

The background of the slide features a network pattern of interconnected nodes and lines. The left side is dark blue with faint, light blue nodes and lines. The right side is a lighter blue with more prominent, darker blue nodes and lines. The overall design is modern and tech-oriented.

# YrkesCo

En skalbar databas för framtidens skolor

# Agenda

- Bakgrund
- Problembeskrivning
- Vår lösning
- Datamodellering
- Implementation
- Queries
- Sammanfattning och nästa steg



# Bakgrund

- Många skolor använder Excelfiler och lärplattformar.
- Risk för fel och dubbelarbete.
- Behov av centraliserad databas.

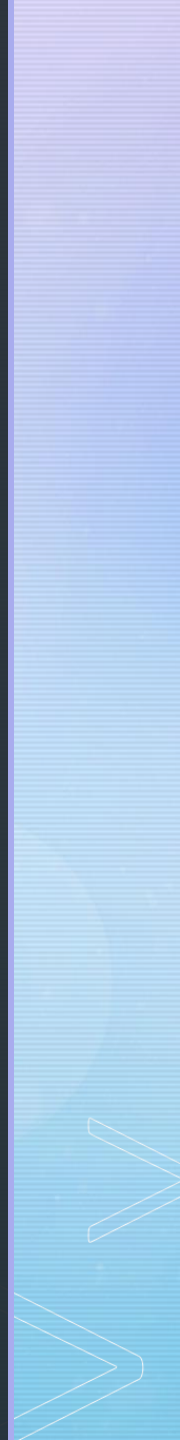


# Problembeskrivning

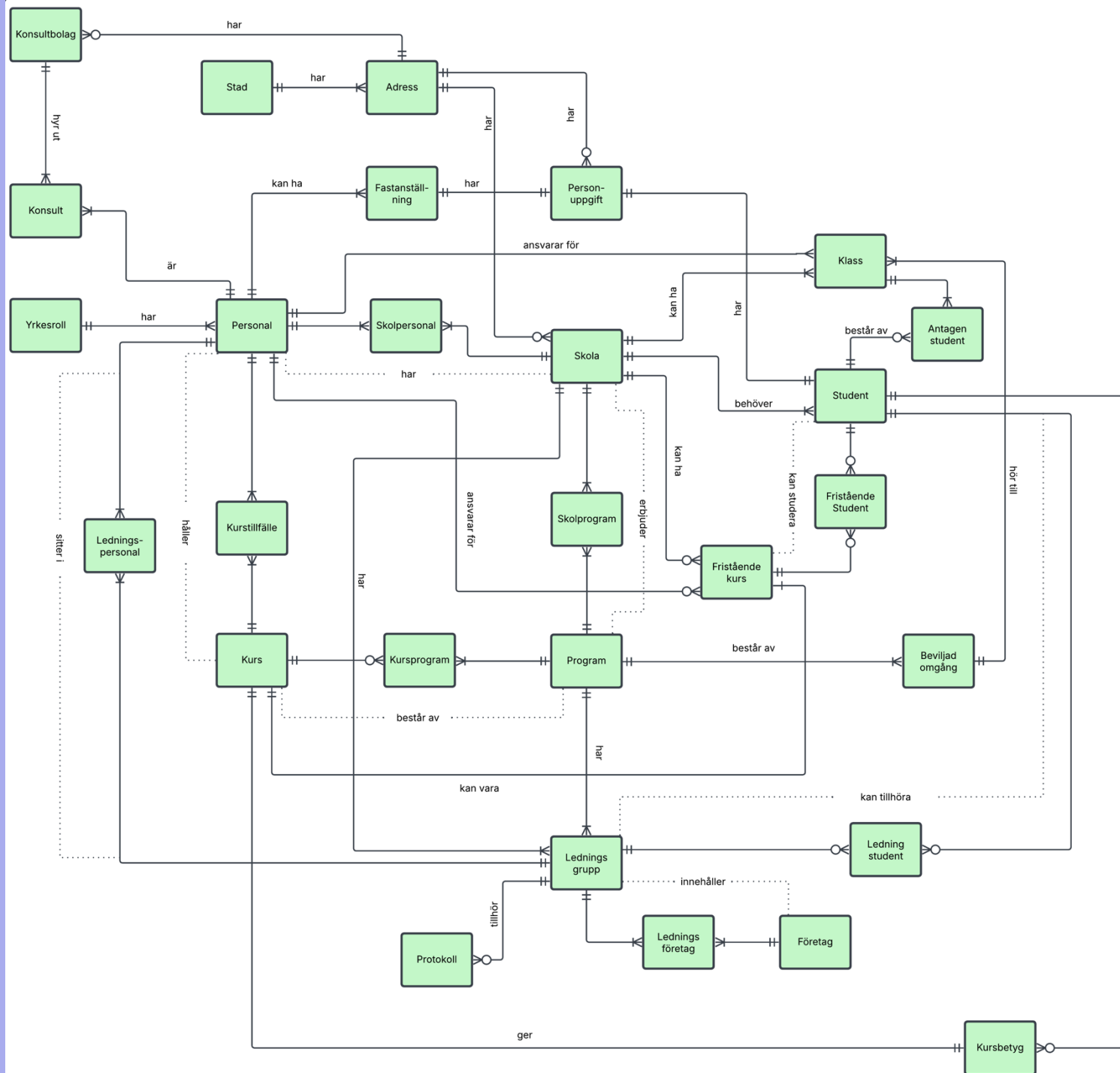
- • Separata system för studenter, kurser och utbildare.
- • Bristande datasäkerhet för känsliga uppgifter.
- • Svårt att skala vid expansion.



