**Mengenal Teori Structured Query Language (SQL)**

Pada modul ini, Anda akan mempelajari dasar-dasar SQL, di antaranya mengenal teori Structured Query Language (SQL), mengidentifikasi Data Definition Language (DDL) dan Data Manipulation Language (DML), mempelajari komponen penyusun Constraints, serta mengenal jenis diagram pada SQL. Namun, sebelum lanjut ke penjelasan lebih dalam tentang SQL, sebaiknya Anda mengetahui sejarahnya terlebih dahulu. Yuk, kita simak di bawah ini.

**Sejarah SQL**

Keberadaan SQL sebagai bahasa yang biasa digunakan dalam pengaksesan data, sangat membantu dalam pengelolaannya secara digital dan memberikan kemudahan pengguna dalam mengakses data tersebut. Namun, tahukah Anda?

Pada tahun 1970, seorang peneliti IBM (International Business Machines) bernama Dr. E. F. Codd berusaha menerapkan matematika ke dunia penyimpanan dan pengambilan data. Codd mencoba mengarahkan pada model data relasional dan bahasa untuk memanipulasi data dalam basis data relasional yang disebut DSL/Alpha.

Dengan cara itu, IBM tertarik dan menugaskan pada Codd sebuah proyek bernama *System*atau yang kita kenal sebagai bahasa R.

Kemudian tim *System*/R mengembangkan versi DSL/Alpha yang disederhanakan menjadi *Square* dan berganti nama menjadi *Sequel,*lalu SQL.

Pada pertengahan 1980-an, SQL mendapatkan momentum yang cukup besar di pasaran sehingga mendapatkan pengawasan dari *American National Standards Institute*(ANSI) dan merilis standar SQL pertamanya pada tahun 1986, diikuti oleh pembaruan pada tahun 1989, 1992, dan 1999.

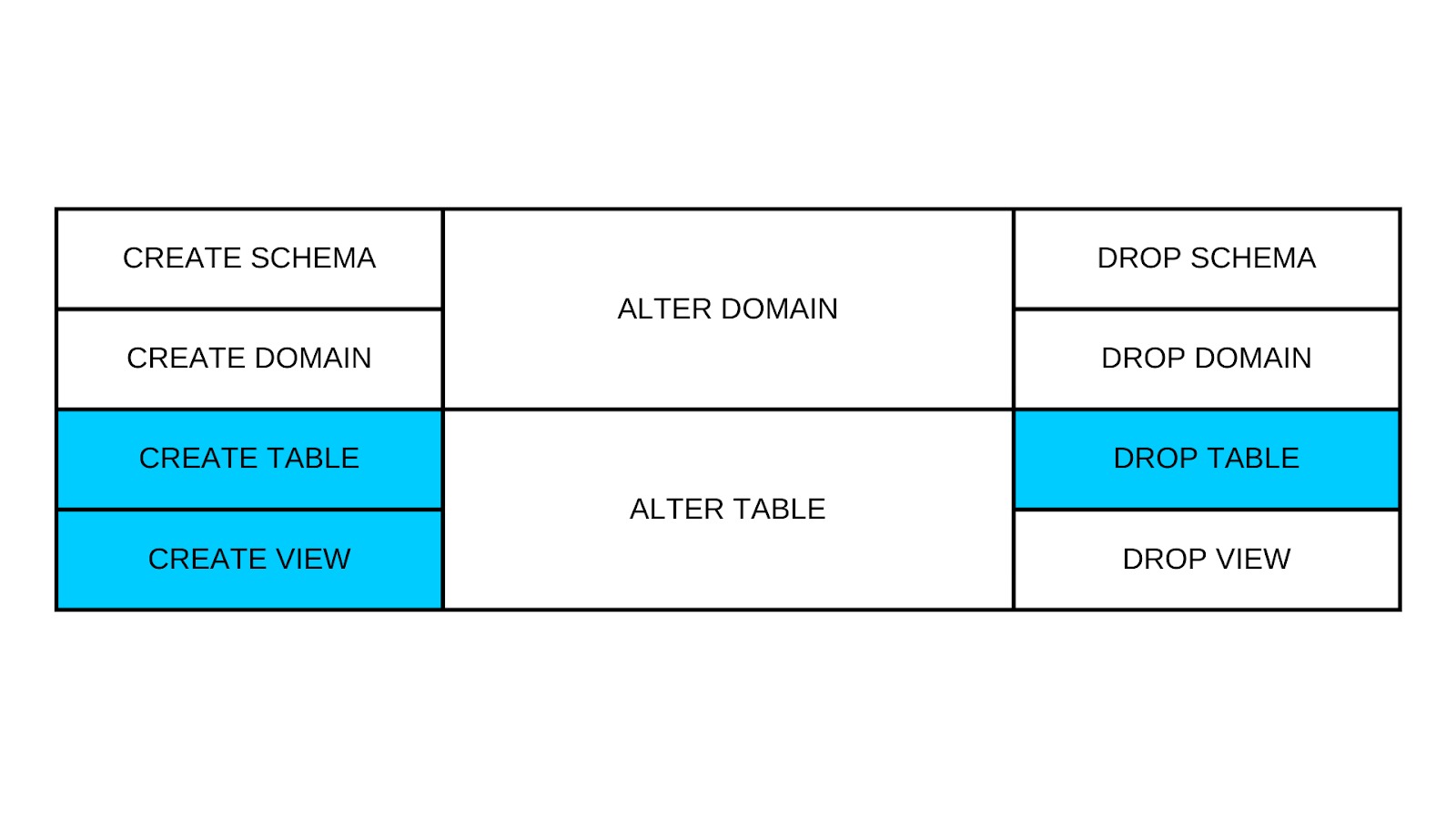
**Pengenalan Data Definition Language (DDL)**

