Ida Nilsson
2018-04-07

Design fördelar

Ren och flexibel kod

Jag har varit noga med att namnge så mycket som möjligt med camel-case i mitt arbete. Jag har försökt att hårdkoda så lite som möjligt, det är endast uppgift nummer två som är hårdkodad. Jag har tänkt efter när jag har valt variabelnamn, så att det verkligen framgår vad som menas med mina namngivningar. Jag har varit konsekvent i namngivningen, alla views börjar med "v_" tex.
Jag har även sett till att allt är namngivet i singular.

Enkelhet

Min databas är uppbyggd på ett sätt som gör att en utomstående snabbt skulle kunna sätta sig in i den, delvis för tydliga variabelnamn men även den lilla mängden tabeller samt tydligheten. Jag känner att jag har fått till det bra med vilka kolumner som ligger i vilka tabeller.

Statistics-tabellen

Här har jag tagit hänsyn till mina foreign-key-options, on delete: cascade. Detta är anledningen till att jag inte har några foreign-keys mellan statistics-tabellen och de andra tabellerna. Skulle jag ta bort en rad i rentinfo, och jag skulle ha en foreign-key till statistics-tabellen så hade min statestik rad försvunnit tillsammans med raden i rentinfo. Jag vill att min statistics-tabell ska innehålla all information som lagras.

Helhet

Mitt upplägg har en helhet där mina lösningar är uppbyggda i enighet, allting fungerar tillsammans. Mina lösningar gör det enkelt att ändra film-status till uthyrd eller inlämnad. Du kan se alla filmer som ägs av firman, filmer som är uthyrda samt vilka filmer som är inte är inlämnade när uthyrningstiden passerar fyra dagar.

En film har ett ID

Att varje film har ett unikt id gör det möjligt för dubbletter. Det ger även en möjlighet för en mer specifik sökning.

Design nackdelar

Hårdkod

I slutprojektet framgick det vilka uppgifter som skulle lösas med views och vilka som skulle lösas med stored procedures. Resultatet av detta var att uppgift nummer två fick lösas med hårdkod, tyvärr. Detta gör att uppgift två inte är flexibel, skulle man vilja att det skulle lösas på ett flexibelt sätt skulle en stored procedure vara önskvärd.

Namngivning

Hade jag gjort detta arbetet igen hade jag valt att använda mig av snake-case, jag märker hur vissa delar av mitt projekt inte kan innehålla stora bokstäver, vilket gör det svårt att läsa. Det hade även varit snyggare med tanke på att mina stored procedures börjar med "sp_" tex.

En anställd kan inte hyra film

Kan ses som en nackdel, men jag personligen ser inget problem i det eftersom personalen skulle kunna registrera sig som kund, dock utan koppling till sin anställning.
Ska en anställd få rabatt på alla sina köp så slås det in i kassan manuellt.

En film kan endast ha en skådespelare

Jag utgår ifrån att en film har en huvudroll och att det är den skådespelaren som är mest intressant i samband med filmen. Skulle jag velat ha flera skådespelare per film så hade min databasstruktur sett annorlunda ut.

En film har ett ID

Att en film har ett ID gör att det inte är lika lätt att få en snabb överblick på hur många filmer det finns av en titel. Vid ett annat upplägg av strukturen hade movie-tabellen kunnat ha en kolumn för antal ex av de filmer av samma titel. En lösning på detta i min struktur hade varit en select, lite mer omständigt, men inte omöjligt.

Avsaknad av felhantering

Min funktion “isLeased” saknar felhantering, detta gör att en film faktiskt skulle kunna bli utlånad flera gånger under en dag. Det finns även inget felmeddelande om filmen redan skulle vara uthyrd. Dock skulle detta ändå inte vara möjligt då filmen inte skulle finnas i butiken. Hade jag haft en struktur där movie-tabellen skulle haft en kolumn för antal ex skulle min nuvarande lösning kanske skapa problem.

Storage Engine

InnoDB

Jag valde att använda mig av InnoDB eftersom det är standard för MySQL. Jag tycker om integriteten för främmande nycklar, då jag kan välja vad som skall hända om jag raderar/uppdaterar något som har en koppling till den främmande nyckeln. I utvecklingssyfte ser jag möjligheten till låsbarhet till radnivå som något mycket positivt då det gör det enkelt att avgränsa, till skillnad från andra storage engines som enbart kan låsa till tabellnivå.
Skulle databasen vidareutvecklas skulle stöd för transaktioner krävas vilket InnoDB stödjer.