# Vår Lösning

- Relationsdatabas som centraliserar information.
  - Separata tabeller för känsliga persondata.
  - Klar för framtida tillväxt och integrationer.
- 

# Konceptuell Modell





# Relationship Statements

- **Skola och Student**

En skola kan ha en eller flera studenter.

En student tillhör en och endast en skola.

- **Program och Kurs**

Ett program består av flera kurser.

En kurs kan finnas i noll eller flera program.

- **Personal och Yrkesroll**

En eller flera personal kan ha samma yrkesroll.

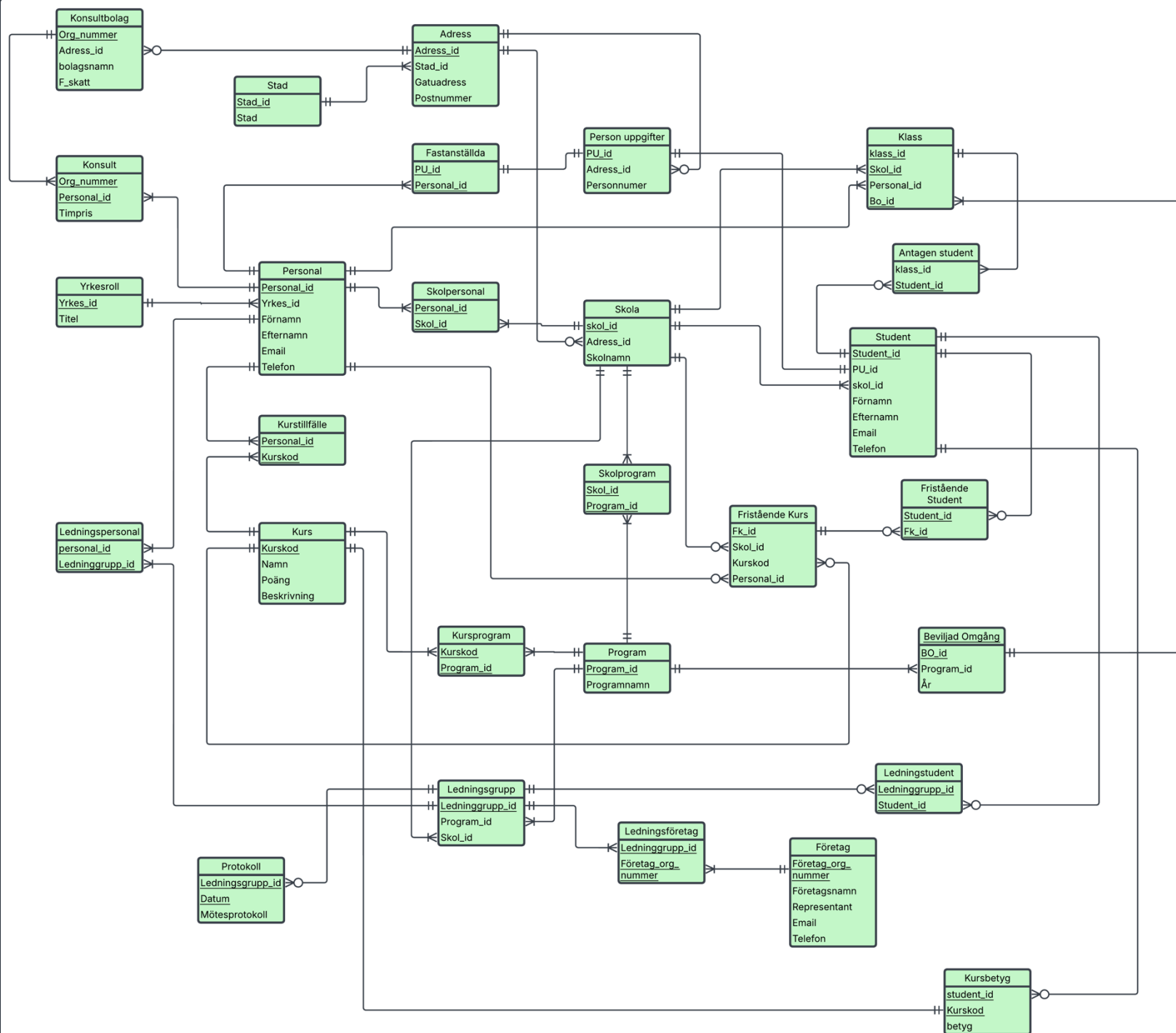
En personal är kopplad till en och endast en yrkesroll.

