



NET2 - 09

Databasdesign

Förra gången

- Mer SQL
 - Joins och relationer
 - Databasteknik 08 -SQL - Joins.pdf
 - Övningar
 - Övningsuppgifter – Relationsdata.pdf

Idag

- Databasdesign
 - Normalisering
 - Primary/Foreign key
 - Constraints
- ER-diagram
 - Lucidchart
- Gruppövning – Bygg bokhandel

Radnummer iSSMS

- [Display Line Numbers in a SQL Server Management Studio Query Window \(mssqltips.com\)](http://mssqltips.com)

Designprocessen

En väl strukturerad databas

- Sparar disk genom att inte dubbellagra data
- Bibehåller datas korrekthet och integritet
- Ger meningsfull åtkomst till data

Designprocessens faser

1. Kravanalys
 - Identifiera vad databasen skall användas till
2. Organisera data i tabeller
3. Ange primärnycklar
4. Analysera relationer
5. Normalisera tabellerna
6. Fylla tabeller med testdata

Kravanalys: vad är syftet med databasen?

- Samla in data
 - Intervjua kommande användare
 - Analysera manuella formulär
 - Analysera befintliga system

Identifiera entiteter

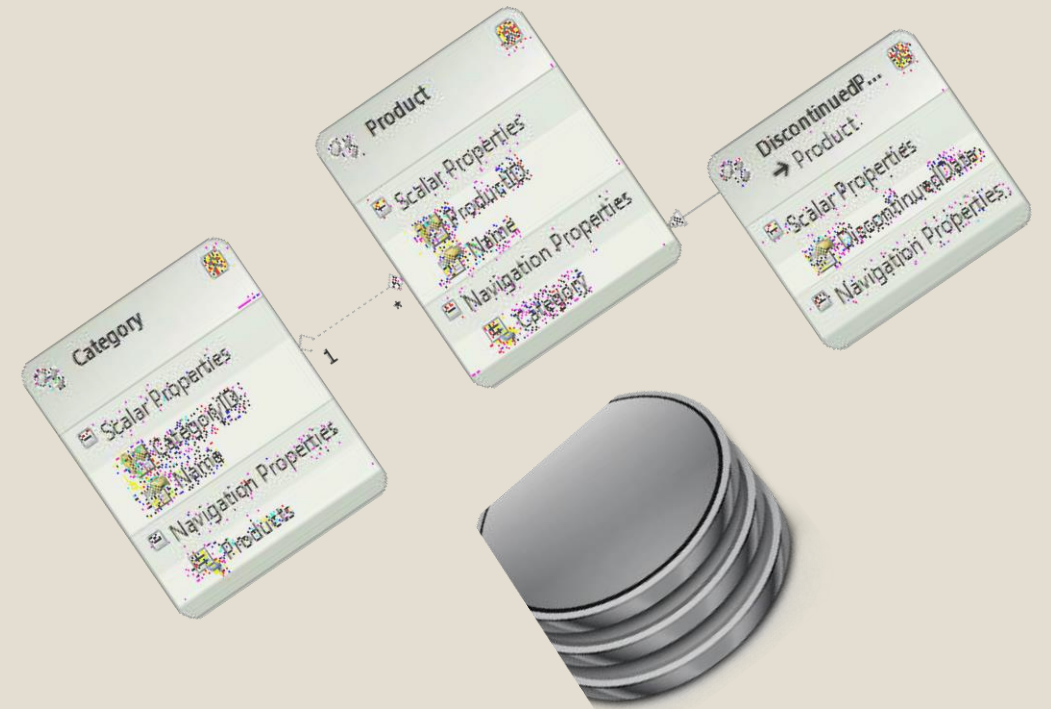
- Entiteter (tabeller) representerar objekt i den verkliga världen
 - Oftast är de substantiv i specifikationen
 - Till exempel:

Vi behöver utveckla ett system som lagrar information om **studenter**, som läser olika **kurser**. Kurserna hålls i olika **städer**. När nya student registrerar sig samlar vi in följande information: namn, student ID nummer, foto och datum

- Entiteter: Student, Kurs, Stad
 - Eller, bättre, på engelska: Student, Course, City

Identifiera kolumner

- Kolumner i tabeller är egenskaper hos entiteterna
 - De har namn och typ
- Till exempel kan studenter ha:
 - Namn (text)
 - Id-nummer (tal eller text)
 - Foto (binärdata)
 - Inskrivningsdatum (datum)



Identifiera kolumnerna

- Kolumner är förtydliganden av entiterna i kravtexten, till exempel:

Vi behöver utveckla ett system som lagrar information om **studenter**, som läser olika **kurser**. Kurserna hålls i olika **städer**. När nya student registrerar sig samlar vi in följande information: namn,
student ID nummer foto datum

- Studenter har följande karakteristika:
 - Namn, student ID nummer, foto, inskrivningsdatum och en lista av kurser där de deltar

Kravanalys: bryt ner delarna

- Ofta används ett Data Dictionary för att hålla ordning
- Bryt ner stora element till små
 - Adress → Gatuadress + Postnummer + Ort + Land
- När detta är gjort är det dags att börja planera den egentliga databasen

Struktur: databasens byggblock

- Skapa en visuell representation av databasen
- Skapa tabeller utifrån våra informationsdelar

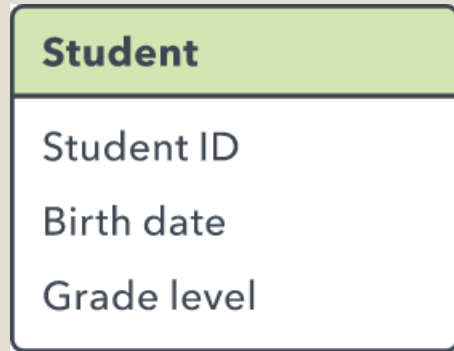
Förnamn	Efternamn	Ålder	Postnummer
Roger	Williams	43	34760
Jessica	Rabbit	32	97453
Luke	Skywalker	56	07645

Tilldela datatyper

- Det är viktigt att alla data har rätt typ
 - Det går att ändra senare, men det är enklast att få det rätt från början
- Några vanliga datatyper:
 - CHAR – text av fast längd
 - VARCHAR – text av varierande längd
 - TEXT – stora textblock
 - INT – heltal
 - FLOAT, DOUBLE – flyttal (tal med decimaler)
 - BLOB – binärdata

Entiteter i ett ER-diagram

- Normalt används en symbol i form av en box med text:



Primärnycklar

- Primärnycklar behöver vara
 - Unika
 - Oföränderliga
 - Alltid ha ett värde (icke-NULL)

Primärnyckel – hur väljer man?

