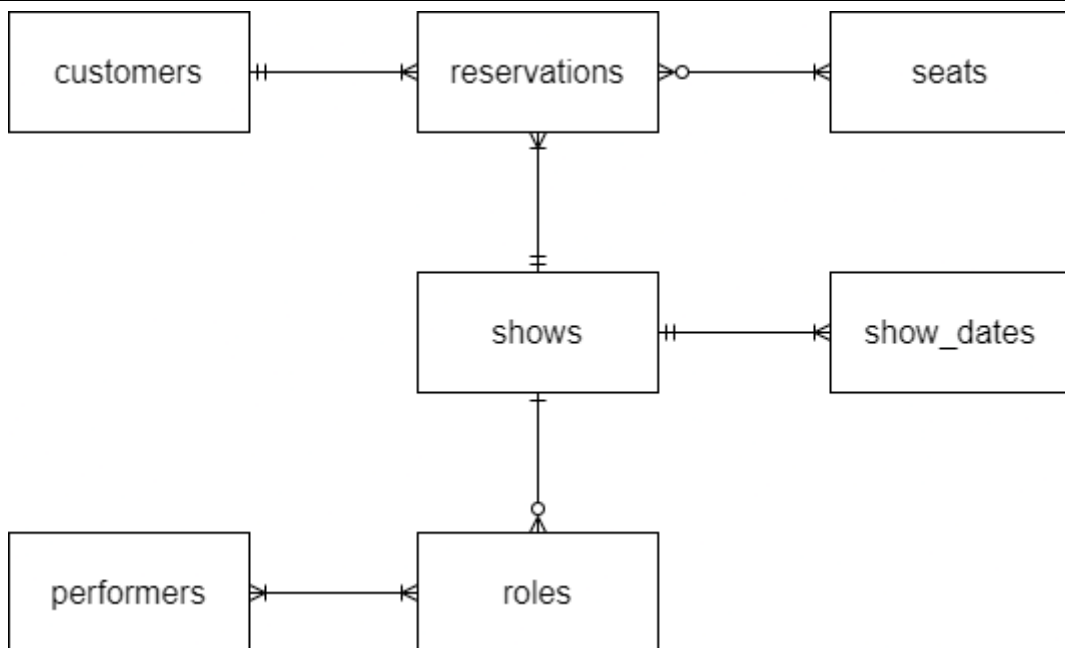




Göteborgs nya kultursatsning är Göteborgs **MINI**opera som kommer hålla föreställningar på en mindre scen med större innovativa satsningar inom musik och teknik. Operan kommer vara MINimal och håller för totalt 86 sittplatser.

MINIoperan behöver en egen databas separat från sin storasyster och detta uppdrag har givits mig. Databasen kommer hantera information från de olika föreställningarna, rollkaraktärer, artister, datumhantering, sittplatser, reservationer och kundinformation.

