DATABAS RAPPORT

By Denisé Larsson

Modeller :
Analysera kravet och notera vilka objekt jag behöver. Objekt som jag valde var:

- Ort
- Utbildning
- Klass (Utbildningsledare tar hand om en klass)
- Anställd(Utbildningsledare och lärare, istället för 2 tabeller)
- Utbildningskurs (Blev en relations tabell, mellan Utbildning och Kurs)
- Kurs
- KlassKurs (Lärare har hand om en klass kurs)
- Studerande
- Betyg (relations tabell mellan klasskurs och studerande)
- Inaktiv studerande (när en elev skrivs ut, ska det föras över hit)
- Inaktiv betyg (här tänkte jag att man inte behöver klarat en hel utbildning för att för att får ett slutbetyg, därför blev det inaktiv betyg som kan hämtas när eleven inte är aktiv längre)

När objektet är framtagen skapade jag en infologisk modell för att kolla vilka relationer dem har till varandra och även kunna visa upp detta för kunden om det var så dem ville har det. Fick feedback från Annika och fick små justera, därefter normaliserade jag för att se att det inte var några dubbel lagringar.Fick lägga till några extra kolumner som till exempel datum till "InAktivBetyg" tabellen etc. Den infologiska modellen är till för att visualisera hur företaget fungerar dvs saker och vilken samband dem har.

PROCEDURE:
Jag valde att lägga till tabell med inaktiva studeranden och inaktiv betyg. Eftersom kravet var att man ska kunna automatiskt överför elevernas information efter avslutat utbildning till en tabell och hämtar betyg när en elev inte längre är aktiv blev de 2 Procedure metoder på detta. Lagrade procedure förbättra prestanda och minska trafiken mellan applikationen och databasservern. När en procedure är skapad kompilera jag bara med EXEC.

CONSTRAINT:
Jag har använt dessa constraint (***PRIMARY KEY, FOREIGN KEY , NOT NULL*)** när jag skapade tabellerna för att vara säker att alla primärnycklar och främmande nycklar knyter ihop. Jag hade kunnat göra det med AlTER TABLE men för att vara säker och inte glömma det så har jag gjort det här. Men när jag skulle INSERT data fick jag lägga till "*SET IDENTITY INSERT*" Studerande ON/OFF" på varje tabell förre/efter, eftersom jag senare valde att ge id nummer till vissa tabeller som hade
"Id INT IDENTITY(1,1) PRIMARY KEY". Använde en **CHECK** för att kolla så att Anstalld Id är mellan 1-1000. Att använda constraints hjälper det oss att skapa en begränsning för ett eller flera fält.

- Primary key: Alla primär nyckel är unika och inte null
- Foreign key: För att anger att en fält är en sekundär nyckeln och den definiera specifika som ska utföras när ett motsvarande primärnyckel värde ändras.

- NOT NULL - presentera frånvaro av ett värde. NULL ska tolkas som att databasen avsiktligt eller oavsiktligt saknar ett värde för kolumnen, därför ska de flesta vanliga operationer man vill göra på NULL misslyckas, eftersom man omöjligen kan vilja utföra en operation på ett värde som inte existerar.

TRIGGERS:
Jag valde att göra en trigger på när man lägger till ett betyg eller ändra så uppdateras datumet automatiskt. Var svårt och komma på något bra här, tänkte först att man kunde använda trigger till flytta elever som har avslutat en utbildning men det blev en procedurer där istället. Med en trigger kan vi utföra en uppdatering automatiskt utan att behöva tänka på. Som då i detta fallet att jag uppdaterade ett betyg och inte behöva tänka på att uppdatera datumet med.

VYER: Här skapade jag en vy som visar lärareId, KursId, start/ slut datum, betyg och vilken elev. Eftersom det också var med i kravet gjorde jag en vy som visar dessa information. Detta är smidigt om man vill se en del eller flera delar av en tabell och får det samlade på ett ställe.

JOIN,GROUP,BY,HAVING m.m:
Valde att visa hur många som studera i en utbildning, vilka kurser och betyg som dem har fått visas.

JDBC - JDBC är en API som tillåter java applikationer för att ansluta till och fråga ett brett spektrum av databaser.
JDBC gör det möjligt för en mjukvaruutvecklare att springa SQL frågor i ett Java-program.
Databasanslutningen och alla nödvändiga fråga översättningar hanteras av JDBC-drivrutinen.
Till exempel samma Java metod kan användas för att fråga en MySQL-databas och en Microsoft SQL Server-databas. Målet är att ge utvecklare "skriv en gång, kör var som helst" -funktionalitet, vilket gör det enkelt att arbeta med olika typer av databaser.

Sammanfattning:
Under projektets gång har den största biten varit att skapa den infologiska modellen och sätta upp rätt data. Har fått rita många infologiska modell för att säkerställa att det blir så bra som möjligt i början. När det kommer till att lägga in data gjordes mycket i början innan db byggdes, märkte även här att jag kunde gjort insert i tabellerna i efterhand när jag behövde en viss data istället för att droppa db och köra den om och om igen efter jag korrigerade data texten.

Jag tror att det är viktigt och analysera korrekt i början annars blir det mycket extra jobb när man väl har skapat databasen. Jag har också valt att lägga in mycket data för att inte begränsa testningen allt för mycket. Fick ändra vissa relationer mellan olika tabeller ett par gånger för att får rätt på data.

När det kommer till att skapa procedure, trigger, vyer, group by etc fick jag testa mig fram och valde utifrån kravet. Hade kunnat skapa hur mycket som helst men kravet är alltid det viktigaste och får med, allt annat görs om det finns tid över.

Koppling mellan databas och java gick smidigt, kul att se kopplingen och vad det faktiskt gör.Ser verkligen fram emot kommande kurser :)

Lärdomar: Det har varit mycket error och hitta rätt kombinationer, men efter alla fel man har gjort vet jag förhoppningsvis när man är ute och jobba vad jag inte borde göra. Många ändringar under projektets gång och många sql kod fick slängas fast jag hade suttit med det länge, det var många gånger lättare och ta bort och skriva om än att försöka lägga till en massa onödigt för att får det och funka.