- Lägg alltid till en extra kolumn för primärnyckeln
 - Använd inte en existerande kolumn, även om värdena är unika (t.ex. personnummer)
 - Avgörs från fall till fall
 - Helst vara ett heltal (Snabbast)
 - Bör deklarerars som primärnyckel
 - Använd **IDENTITY** för att få auto-inkrement
 - Lägg primärnyckeln som första kolumn

Identifiera relationer (relationships)

- Relationer/förhållanden är beroenden mellan entiteter:

Vi behöver utveckla ett system som lagrar information om **studenter**, som **läser** olika **kurser**. **Kurserna hålls i** olika **städer**. När nya student registrerar sig samlar vi in följande information: namn, student ID nummer, foto och datum

- "Studenter läser kurser" – många-till-många förhållande
- "Kurser hålls i städer" – många-till-en (eller många-till-många) förhållande

Skapa relationer mellan entiteter





- Det finns olika sorters relationer
- Kallas **kardinalitet**
 - En till en
 - En till många
 - Många till många

Vanligaste symboler för kardinalitet

- Ring: "noll"
- Streck: "ett"
- Kråkfot: många

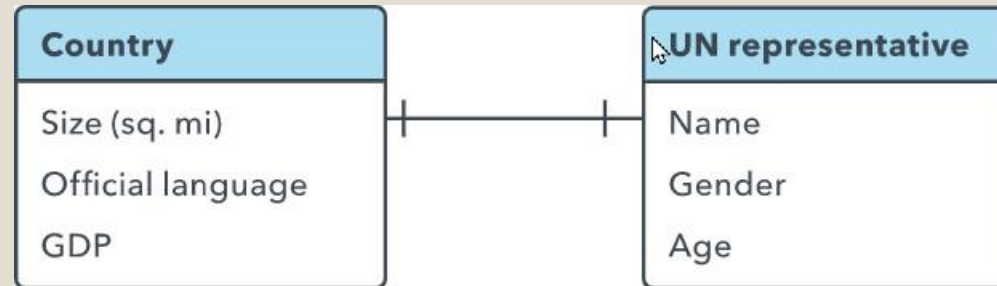


Symboler för kardinalitet

Beskrivning	Symbol
Ring och streck: Minsta nolla, maximalt en (valfritt)	
Streck och streck: Minst ett, maximalt ett (obligatoriskt)	
Ring och kråkfot: Lägsta nolla, maximalt många (valfritt)	
Streck- och kråkfotsfot: Minst en, maximalt många (obligatoriskt)	

Relationer: en-till-en

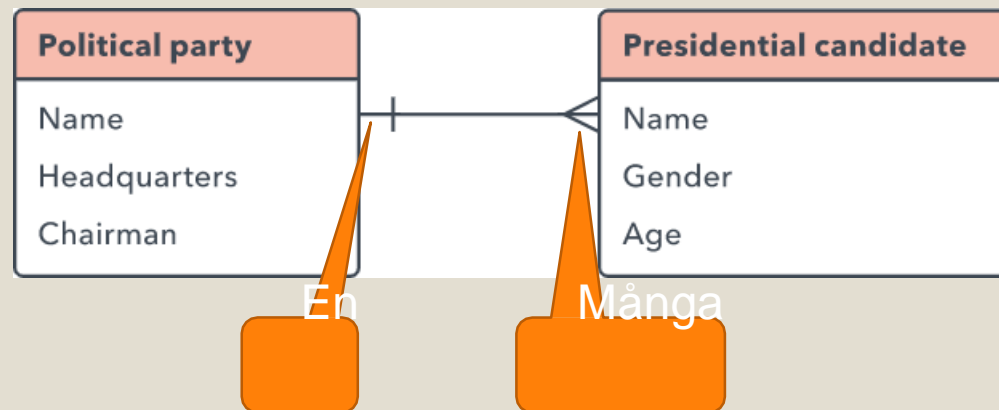
- För varje post i tabell A finns en post i tabell B



- Vanligen är det bättre att slå ihop detta till en tabell

Relationer: en-till-många

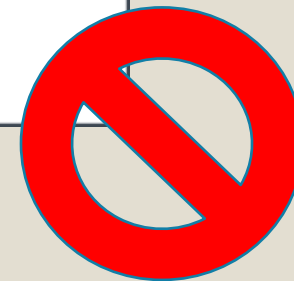
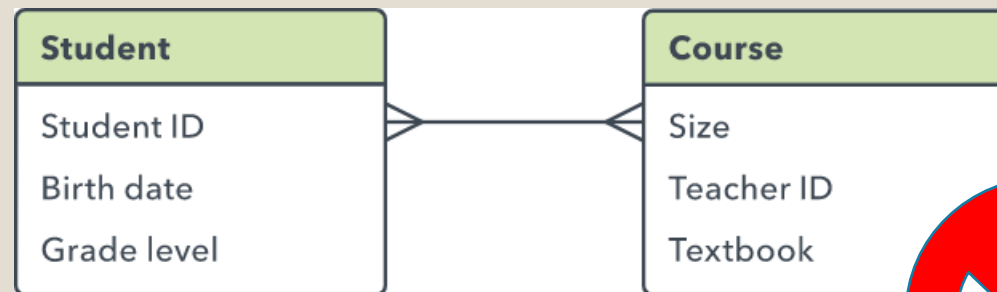
- En rad i tabell A hör ihop med många rader i tabell B
 - En beställare kan ha flera ordrar
 - En kund kan ha lånat flera böcker på biblioteket
 - Ett parti kan ha många presidentkandidater



- Implementering:
 - Ta PK från "en"-tabellen och lägg den som FK-fält i "många"-tabellen

Relationer: många-till-många

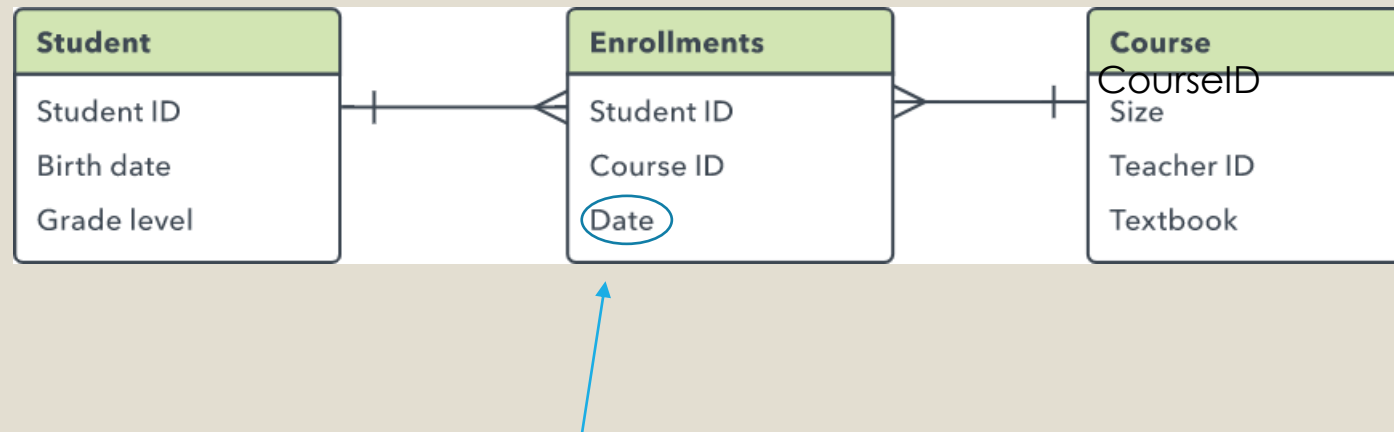
- En rad i tabell A hör ihop med många rader i tabell B, **och**
- En rad i tabell B hör ihop med många rader i tabell A
 - En student läser många kurser
 - En kurs blir läst av många studenter