- **Konsult och Konsultbolag**

En konsult är kopplad till ett och endast ett konsultbolag.

Ett konsultbolag kan ha en eller flera konsulter.

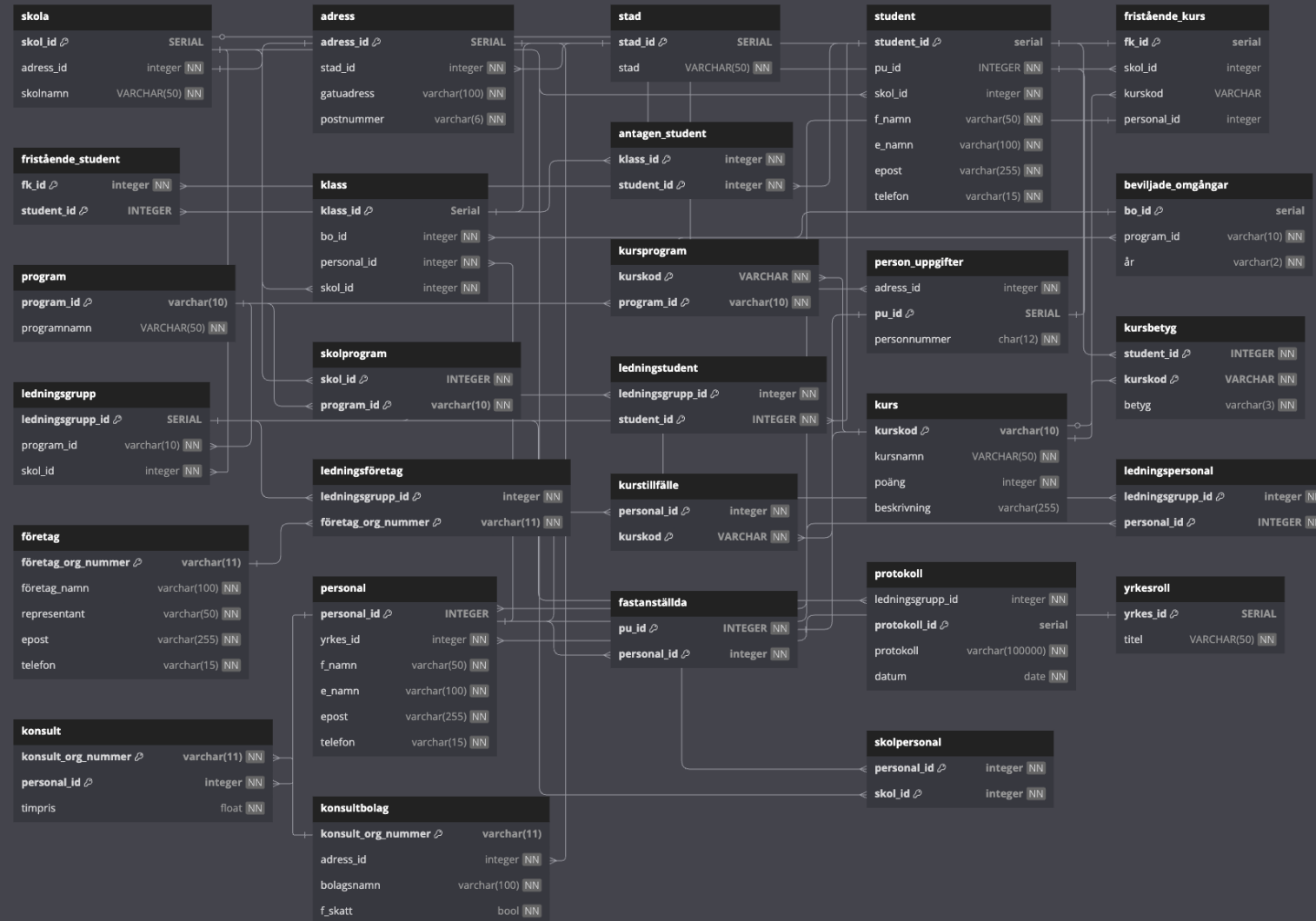
# Logisk Modell





# Fysisk Modell

```
1 Table skola {
2   skol_id SERIAL [primary key]
3   adress_id integer [not null]
4   skolnamn VARCHAR(50) [not null]
5 }
6
7 Table adress {
8   adress_id SERIAL [primary key]
9   stad_id integer [not null]
10  gatadress varchar(100) [not null]
11  postnummer varchar(6) [not null]
12 }
13
14 Table stad {
15   stad_id SERIAL [primary key]
16   stad VARCHAR(50) [not null]
17 }
18
19 Table student {
20   student_id serial [primary key]
21   pu_id INTEGER [not null, ref: > person_uppgifter]
22   skol_id integer [not null, ref: > skola]
23   f_namn varchar(50) [not null]
24   e_namn varchar(100) [not null]
25   epost varchar(255) [not null]
26   telefon varchar(15) [not null]
27 }
28
29 Table fristående_kurs {
30   fk_id serial [primary key]
31   skol_id integer [ref: > skola]
32   kurskod VARCHAR [ref: > kursprogram]
33   personal_id integer [ref: > fristående_student]
34 }
35
36 Table fristående_student {
37   fk_id integer [not null, ref: > fristående_kurs]
38   student_id INTEGER [ref: > student]
39   indexes {
40     (fk_id, student_id) [pk]
41   }
42 }
```



# Normaliserad data

- Alla attribut innehåller atomära värden.
- Alla rader är unika.
- Alla icke-nyckel-attribut är fullt funktionellt beroende av hela primärnyckeln
- Inga transitiva beroenden