Pada dasarnya, SQL digunakan dalam manajemen basis data relasional (*database relational*). Dengan menggunakan SQL, Anda dapat mengakses, menghapus, dan mengubah data, serta menjalankan query. Hingga saat ini, sebagian besar server basis data yang tersedia men-*support*SQL untuk menjalankan manajemen datanya. Maka dari itu, SQL merupakan bahan yang masih hangat diperbincangkan karena masih mengeluarkan inovasi dan masih dibutuhkan oleh publik yang bergelut dengan dunia data.

Dalam penggunaan SQL ada beberapa jenis perintah yang digunakan. Perintah tersebut dibutuhkan dalam mengakses dan memanajemen data dalam basis data. Salah satunya adalah *Data Definition Language*(DDL).

DDL merupakan subperintah pada SQL yang dimanfaatkan guna membangun kerangka basis data.

Berikut adalah pernyataan (*statements*) yang utama dari DDL.



Dalam kelas dasar ini Anda hanya akan mempelajari CREATE TABLE, DROP TABLE, dan CREATE VIEW yang akan dibahas lebih lanjut pada modul berikutnya.

**Pengenalan Data Manipulation Language (DML)**

Kemudian, jenis perintah SQL yang lain adalah *Data Manipulation Language*(DML). DML merupakan subperintah pada SQL yang dimanfaatkan dalam manipulasi basis data yang sudah dibuat. Pada dasarnya, dalam DML terdapat empat perintah berikut dengan fungsinya.

INSERT : Digunakan untuk memasukkan data baru dalam tabel. Perintah ini dapat dijalankan saat database dan tabel telah selesai dibuat.

SELECT : Digunakan untuk mengambil, lalu menampilkan data dari tabel atau sejumlah tabel dengan memanfaatkan relasi.

UPDATE : Digunakan untuk memperbarui data dalam tabel.