- **Detta kan vi inte hantera i relationsdatabas**

Relationer: många-till-många

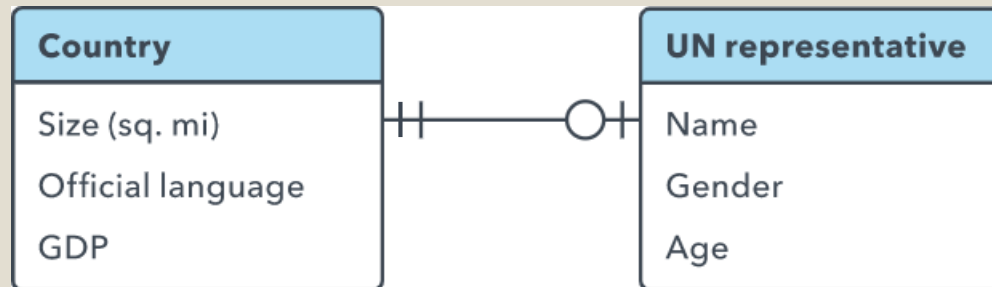
- En många-till-många relation måste lösas upp
- Två en-till-många
- Vi skapar en **upplösningstabell (junction table)**



- Upplösningstabellen kan ha extra data. Eller inte

Obligatorisk eller inte?

- Vi behöver också se på båda sidorna i en relation:
- Måste det finnas några poster?
 - Ett land måste finnas för att ha en FN-representant
 - Det omvända är inte sant



Normalisering

- **Normalisering** är en process som används för att avlägsna fel i en datamodell
- Syftet med normalisering är att minimera redundans (dubbletter) och andra anomalier i en databas.
- Beskriver ett antal **normalformer** som består av en uppsättning regler som beskriver hur en tabellstruktur ska och inte ska utformas.

Från Wikipedia



The image is a screenshot of the Swedish Wikipedia page for 'Normalform (databaser)'. On the left is the Wikipedia logo and a sidebar with navigation links. The main content area has the title 'Normalform (databaser)' with a '[redigera | redigera wikitext]' link. The text explains that normal forms are used in relational databases to prevent anomalies and maintain integrity, listing 1NF through 6NF and BCNF. It mentions that 1NF is the minimum requirement for a database, while higher forms increase strictness. The text is partially cut off at the bottom.

WIKIPEDIA
Den fria encyklopedin

Huvudsida
Introduktion
Deltagarportalen
Bybrunnen
Senaste ändringarna
Slumpartikel (-bot)
Ladda upp filer
Stöd Wikipedia
Kontakta Wikipedia
Hjälp

Normalform (databaser) [\[redigera | redigera wikitext\]](#)

Normalformer är i samband med [relationsdatabaser](#) ett systematiskt sätt att se till att databasstrukturen är lämplig för normala frågedatabaser så att inga oönskade anomalier vid insättning, uppdatering eller borttagning kan ske, och därmed att skydda databasens integritet. De vanligaste är 1NF, 2NF, 3NF och Boyce-Codds normalform (BCNF). Inte lika ofta implementerade är 4NF, 5NF och 6NF. Dessa anger, i ökande grad av strikthet, ett antal krav på databasens utseende. 1NF räcker för skapa en databas, men vid lägre normalform ökar risken för att [anomalier](#) när data uppdateras.

För tabeller som ingår i så kallade stjärnscheman i [informationslager](#)

[https://sv.wikipedia.org/wiki/Normalform_\(databaser\)](https://sv.wikipedia.org/wiki/Normalform_(databaser))

Första normalformen (1NF)

1. Det måste finnas en **primärnyckel** i varje tabell.
2. Varje kolumnvärde måste vara odelbar. (*högst **ett värde** per ruta*).
3. Dessutom får kolumnerna inte upprepas

Id	Namn	Befattning	Färdigheter
A1	Nina Larsson	Programmerare	C#, Java
A2	Bengt Svensson	DBA	MS SQL Server, MySQL
A3	Erik Persson	Programmerare	Java, C++
A4	Camilla Blom	Sekreterare	Office, Kopiatorreparation



1NF: Varje attribut måste vara odelbart

Nu måste vi ta bort återkommande grupper

Id	Namn	Befattning	Färdighet1	Färdighet2
A1	Nina Larsson	Programmerare	C#	Java
A2	Bengt Svensson	DBA	MS SQL Server	MySQL
A3	Erik Persson	Programmerare	Java	C++
A4	Camilla Blom	Sekreterare	Office	Kopiatorreparation

1NF: Högst ett värde per ruta

Nu har vi "atomära" värden men mycket redundans

Id	Namn	Befattning	Färdighet
A1	Nina Larsson	Programmerare	C#
A1	Nina Larsson	Programmerare	Java
A2	Bengt Svensson	DBA	MS SQL Server
A2	Bengt Svensson	DBA	MySQL
A3	Erik Persson	Programmerare	Java
A3	Erik Persson	Programmerare	C++
A4	Camilla Blom	Sekreterare	Office
A4	Camilla Blom	Sekreterare	Kopiatorreparation

1NF: Dela upp tabellen

Id	Namn	Befattning
A1	Nina Larsson	Programmerare
A1	Nina Larsson	Programmerare
A2	Bengt Svensson	DBA
A2	Bengt Svensson	DBA
A3	Erik Persson	Programmerare
A3	Erik Persson	Programmerare
A4	Camilla Blom	Sekreterare
A4	Camilla Blom	Sekreterare

Personal

Personalld	Färdighet
A1	C#
A1	Java
A2	MS SQL Server
A2	MySQL
A3	Java
A3	C++
A4	Office
A4	Kopiatorreparation

Främmande nyckel

Färdigheter

Andra normalformen (2NF)

- Alla attribut (kolumner) beror helt och enbart på primärnyckeln
- Attribut beror på primärnyckeln direkt
 - Inte indirekt via något annat attribut
- Till exempel "Ålder" beror på "Födelsedatum", vilket i sin tur beror på "StudentId" → bryter mot 2NF

2NF – exempel

Sammanfatt primärnyckel

Id	Ort	Namn	Befattning
A1	Stockholm	Nina Larsson	Programmerare
A1	Göteborg	Nina Larsson	Programmerare
A2	Stockholm	Bengt Svensson	DBA
A3	Göteborg	Erik Persson	Programmerare
A4	Stockholm	Camilla Blom	Sekreterare

Alla attribut som inte är en del av primärnyckeln är helt funktionellt beroende av hela primärnyckeln

2NF

Dela upp tabellen

Id	Namn	Befattning
A1	Nina Larsson	Programmerare
A2	Bengt Svensson	DBA
A3	Erik Persson	Programmerare
A4	Camilla Blom	Sekreterare

Sammanfatt

Personalld	Ort
A1	Stockholm
A1	Göteborg
A2	Stockholm
A3	Göteborg
A4	Stockholm

Tredje normalformen (3NF)

- **Definition:**

2NF plus att inget icke-nyckelattribut får vara funktionellt beroende av något annat icke-nyckelattribut.