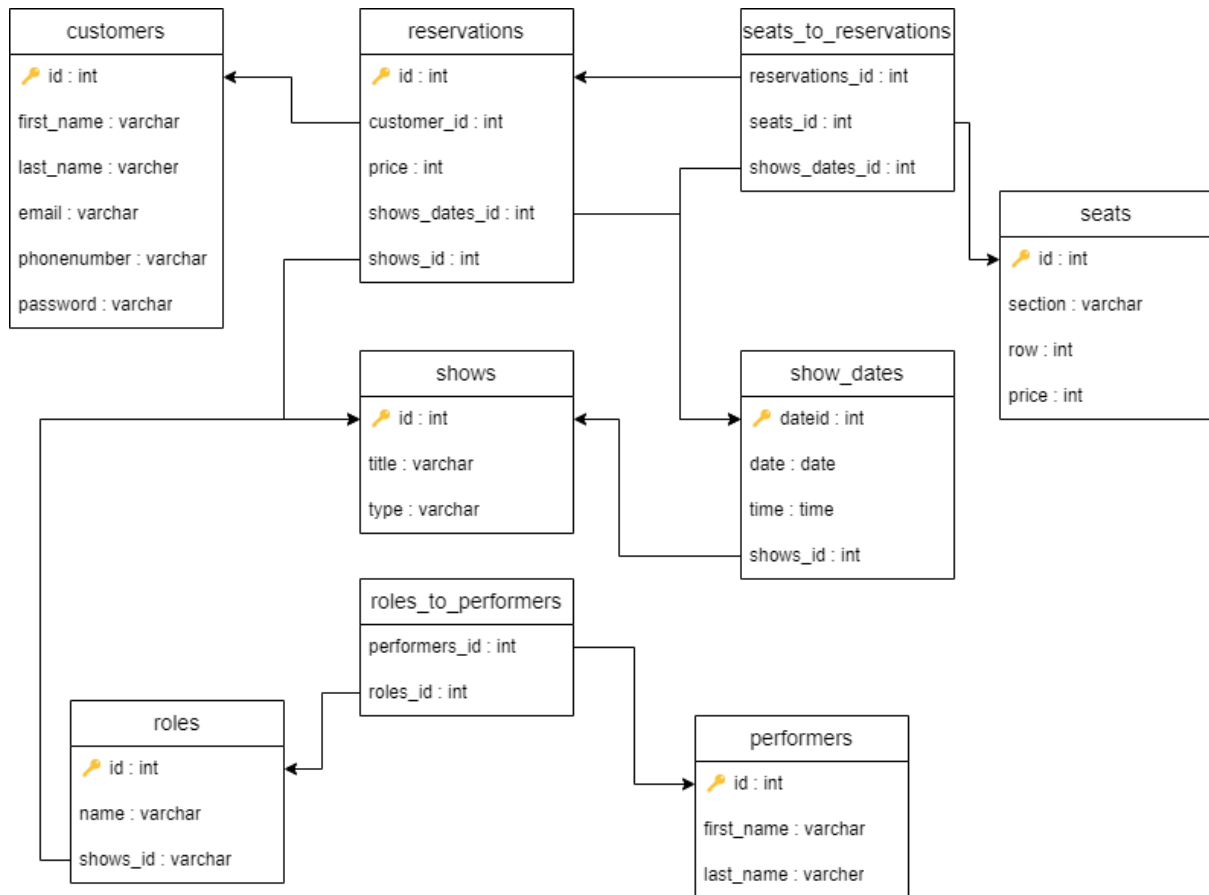
ER-diagram



Vilka entiteter är nödvändiga för att kunna skapa ett mindre bokningssystem var min första fråga som behövde besvaras? Detta resulterade i tabellerna : customers, reservations, seats, shows, roles och performers. När väl tabellerna skapats med attributerna som jag avgjort som nödvändiga insåg jag att datum och tider fungerade bättre som en egen tabell med unika id:n som primary key.



Kopplingsdiagram




Databasen består av nio tabeller varav två tabeller är kopplingstabeller. För att lättare avgöra huruvida en seat är bokad eller inte har jag även kopplat show_dates till kopplingstabellen seats_to_reservations.

Databasen ska ha kommandon utifrån CRUD - create, read, update & delete. Jag har begränsat arbetet med CRUD att enbart behandla användarens version av programmet. Detta innebär i korthet att datan som finns i databasen som showerna, dess roller, artisterna samt säten och datum och tid inte kan skapas, tas bort eller uppdateras från applikationen.

Problem under arbetets gång

Jag hade svårigheter vid borttagandet av reservationer och hur sätena skulle uppdateras så att de skulle visas som lediga igen. Jag testade med olika SQL-kommandon innan jag kom fram till att jag kunde använda relationerna mellan tabellerna; reservations och seats_to_reservations då den senare har foreign key som pekar mot reservations.id som är tabellens primary key. Vid DELETE eller UPDATE av reservations.id påverkas även "barntabellen"¹ vid användandet av CASCADE.



En utmaning jag hade under programmets början men en utmaning som har gett mig mycket mer förståelse för är hur datahantering fungerar mellan databas och applikationen men främst skapande av klassobjekt som håller information nödvändig för programmet. Ett exempel är tabellen 'shows' som har attributerna id (primary key), title och type och tabellen 'show_dates' håller information över datum och tid med unika id:n och en foreign key som pekar mot 'shows'. Med hjälp av INNER JOIN i SQL-kommando kan jag hämta data från flera tabeller och skapa ett klassobjekt som binder ihop datan från de båda tabellerna. Jag insåg hur låst jag var med en felaktig inställning att ett klass måste dela alla tabellernas attribut.

```
var sql = (`SELECT shows.id, shows.title, shows.type,
show_dates.dateid, show_dates.date, show_dates.time
FROM shows
INNER JOIN show_dates ON shows.id = show_dates.shows_id;`);
```

```
class ShowToDates
{
    2 references
    public int ShowId {get; set;}
    6 references
    public int DateTimeId {get; set;}
    5 references
    public string Title {get; set;}
    2 references
    public string Type {get; set;}
    6 references
    public DateTime Date {get; set;}
    4 references
    public TimeOnly Time {get; set;}
```

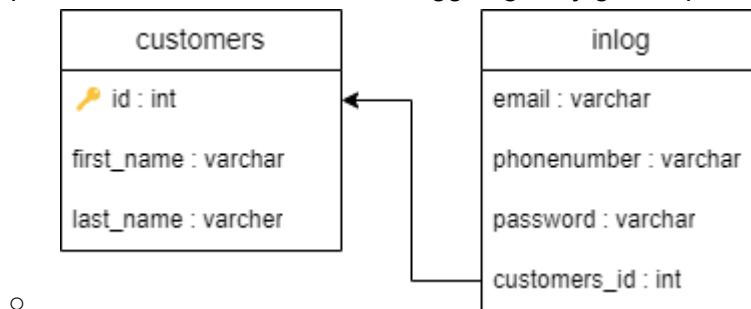
¹ <https://www.javatpoint.com/mysql-on-delete-cascade>

