

Les 7 — Lesstof

Supabase Setup + eerste queries

Vak: Technical Foundations
Cursus: AI Developer
Volgende les: Les 8 — Supabase CRUD

Vak: Technical Foundations **Vorige les:** Les 6 — Next.js QuickPoll deel 2 **Volgende les:** Les 8 — Supabase CRUD + Mutations

Inhoud

1. [Wat is Supabase](#1-wat-is-supabase)
2. [Supabase vs alternatieven](#2-supabase-vs-alternatieven)
3. [Account + project aanmaken](#3-account--project-aanmaken)
4. [Postgres basics](#4-postgres-basics)
5. [Tabel aanmaken in Supabase](#5-tabel-aanmaken-in-supabase)
6. [Foreign keys + relaties](#6-foreign-keys--relaties)
7. [RLS — Row Level Security (intro)](#7-rls--row-level-security-intro)
8. [SQL Editor — eerste queries](#8-sql-editor--eerste-queries)
9. [Supabase client in Next.js](#9-supabase-client-in-nextjs)
10. [Eerste SELECT vanuit Server Component](#10-eerste-select-vanuit-server-component)

1. Wat is Supabase

Supabase = open-source Backend-as-a-Service. Geef je een Postgres database + auth + storage + realtime + functions via één dashboard. Gratis tier voor leren + kleine projecten.

Wat krijg je

- **Postgres database** — echte SQL, geen NoSQL
- **Auth** — login, signup, magic link, social (Les 9-10)
- **Storage** — files uploaden (later)

- **Realtime** — subscribe op DB changes (later)
- **Edge Functions** — serverless deno-runtime (later)
- **Dashboard** — visuele tabel-editor, SQL editor, auth-management

Open source

Volledige code op GitHub. Self-hostbaar als je wilt. Geen vendor lock-in.

2. Supabase vs alternatieven

	Supabase	Firebase	Planetscale	Neon
Database	Postgres	NoSQL (Firestore)	MySQL	Postgres
Auth	✓	✓	✗	✗
Pricing	Free tier ruim	Pay-per-read	Free tier	Free tier
Open source	✓	✗	✗	Onderdelen

Voor deze leerlijn: Supabase. Postgres + auth + dashboard alles-in-één. Geen extra services nodig voor 90% van je app.

3. Account + project aanmaken

Account

Ga naar supabase.com → Sign Up (mag met GitHub).

Project

1. New project
2. Org kiezen (default = je naam)
3. **Naam:** quickpoll
4. **Database password:** sla op (lange random string)
5. **Region:** Europe (Frankfurt) voor NL gebruikers
6. **Plan:** Free
7. Wacht 1-2 min op provisioning

Wat krijg je

- **Project URL:** `https://abc123.supabase.co`
- **Anon key:** publieke API key (mag in client)

- **Service role key:** admin key (NIET in client!)

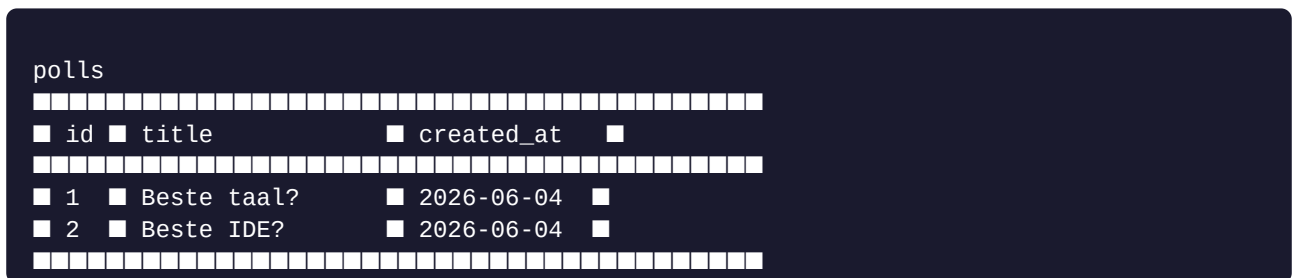
Beide te vinden via Settings → API.

4. Postgres basics

Wat is Postgres

Relationele SQL database. Sinds 1986, krachtige + betrouwbaar. Industrie-standaard.