# Implementation i Postgres

- Tabeller skapade i Postgres.
- Testdata insatt för alla tabeller.

```
INSERT INTO skola (adress_id, skolnamn) VALUES
(1,'YrkesCo Göteborg'),
(2,'YrkesCo Stockholm');

INSERT INTO personuppgift (adress_id, personnummer) VALUES
(5, '200001011234'), (6, '200002031238'), (7, '200003051245'), (8, '200004071212'), (9, '200005091299'), --GBG
(10, '200006111234'), (11, '200007131211'), (12, '200008151298'), (13, '200009171287'), (14, '200010191222'),
(15, '200011211265'), (16, '200012231267'), (17, '200001251278'), (18, '200002271289'), (19, '200003291234'),
(20, '200004301276'), (21, '200005021233'), (22, '200006041266'), (23, '200007061278'), (24, '200008081299'), --STHLM
(25, '200009101212'), (26, '200010121211'), (27, '200011141234'), (28, '200012161243'), (29, '200001181277'),
(30, '200002201244'), (31, '200003221287'), (32, '200004241299'), (33, '200005261212'), (34, '200006281211'),
(35, '198001011111'), (36, '198102031123'), (40, '198203051234'), (41, '198304071278'), (42, '198405091211'), --Personal
(37, '198506111299'), (38, '198607131234'), (43, '198708151267'), (39, '198809171299'), (44, '198910191278');

INSERT INTO program (program_id, programnamn) VALUES
('NET', 'Systemutvecklare .NET'),
('JAVA', 'Systemutvecklare Java'),
('DE', 'Data engineer'),
('APP', 'iOS/Android Developer'),
('IoT', 'Systemutvecklare Inbyggda system'),
('UX', 'UX-designer'),
('DevOps', 'DevOps Engineer');

INSERT INTO skolprogram (skol_id, program_id) VALUES
(1, 'NET'), (1, 'JAVA'), (1, 'DE'), (1, 'APP'), (1, 'IoT'), -- GBG
(2, 'JAVA'), (2, 'DE'), (2, 'IoT'), (2, 'UX'), (2, 'DevOps'); -- STHLM

INSERT INTO beviljad_omgång (program_id, år) VALUES
('NET', '24'), ('NET', '25'), ('NET', '26'),
('JAVA', '24'), ('JAVA', '25'), ('JAVA', '26'),
('DE', '24'), ('DE', '25'), ('DE', '26'),
('APP', '24'), ('APP', '25'), ('APP', '26'),
('IoT', '24'), ('IoT', '25'), ('IoT', '26'),
('UX', '24'), ('UX', '25'), ('UX', '26'),