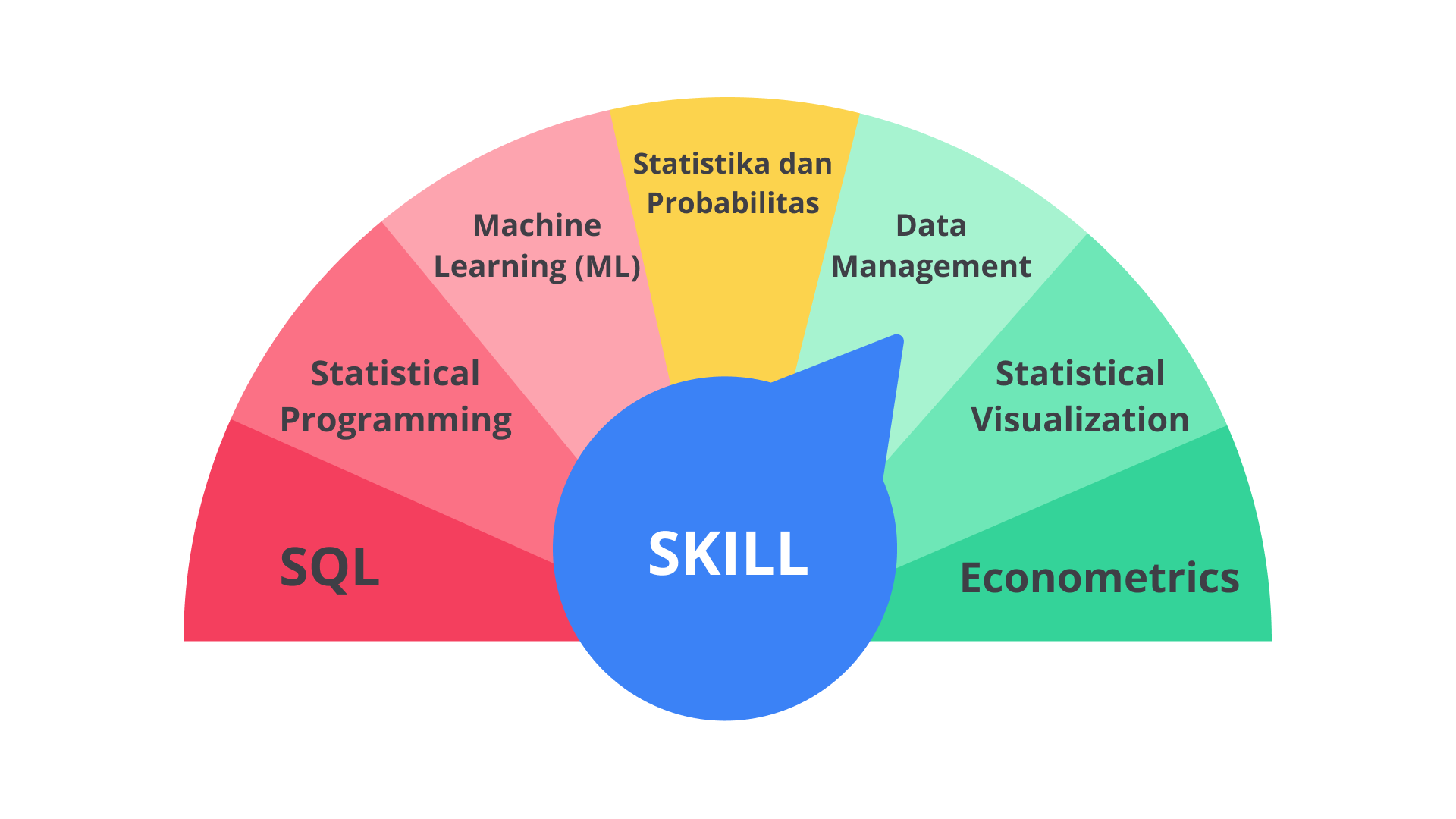
DELETE : Digunakan untuk menghapus data dari tabel.

**Fungsi Structured Query Language**

Setelah Anda mempelajari jenis-jenis perintah yang digunakan dalam SQL, ada baiknya pula Anda mengetahui fungsi SQL tersebut.

Pada dasarnya, SQL memiliki fungsi untuk mengubah ataupun mengakses basis data atau melakukan queryyang dibutuhkan. Adapun selain itu SQL juga berfungsi sebagai penghubung antara aplikasi dengan basis data karena ia hampir didukung oleh berbagai macam basis data.