- Alltså

**Attributen får inte vara beroende av någonting annat än nyckeln.
(Nothing but the key)**

3NF

Attributen får inte vara beroende av något annat än nyckeln



Id	Namn	Befattning	Avdelning	Avdelningschef
A1	Nina Larsson	Programmerare	Utveckling	Mahmud Al Hakim
A2	Bengt Svensson	DBA	Utveckling	Mahmud Al Hakim
A3	Erik Persson	Programmerare	Utveckling	Mahmud Al Hakim
A4	Camilla Blom	Sekreterare	Ekonomi	Johanna Eriksson

3NF – dela upp tabellen

Id	Namn	Befattning	AvdelningId
A1	Nina Larsson	Programmerare	Av1
A2	Bengt Svensson	DBA	Av1
A3	Erik Persson	Programmerare	Av1
A4	Camilla Blom	Sekreterare	Av2

Personal

Id	Avdelning	Avdelningschef
Av1	Utveckling	Mahmud Al Hakim
Av2	Ekonomi	Johanna Eriksson

Avdelning

Boyce-Codd Normalform (BCNF)

- Relationen måste vara i den tredje normalformen...

och

- Alla funktionella beroenden måste ha en supernyckel på den vänstra sidan.

BCNF – exempel

Id	Namn	Befattning	AvdelningId
A1	Nina Larsson	Programmerare	Av1
A2	Bengt Svensson	DBA	Av1
A3	Erik Persson	Programmerare	Av1
A4	Camilla Blom	Sekreterare	Av2
A1	Nina Larsson	DBA	Av1

Nina har bytt roll...
Vi vill inte ta bort
gammal information i
databasen
och vi får inte heller
skapa en ny post!

BCNF – dela upp tabellen

Nu kan vi lagra mer info om personal oberoende av befattning och avdelning

Id	Namn
A1	Nina Larsson
A2	Bengt Svensson
A3	Erik Persson
A4	Camilla Blom

Personal

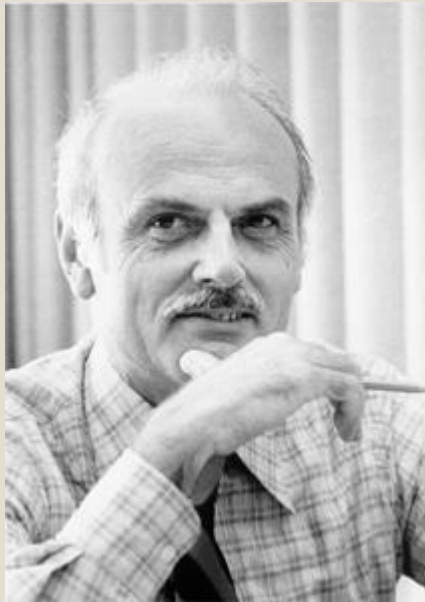
Sammanfatt

Datum	Personal	Befattning	AvdelningId
150101	A1	Programmerare	Av1
150201	A2	DBA	Av1
150315	A3	Programmerare	Av1
150401	A4	Sekreterare	Av2
200901	A1	DBA	Av1

Befattning

Upphovsmännen till relationsdatabaser

Edgar F. Codd (1923 – 2003)



- År 1970 introducerade Codd 1NF och ett år senare 2NF

• Raymond F. Boyce (1946–1974)



- År 1974 definierade Boyce och Codd BCNF

Normalisering

"Each attribute must represent a fact about

The key,

the whole key,

and nothing but the key,

so help me Codd."

Sammanfattning

- Alla relationer knyts ihop av nycklar, som vanligen är ett unikt Id.
- Ingen data ska finnas på med än en plats.
- All data ska vara uppdelad i separata delar
- Målet är att få en databas att vara:
 - Inte innehålla upprepningar
 - Minnessnål
 - Korrekta värden
 - Uppfylla kravspecifikationen
 - Hållas så enkel som möjligt
 - Säker!
 - Regler för datas integritet

Regler för datas integritet

- NOT NULL – ett fält måste ha ett värde
- Referensintegritet – en FK i en tabell måste motsvaras av PK i en annan tabell
 - Det går inte att ta bort post i förälder som har poster i barntabell som pekar på sig
 - Kan inte ta bort order som har orderrader
- Diverse affärsregler
 - Pris måste vara > 0
 - Mötestid måste vara under kontorstid

Data-integritet

- SQL för att styra datas integritet
 - NOT NULL
 - Värdet får inte vara NULL
 - IDENTITY
 - Räknar upp Id med 1 automatiskt
 - PRIMARY KEY
 - Anger att det är en unik identifierare, som inte får vara NULL, och måste vara unik
 - CONSTRAINT, FOREIGN KEY och REFERENCES
 - Anger relationerna mellan tabellerna, och ser till att det inte går att ta bort saker som relaterar till data.
 - UNIQUE
 - En kolumn som inte får innehålla dubletter

NOT NULL

```
CREATE TABLE Cities  
(  
    Id INT IDENTITY,  
    CityName varchar(255) NOT NULL,  
)
```

Detta kommer inte gå att köra:

```
INSERT INTO Cities(CityName) VALUES ('Stockholm'), (NULL)
```

Identity

```
CREATE TABLE [Cities] (  
    [Id] INT IDENTITY,  
    [CityName] varchar(255),  
);
```

Primary Key

```
CREATE TABLE [Cities] (  
    [Id] INT IDENTITY,  
    [CityName] varchar(255),  
    PRIMARY KEY ([Id])  
);
```

Fungerar också:

```
CREATE TABLE [Cities] (  
    [Id] INT IDENTITY PRIMARY KEY,  
    [CityName] varchar(255),  
);
```


CONSTRAINT, FOREIGN KEY och REFERENCES

- En constraint är en regel eller begränsning
- Används för att tala om beroenden och relationer mellan tabeller
- Syntax:

```
CONSTRAINT [Eget påhittat namn]
    FOREIGN KEY ([Kolumnen I tabellen som anger relationen])
    REFERENCES [Den andra tabellen]([Den andra tabellens kolumn])
```

- Exempel

```
CREATE TABLE [ParkingHouses] (
    [Id] INT IDENTITY,
    [HouseName] Varchar(255),
    [CityId] INT,
    PRIMARY KEY ([Id]),
    CONSTRAINT [FK_ParkingHouses.CityId]
        FOREIGN KEY ([CityId])
        REFERENCES [Cities]([Id])
);
```

