Karena saya pasti menggunakan Midtrans, apakah perlu adanya tabel payment, payment\_method, dan payment\_provider? Karena saya masih kebingungan dengan 3 tabel itu.

Jawaban:

Memahami kebingungan kamu mengenai tabel **payment**, **payment\_method**, dan **payment\_provider**, mari kita ulas satu per satu untuk memastikan apakah tabel-tabel ini diperlukan dalam integrasi dengan **Midtrans** atau apakah bisa disederhanakan.

**1. Tabel payment (Peran Utama: Mencatat Pembayaran)**

Tabel **payment** bertanggung jawab untuk mencatat semua detail pembayaran yang dilakukan oleh pengguna. Ini adalah tempat utama untuk menyimpan informasi yang berhubungan langsung dengan transaksi pembayaran.

**Apa yang disimpan di payment?**

* **payment\_id**: ID unik untuk setiap entri pembayaran.
* **transaction\_id**: Relasi ke transaksi yang sedang dibayar (misalnya, satu transaksi bisa memiliki satu atau lebih pembayaran jika ada cicilan).
* **payment\_status**: Status pembayaran seperti pending, success, atau failed.
* **payment\_amount**: Jumlah uang yang dibayar pada pembayaran ini.
* **service\_fee**: Biaya layanan yang dikenakan pada pembayaran.
* **total\_payment**: Jumlah total yang dibayar, termasuk biaya layanan.
* **payment\_time**: Waktu pembayaran dilakukan.
* **payment\_method**: Metode pembayaran (misalnya, kartu kredit, bank transfer, dll).
* **response\_data**: Menyimpan data respons dari Midtrans (atau penyedia pembayaran lainnya).

**Peran payment dalam sistem:**

Tabel **payment** penting karena ia mencatat **proses pembayaran yang terkait dengan transaksi**. Misalnya, setelah pengguna memilih produk/tiket dan melakukan pembayaran, detail pembayaran tersebut akan disimpan di sini.

**✅ Apakah perlu?**

Ya, tabel **payment** diperlukan untuk menyimpan **informasi pembayaran** yang telah dilakukan oleh pengguna. Ini sangat berguna ketika kamu perlu melacak status pembayaran atau menampilkan riwayat pembayaran pengguna.

**2. Tabel payment\_method (Metode Pembayaran yang Tersedia)**

Tabel **payment\_method** mencatat **metode pembayaran** yang dapat digunakan oleh pengguna untuk melakukan pembayaran. Ini mencakup informasi tentang berbagai jenis metode pembayaran yang didukung, seperti **kartu kredit**, **bank transfer**, **e-wallet**, atau **QRIS**.

**Apa yang disimpan di payment\_method?**

* **method\_id**: ID unik untuk metode pembayaran.
* **method\_name**: Nama metode pembayaran (misalnya, Credit Card, Bank Transfer, QRIS).
* **method\_code**: Kode unik yang digunakan untuk mengidentifikasi metode pembayaran.
* **method\_icon**: Ikon yang terkait dengan metode pembayaran (untuk UI).
* **service\_fee**: Biaya layanan tambahan yang terkait dengan metode pembayaran.
* **minimum\_payment**: Pembayaran minimum yang diperlukan untuk menggunakan metode ini.
* **maximum\_payment**: Pembayaran maksimum yang dapat dilakukan dengan metode ini.
* **is\_active**: Menandakan apakah metode pembayaran aktif atau tidak.

**Peran payment\_method dalam sistem:**

Tabel **payment\_method** memberikan fleksibilitas dalam mengelola berbagai **metode pembayaran** yang dapat dipilih pengguna. Kamu bisa menambahkan berbagai jenis metode pembayaran dari Midtrans, seperti kartu kredit, bank transfer, atau e-wallet, dan setiap transaksi akan memiliki metode pembayaran yang terkait.

**✅ Apakah perlu?**

Jika kamu hanya ingin mendukung pembayaran dengan **Midtrans**, kamu tidak perlu membuat tabel payment\_method yang rumit. **Midtrans** sudah menyediakan API yang dapat menangani berbagai metode pembayaran (kartu kredit, e-wallet, transfer bank, dll). Namun, tabel ini masih bisa berguna jika kamu ingin:

* **Mendukung berbagai metode pembayaran** selain yang disediakan oleh Midtrans.
* **Menampilkan opsi pembayaran kepada pengguna** di UI dan melacaknya di database.

Jika kamu hanya akan menggunakan Midtrans dengan sedikit variasi metode pembayaran, tabel ini mungkin tidak perlu terlalu banyak data dan bisa cukup sederhana.

**3. Tabel payment\_provider (Penyedia Pembayaran)**

Tabel **payment\_provider** mencatat **informasi tentang penyedia pembayaran** (misalnya, **Midtrans**, **Doku**, **Xendit**, dll). Ini berguna jika kamu berencana untuk mendukung **lebih dari satu penyedia pembayaran**.