**Tujuh Skill Data yang Dibutuhkan Tahun 2023**



1. **SQL**  
   Bahasa yang paling umum digunakan dalam dunia data untuk berinteraksi dengan basis data adalah SQL (Structured Query Language). SQL menempati urutan pertama dalam *skill* yang dibutuhkan karena hampir semua analis data perlu menggunakannya untuk mengakses data dari basis data perusahaan.
2. ***Statistical Programming***  
   Dalam dunia data tentunya sudah tidak asing lagi dengan bahasa pemrograman Python atau R. Kedua bahasa tersebut dapat Anda pilih salah satunya. Terkait perdebatan tentang bahasa yang lebih baik, hingga saat ini  masih terus berlangsung. Namun, keduanya memiliki persamaan, yaitu untuk membersihkan, menganalisis, serta menampilkan kumpulan data besar dan domainnya.
3. ***Machine Learning* (ML)**  
   Keahlian ini mampu mengembangkan algoritma yang memungkinkan komputer untuk memiliki kemampuan yang mirip seperti manusia dalam mengidentifikasi pola pada sebuah data. Pola ini selanjutnya akan digunakan untuk menghasilkan sebuah prediksi. Semakin banyak jumlah data yang digunakan maka semakin baik prediksi yang dihasilkan.
4. **Statistika dan Probabilitas**  
   Biasanya statistika terkenal dengan lingkup matematika dan sains yang berhubungan dengan data numerik dan penyajian serta interpretasinya. Saat ini, hal yang berhubungan dengan data sudah tidak asing lagi bagi seorang *data analyst*dan *data scientist*. Namun, tentunya sebelum mengemban amanah seorang *data analyst*atau *data scientist*, memiliki pemahaman yang kuat tentang statistika dan probabilitas dapat membantu Anda dalam mengerjakan hal berikut.
   * Menemukan tren dan pola dalam data.
   * Mengurangi prasangka, kekeliruan, atau kesalahan logis yang mengaburkan analisis Anda.
5. ***Data Management***  
   Kemampuan *data management*digunakan untuk menggambarkan proses yang terlibat dalam memperoleh, mengklasifikasikan, serta menyimpan informasi dengan cara yang tepat waktu, aman, dan murah.
6. ***Data Visualization***  
   Mampu melakukan visualisasi data menggunakan *chart,*diagram, *maps,* dan representasi visual lainnya dapat membantu menjelaskan temuan. Biasanya kemampuan ini dapat Anda pelajari melalui *tools*Tableau untuk data dan visualisasi.
7. ***Econometrics***  
   Pekerjaan di industri keuangan, seperti bank, membutuhkan analis data untuk memahami ekonometrika dengan kuat.

**Data Definition Language**

Data Definition Language (DDL) merupakan sub-perintah pada SQL yang dimanfaatkan guna membangun kerangka basis data.

**Pengantar Data Definition Language (DDL)**

Dalam materi sebelumnya, Anda telah mengenal definisi Data Definition Language (DDL) serta statement yang termasuk ke dalamnya.

