

Veckans Agenda V.20

Del 4, sista delen av fullstack kursen:

Fullstack-utveckling Backend och Frontend med React, SQL och express

15 Maj > 8 Juni

Måndag:

Säg hej till Oskar

- Dokumentation github
- Kort sammanfattning SQL
- Postgres (postgresql) , genomgång > öva

Tisdag:

- Repetering, vad gjorde vi igår
- postgres, react och express (PERN), genomgång > Richard visar

Onsdag:

- Richard&Betty Drop-in, studieplanering-rester (undvik sommarstudier)

Torsdag:

- Självstudier, rekommenderat är att sätta ihop en enkel PERN lösning inkl formulär som kan "postas" in till databas

Fredag:

- Ej handledning pga klämdag, extra handledning blir på Onsdag 13-15



Github Dokumentation

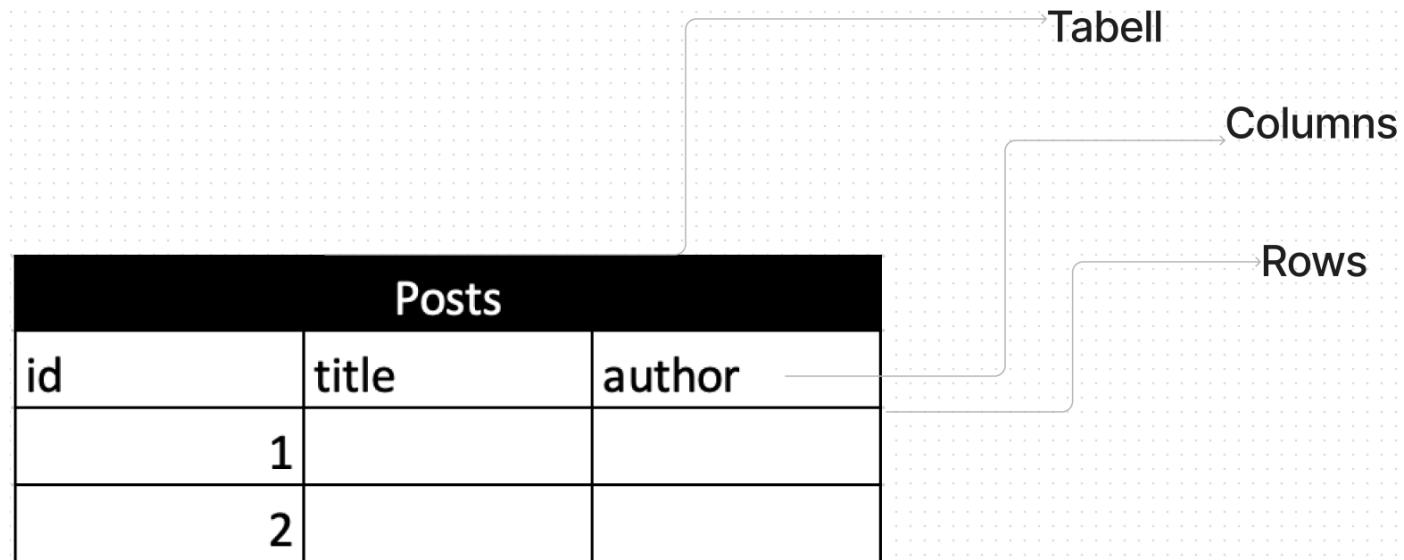
<https://github.com/RichardITHS/2023-W20>



IT-HÖGSKOLAN

Här startar din IT-karriär.

Lite SQL repetition



The diagram illustrates a SQL table structure. A table with three columns (id, title, author) and three rows (header, row 1, row 2) is shown. Labels with arrows point to the table, columns, and rows.

Posts		
id	title	author
1		
2		

Labels and arrows:

- Tabell**: Points to the entire table structure.
- Columns**: Points to the header row (id, title, author).
- Rows**: Points to the data rows (1, 2).