**Apa yang disimpan di payment\_provider?**

* **provider\_id**: ID unik untuk penyedia pembayaran.
* **provider\_name**: Nama penyedia pembayaran (misalnya, Midtrans).
* **provider\_logo**: Logo penyedia pembayaran.
* **provider\_desc**: Deskripsi penyedia pembayaran.
* **is\_active**: Menandakan apakah penyedia pembayaran aktif atau tidak.

**Peran payment\_provider dalam sistem:**

Tabel **payment\_provider** berguna jika kamu berencana untuk mengintegrasikan **beberapa penyedia pembayaran** dalam satu aplikasi. Namun, jika hanya menggunakan **Midtrans**, tabel ini bisa jadi **tidak diperlukan**, karena kamu hanya akan memiliki satu penyedia pembayaran yang digunakan.

**✅ Apakah perlu?**

Jika kamu **hanya menggunakan Midtrans** dan tidak berniat menambah penyedia pembayaran lain, maka **payment\_provider** bisa dihilangkan atau disederhanakan, karena hanya akan mengarah ke satu penyedia (Midtrans).

**🎯 Ringkasan Keperluan Tabel-tabel ini:**

| **Tabel** | **Keterangan** | **Perlu atau Tidak?** |
| --- | --- | --- |
| **payment** | Menyimpan data pembayaran (satu transaksi bisa memiliki satu atau lebih pembayaran). | **Perlu** |
| **payment\_method** | Menyimpan berbagai metode pembayaran (misalnya, kartu kredit, bank transfer, e-wallet). | **Perlu** jika ada berbagai metode yang didukung (misalnya selain Midtrans). Jika hanya Midtrans, bisa disederhanakan. |
| **payment\_provider** | Menyimpan penyedia pembayaran (misalnya, Midtrans, Doku, Xendit). | **Tidak perlu** jika hanya menggunakan Midtrans. |

**💡 Simpulan:**

* **payment** adalah **wajib** karena mencatat pembayaran yang terjadi untuk transaksi.
* **payment\_method** dan **payment\_provider** adalah **opsional**. Jika kamu hanya menggunakan **Midtrans** untuk pembayaran, kamu bisa mempertimbangkan untuk **menyederhanakan** atau **menghapus** tabel ini.

Jadi bagaimana dengan jika saya memang menggunakan Midtrans, apakah perlu ada revisi pada 3 tabel tersebut, atau ada revisi total untuk semua tabel di database saya? schema.prisma (database baturraden\_tourism)

Jawaban:

Dari **schema.prisma** yang kamu berikan, kita bisa lihat bahwa tabel **payment**, **payment\_method**, dan **payment\_provider** memiliki **peran khusus** dalam menangani transaksi pembayaran. Sekarang mari kita evaluasi apakah **perlu ada revisi** atau tidak pada ketiga tabel tersebut jika kamu hanya menggunakan **Midtrans**.

**Peran Tabel-Tabel dalam Proses Pembayaran:**

**1. Tabel payment**

Tabel **payment** adalah tabel yang mencatat **detail pembayaran** terkait dengan transaksi. Setiap entri mencatat informasi terkait pembayaran yang dilakukan untuk transaksi tertentu.

**Kebutuhan untuk payment:**

* **payment\_id**: ID unik untuk setiap pembayaran.
* **transaction\_id**: Relasi ke transaksi yang sedang dibayar (satu transaksi bisa memiliki beberapa pembayaran).
* **payment\_status**: Status pembayaran seperti pending, success, atau failed.
* **payment\_time**: Waktu pembayaran dilakukan.
* **payment\_method**: Metode pembayaran yang digunakan, seperti kartu kredit atau e-wallet.
* **response\_data**: Data respons dari Midtrans (untuk menyimpan informasi lebih lanjut tentang transaksi pembayaran, seperti URL untuk pembayaran, kode transaksi, dll.).

**Apakah perlu?**

* Ya, **payment tetap diperlukan** karena mencatat **detail pembayaran** yang dilakukan untuk setiap transaksi. Jika kamu menggunakan Midtrans untuk memproses pembayaran, kamu tetap perlu mencatat status pembayaran dan menghubungkannya dengan transaksi tertentu.

**2. Tabel payment\_method**

Tabel **payment\_method** menyimpan informasi tentang **metode pembayaran** yang dapat dipilih pengguna, seperti **kartu kredit**, **transfer bank**, **QRIS**, dan lain-lain.

**Kebutuhan untuk payment\_method:**

* **method\_id**: ID unik untuk metode pembayaran.
* **method\_name**: Nama metode pembayaran, seperti Credit Card, Bank Transfer, QRIS.
* **method\_code**: Kode unik untuk metode pembayaran.
* **is\_active**: Status apakah metode pembayaran aktif atau tidak.