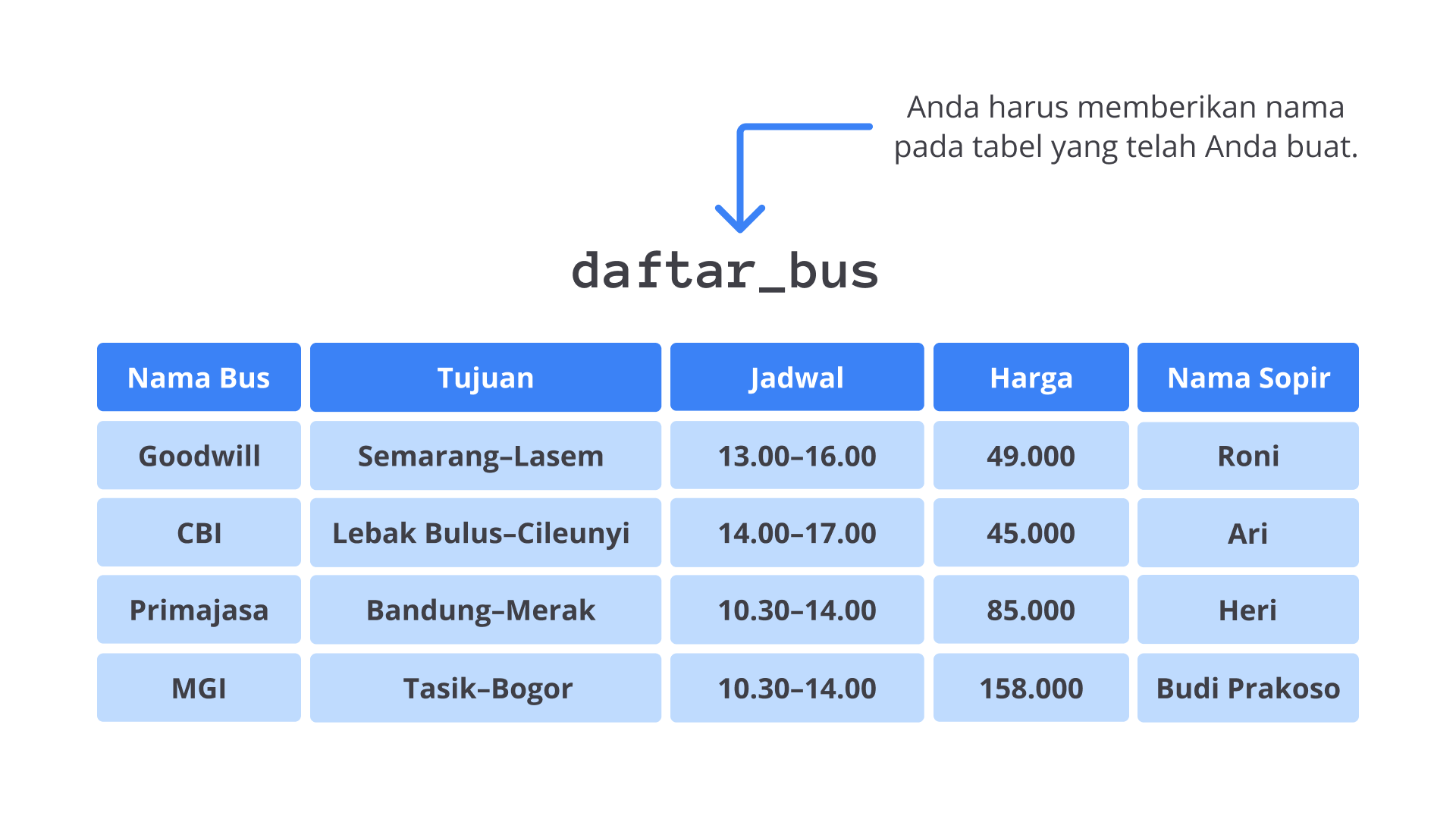
Pada bagian ini, Anda akan mempelajari lebih lanjut terkait CREATE TABLE, DROP TABLE, dan VIEW.

**CREATE TABLE**

**Mengenal Tabel**

Sebelum mempelajari terkait detail tabel, baris, dan kolom, mari kita kilas balik terkait basis data terlebih dahulu.