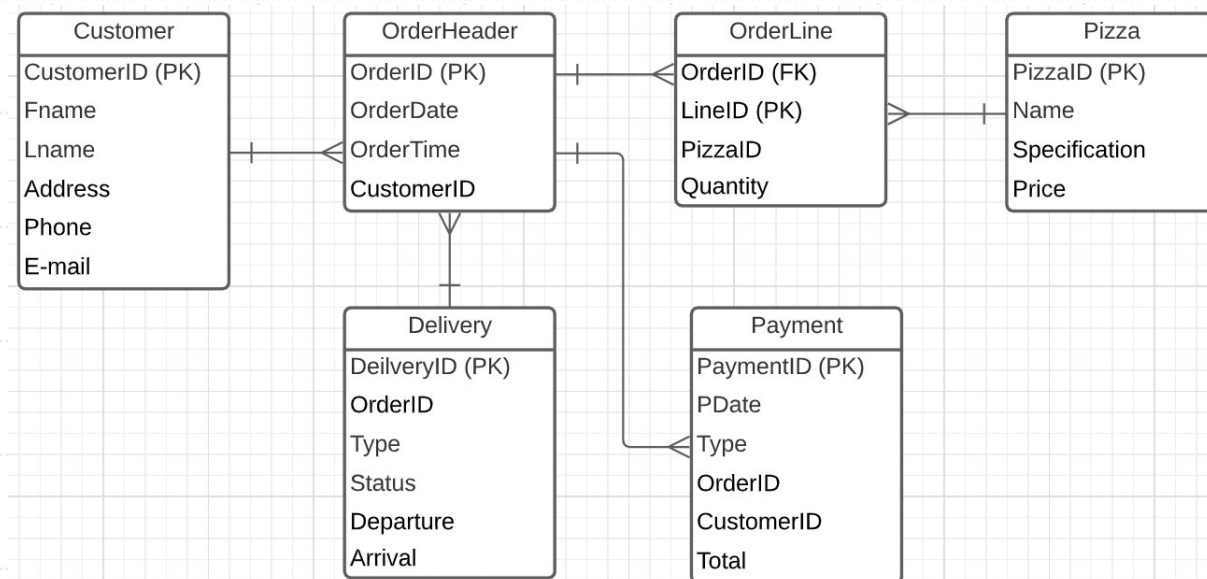


IT-HÖGSKOLAN

Här startar din IT-karriär.

Lite SQL repetition

Relation med flera tabeller, exempel



Lite SQL repetition- Datatyper

String datatypes:

- VARCHAR(size), MySQL
- Character varying, postgres
- Text

Number datatypes:

- Int (-2147483648 to 2147483647)
- Integer (samma som ovan)
- BigInt (-9223372036854775808 to 9223372036854775807)

Date and time datatypes:

- Date (YYYY-MM-DD)
- TIMESTAMP (YYYY-MM-DD hh:mm:ss)
- YEAR (1901 to 2155, znd 0000)

TRUE OF FALSE , Boolean

Se här för postgres datatyper:

<https://www.postgresqltutorial.com/postgresql-tutorial/postgresql-data-types/>



IT-HÖGSKOLAN

Här startar din IT-karriär.

Lite SQL repetition - Begränsningar

PRIMARY KEY (primärnycklar) och **AUTOINCREMENT** (automatisk generering av numeriska värden)

Varje rad i en tabell kan ha (och bör ofta ha) ett unikt värde som identifierar raden.

Värden som unikt identifierar rader kallas för primärnycklar.

En kolumn kan göras till en primärnyckel via nyckelordet PRIMARY KEY.

I postgres är "serial" motsvarigheten till AUTOINCREMENT, fyller samma funktion

UNIQUE (unika värden som inte är primärnycklar)

PRIMARY KEY-kolumner måste vara unika, men ibland vill man ha ytterligare kolumner som är unika.

Till exempel skulle name-kolumnen i cities-tabellen kunna vara unik.

Ett annat exempel skulle kunna vara en users-tabell innehållandes användare, där varje e-postadress (värden i en email-kolumn) bara får förekomma en gång.