('DevOps', '24'), ('DevOps', '25'), ('DevOps', '26');
```

```
CREATE TABLE IF NOT EXISTS "skola" (
  "skol_id" SERIAL PRIMARY KEY,
  "adress_id" INTEGER NOT null REFERENCES "adress",
  "skolnamn" VARCHAR(50) NOT NULL
);

CREATE TABLE IF NOT EXISTS "personuppgift" (
  "pu_id" SERIAL PRIMARY KEY,
  "adress_id" INTEGER NOT null REFERENCES "adress",
  "personnummer" CHAR(12) NOT NULL
);

CREATE TABLE IF NOT EXISTS "program" (
  "program_id" VARCHAR(10) PRIMARY KEY,
  "programnamn" VARCHAR(50) NOT NULL
);

CREATE TABLE IF NOT EXISTS "skolprogram" (
  "skol_id" INTEGER NOT NULL REFERENCES "skola",
  "program_id" VARCHAR(10) NOT NULL REFERENCES "program",
  PRIMARY KEY ("skol_id", "program_id")
);

CREATE TABLE IF NOT EXISTS "beviljad_omgång" (
  "bo_id" SERIAL PRIMARY KEY,
  "program_id" VARCHAR(10) NOT NULL REFERENCES "program",
  "år" VARCHAR(2) NOT NULL
);
```

# Exempel på Queries

-- Hämta studenter och vilken skola de tillhör

```
SELECT
  st.förnamn,
  st.efternamn,
  sk.skolnamn
FROM student st
JOIN skola sk ON st.skol_id = sk.skol_id;
```

A: förnamn	A: efternamn	A: skolnamn
Olivia	Ohlsson	YrkesCo Göteborg
Niklas	Nordin	YrkesCo Göteborg
Maria	Månsson	YrkesCo Göteborg
Lars	Lind	YrkesCo Göteborg
Karin	Karlsson	YrkesCo Göteborg
Johan	Johansson	YrkesCo Göteborg
Ida	Ivarsson	YrkesCo Göteborg
Henrik	Hult	YrkesCo Göteborg
Greta	Gran	YrkesCo Göteborg
Filip	Fransson	YrkesCo Göteborg
Elin	Ek	YrkesCo Göteborg
David	Dahl	YrkesCo Göteborg
Carla	Carlsson	YrkesCo Göteborg
Björn	Berg	YrkesCo Göteborg
Anna	Andersson	YrkesCo Göteborg
Östen	Öberg	YrkesCo Stockholm
Örjan	Öst	YrkesCo Stockholm
Älva	Ädel	YrkesCo Stockholm
Åke	Ångström	YrkesCo Stockholm
Zandra	Zed	YrkesCo Stockholm
Ylva	Yng	YrkesCo Stockholm
Xander	Xen	YrkesCo Stockholm
Wilma	Wall	YrkesCo Stockholm
Viktor	Vik	YrkesCo Stockholm
Ulrika	Ulfsson	YrkesCo Stockholm
Tobias	Torn	YrkesCo Stockholm
Sara	Sand	YrkesCo Stockholm
Rikard	Ryd	YrkesCo Stockholm
Quinn	Quick	YrkesCo Stockholm
Patrik	Persson	YrkesCo Stockholm

