name, Item_ID, player*), gLevel(nrOfMonster, levelName, level_ID), GAME(game_ID, player*, score);

For spiller så har jeg valgt å bruke en helt vanlig INT som har AUTO_INCREMENT på seg som Primær Nøkkel. Dette er fordi dette gir et helt enkelt grunnlag for kobling fra spiller/avatar til alt annet. Siden den har AUTO_INCREMENT vil den også alltid være unik etter som AUTO_INCREMENT aldri lager duplikater i en tabell. Spillet jeg har basert meg ut fra er egentlig et platforming spill, derfor er det heller ikke veldig naturlig å bruke epost eller navn som Primær Nøkkel. Om dette skulle vært et mulitplayer spill kunne en Epost vært fint og om dette hadde vært en MMO så vill jeg nok brukt en epost som unik for spilleren og ha koblet denne spilleren til en samling spillavatarer som kunne ha AUTO_INCREMENT på seg for å genere helt unike nøkler innad i spillet. Jeg har også lagt til en ENUM active/inactive for å brukes til å se hvem spiller som skal oppdateres i en potensiell simulering/bruk av dette spillet siden det egentlig bare skal være en spiller av gangen. Dette blir altså mer som om spillerne lagrer hvem saveslot som ting skal lagres på akkurat nå.

I Inventory så er Item_ID Primær nøkkel. Orginalt ønsket jeg å finne en måte å koble hele tabellen Inventory til spilleren og la hvert enkelt Item ha ID-er som er samme basert på om de var kopier av hverandre og så kunne relatere det til en tabell som inneholdte informasjonen om alle disse ID-ene. Ettersom jeg ikke fant en fornuftig måte å gjøre dette så har jeg da et Inventory som er felles for alle spillere. Derfor må alle items ha egene ID-er og jeg kobler heller de sammen med spilleren med en Fremmed Nøkkel som referer til spilleren sin Primær Nøkkel. For å kompansere for mangelen for mangelen på en tabell som inneholder informasjon om de forskjellige itemene så har jeg lagt in en ENUM som inneholder noen enkle typer man kan bruke.

GLevel er ikke koblet opp til noen tabeller. Dette har jeg gjort litt på grunnlag av at i realiteten trenger ikke level å være koblet opp til noe i den forstan at levelene vil være det samme vær gang det kjøres igjennom og det eneste som trenges er at det blir gitt info til andre tabeller som enkelt kan gjøres med spørringer eller lignede. Si en spiller gjennomfører level 2 og får X poeng. Dette kan da sendes inn til GAME med en UPDATE spørring.
gLevel har navn for å ha noe annet og litt enklere å søke på enn level_ID. levelName kan fint være unik for det spillet jeg opererer med ettersom det kun kommer til å være en instans til enhver tid, men jeg føler det er en god ide å innkludere ID også slik at det er en innelagt base for flerspiller muligheter som multiplayer.

Det kan også være fint for tilfeller hvor spillere spiller igjennom et level flere ganger. Da vil levelet ha samme navn men ID-ene vil være forskjellige hver gang. Dette fører da derimot til en del Redunant Data, som antagligvis ikke har noen effektiv bruk.

For GAME, så har jeg egentlig ikke deklarert game_ID som Primær Nøkkel, men den har blitt det ettersom jeg satt en AUTO_INCREMENT på den og så koblet den opp til Fremmed Nøkkelen player. I prosessen har den da blitt en Primær Nøkell. Dette passer helt fint ettersom da vil være gjennomspilling ha en unik ID som er koblet opp til en unik spiller ID.

Score vil kunne lagre den total scoren som spilleren har og HIghscore i Player vil så bli oppdatert til til den som er størst for hver gjennomspilling gjennom en IF score >maxScore THEN SET maxScore = score END IF;. Eller noe lignende.

```
CREATE DATABASE ColorRaves;
USE ColorRaves;

CREATE TABLE Player
        (Player_Name VARCHAR(40),
         Player_ID INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY);

CREATE TABLE Inventory
        (Item_Name VARCHAR(40),
         Item_ID INT PRIMARY KEY AUTO_INCREMENT,
         player INT UNSIGNED NOT NULL, FOREIGN KEY(player) REFERENCES Player
        (Player_ID) ON UPDATE CASCADE ON DELETE CASCADE);

CREATE TABLE gLevel
( nrOfMonsters TINYINT,
levelName VARCHAR(40),
 level_ID INT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY);

 CREATE TABLE GAME
        (game_ID INT UNSIGNED NOT NULL AUTO_INCREMENT,
         player INT UNSIGNED NOT NULL,
        FOREIGN KEY(player) REFERENCES Player (Player_ID) ON UPDATE CASCADE ON
        DELETE CASCADE,
        PRIMARY KEY(game_ID, player));

ALTER TABLE GAME ADD COLUMN score INT;

 ALTER TABLE Player ADD COLUMN state ENUM('active', 'inactive');

 ALTER TABLE Inventory ADD COLUMN item_type ENUM('weapon','power up', 'other');

 ALTER TABLE Player ADD COLUMN HighScore INT;
```