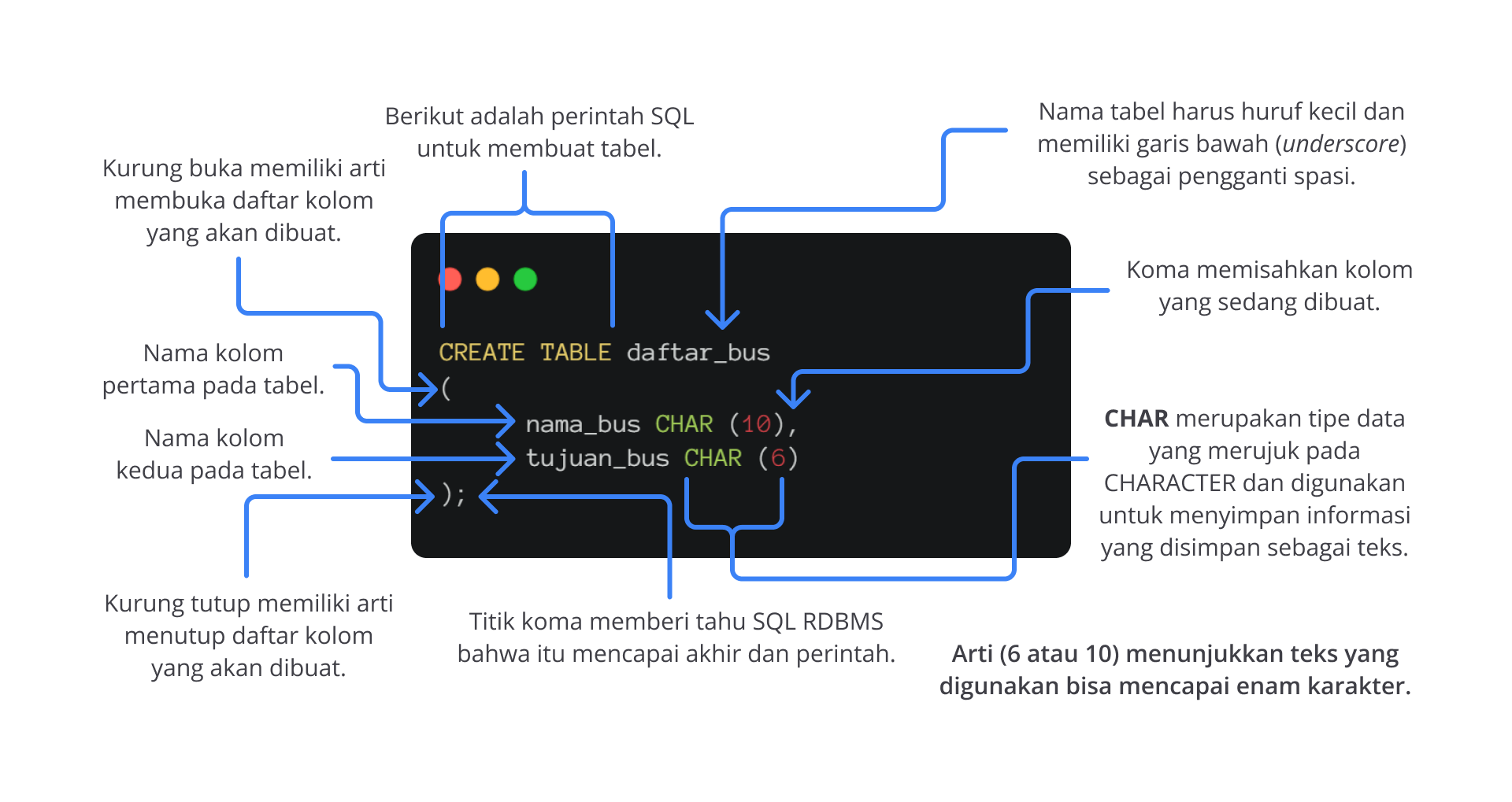
Dalam Modul Pengenalan Data dan Basis Data, dijelaskan bahwa basis data terdiri dari tabel-tabel yang saling berelasi dan terdiri dari kategori-kategori yang telah dibuat.



Mari kita beranjak ke materi inti, yaitu CREATE TABLE. Kita analogikan bahwa Anda merupakan seorang kondektur yang telah diberi jadwal oleh perusahaan dan harus bekerja lintas kota dengan jadwal yang tertera pada tabel di bawah ini. Katakanlah Anda kesulitan mengingat jadwal tersebut sehingga Anda membuat tabel agar tidak melewatkan tugas.

