

# Databaser

Datamodellering

*[christoffer.wallenberg@zocom.se](mailto:christoffer.wallenberg@zocom.se)*

**ZoCom**

# **Datamodellering**

A **data model** is an *abstract model* that organizes elements of data and *standardizes* how they relate to one another and to properties of the *real world entities*.

*Wikipedia*

**Utmaningen**

**Verkligheten är komplex!**

Verkligheten



Modell

## Ex. Bil

- Vilka *egenskaper* karaktäriserar en bil?
- Vilka *egenskaper* har alla bilar?
- Hur kategoriserar vi de egenskaper som skiljer bilar åt?

# Bil



En bil



En bil...?



Ehhh...?

# Bil-objekt

```
let car = {  
  wheels: 4,  
  fuel: "electric",  
  steeringwheel: true // behövs?  
}
```



# Övning

Gör ett **cykel-objekt** i JS.

- Vilka *egenskaper* karaktäriserar en cykel?
- Vilka *egenskaper* har alla cyklar?
- Hur kategorisera de egenskaper som skiljer cyklar åt?

# I vilken form kommer data?

- Strings
- Numbers
- Objects
- Arrays
- Booleans
- Binary ( filer, bilder etc )

# Vad är en databas?

- Är en samling av organiserad data
- Går att spara, ta bort, hämta alla data, skapa nytt
- Används för att spara data som ska kunna användas igen

# Databastyper

- **SQL databaser** ( *relationella* )  
ex. mySql, mariaDb, sqLite
- **noSQL databaser** ( *document* )  
ex. MongoDB, CouchDb, Firestore
- **Graph databaser** ( *nodes* )  
Neo4j, arrangoDb, Azure Cosmos

# Relationsdatabas

- Ex. PostgreSQL
- Består av tabeller för att spara data (Tänk Excel)
- Använder SQL som språk för att söka i databasen
- Sparar data i ett tabellformat
- Funkar bra när alla data ska ha samma struktur

# Relationsdatabaser

```
SELECT Student.s_id, Student.s_name, MARKS.score, MARKS.status  
FROM Student  
INNER JOIN MARKS ON Student.s_id=MARKS.s_id;
```

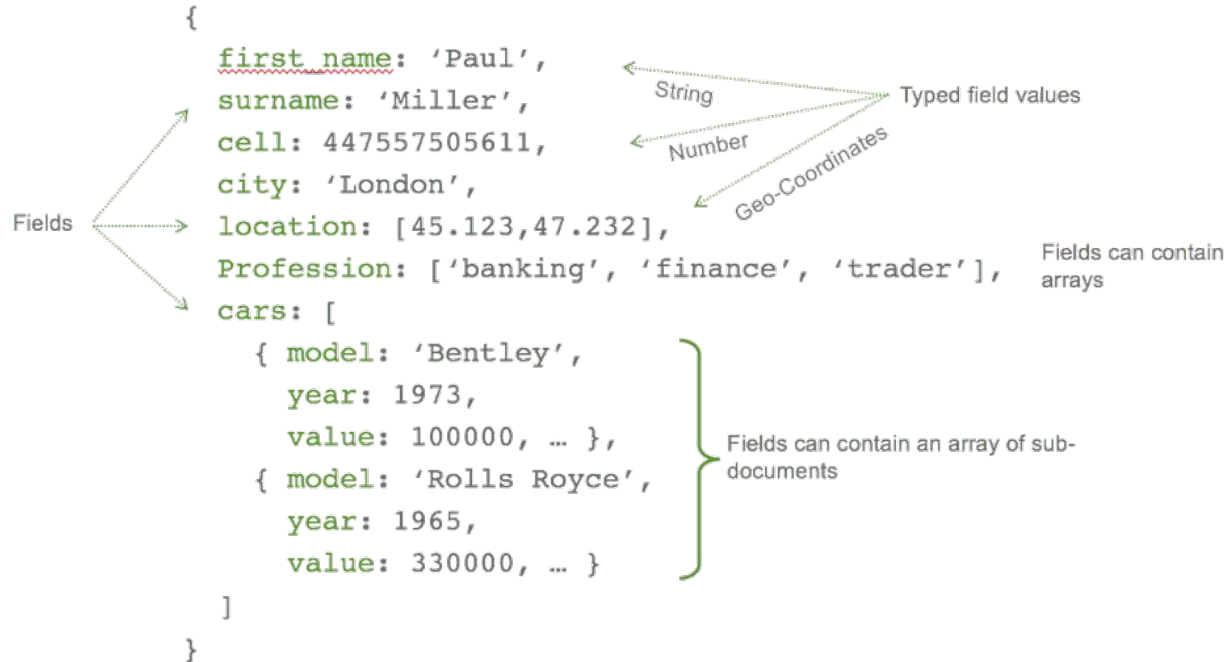
Student		
	s_id	s_name
1	1	Jack
2	2	Rithvik
3	3	Jaspreet
4	4	Praveen
5	5	Bisa
6	6	Suraj