NOT NULL (obligatoriska värden)

Celler kan som utgångspunkt sakna värden, vilket i SQL innebär att de innehåller NULL

DEFAULT (standardvärden)

Om en kolumn önskas ha ett standardvärde kan nyckelordet DEFAULT användas

FOREIGN KEY (sekundärnycklar)

Foreign keys (sekundärnycklar) är en SQL-funktionalitet som kan koppla ihop tabeller via primary keys (primärnycklar)



Lite SQL repetition - Begränsningar

PRIMARY KEY (primärnycklar) och **AUTOINCREMENT** (automatisk generering av numeriska värden)

Varje rad i en tabell kan ha (och bör ofta ha) ett unikt värde som identifierar raden.

Värden som unikt identifierar rader kallas för primärnycklar.

En kolumn kan göras till en primärnyckel via nyckelordet PRIMARY KEY.

I postgres är "serial" motsvarigheten till AUTOINCREMENT, fyller samma funktion

UNIQUE (unika värden som inte är primärnycklar)

PRIMARY KEY-kolumner måste vara unika, men ibland vill man ha ytterligare kolumner som är unika.

Till exempel skulle name-kolumnen i cities-tabellen kunna vara unik.

Ett annat exempel skulle kunna vara en users-tabell innehållandes användare, där varje e-postadress (värden i en email-kolumn) bara får förekomma en gång.

NOT NULL (obligatoriska värden)

Celler kan som utgångspunkt sakna värden, vilket i SQL innebär att de innehåller NULL

DEFAULT (standardvärden)

Om en kolumn önskas ha ett standardvärde kan nyckelordet DEFAULT användas

FOREIGN KEY (sekundärnycklar)

Foreign keys (sekundärnycklar) är en SQL-funktionalitet som kan koppla ihop tabeller via primary keys (primärnycklar)



Lite SQL repetition - Begränsningar

Kort sammanfattning:

- PRIMARY KEY används för värden som unikt identifierar rader
- FOREIGN KEY låter primärnycklar “koppla ihop” tabeller
- Foreign key-funktionaliteten behöver aktiveras för att fungera som den ska
- AUTOINCREMENT (serial i postgres) och DEFAULT kan användas för att sätta värden automatiskt
- UNIQUE och NOT NULL möjliggör ytterligare värdebegränsningar
- IS NULL behöver för att hitta NULL-värden



IT-HÖGSKOLAN

Här startar din IT-karriär.

Lite SQL repetition – SQL Kommandon

Vanligaste kommandon vi använder i SQL:

- SELECT, hämtar data från en eller flera tabeller
- UPDATE, Ändra befintlig data
- INSERT INTO, Sätta in ny data i en tabell
- DELETE FROM, Ta bort data från tabell



IT-HÖGSKOLAN

Här startar din IT-karriär.

Lite SQL repetition – SQL Kommandon

CREATE TABLE

När vi skapar en tabell ska vi ange tabellnamn, vilka kolumner vi ska ha samt vilka datatyper de har

```
CREATE TABLE Persons (  
  id SERIAL PRIMARY KEY,  
  FirstName varchar(255),  
  LastName varchar(255),  
  Address varchar(255),  
  City varchar(255)  
);
```



IT-HÖGSKOLAN

Här startar din IT-karriär.

Lite SQL repetition – SQL Kommandon

INSERT INTO

När vi ska sätta in data behöver givetvis ange vilken tabell och vilka värden

Sätta in data:

```
INSERT INTO Persons (FirstName, LastName, Address, City)  
VALUES ('Smiths', 'Janes', '456 Elm St', 'Shelbyville');
```



IT-HÖGSKOLAN

Här startar din IT-karriär.

Lite SQL repetition – SQL Kommandon

UPDATE

När vi ska uppdatera data behöver vi ange id

Uppdatera data:

UPDATE persons

SET FirstName = 'Richard'

WHERE id = 2



IT-HÖGSKOLAN

Här startar din IT-karriär.