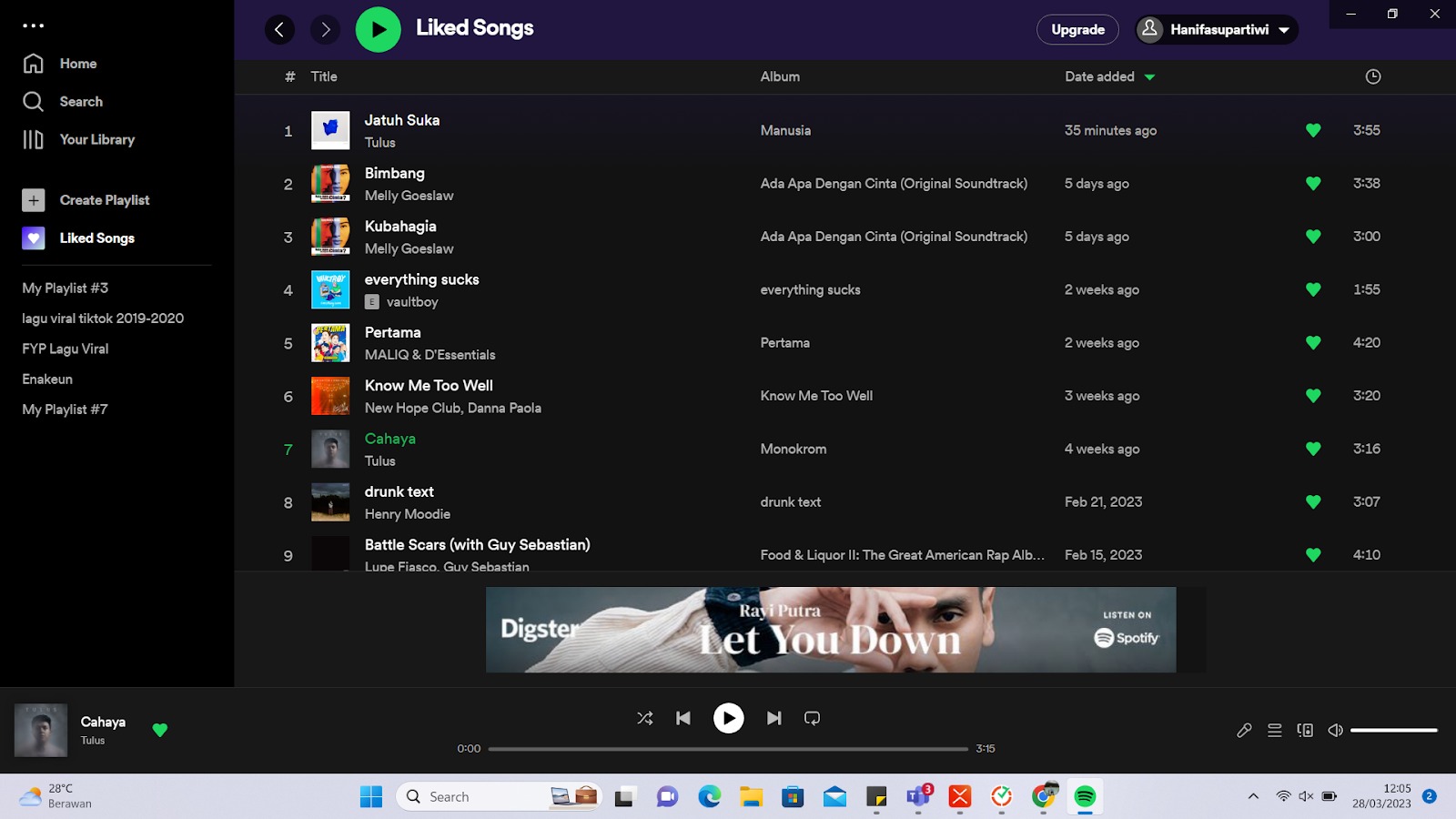
Di bawah ini merupakan jadwal dinas Anda sebagai kondektur bus yang telah disederhanakan dari tabel sebelumnya.

Perhatikan contoh berikut. Ini adalah implementasi dari tabel di atas sebelum Anda mencobanya pada DB Browser SQLite.



**Implementasi Tabel**

Setelah mempelajari CREATE TABLE pada materi sebelumnya, dalam materi ini Anda akan mengeksplorasi implementasi dari tabel yang sangat beragam.



Gambar di atas merupakan tampilan dari aplikasi Spotifyyang terdiri dari beberapa kategori, di antaranya *title,*album*, date added,*dan durasi.