-- Hämta alla skolor och deras adresser

```
SELECT
  s.skolnamn,
  a.gatuadress,
  a.postnummer
FROM skola s
JOIN adress a ON s.adress_id = a.adress_id;
```

A: skolnamn	A: gatuadress	A: postnummer
YrkesCo Göteborg	Gatan 1	41101
YrkesCo Stockholm	Vägen 2	11122

```
-- Konsulter, vilket företag de tillhör och vilka klasser de undervisar
SELECT
  p.förnamn || ' ' || p.efternamn AS konsult,
  k.timpris,
  kb.bolagsnamn,
  STRING_AGG(ku.kursnamn, ' ') AS kurser
FROM konsult k
JOIN personal p ON k.personal_id = p.personal_id
JOIN konsultbolag kb ON k.org_nummer = kb.org_nummer
JOIN kurstillfälle kt ON p.personal_id = kt.personal_id
JOIN kurs ku ON kt.kurskod = ku.kurskod
GROUP BY p.förnamn, p.efternamn, k.timpris, kb.bolagsnamn;
```

```
-- Alla antagna studenter, deras klass, utbildningsledare och program
SELECT
  st.förnamn || ' ' || st.efternamn AS student,
  kl.klass_id,
  p.förnamn AS ledare_förnamn,
  p.efternamn AS ledare_efternamn,
  pr.programnamn
FROM antagen_student st
JOIN klass kl ON st.klass_id = kl.klass_id
JOIN personal p ON kl.personal_id = p.personal_id
JOIN beviljad_omgång bo ON kl.bo_id = bo.bo_id
JOIN program pr ON bo.program_id = pr.program_id
JOIN student st ON st.student_id = st.student_id;
```

student	123 klass_id	A: ledare_förnamn	A: ledare_efternamn	A: programnamn
Anna Andersson	1	Eva	Ekström	Systemutvecklare .NET
Filip Fransson	1	Eva	Ekström	Systemutvecklare .NET
Karin Karlsson	1	Eva	Ekström	Systemutvecklare .NET
Björn Berg	2	Eva	Ekström	Systemutvecklare Java
Greta Gran	2	Eva	Ekström	Systemutvecklare Java
Lars Lind	2	Eva	Ekström	Systemutvecklare Java
Patrik Persson	3	Anna	Andersson	Systemutvecklare Java
Ulrika Ulfsson	3	Anna	Andersson	Systemutvecklare Java
Zandra Zed	3	Anna	Andersson	Systemutvecklare Java
Elin Ek	4	Eva	Ekström	Data engineer
Johan Johansson	4	Eva	Ekström	Data engineer
Rikard Ryd	5	Anna	Andersson	Data engineer
Wilma Wall	5	Anna	Andersson	Data engineer
Älva Ädel	5	Anna	Andersson	Data engineer
David Dahl	6	Fredrik	Frid	iOS/Android Developer
Ida Ivarsson	6	Fredrik	Frid	iOS/Android Developer
Carla Carlsson	7	Fredrik	Frid	Systemutvecklare Inbyggda system
Henrik Hult	7	Fredrik	Frid	Systemutvecklare Inbyggda system
Maria Månsson	7	Fredrik	Frid	Systemutvecklare Inbyggda system
Quinn Quick	8	Anna	Andersson	Systemutvecklare Inbyggda system
Viktor Vik	8	Anna	Andersson	Systemutvecklare Inbyggda system
Åke Ångström	8	Anna	Andersson	Systemutvecklare Inbyggda system
Sara Sand	9	Johan	Johansson	UX-designer
Xander Xen	9	Johan	Johansson	UX-designer
Tobias Torn	10	Johan	Johansson	DevOps Engineer
Ylva Yng	10	Johan	Johansson	DevOps Engineer

A: konsult	123 timpris	A: bolagsnamn	A: kurser
Karl Kust	950	TechKonsult AB	UX Design, Python, Projektledning IT
Linda Lund	900	TechKonsult AB	UX Design, Python, Projektledning IT
Magnus Myra	1,000	IT-Resurs AB	Molntjänster AWS, Objektorienterad programmering
Nina Norr	1,100	IT-Resurs AB	Molntjänster AWS, Objektorienterad programmering



# Sammanfattning

- Centraliserad, effektiv och säker databas.
- Förbättrad datakvalitet och GDPR-säkerhet.
- Skalbar för framtida behov.