Lite SQL repetition – SQL Kommandon

DELETE

När vi ska ta bort data behöver vi ange id

Ta bort data:
DELETE FROM persons
WHERE id = 2



IT-HÖGSKOLAN

Här startar din IT-karriär.

Lite SQL repetition – SQL Kommandon

DELETE TABLE

Om vi önskar ta bort en hel tabell använder vi DROP

```
DROP TABLE IF EXISTS persons;
```



IT-HÖGSKOLAN

Här startar din IT-karriär.

Lite SQL repetition – SQL Kommandon

SELECT

För att kunna hämta ut och arbeta med data behöver man känna till vilka Kolumner och tabeller som finns

- SELECT, hämtar data från en eller flera tabeller
- FROM, Från vilka tabeller
- WHERE, Begränsa data man vill hämta
- GROUP BY, Välja ut specifik data från olika kolumner och gruppera
- ORDER BY, Sortera

Hämta data:

```
SELECT * FROM public.persons  
ORDER BY id ASC
```



IT-HÖGSKOLAN

Här startar din IT-karriär.

Vad är Postgres(SQL)

- PostgreSQL är ett avancerat relationsdatabassystem i företagsklass med öppen källkod och stöder både SQL (relationella) och JSON (icke-relationella) frågor.
- PostgreSQL är en mycket stabil databas som backas upp av mer än 20 års utveckling av open source-gemenskapen.
- PostgreSQL används som en primär databas för många webbapplikationer såväl som mobil- och analysapplikationer.
- PostgreSQL:s community uttalar PostgreSQL som /'poustgres ,kju: 'el/.



Historia

- PostgreSQL-projektet startade 1986 vid Berkeley Computer Science Department, University of California.
- Projektet hette ursprungligen POSTGRES, med hänvisning till den äldre Ingres-databasen som också utvecklades i Berkeley. Målet med POSTGRES-projektet var att lägga till de minimala funktioner som behövs för att stödja flera datatyper.
- 1996 döptes POSTGRES-projektet om till PostgreSQL för att tydligt illustrera dess stöd för SQL. Idag förkortas PostgreSQL vanligtvis som Postgres.
- Sedan dess har PostgreSQL Global Development Group, en dedikerad gemenskap av bidragsgivare, fortsatt att släppa det fria databasprojektet med öppen källkod.
- Ursprungligen designades PostgreSQL för att köras på UNIX-liknande plattformar. Och sedan utvecklades PostgreSQL och kördes på olika plattformar som Windows, macOS och Solaris.



Vanliga användningsfall av PostgreSQL

- 1) En robust databas i LAPP-stacken
 - LAPP står för Linux, Apache, PostgreSQL och PHP (eller Python och Perl). PostgreSQL används främst som en robust back-end-databas som driver många dynamiska webbplatser och webbapplikationer.
- 2) Transaktionsdatabas för allmänna ändamål
 - Både stora företag och nystartade företag använder PostgreSQL som primära databaser för att stödja sina applikationer och produkter.
- 3) Geospatial databas
 - PostgreSQL med PostGIS-tillägget stöder geospatiala databaser för geografiska informationssystem (GIS).



Många populära språk stöds

- Python
- Java
- C#
- C/C+
- Ruby
- JavaScript (Node.js)
- Perl
- Go
- Tcl
- PHP



IT-HÖGSKOLAN

Här startar din IT-karriär.



Nu...Installation av postgresql och pgAdmin

<https://github.com/RichardITHS/2023-W20/blob/main/W20-Postgres.md>



IT-HÖGSKOLAN

Här startar din IT-karriär.



Back-End Node.js-Express-postgres



IT-HÖGSKOLAN

Här startar din IT-karriär.

Först och främst...

Nu ska arbeta med 2 "mappar" i samma projekt

Backend
Frontend

Men vi börjar med backend..



IT-HÖGSKOLAN

Här startar din IT-karriär.