Vidareutveckling

Då programmet är designat utifrån användarens perspektiv, behövs vidareutveckling för en administrativ version av programmet. Sedan är det inte logiskt att informationen om de olika showera hämtas direkt från databasen exempelvis vid menyhantering som hämtar ut alla showers titlar. Detta var ett val från min sida enbart för att arbeta mer åt databasen, det vill säga att skicka datan sinsemellan databasen och applikationen.

Sammanställning för vidare utvecklingsmöjligheter :

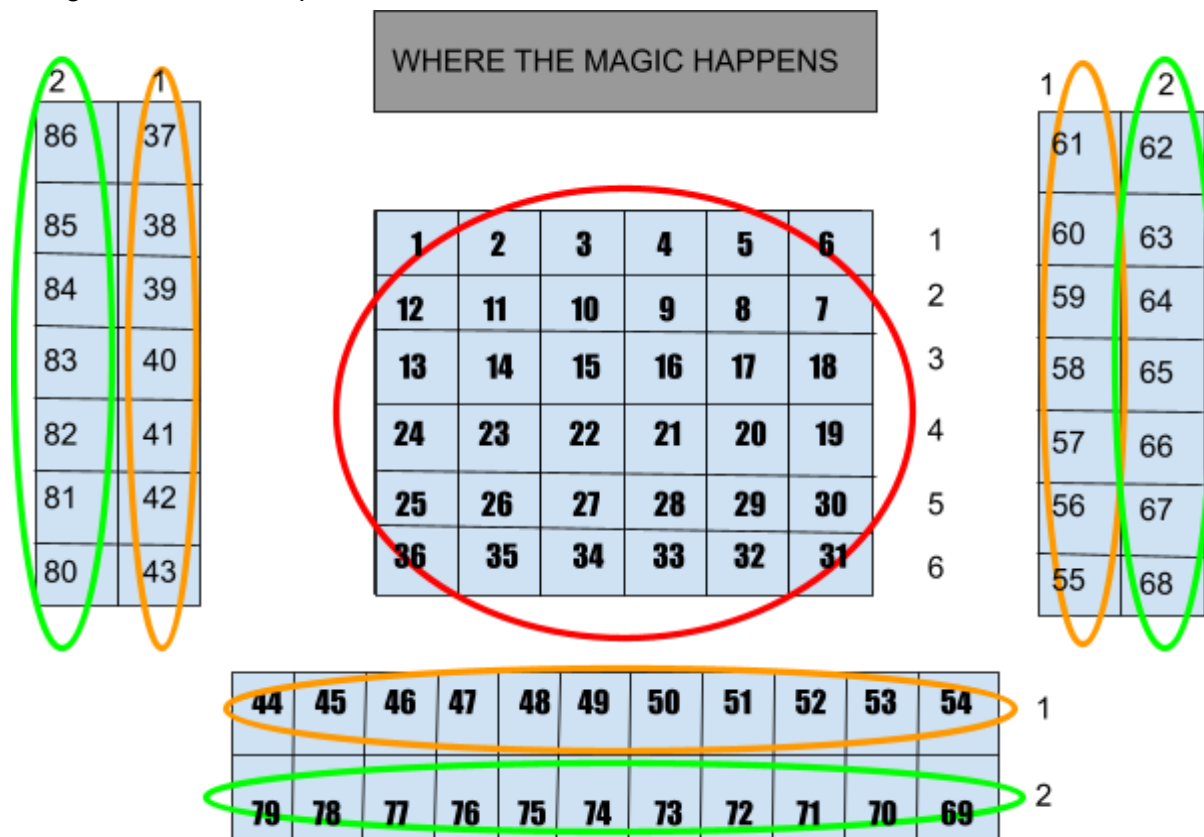
- Skapa en tabell som tillåter flera inloggningsmöjligheter per användare,



- Komma på en design som möjliggör bokning av säten på balkongen (se bilaga) samt olika prisnivåer av säten,
- Lägga till kod i programmet som ser över datum så inte tidigare shower kan bokas och att shower längre fram i tiden släpps tillgång till via förbestämt datum,
- Skapa administrativ version av programmet,
- Se över SQL-kommandon och låta databasen utföra stored procedures.

Bilaga

Design över sätena i operan :



Olika priser beroende av säte och gästens ålder och ekonomiska situation :

PRISER	ORDINARIE	BARN	STUDENT	PENSIONÄR
PARKETT	450	225	338	315
BALKONG R 1	380	190	285	266
BALKONG R 2	350	175	263	245