**Apakah perlu?**

* Jika kamu hanya menggunakan **Midtrans** dan **metode pembayaran yang disediakan oleh Midtrans**, tabel ini **tidak terlalu diperlukan** dalam bentuk kompleks. Kamu hanya perlu mencatat **metode pembayaran yang didukung** oleh Midtrans, dan bisa langsung menghubungkannya ke transaksi.
* Namun, jika kamu ingin **menambah metode pembayaran lain** selain yang disediakan oleh Midtrans (misalnya, beberapa opsi tambahan di masa depan), kamu bisa menggunakan tabel ini untuk mempermudah pengelolaan dan menampilkan opsi pembayaran kepada pengguna.

**3. Tabel payment\_provider**

Tabel **payment\_provider** menyimpan informasi tentang penyedia **metode pembayaran**, seperti **Midtrans**.

**Kebutuhan untuk payment\_provider:**

* **provider\_id**: ID unik untuk penyedia layanan pembayaran.
* **provider\_name**: Nama penyedia layanan pembayaran (misalnya, Midtrans).
* **is\_active**: Status apakah penyedia pembayaran aktif atau tidak.

**Apakah perlu?**

* Jika kamu hanya menggunakan **Midtrans**, tabel **payment\_provider** bisa jadi **tidak diperlukan**. Semua data terkait penyedia (Midtrans) dapat langsung disimpan di dalam tabel **payment\_method**.
* **Jika ingin mendukung lebih dari satu penyedia pembayaran**, misalnya **Midtrans**, **Doku**, atau **Xendit**, tabel ini akan berguna untuk mencatat penyedia layanan pembayaran yang digunakan.

**Rekomendasi Revisi Tabel-Tabel:**

1. **Tabel payment**: **Tetap diperlukan**. Ini sangat penting untuk mencatat semua detail pembayaran, termasuk data dari Midtrans (status, waktu, dan amount pembayaran).
2. **Tabel payment\_method**: Jika hanya **Midtrans** yang digunakan, kamu bisa **menyederhanakan** tabel ini atau bahkan **menghapusnya**, dan langsung simpan metode pembayaran (misalnya, credit\_card, bank\_transfer, dll.) di **tabel payment**. Namun, jika berencana menambah berbagai metode pembayaran lainnya, tabel ini akan tetap berguna.
3. **Tabel payment\_provider**: Jika hanya menggunakan **Midtrans**, tabel ini **tidak diperlukan** dan bisa **dihapus**. Namun, jika ada rencana untuk menambah penyedia pembayaran lain, kamu bisa **mempertahankan** tabel ini.

**Revisi Total pada Tabel-tabel (Jika Hanya Menggunakan Midtrans):**

Jika kamu hanya berencana menggunakan **Midtrans**, kamu bisa melakukan beberapa **penyederhanaan** berikut:

**1. Simpan hanya metode pembayaran yang digunakan:**

* Di **tabel payment\_method**, kamu bisa hanya mencatat beberapa opsi pembayaran yang disediakan oleh Midtrans (misalnya, credit\_card, bank\_transfer, dan e\_wallet).

**2. Menghapus payment\_provider jika hanya menggunakan satu penyedia (Midtrans).**

**Contoh Penyederhanaan Struktur Tabel payment dan payment\_method:**

1. **Tabel payment\_method** (jika hanya satu penyedia seperti Midtrans):

|  |
| --- |
| prisma  model payment\_method {  method\_id Int @id @default(autoincrement())  method\_name String @db.VarChar(50)  method\_code String @db.VarChar(50)  is\_active Boolean? @default(true)  payments payment[] // Relasi ke tabel payment  } |

1. **Tabel payment** (sederhana, hanya mencatat informasi yang dibutuhkan):

|  |
| --- |
| prisma  model payment {  payment\_id Int @id @default(autoincrement())  transaction\_id Int?  method\_id Int?  reference\_number String @unique(map: "reference\_number") @db.VarChar(50)  payment\_amount Decimal @db.Decimal(10, 2)  service\_fee Decimal @db.Decimal(10, 2)  total\_payment Decimal? @db.Decimal(10, 2)  payment\_status PaymentStatus? @default(pending)  payment\_time DateTime? @default(now()) @db.Timestamp(0)  payment\_method payment\_method? @relation(fields: [method\_id], references: [method\_id])  transaction transaction? @relation(fields: [transaction\_id], references: [transaction\_id], onDelete: Cascade, onUpdate: NoAction)  } |

**Kesimpulan:**

* **Tabel payment** tetap diperlukan untuk mencatat detail pembayaran.
* **Tabel payment\_method** dapat disederhanakan atau dipertahankan jika kamu berencana untuk menambah berbagai metode pembayaran selain Midtrans.
* **Tabel payment\_provider** bisa **dihapus** jika hanya menggunakan **Midtrans**.

Apakah kamu ingin saya bantu dengan implementasi **Midtrans API** atau melanjutkan ke bagian lain?

Saya memang memiliki rasa penasaran cukup tinggi, jadi mengapa payment\_method ini disederhakan dan payment\_provider ini dihilangkan jika saya memang menggunakan Midtrans?