Paket Node.js för back-end

- **Express**

npm i express

(Ramverk i Node för att skapa en server, kan lägga till rutter samt mellanprogram)

- **Body-Parser**

npm i body-parser

(Body-parser är en middleware-modul för Express.js som används för att hantera HTTP POST-förfrågningar. Body-parser extraherar hela bodyn av en inkommande förfrågan och gör den tillgänglig på req.body)

- **Cors**

npm i cors

(CORS står för Cross Origin Resource Sharing och är en HTTP-funktion som gör det möjligt för ett webbprogram som körs i en domän att komma åt resurser i en annan domän)

Detta är nödvändigt eftersom webbläsare implementerar en säkerhetsbegränsning som kallas origin principen som förhindrar att en webbsida anropar API:er i en annan domän)

- **Dotenv**

npm i dotenv

(Dotenv är en modul som används för att läsa in nyckel-värde-par från en .env-fil och kan ställa in dem som miljövariabler)

- **Pg**

npm i pg

(Paket för att kunna hantera och kommunicera med postgres)

- **Nodemon**

npm i nodemon --save-dev (glöm ej att lägga till startskriptet i package.json, bör ni kunna nu 😊)

(Nodemon är ett verktyg som hjälper till att utveckla Node.js-baserade applikationer genom att automatiskt starta om node-applikationen när filändringar i katalogen upptäcks)



IT-HÖGSKOLAN

Här startar din IT-karriär.



Kom ihåg



IT-HÖGSKOLAN

Här startar din IT-karriär.

ENV

Skapa en .env-fil i backend-mappen med följande innehåll:

HOST=localhost

USER=postgres

PASSWORD=

DATABASE=postgres (eller vad ni döpt databasen till)

PORT=5432

(PASSWORD vara lösenordet som sattes under installationen av PostgreSQL.)



IT-HÖGSKOLAN

Här startar din IT-karriär.

.gitignore

Vi vill ej ha med node modules eller känsliga inloggningsuppgifter på github.
Lägg därför till en .gitignore fil i backend mappen och skriv:

```
.env  
node_modules
```



IT-HÖGSKOLAN

Här startar din IT-karriär.



Learning by doing

<https://github.com/RichardITHS/2023-W20/blob/main/W20-Postgres.md>

Jag visar först 😊



IT-HÖGSKOLAN

Här startar din IT-karriär.



Front-End React-axios



IT-HÖGSKOLAN

Här startar din IT-karriär.



**Kom ihåg, nu ska ni arbeta med
frontend i en annan mapp..
Ha koll på terminalen**



IT-HÖGSKOLAN

Här startar din IT-karriär.

Grundläggande rek paket för front-end

- **Vite med React**

npm init vite@latest .

(Välj att installera med React utan typescript eller med, det avgör ni 😊)

- **Axios**

npm i axios

(Ett smidigt sätt istället för FETCH, glöm inte att alla anslutningar skickas till servern och dess port, ej databasens port..)

- **React Router:**

npm i react-router-dom

(Flera sidor kräver en rutthanterare, detta kan ni nu redan 😊)

- **Ni behöver använda följande react hooks:**

useState (Hanterar tillstånd och kan "sätta ny data eller tillstånd")

useEffect (Låter dig synkronisera en komponent med ett externt system, Några exempel på (sido)effekter är att hämta data, direkt uppdatera DOM och timers etc)



IT-HÖGSKOLAN

Här startar din IT-karriär.

Axios Exempel i React

```
import './App.css'
import { useState, useEffect } from 'react';
import axios from 'axios';

function App() {

  //Hämta data
  const [data, setData] = useState([]);
  const [success, setSuccess] = useState(false);

  useEffect(() => {
    axios.get('http://localhost:8800/persons/')
      .then(response => {
        setData(response.data);
      })
      .catch(() => {
        // handle error
      });
  }, []);

  //Posta data
  const [formData, setFormData] = useState({
    FirstName: '',
    LastName: '',
    Address: '',
    City: ''
  });
```



IT-HÖGSKOLAN

Här startar din IT-karriär.



Testa er fram och se på lektion i
efterhand, repot kommer EJ att
läggas upp 😊



IT-HÖGSKOLAN

Här startar din IT-karriär.