Tabellen + kolommen



```
polls
┌───────────┬───────────┬───────────┬───────────┐
│ id │ title │ created_at │ │
├───────────┴───────────┴───────────┴───────────┤
│ 1 │ Beste taal? │ 2026-06-04 │ │
│ 2 │ Beste IDE? │ 2026-06-04 │ │
└───────────┴───────────┴───────────┴───────────┘
```

Veelgebruikte types

Type	Wanneer
bigserial	Auto-increment ID
uuid	Random UUID (default gen_random_uuid())
text	Variabele tekst
integer	Geheel getal
boolean	true/false
timestamp	Datum + tijd
jsonb	JSON (queryable)

Constraints

- `primary key` — unieke identifier
- `not null` — verplicht veld
- `unique` — geen duplicaten
- `references` — foreign key
- `default` — automatische waarde bij niet meegegeven

5. Tabel aanmaken in Supabase

Via Dashboard

1. Table Editor → New table
2. Naam: polls
3. Kolommen:
 - id — bigserial — primary key
 - title — text — not null
 - created_at — timestamp — default now()
1. Save

Via SQL Editor

```
create table polls (  
  id          bigserial primary key,  
  title       text not null,  
  created_at  timestamp default now()  
);
```

SQL is sneller + reproduceerbaar. Voor productie altijd via SQL (migrations).

6. Foreign keys + relaties

Voorbeeld — polls + options

Een poll heeft meerdere opties. Twee tabellen, relatie via poll_id:

```
create table options (  
  id          bigserial primary key,  
  poll_id     bigint not null references polls(id) on delete cascade,  
  text        text not null,  
  votes       integer default 0  
);
```

references polls(id) = foreign key naar polls.id. on delete cascade = poll verwijderd → opties ook weg.

One-to-many

polls → options (één poll, meerdere opties)

Querying

```
-- Alle polls met opties
select p.title, o.text, o.votes
from polls p
join options o on o.poll_id = p.id;
```

In Supabase JS:

```
await supabase.from("polls").select("*", options(*));
```

Automatisch een join.

7. RLS — Row Level Security (intro)

RLS = Postgres-feature waarmee je per-row regels stelt over wie wat mag.

Zonder RLS

Iedereen met je anon key kan ALLE data lezen + schrijven. Voor demo OK, voor productie disaster.

Met RLS

Per tabel: enable RLS + maak policies.

```
alter table polls enable row level security;

create policy "polls zijn publiek leesbaar"
on polls for select using (true);

create policy "alleen ingelogde users mogen polls maken"
on polls for insert with check (auth.uid() is not null);
```

Voor vandaag

In deze les: tabellen met RLS uit OF met open policies (using (true)). Dieper in Les 9-10 (auth + per-user RLS).

```
-- Voor demo — alle access open
alter table polls enable row level security;
create policy "demo open" on polls for all to anon using (true) with check (true);
```

8. SQL Editor — eerste queries

Insert

```
insert into polls (title) values ('Beste programmeertaal?')
returning id;
```

returning id = krijg de nieuwe id terug.

Select

```
-- Alle polls
select * from polls;

-- Specifieke poll
select * from polls where id = 1;

-- Sorted op nieuwste
select * from polls order by created_at desc limit 10;
```

Update

```
update polls set title = 'Nieuwere titel' where id = 1;
```

Delete

```
delete from polls where id = 1;
```

Aggregates

```
select count(*) from polls;
select count(*) as total, date_trunc('day', created_at) as day
from polls group by day;
```

9. Supabase client in Next.js

Install

```
pnpm add @supabase/supabase-js
```

`.env.local`

```
NEXT_PUBLIC_SUPABASE_URL=https://abc123.supabase.co
NEXT_PUBLIC_SUPABASE_ANON_KEY=eyJ...
```

NEXT_PUBLIC_ prefix want anon key mag in client (RLS beschermt data).

Client setup

```
// lib/supabase.ts
import { createClient } from "@supabase/supabase-js";

export const supabase = createClient(
  process.env.NEXT_PUBLIC_SUPABASE_URL!,
  process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY!,
);
```

In Les 9-10 vervangen we dit door SSR-safe variant (@supabase/ssr). Voor nu volstaat dit.

10. Eerste SELECT vanuit Server Component

```
// app/page.tsx
import { supabase } from "@/lib/supabase";

export default async function Home() {
  const { data: polls, error } = await supabase
    .from("polls")
    .select("*")
    .order("created_at", { ascending: false });

  if (error) return <p>Error: {error.message}</p>;

  return (
    <ul>
      {polls?.map((poll) => (
        <li key={poll.id}>
          <a href={` /poll/${poll.id} `}>{poll.title}</a>
        </li>
      ))}
    </ul>
  );
}
```

Wat hier gebeurt

- Server Component (geen "use client")
- `await supabase.from(...).select()` rechtstreeks in component
- Server fetcht data, retournt HTML naar client

-
- Geen extra API route, geen useEffect

In Les 8: we voegen INSERT toe (create polls) + de /create page.

Bronnen

- **Supabase docs:** <https://supabase.com/docs>
- **Supabase voor Next.js:** <https://supabase.com/docs/guides/getting-started/quickstarts/nextjs>
- **Postgres tutorial:** <https://www.postgresqltutorial.com/>
- **RLS guide:** <https://supabase.com/docs/guides/database/postgres/row-level-security>
- **JS client docs:** <https://supabase.com/docs/reference/javascript/select>
- **SQL crash course:** <https://sqlbolt.com>