Jawaban:

Itu adalah pertanyaan yang sangat bagus! Memang, untuk sistem pembayaran seperti **Midtrans**, ada alasan mengapa saya menyarankan untuk **menyederhanakan** atau **menghilangkan** beberapa tabel seperti payment\_method dan payment\_provider. Mari kita jelaskan alasan di baliknya:

**1. Tabel payment\_method**

Tabel **payment\_method** biasanya digunakan untuk menyimpan berbagai metode pembayaran yang tersedia untuk pengguna, seperti:

* **Kartu Kredit**
* **Bank Transfer**
* **E-wallet**
* **QRIS**, dll.

Namun, jika kamu **hanya menggunakan Midtrans** sebagai penyedia pembayaran utama, maka semua metode pembayaran yang ingin kamu tawarkan kepada pengguna **sudah disediakan oleh Midtrans** melalui API mereka.

**Kenapa bisa disederhanakan?**

1. **Midtrans sudah mengelola berbagai metode pembayaran untukmu**. Misalnya, jika pengguna memilih **kartu kredit**, **bank transfer**, atau **e-wallet**, Midtrans akan menangani semua ini melalui satu API yang sama. Oleh karena itu, **tidak perlu tabel payment\_method yang menyimpan data metode pembayaran** secara manual.
2. **Informasi tentang metode pembayaran** (seperti kartu kredit, bank transfer, dll.) dapat langsung diambil dari **Midtrans API** saat transaksi dilakukan. Kamu bisa mendapatkan informasi tentang metode pembayaran yang digunakan langsung dari Midtrans (misalnya, menggunakan payment\_type di respons transaksi).

**💡 Solusi Penyederhanaan:**

Kamu bisa langsung menyimpan metode pembayaran **di dalam tabel payment** atau hanya menyimpan **payment\_status** dan informasi terkait, tanpa memerlukan tabel terpisah untuk metode pembayaran.

**2. Tabel payment\_provider**

Tabel **payment\_provider** menyimpan informasi tentang penyedia layanan pembayaran (misalnya **Midtrans**, **Doku**, **Xendit**, dll.). Biasanya, tabel ini berguna ketika kamu ingin mendukung lebih dari satu penyedia pembayaran.

**Kenapa bisa dihilangkan?**

Jika kamu hanya berencana **menggunakan satu penyedia pembayaran**, yaitu **Midtrans**, maka tabel **payment\_provider** **tidak diperlukan**. Midtrans sudah menyederhanakan seluruh proses pembayaran, jadi kamu tidak memerlukan tabel yang memisahkan informasi tentang penyedia layanan pembayaran.

1. **Satu penyedia pembayaran**: Jika hanya ada **Midtrans** sebagai penyedia pembayaran, tidak perlu memisahkan data penyedia pembayaran di tabel berbeda.
2. **Keuntungan dari menghapusnya**: Mengurangi kerumitan dan kompleksitas database jika hanya menggunakan satu penyedia. Cukup simpan informasi metode pembayaran di dalam tabel **payment\_method** dan **payment**, yang mengarah langsung ke transaksi.

**Apakah Harus Menggunakan Pendekatan ini?**

Pendekatan ini akan **lebih efisien** dalam hal penyimpanan dan pengelolaan data, karena kamu hanya akan menyimpan informasi yang **relevan** dengan Midtrans. Jika di masa depan kamu **berencana untuk menambahkan lebih banyak penyedia pembayaran**, kamu bisa **menambahkannya** kembali dengan mudah.

Namun, jika kamu **membutuhkan fleksibilitas** untuk mendukung beberapa penyedia pembayaran sejak awal, kamu bisa **mempertahankan tabel payment\_method dan payment\_provider**, dan nanti bisa menambahkan logika untuk menangani lebih dari satu penyedia.

**Contoh Alur Penggunaan Midtrans tanpa payment\_method dan payment\_provider:**

1. **Pembayaran dimulai** → **Midtrans API** menerima pembayaran dan memprosesnya.
2. **Status pembayaran** disimpan di **payment** dengan informasi seperti payment\_status, payment\_amount, dll.
3. Jika diperlukan, kamu bisa **mendapatkan informasi pembayaran** melalui API Midtrans, tanpa perlu menyimpan metode pembayaran secara terpisah.

**Ringkasan:**

* Jika **hanya menggunakan Midtrans**, kamu bisa **menyederhanakan** dan menghindari kompleksitas dengan menggabungkan data **metode pembayaran** langsung di dalam tabel **payment** dan **payment\_method**.
* **payment\_provider** bisa dihilangkan karena hanya ada satu penyedia pembayaran (Midtrans).
* **Mengurangi redundansi data** dan membuat struktur lebih mudah dikelola jika kamu hanya berencana menggunakan **Midtrans**.