UNIQUE

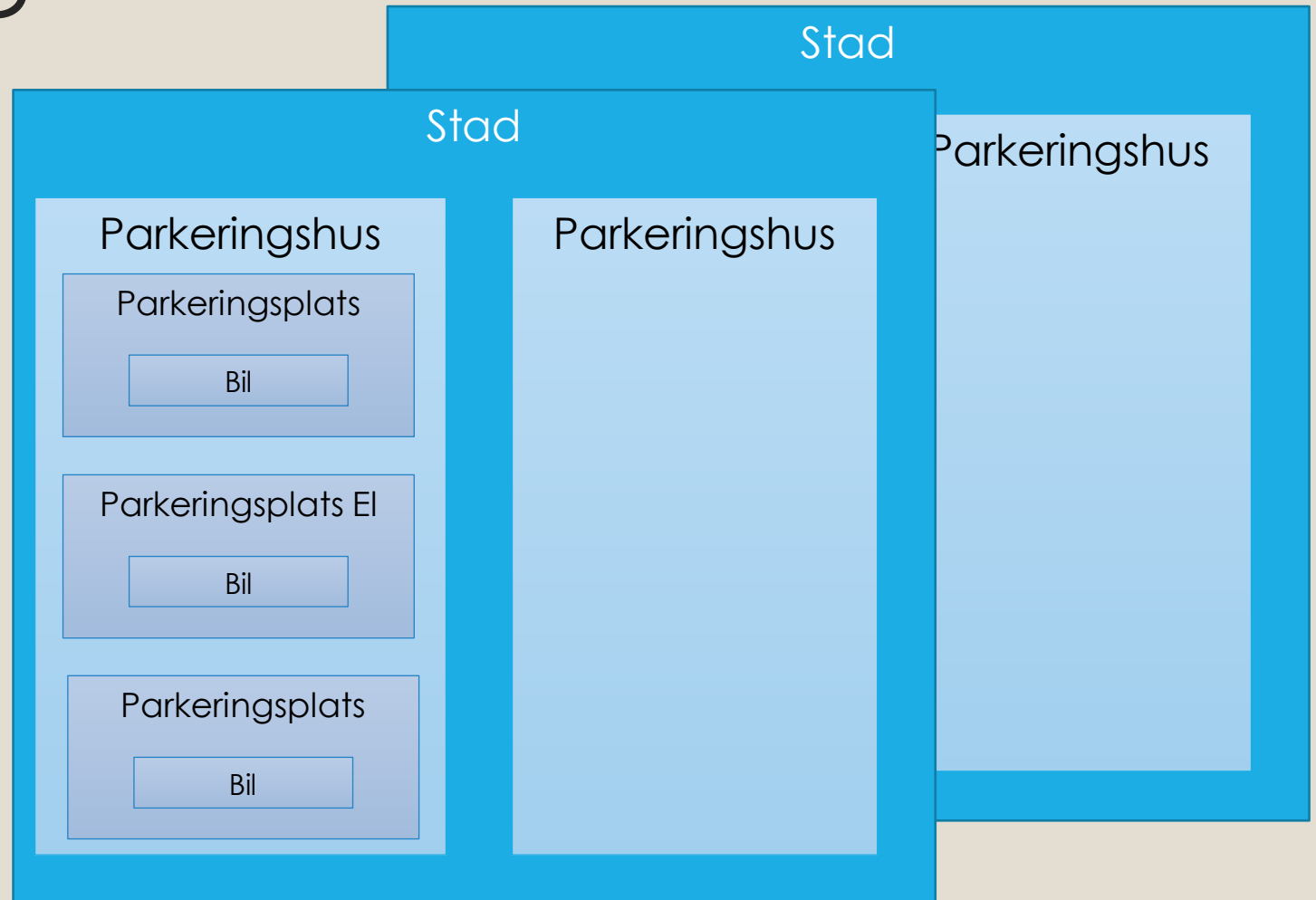
- Ibland vill man att värden i en kolumn INTE har dubletter.
- Exempel på data som borde vara unika:
 - Personnummer
 - Registreringsskyltar
 - Hästnamn på ATG
- Exempel på saker som borde vara unika, men inte är det:
 - Författarnamn
 - Filmtitlar
- Exempel

`CREATE TABLE Cars`

```
(  
    Id INT IDENTITY,  
    Plate varchar(255) UNIQUE,  
    Make varchar(255)  
)
```

Code along

- Deluxe parkering
 - Fyra tabeller:
 - Städer
 - Parkeringshus
 - Parkeringsplatser
 - Bilar
 - Fil: DeluxeParkeringSQL.sql