INSERT INTO gLevel VALUES(6, 'level 1',NULL),(10,'level 2', NULL),(2,'level 3', NULL),(25,'level 4',NULL);

nrOfMonsters	levelName	level_ID
6	level 1	1
10	level 2	2
2	level 3	3
25	level 4	4
NULL	NULL	NULL

INSERT INTO PLAYER VALUES ('Adrian',NULL,'inactive',76), ('Oda',NULL,'inactive',80), ('Berd',NULL,'inactive',64), ('Emma',NULL,'active',NULL);

Player_Name	Player_ID	state	HighScore
Adrian	1	inactive	76
Oda	2	inactive	80
Berd	3	inactive	64
Emma	4	active	NULL
NULL	NULL	NULL	NULL

Når jeg brukte setningen
UPDATE PLAYER SET HighScore = (SELECT max(score) FROM GAME WHERE player = 4) where Player_ID = 4;
Så ble Emma sin HighScore endret riktig:

Player_Name	Player_ID	state	HighScore
Adrian	1	inactive	76
Oda	2	inactive	80
Berd	3	inactive	64
Emma	4	active	33
NULL	NULL	NULL	NULL

```
INSERT INTO Inventory VALUES
        ('Turbo Boost',NULL,4, 'power up'),
        ('Ray Gun',NULL,1, 'weapon'),
  ('Shoes',NULL,1, 'other'),
 ('Potion',NULL,3, 'power up'),
  ('Wave Grenade',NULL,2, 'weapon'),
  ('car',NULL,4, 'other');
```

Item_Name	Item_ID	player	item_type
Turbo Boost	1	4	power up
Ray Gun	2	1	weapon
Shoes	3	1	other
Potion	4	3	power up
Wave Grenade	5	2	weapon
car	6	4	other
NULL	NULL	NULL	NULL

```
INSERT INTO GAME VALUES
        (NULL, 1, 43)
        (NULL, 2, 50),
  (NULL, 2, 20),
  (NULL, 2, 80),
  (NULL, 3, 16),
  (NULL, 3, 64),
  (NULL, 4, 33),
  (NULL, 4, 33),
  (NULL, 1, 76);
```

game_ID	player	score
1	1	43
2	2	50
3	2	20
4	2	80
5	3	16
6	3	64
7	4	33
8	4	33
9	1	76
NULL	NULL	NULL

For å få high score lista så brukte jeg kun
SELECT* FROM GAME ORDER BY score DESC;
Som gir:

game_ID	player	score
4	2	80
9	1	76
6	3	64
2	2	50
1	1	43
7	4	33
8	4	33
3	2	20
5	3	16
NULL	NULL	NULL

Man kan gjøre det samme og få med navn ved å
SELECT Game.score, player.Player_Name FROM GAME LEFT JOIN player ON game.player = player.Player_ID ORDER BY score DESC;

score	Player_Name
80	Oda
76	Adrian
64	Berd
50	Oda
43	Adrian
33	Emma
33	Emma
20	Oda
16	Berd

For å få en DISTINCT highscore liste så har jeg jo alt lagt inn HighScore i player så jeg trenger egentlgi bare bruke
SELECT Player_Name, HighScore FROM Player ORDER BY HighScore DESC;

Player_Name	HighScore
Oda	80
Adrian	76
Berd	64
Emma	33

Men man skal kunne bruke noe a la

SELECT Player_Name, MAX(score) FROM GAME GROUP BY Player ORDER BY score DESC;

Om jeg hadde hatt spiller navn i Game, det er sikker mulig å finne spiller navn fra Foreign key men vet ikke helt hvordan det gjøres.

Ved å bruke
DELETE FROM Player where Player_ID = 4;

Slettet jeg 'Emma' fra spiller listen. Den raden forsvant da fra Player, alle rader tilhørende denne Primær Nøkkelen forsvinner da fra GAME og Inventory ettersom de begge har spiller som en Fremmed Nøkkel;