Sebelum itu, apakah pada tabel payment, payment\_method, dan payment\_provider perlu dirombak lagi atau tidak usah? Karena memang aku menggunakan Midtrans.

Jawaban:

Pertanyaanmu sangat penting dan menunjukkan kamu ingin sistem yang **rapi, efisien, dan realistis**.

**✳️ Jawaban Ringkas:**

**✅ payment tetap digunakan.**

**⚠️ payment\_method bisa disederhanakan.**

**❌ payment\_provider boleh dihapus.**

**🔍 Alasan Detail Sesuai Midtrans Integration**

**1. ✅ Tetap Gunakan payment**

payment berfungsi sebagai **log resmi pembayaran yang sukses**, dan **wajib** digunakan, karena:

* Midtrans tidak menyimpan transaksi selamanya → kamu butuh log lokal
* Admin bisa melihat siapa bayar berapa, kapan, pakai apa
* Bisa digunakan untuk laporan, rekap, export

**2. ⚠️ payment\_method: Opsional, tapi bisa disederhanakan**

Midtrans mengirim:

|  |
| --- |
| json  {  "payment\_type": "qris"  } |

Kamu cukup simpan ini dalam kolom biasa seperti:

|  |
| --- |
| prisma  payment\_type String // misalnya: 'qris', 'gopay', 'bank\_transfer' |

Jika kamu ingin menampilkan ikon dan label pembayaran di UI (contoh: QRIS = "QR Code Indonesia Standard"), kamu **tidak perlu tabel payment\_method di database**, cukup mapping statis di frontend/backend.

**3. ❌ payment\_provider tidak perlu**

Karena kamu **tidak mendukung multi-payment gateway** seperti Xendit, Doku, dsb, tabel ini **tidak ada gunanya** dan **boleh dihapus**.

**✨ Opsi Refactor Paling Minimal dan Efisien**

Berikut versi tabel payment yang cocok untuk Midtrans:

|  |
| --- |
| prisma  CopyEdit  model payment {  payment\_id Int @id @default(autoincrement())  transaction\_id Int  reference\_number String @unique @db.VarChar(50)  payment\_type String @db.VarChar(50) // ← dari Midtrans  payment\_amount Decimal @db.Decimal(10, 2)  payment\_status PaymentStatus @default(pending)  payment\_time DateTime @default(now()) @db.Timestamp(0)  response\_data Json  transaction transaction @relation(fields: [transaction\_id], references: [transaction\_id], onDelete: Cascade)  @@index([transaction\_id])  } |

**🧠 Penjelasan untuk Dosen:**

Karena saya hanya menggunakan satu payment provider, yaitu Midtrans, maka sistem tidak memerlukan tabel payment\_provider.  
Metode pembayaran (payment\_type) seperti QRIS, ShopeePay, atau BCA Virtual Account dikirim langsung oleh Midtrans melalui webhook, sehingga cukup disimpan dalam kolom payment\_type tanpa perlu relasi ke tabel payment\_method.  
Hal ini membuat sistem lebih ringan dan tetap cukup informatif untuk admin dan laporan pembayaran.

**🔧 Mau saya bantu:**

* Refactor schema.prisma langsung ke versi efisien ini?
* Atau bantu membuat script migrasi untuk menghapus payment\_provider dan simplify payment\_method?

Tinggal bilang, saya bantu.

Sebelum masuk ke temp\_order lagi, jika:   
- ✅ payment tetap digunakan.   
- ⚠️ payment\_method bisa disederhanakan.   
- ❌ payment\_provider boleh dihapus. Bagaimana perombakan pada schema.prisma?

Jawaban:

Berikut ini adalah **revisi schema.prisma minimalis dan optimal khusus untuk Midtrans**, berdasarkan poin:

✅ payment tetap digunakan  
⚠️ payment\_method disederhanakan  
❌ payment\_provider dihapus sepenuhnya

**✅ Perubahan Final:**

**🔁 Ganti payment model seperti ini:**

|  |
| --- |
| prisma  CopyEdit  model payment {  payment\_id Int @id @default(autoincrement())  transaction\_id Int  reference\_number String @unique @db.VarChar(50)  payment\_type String @db.VarChar(50) // ← contoh: 'qris', 'gopay'  payment\_amount Decimal @db.Decimal(10, 2)  payment\_status PaymentStatus @default(pending)  payment\_time DateTime @default(now()) @db.Timestamp(0)  response\_data Json  transaction transaction @relation(fields: [transaction\_id], references: [transaction\_id], onDelete: Cascade)  @@index([transaction\_id])  } |

**🗑 Hapus model berikut dari schema:**

|  |
| --- |
| prisma  model payment\_method { ... }  model payment\_provider { ... } |

Dan juga hapus relasi ini dari model payment yang sebelumnya:

|  |
| --- |
| prisma  method\_id Int?  method payment\_method? @relation(...) // hapus ini  @@index([method\_id]) // hapus ini |