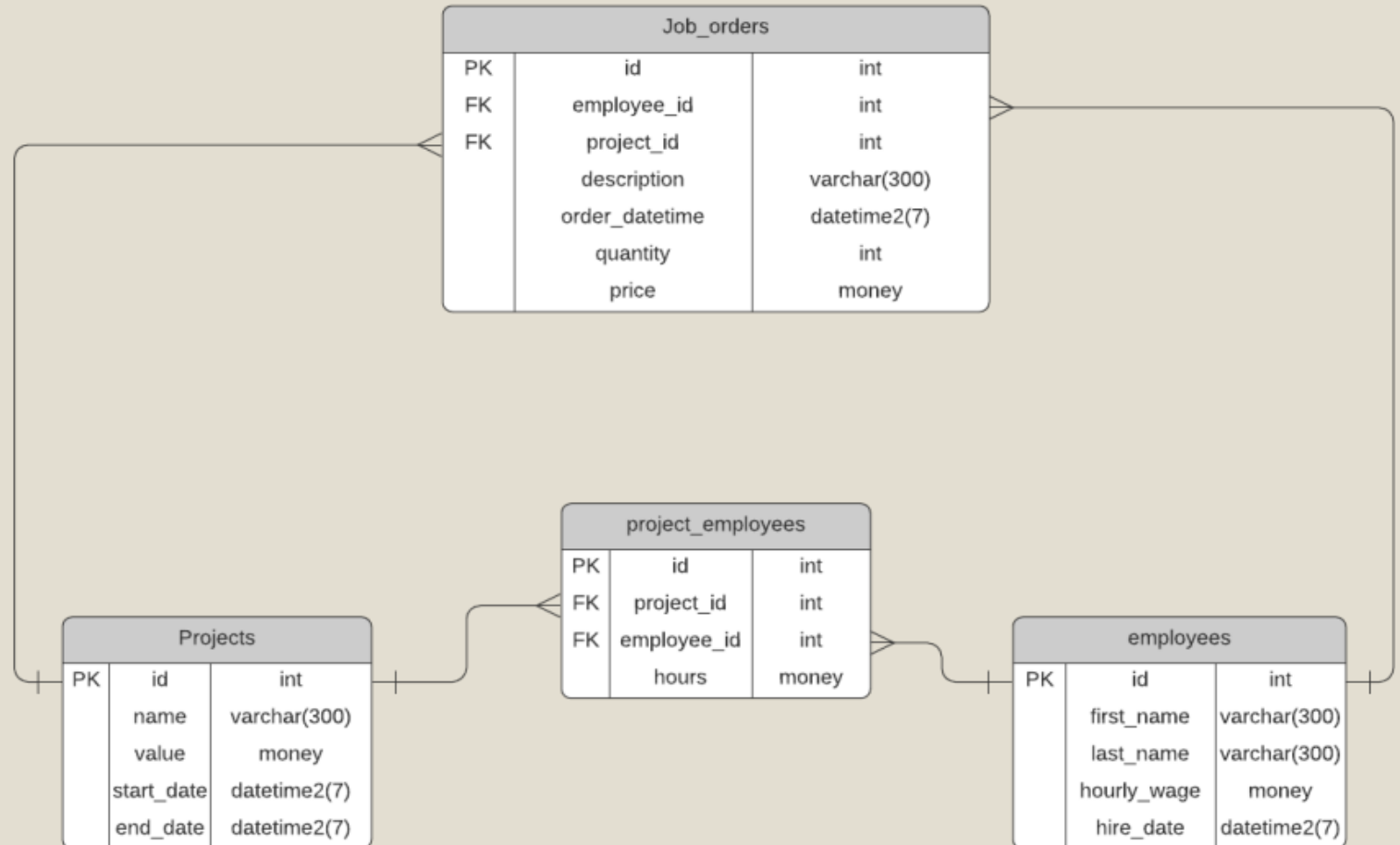
Modellering med Lucidchart

Vi använder Lucidchart ER Diagram

Codey's Construction

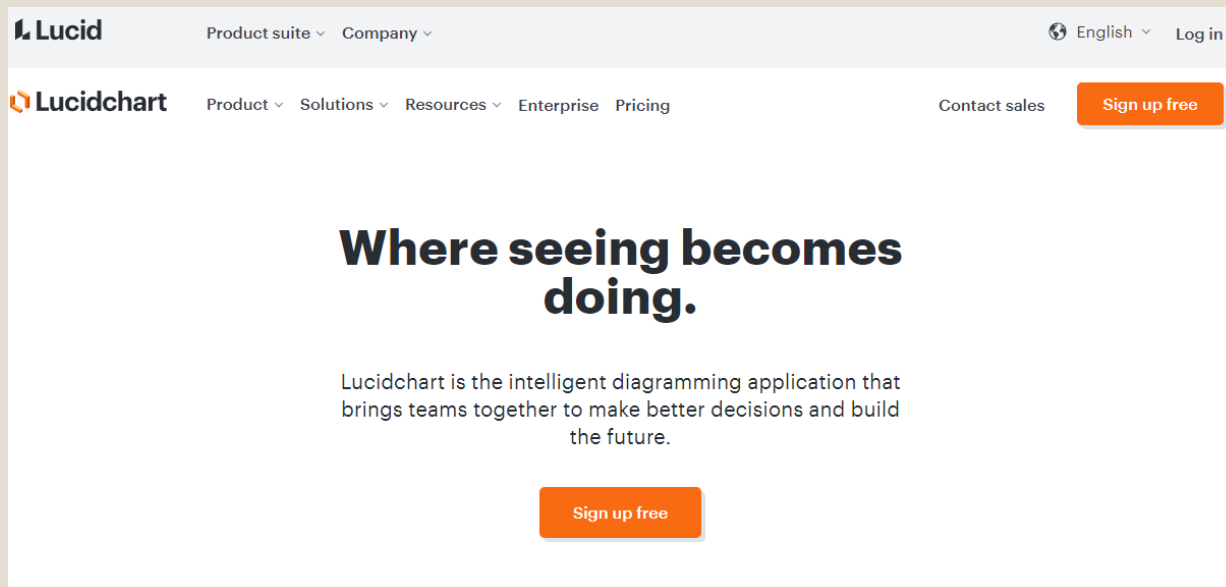
Claes Engelin | January 19, 2022

Customers		
name		varchar(255)
industry		Varchar(255)
project1_id		int
project1_feedback		text
project2_id		int
project2_feedback		text
contact_person_id		int
contact_person_and_role		varchar(300)
phone_number		varchar(12)
address		varchar(255)
city		varchar(255)
postal_code		varchar(10)

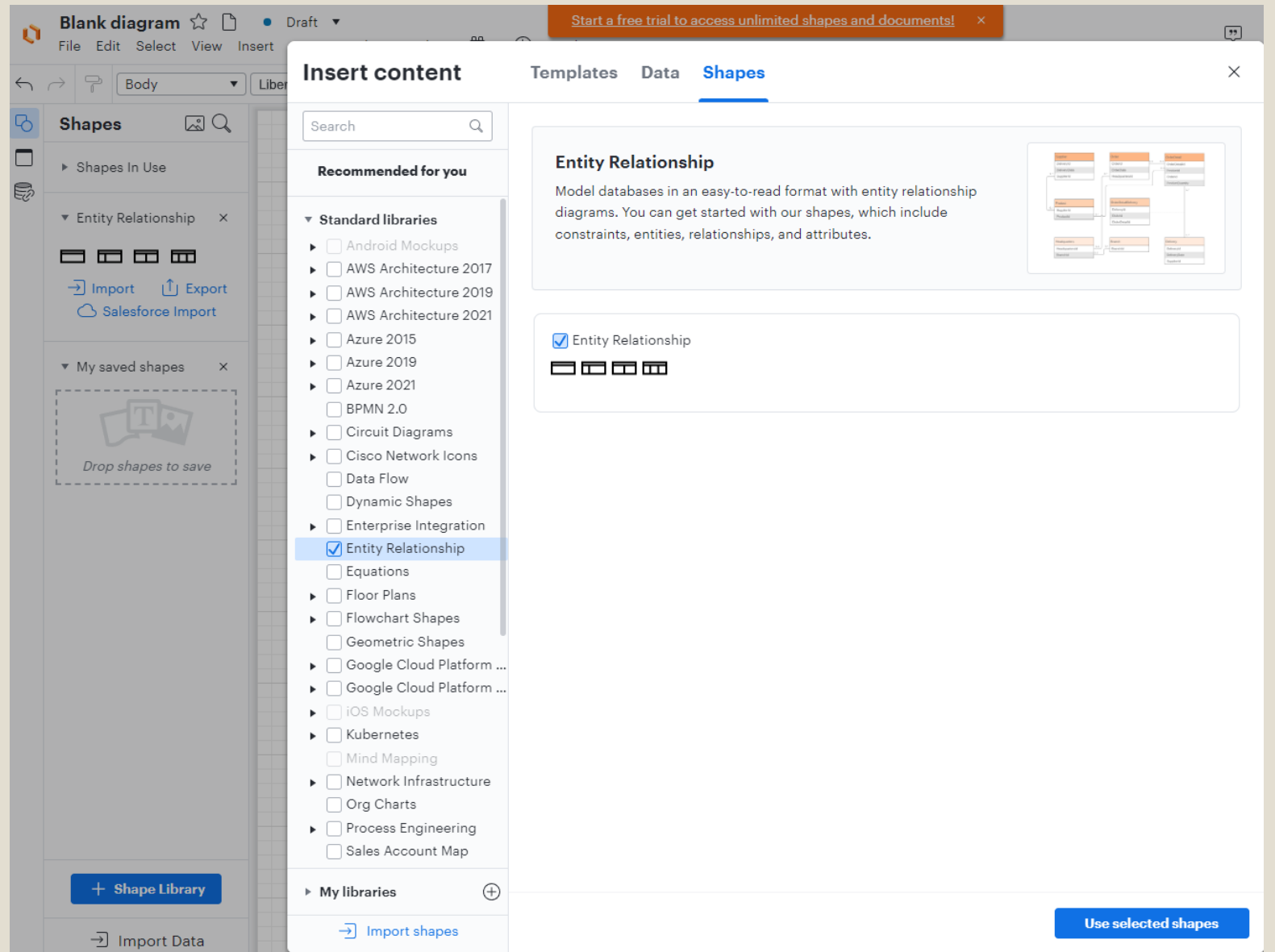


Lucidchart

- Gratis, men bara tre samtidiga document kan redigeras
- Undvik Free trial, ger saker som sedermera försvinner, när tiden gått ut.
- Skapa konto på <https://www.lucidchart.com/>



