Prisma

A schema-first ORM with a type-safe client.

01 Install & initialize

```
npm install --save-dev prisma@7
npm install @prisma/client@7 dotenv
npm install @prisma/adapter-pg
npx prisma init
```

Prisma 7 needs Node.js 20.19+, 22.12+ or 24+. init creates prisma/schema.prisma, prisma.config.ts and .env.

02 prisma.config.ts

```
import 'dotenv/config'
import { defineConfig } from 'prisma/config'
export default defineConfig({
  migrations: {
    seed: 'npx tsx prisma/seed.ts',
  },
  datasource: {
    url: process.env.DATABASE_URL,
  },
})
```

Prisma 7 no longer loads .env itself, hence dotenv. The CLI reads the database URL from here, not from the schema.

03 Generator & datasource

```
generator client {
  provider = "prisma-client"
  output   = "../generated/prisma"
}
datasource db {
  provider = "postgresql"
}
```

prisma-client replaces prisma-client-js and needs output. Run npx prisma generate after schema changes.

04 Models & enums

```
model User {
  id    String  @id @default(cuid())
  email String  @unique
  name  String? // optional
  role  Role    @default(USER)
  posts Post[]  // list: one-to-many
}
enum Role {
  USER
  ADMIN
}
```

05 Field & block attributes

`@id / @@id`	Primary or composite key
`@default(...)`	autoincrement(), now()
`@unique / @@unique`	Unique field or combo
`@updatedAt`	Set on every update
`@map / @@map`	DB column or table name
`@@index([a, b])`	Index one or more fields

06 One-to-many & one-to-one

```
model Post {
  id       String @id @default(cuid())
  title    String
  authorId String
  author   User   @relation(
    fields: [authorId], references: [id])
}
```

Keep @relation(...) on one line; it is wrapped here to fit. One-to-one: a @unique key, and post Post? on User.

07 Many-to-many & self-relations

```
model Tag {
  id    String @id @default(cuid())
  name  String @unique
  posts Post[]   // Post has tags Tag[]
}
// Self-relation fields on User
followedBy User[] @relation("UserFollows")
following  User[] @relation("UserFollows")
```

Prisma manages the join table. Self-relations need a relation name. For extra columns, write the join model yourself.

08 Referential actions

Cascade	Delete children too
Restrict	Block while children exist
NoAction	No automatic action
SetNull	Set the key to null
SetDefault	Set the key to its default

Set onDelete or onUpdate inside @relation(...).

09 Prisma Client

```
const adapter = new PrismaPg({
  connectionString: process.env.DATABASE_URL
})
export const prisma = new PrismaClient({
  adapter,
  log: ['query', 'error'],
})
```

Prisma 7 needs a driver adapter (@prisma/adapter-pg). Import PrismaClient from ./generated/prisma/client.

Prisma

Queries, transactions and migrations.

10 Create records

```
const user = await prisma.user.create({
  data: {
    email: 'alice@example.com',
    posts: { create: { title: 'Hello' } },
  },
  include: { posts: true },
})
```

Nested writes run in one transaction. createMany({ data: [...], skipDuplicates: true }) inserts in bulk.

11 Find, sort & paginate

```
const users = await prisma.user.findMany({
  where: { role: 'ADMIN' },
  select: { id: true, email: true },
  orderBy: { email: 'asc' },
  take: 20,
  skip: 40,
})
```

findUnique needs a unique field. include: { posts: true } adds relations. Cursor paging: cursor: { id }, skip: 1.

12 Filter operators

`contains`	Substring match
`startsWith`	Prefix; endsWith for suffix
`in`	Value is in a list
`gt gte lt lte`	Number and date ranges
`not`	Not equal, also not: null
`some/every/none`	To-many relation filters
`AND / OR / NOT`	Combine conditions

13 Update & delete

```
await prisma.user.update({
  where: { email },
  data: { name: 'Jane Doe' },
})
await prisma.post.deleteMany({
  where: { authorId: user.id },
})
```

upsert({ where, update, create }) inserts when nothing matches. Numbers take { increment: 1 }.

14 Aggregate & group

```
const stats = await prisma.post.aggregate({
  _count: { _all: true },
  _avg: { viewCount: true },
})
const byAuthor = await prisma.post.groupBy({
  by: ['authorId'], _count: { id: true },
})
```

Also _sum, _min and _max. count() counts rows.

15 Transactions

```
await prisma.$transaction(async (tx) => {
  const user = await tx.user.create({
    data: { email },
  })
  await tx.post.create({
    data: { title, authorId: user.id },
  })
})
```

A throw rolls it all back. Batch form: $transaction([q1, q2]). Options: timeout, maxWait, isolationLevel.

16 Raw SQL

```
const rows = await prisma.$queryRaw<User[]>`
  SELECT * FROM "User"
  WHERE email = ${email}`
```

Values become parameters, safe from SQL injection ($queryRawUnsafe is not). $executeRaw returns a row count.

17 Errors

```
try {
  await prisma.user.create({ data })
} catch (e) {
  if (e instanceof
    Prisma.PrismaClientKnownRequestError &&
    e.code === 'P2002') {
    throw new Error('Email already taken')
  }
  throw e
}
```

Import Prisma from your generated client. P2002: unique constraint, P2003: foreign key, P2025: record not found.

18 Migrations & seeding

```
npx prisma migrate dev --name init
npx prisma generate
npx prisma migrate deploy   # production
npx prisma migrate status
npx prisma migrate reset    # drops data
npx prisma db seed
npx prisma db push          # no migration
npx prisma studio
```

Prisma 7: migrate dev no longer runs generate or the seed, so run both yourself. Don't edit a migration after it is applied.