**📌 Penyesuaian Tambahan (opsional):**

Kalau sebelumnya kamu pakai:

|  |
| --- |
| ts  method\_code  method\_id  method\_name |

Sekarang cukup gunakan:

|  |
| --- |
| ts  payment\_type // ini didapat langsung dari Midtrans, misalnya 'qris', 'gopay', 'bank\_transfer' |

**🧠 Penjelasan Untuk Dosen (Siap Pakai)**

Karena sistem ini menggunakan Midtrans sebagai satu-satunya payment gateway, maka saya tidak perlu mengelola metode pembayaran secara manual melalui tabel payment\_method atau payment\_provider.  
Midtrans sudah menyediakan jenis pembayaran dalam response callback-nya (seperti QRIS, Bank Transfer, GoPay), sehingga saya cukup menyimpan informasi tersebut dalam kolom payment\_type.  
Pendekatan ini membuat sistem lebih sederhana, cepat dikembangkan, dan tetap cukup informatif untuk kebutuhan pencatatan pembayaran di admin.

**🧪 Siap Migrasi?**

Kalau kamu sudah edit schema.prisma, cukup jalankan:

bash

CopyEdit

npx prisma migrate dev --name simplify\_payment\_for\_midtrans

Jika kamu ingin saya bantu buatkan **full versi schema baru siap paste**, tinggal bilang ya 🙌 Mau lanjut ke temp\_order setelah ini juga bisa langsung lanjut.

Sebentar, aku jadi penasaran. Jika memang payment\_method dan payment\_provider ini dihapus, kemudian ditambahkan dengan temp\_order dan temp\_order\_detail. Bagaimana alur keseluruhanya dari awal sampai akhir? Karena schema.prisma ku sudah pasti ada perombakan lagi.

Jawaban:

Pertanyaanmu sangat tepat — dan kamu **harus tahu gambaran besar alur sistem secara end-to-end** terutama setelah kita menyederhanakan payment\_\* dan menambahkan temp\_order. Ini juga sangat penting untuk dokumentasi dan sidang.

**🔁 ALUR KESELURUHAN SISTEM (FULL CYCLE)**

**🎯 Fokus: Transaksi tiket wisata menggunakan Midtrans + temp\_order**

**🧑 1. User Login dan Masuk ke Halaman Tiket**

* User melihat daftar tiket dari database ticket.

**🛒 2. User Klik “Beli Tiket”**

* User memilih tiket dan memasukkan jumlah & tanggal kunjungan.
* Jika membeli lebih dari 1, user juga bisa mengisi nama tamu lain (visitor\_name).

✅ **Yang dilakukan sistem:**

* Generate order\_id = ORDER-${timestamp}
* Simpan ke:
  + temp\_order → user\_id, visit\_date, order\_id
  + temp\_order\_detail → tiket, quantity, harga, visitor\_name
* Redirect ke Snap Midtrans menggunakan token dari /api/payment/tokenize

**💳 3. User Membayar Lewat Snap Midtrans**

* Midtrans akan:
  + Menyimpan status transaksi (success, pending, failed)
  + Mengirim **callback** ke pages/api/admin/transaction/callback.ts

**📨 4. Midtrans Kirim Callback**

Callback akan membawa:

|  |
| --- |
| json  {  "order\_id": "ORDER-...",  "transaction\_status": "settlement",  "gross\_amount": "5000",  "payment\_type": "qris"  } |

✅ **Yang dilakukan sistem (di callback.ts):**

1. Cari temp\_order berdasarkan order\_id
2. Ambil user\_id, visit\_date, dan semua detail tiket dari temp\_order\_detail
3. Buat entri baru ke:
   * transaction (1 baris)
   * transaction\_detail (per tiket)
4. Buat entri ke:
   * payment dengan payment\_type, amount, dan status = success
5. Generate QR dan simpan ke ticket.url\_qr
6. Opsional: Hapus temp\_order & temp\_order\_detail

**📃 5. User Melihat Halaman Riwayat Transaksi**

* Menampilkan data dari tabel transaction (yang status-nya paid)
* Dapat melihat:
  + Tanggal
  + Nama tiket
  + Status
  + Tombol Download Tiket (jika status = paid)

**🧾 6. Admin Melihat Halaman Pembayaran**

* Menampilkan data dari tabel payment
* Dapat melihat:
  + Reference (order\_id)
  + Jenis pembayaran (payment\_type)
  + Nominal
  + Status (success/pending)
  + Tanggal

**🧱 Struktur Tabel yang Digunakan:**

| **Fungsi** | **Tabel** |
| --- | --- |
| Data tiket wisata | ticket, spot\_tourism |
| Data user | users |
| Penyimpanan awal transaksi | temp\_order, temp\_order\_detail |
| Data final transaksi | transaction, transaction\_detail |
| Log pembayaran | payment |

**✍️ Penjelasan untuk Dokumentasi Tugas Akhir:**

Sistem ini dirancang dengan pendekatan transaksi dua fase: fase sementara (temp\_order) dan fase final (transaction). Karena integrasi pembayaran menggunakan Midtrans bersifat asinkron (callback-based), maka data transaksi awal disimpan ke dalam temp\_order dan temp\_order\_detail sebelum redirect ke Snap.  
Setelah pembayaran berhasil, sistem akan menerima callback dan menyelesaikan transaksi ke tabel transaction dan payment. Dengan pola ini, integritas data tetap terjaga dan sistem dapat memberikan log pembayaran yang akurat kepada admin maupun pengguna.