Välj entity relationship



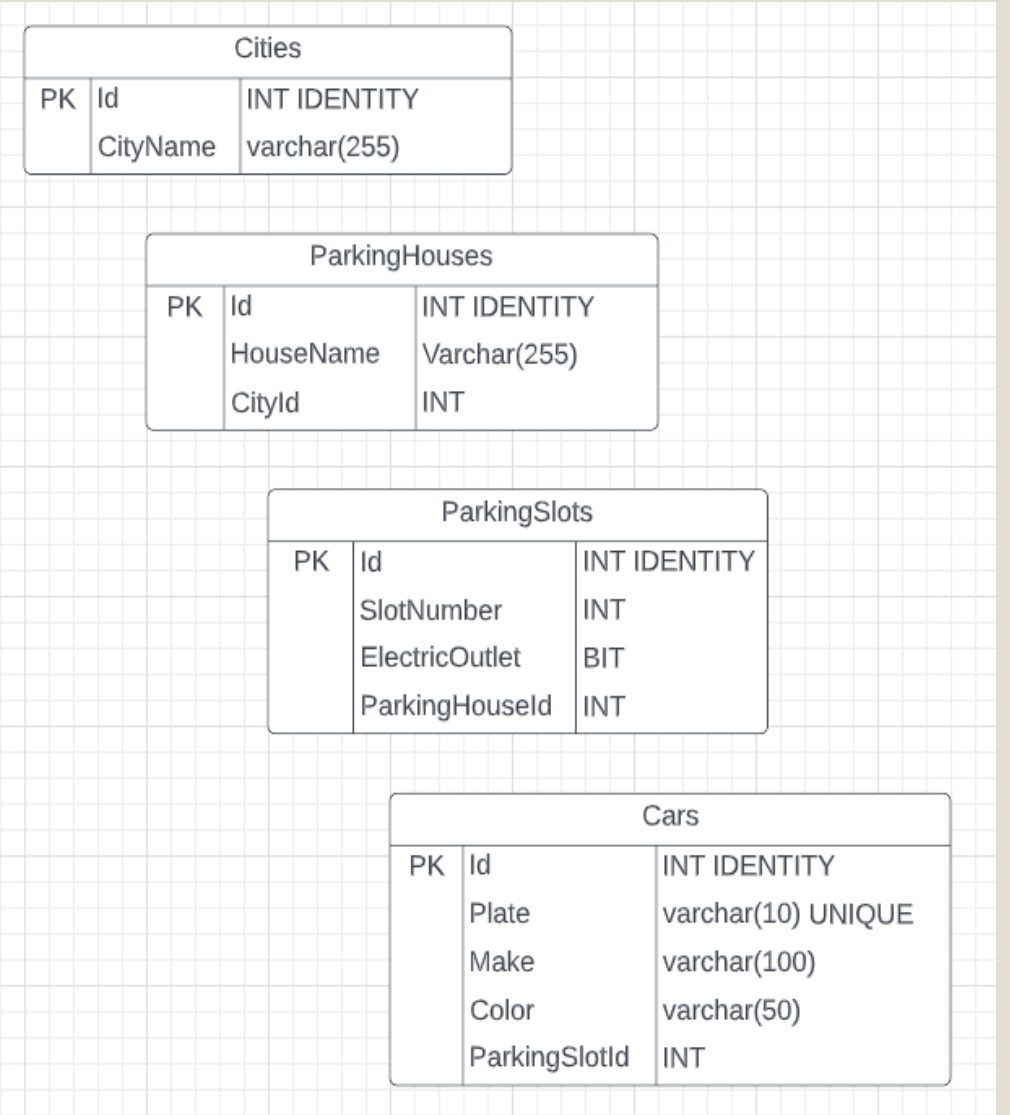
```
CREATE TABLE [Cities] (
    [Id] INT IDENTITY,
    [CityName] varchar(255),
    PRIMARY KEY ([Id])
);
```

```
CREATE TABLE [ParkingHouses] (
    [Id] INT IDENTITY,
    [HouseName] Varchar(255),
    [CityId] INT,
    PRIMARY KEY ([Id])
);
```

```
CREATE TABLE [ParkingSlots] (
    [Id] INT IDENTITY,
    [SlotNumber] INT,
    [ElectricOutlet] BIT,
```

```
    [ParkingHouseId] INT,
    PRIMARY KEY ([Id])
);
```

```
CREATE TABLE [Cars] (
    [Id] INT IDENTITY,
    [Plate] varchar(10) UNIQUE,
    [Make] varchar(100),
    [Color] varchar(50),
    [ParkingSlotId] INT,
    PRIMARY KEY ([Id])
);
```




```

CREATE TABLE [Cities] (
    [Id] INT IDENTITY,
    [CityName] varchar(255),
    PRIMARY KEY ([Id])
);

CREATE TABLE [ParkingHouses] (
    [Id] INT IDENTITY,
    [HouseName] Varchar(255),
    [CityId] INT,
    PRIMARY KEY ([Id]),
    CONSTRAINT [FK_ParkingHouses.CityId]
        FOREIGN KEY ([CityId])
            REFERENCES [Cities]([Id])
);

CREATE TABLE [ParkingSlots] (
    [Id] INT IDENTITY,
    [SlotNumber] INT,
    [ElectricOutlet] BIT,

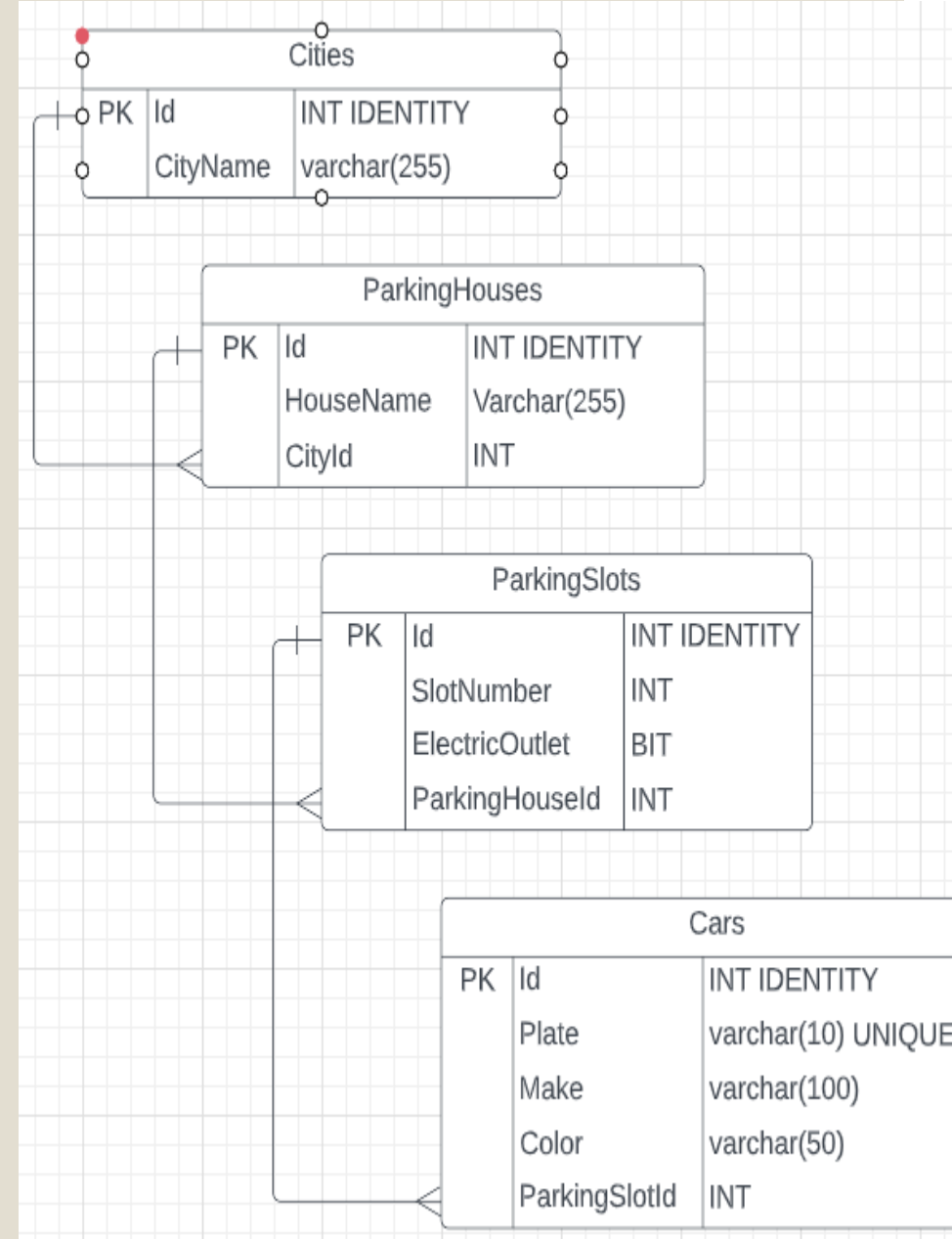
```

```

    [ParkingHouseId] INT,
    PRIMARY KEY ([Id]),
    CONSTRAINT
        [FK_ParkingSlots.ParkingHouseId]
            FOREIGN KEY ([ParkingHouseId])
                REFERENCES [ParkingHouses]([Id])
);

CREATE TABLE [Cars] (
    [Id] INT IDENTITY,
    [Plate] varchar(10) UNIQUE,
    [Make] varchar(100),
    [Color] varchar(50),
    [ParkingSlotId] INT,
    PRIMARY KEY ([Id]),
    CONSTRAINT [FK_Cars.ParkingSlotId]
        FOREIGN KEY ([ParkingSlotId])
            REFERENCES [ParkingSlots]([Id])
);