MARKS				
	school_id	s_id	score	status
1	1004	1	23	fail
2	1008	6	95	pass
3	1012	2	97	pass
4	1016	7	67	pass
5	1020	3	100	pass
6	1025	8	73	pass
7	1030	4	88	pass
8	1035	9	13	fail
9	1040	5	16	fail
10	1050	10	53	pass

# noSQL-databas

- Ex. MongoDB
- Är dokument-driven och sparar i annat format än tabeller. Exempelvis JSON
- noSQL-databas är allt som inte använder SQL som frågespråk
- Sparar i ett ostrukturerat format
- Funkar bra när all data har en varierande struktur

# Dokumentdatabaser





# Men...hur ska man tänka då?

- Utgå ifrån gränssnittet!  
*Vilken data behövs?*
- Behövs “känslig” data?  
*Ex: lösenord*
- Tänk på objekten i “verkligheten”  
*Vilka egenskaper har ex. en tröja?*

# Lowdb

How low can you go?


- Liten JSON-databas för mindre projekt
- Sparar allt i en JSON-fil så det behövs ingen databasserver
- Använder sig av promise, yay!
- Installeras via **npm**

# Lowdb

Låt oss kika på lite exempel

# Skapa en ny databas

```
database.defaults({ insults: [], count: 0 }).write();
```



Skriver till  
databasen

Skapar en ny databas med som innehåller två egenskaper: **insults** och **count**.

# Hämta från databasen

```
database.get('insults');
```

```
database.read();
```

**get()** hämtar allt som ligger i **insults**.

**read()** hämtar allt innehåll i databasen.

# Skriva till databasen

```
database.get('insults')  
    .push({ insult: insult, play: play })  
    .write();
```

Lägger till ett objekt i arrayen **insults**

# Söka i databasen

```
database.get('insults')  
    .find({'play': 'Rickard III'});
```

Letar i **insults** efter en pjäs som heter Rickard III och returnerar **första** träffen

# Söka i databasen

```
database.get('insults')  
    .filter({'play': 'Rickard III'});
```

Letar i **insults** efter en pjäs som heter Rickard III och returnerar **alla** träffar



# Uppdatera

```
database.get('insults')  
    .find({ 'play': 'Rickard III'})  
    .assign({ 'play': 'Julius Ceasar'})  
    .write();
```

Letar i **insults** efter en pjäs som heter Rickard III och skriver över den med **Julius Ceasar** istället.

A person with a mustache and glasses, wearing a green t-shirt, is holding a vintage computer keyboard. They are sitting at a desk with two computer monitors in the background. The entire image is overlaid with a semi-transparent green filter. The text "Lets code!" is written in large white letters, and "With lowdb" is written in smaller white letters below it.

# Lets code!

With lowdb