Kalau kamu mau, aku bisa bantu gambar diagram alir (flowchart) alur ini atau versi **urutan CRUD + tabel** agar bisa langsung kamu masukkan ke laporan tugas akhir. Mau dibantu juga?

Oke, aku kirimkan schema.prisma lagi, tapi schema.prisma ini masih yang lama belum ada perubahan. Jadi bagaimana alurnya jika memang penghapusan payment\_method dan payment\_provider, dan ditambahnya temp\_order dan temp\_order\_detail nanti?

Jawaban:

|  |
| --- |
| generator client {  provider = "prisma-client-js"  output = "@/src/generated/prisma"  }  datasource db {  provider = "mysql"  url = env("DATABASE\_URL")  }  model category\_tourism {  category\_id String @id @db.Char(5)  category\_name String @db.VarChar(20)  category\_desc String? @db.VarChar(150)  category\_icon String? @db.VarChar(255)  spots spot\_tourism[]  }  model facility {  facility\_id String @id @db.Char(10)  facility\_name String @db.VarChar(100)  facility\_icon String? @db.VarChar(255)  facility\_tourisms facility\_tourism[]  }  model facility\_tourism {  facility\_tourism\_id String @id @db.Char(10)  spot\_id String? @db.Char(10)  facility\_id String? @db.Char(10)  spot spot\_tourism? @relation(fields: [spot\_id], references: [spot\_id], onDelete: Cascade)  facility facility? @relation(fields: [facility\_id], references: [facility\_id], onDelete: Cascade)  @@index([facility\_id])  @@index([spot\_id])  }  model gallery {  gallery\_id String @id @db.Char(10)  spot\_id String? @db.Char(10)  gallery\_img String? @db.VarChar(255)  gallery\_caption String? @db.VarChar(255)  spot spot\_tourism? @relation(fields: [spot\_id], references: [spot\_id], onDelete: Cascade)  @@index([spot\_id])  }  model operating\_hours {  operating\_id String @id @db.Char(10)  spot\_id String? @db.Char(10)  operating\_day OperatingDay  hours\_open DateTime @db.Time(0)  hours\_closed DateTime @db.Time(0)  spot spot\_tourism? @relation(fields: [spot\_id], references: [spot\_id], onDelete: Cascade)  @@index([spot\_id])  }  model payment {  payment\_id Int @id @default(autoincrement())  transaction\_id Int  reference\_number String @unique @db.VarChar(50)  payment\_type String @db.VarChar(50)  payment\_amount Decimal @db.Decimal(10, 2)  payment\_status PaymentStatus @default(pending)  payment\_time DateTime @default(now()) @db.Timestamp(0)  response\_data Json  transaction transaction @relation(fields: [transaction\_id], references: [transaction\_id], onDelete: Cascade)  @@index([transaction\_id])  }  model reviews {  reviews\_id String @id @db.Char(10)  spot\_id String? @db.Char(10)  user\_id String? @db.Char(10)  reviews\_rating Int? @db.TinyInt  reviews\_desc String? @db.Text  reviews\_created\_at DateTime? @default(now()) @db.Timestamp(0)  spot spot\_tourism? @relation(fields: [spot\_id], references: [spot\_id], onDelete: Cascade)  user users? @relation(fields: [user\_id], references: [user\_id], onDelete: Cascade)  @@index([spot\_id])  @@index([user\_id])  }  model spot\_tourism {  spot\_id String @id @db.Char(10)  category\_id String? @db.Char(5)  village\_id String? @db.Char(10)  spot\_name String @db.VarChar(60)  spot\_desc String? @db.Text  spot\_address String? @db.VarChar(200)  spot\_maps String? @db.VarChar(255)  spot\_thumbnail String? @db.VarChar(100)  spot\_rating Decimal? @default(0.00) @db.Decimal(3, 2)  spot\_contact String? @db.VarChar(15)  facility\_tourisms facility\_tourism[]  galleries gallery[]  operating\_hours operating\_hours[]  reviews reviews[]  category category\_tourism? @relation(fields: [category\_id], references: [category\_id])  village village? @relation(fields: [village\_id], references: [village\_id])  tickets ticket[]  @@index([category\_id])  @@index([village\_id])  }  model ticket {  ticket\_id String @id @db.Char(10)  spot\_id String? @db.Char(10)  ticket\_name String? @db.VarChar(100)  ticket\_price Decimal @db.Decimal(10, 2)  ticket\_desc String? @db.Text  ticket\_stock Int? @default(0)  code String @db.VarChar(10)  url\_qr String @db.Text  spot spot\_tourism? @relation(fields: [spot\_id], references: [spot\_id], onDelete: Cascade)  transaction\_details transaction\_detail[]  temp\_order\_details temp\_order\_detail[]  @@index([spot\_id])  }  model transaction {  transaction\_id Int @id @default(autoincrement())  user\_id String? @db.Char(10)  transaction\_code String @unique @db.VarChar(50)  total\_price Decimal @db.Decimal(10, 2)  transaction\_status TransactionStatus? @default(pending)  transaction\_date DateTime? @default(now()) @db.Timestamp(0)  visit\_date DateTime @db.Date  payments payment[]  user users? @relation(fields: [user\_id], references: [user\_id], onDelete: Cascade)  transaction\_details transaction\_detail[]  @@index([user\_id])  }  model transaction\_detail {  transaction\_detail\_id Int @id @default(autoincrement())  transaction\_id Int?  ticket\_id String? @db.Char(20)  detail\_quantity Int  detail\_price Decimal @db.Decimal(10, 2)  detail\_subtotal Decimal? @db.Decimal(10, 2)  is\_used TicketStatus @default(unverified)  transaction transaction? @relation(fields: [transaction\_id], references: [transaction\_id], onDelete: Cascade)  ticket ticket? @relation(fields: [ticket\_id], references: [ticket\_id], onDelete: Cascade)  @@index([ticket\_id])  @@index([transaction\_id])  }  model users {  user\_id String @id @db.Char(10)  name String @db.VarChar(100)  email String @unique @db.VarChar(100)  password String @db.VarChar(255)  phone\_number String? @db.VarChar(20)  address String? @db.Text  role Role  reviews reviews[]  transactions transaction[]  temp\_orders temp\_order[]  }  model village {  village\_id String @id @db.Char(10)  village\_name String @db.VarChar(100)  village\_city String @db.VarChar(100)  spots spot\_tourism[]  }  model temp\_order {  temp\_order\_id Int @id @default(autoincrement())  order\_id String @unique @db.VarChar(50)  user\_id String @db.Char(10)  visit\_date DateTime @db.Date  created\_at DateTime @default(now()) @db.Timestamp(0)  user users @relation(fields: [user\_id], references: [user\_id], onDelete: Cascade)  details temp\_order\_detail[]  @@index([user\_id])  }  model temp\_order\_detail {  temp\_order\_detail\_id Int @id @default(autoincrement())  temp\_order\_id Int  ticket\_id String @db.Char(20)  visitor\_name String? @db.VarChar(100)  quantity Int @default(1)  price Decimal @db.Decimal(10, 2)  subtotal Decimal @db.Decimal(10, 2)  temp\_order temp\_order @relation(fields: [temp\_order\_id], references: [temp\_order\_id], onDelete: Cascade)  ticket ticket @relation(fields: [ticket\_id], references: [ticket\_id], onDelete: Cascade)  @@index([temp\_order\_id])  @@index([ticket\_id])  }  enum OperatingDay {  Senin  Selasa  Rabu  Kamis  Jumat  Sabtu  Minggu  }  enum PaymentStatus {  pending  success  failed  }  enum TransactionStatus {  pending  paid  cancelled  }  enum Role {  admin  user  tourism\_owner  }  enum TicketStatus {  unverified  verified  } |