```



Demo – Lucidchart

- <https://www.lucidchart.com/>

Gruppövning – Bygg bokhandel

- Vi ska bygga en databas för en bokhandel:
 - Du får i uppgift att designa den i SQL-server med nycklar, relationer, integritetsvillkor och ett ER-diagram. Nedan finns några minimikrav på tabellerna:
- Tabell: **Författare** I tabellen författare vill vi ha en "Identitetskolumn" (identity) kallad Id som PK. Därutöver vill vi ha kolumnerna: Förnamn, Efternamn och Födelsedatum i passande datatyper.
- Tabell: **Böcker** I tabellen böcker vill vi ha Id som primärnyckel med lämpliga constraints. Utöver det vill vi ha kolumnerna: Titel, ISBN, Språk, Pris, och Utgivningsdatum av passande datatyper. Sist vill vi ha en FK-kolumn "FörfattareID" som pekar mot tabellen "Författare". Utöver dessa kolumner får du gärna lägga till egna med information som du tycker kan vara bra att lagra om varje bok
- Tabell: **Butiker** Utöver ett identity-ID så behöver tabellen kolumner för att lagra butiksnamn samt adressuppgifter
- Tabell: **LagerSaldo** I denna tabell vill vi ha 3 kolumner: ButikID som kopplas mot Butiker, BokID som kopplas mot böcker, samt Antal som säger hur många exemplar det finns av en given bok i en viss butik.

Lite förutsättningar

- Ingen information om författare, bok eller butik får finnas på fler ställen
- En författare kan ha skrivit flera böcker
- En bok kan finnas i flera butiker, i olika antal
- Skriv SQL-frågor för att testa detta:
 - Det ska inte gå att ta bort en författare, som har en bok.
 - Det ska inte gå att ta bort en butik som har böcker
 - Det ska inte gå att lägga till en bok med en författare som inte finns inlagd.
- Testdata: För test och demonstration vill vi ha minst 3 butiker, 4 författare, 10 boktitlar med tillhörande saldo (alla böcker ska finnas i alla butiker, men i olika antal).
- Minst **tre** frågor, som hämtar data från flera tabeller
 - Exempel:
 - Hur stort lagersaldo har varje författare, per bok?
 - Hur många böcker har varje författare skrivit?
 - Hur mycket pengar skulle det kosta om man köpte en författares alla böcker, i alla butiker?
 - Vilken bok finns det flest av, i varje butik?

Mer förutsättningar

- Databasen ska ha ett ER-diagram.
 - Du väljer själv om du vill börja skapa databasen i Lucidchart, och sedan använda den (med eventuella justeringar) i SSMS eller om du först skapar tabellerna i SSMS, och sedan försöker göra ER-diagramet så det överensstämmer
- Kort redovisning på måndag
- Jobba i grupperna.
- Titta igenom tidigare material, för att repetera.

Länkar

- [En webbkurs om databaser \(databasteknik.se\)](http://databasteknik.se)
- [SQL PRIMARY KEY Constraint \(w3schools.com\)](http://w3schools.com)
- [SQL FOREIGN KEY Constraint \(w3schools.com\)](http://w3schools.com)
- <https://www.lucidchart.com/>
- Kolla videon på: [ER Diagram \(ERD\) - Definition & Overview | Lucidchart](https://www.youtube.com/watch?v=UgT81311880)
- [MicroNugget: How to Normalize Databases - YouTube](https://www.youtube.com/watch?v=UgT81311880)
- [What is Database Normalization in SQL Server? \(sqlshack.com\)](http://sqlshack.com)