Dokumen schema.prisma milikmu telah berhasil direvisi sesuai kebutuhan integrasi Midtrans dan penggunaan temp\_order. Berikut perubahan utamanya:

**✅ Perubahan Utama:**

1. **Menghapus:**
   * payment\_method
   * payment\_provider
2. **Memodifikasi:**
   * payment → menambahkan payment\_type, menyederhanakan strukturnya
3. **Menambahkan:**
   * temp\_order
   * temp\_order\_detail

**🔁 Alur Sistem (Sesuai Struktur Baru)**

| **Tahap** | **Penjelasan** | **Tabel yang Digunakan** |
| --- | --- | --- |
| 1 | User memilih tiket dan jumlah | ticket |
| 2 | Simpan data order sementara + detail | temp\_order, temp\_order\_detail |
| 3 | Redirect ke Snap Midtrans (menggunakan order\_id) | - |
| 4 | Midtrans kirim callback ke server | transaction, transaction\_detail, payment |
| 5 | QR Code digenerate dan disimpan | ticket.url\_qr |
| 6 | Riwayat transaksi muncul di halaman user | transaction, payment |
| 7 | Admin memantau pembayaran | payment, transaction |

Kalau kamu siap, kita bisa lanjut ke:

* Membuat API penyimpanan ke temp\_order
* Menyesuaikan callback transaction/callback.ts agar ambil dari temp\_order
* Atau lanjut ke pembuatan halaman admin & user

Silakan pilih arah selanjutnya.