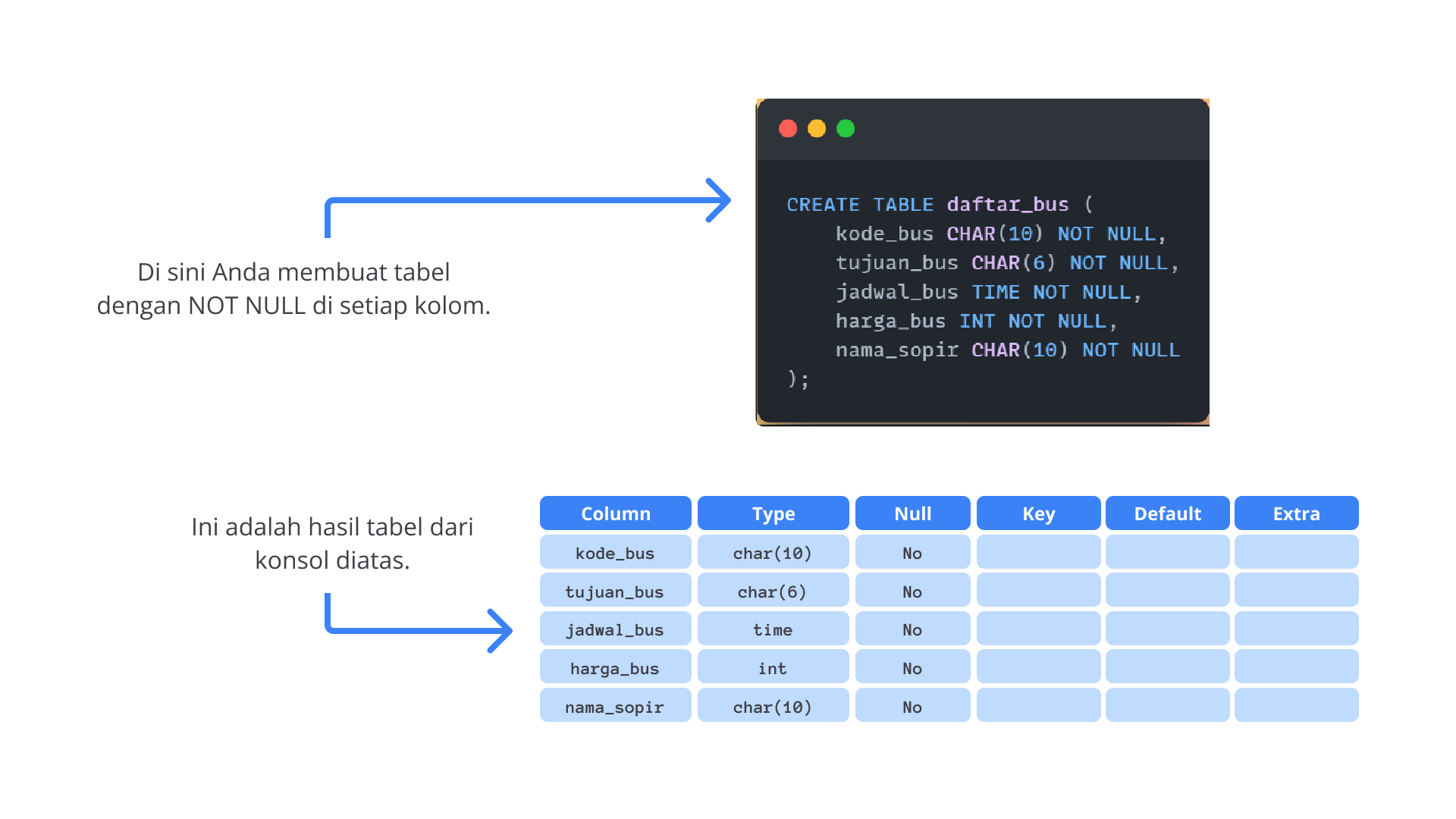
Gambar tersebut merupakan implementasi tabel pada sebuah aplikasi yang telah disusun sangat rapi dan menarik sehingga pembaca mudah dalam melihatnya.

**Tipe Data yang Didukung oleh Tabel**

Sebelumnya Anda telah mempelajari cara membuat tabel dengan tipe-tipe data pada SQL yang telah disampaikan juga pada pada Modul 1. Namun, apakah Anda tahu? Ada dua jenis tipe data yang harus Anda ketahui, yaitu NULL dan NOT NULL. Apakah itu?

NULL adalah sebuah tipe data spesial yang menunjukkan sebuah variabel belum memiliki nilai atau data dan NOT NULL adalah kebalikannya. Pada materi ini, Anda akan mengenal lebih dalam mengenai NOT NULL yang memiliki fungsi untuk memastikan bahwa nilai pada kolom tersebut tidak boleh kosong.

Perhatikan contoh NOT NULL di bawah ini dengan saksama.



**Membuat Tabel dengan SQL**

Setelah Anda mengenal dan mempelajari CREATE TABLE, saatnya Anda menguji pemahaman terkait materi yang telah dipelajari. Sebagaimana yang telah disebutkan pada modul sebelumnya, kita akan menggunakan DB Browser SQLite untuk latihan di kelas ini.

**DROP TABLE**

**Mengenal DROP TABLE**

Seiring waktu, tabel pada basis data akan berubah sesuai kebutuhannya, baik bertambah maupun berkurang. Apabila untuk menambahkan tabel Anda telah mempelajari pada materi sebelumnya, yaitu CREATE TABLE; nah,berkebalikan dengan menambahkan (CREATE TABLE), DROP TABLE berfungsi untuk menghapus tabel dari basis data.

**Catatan:**

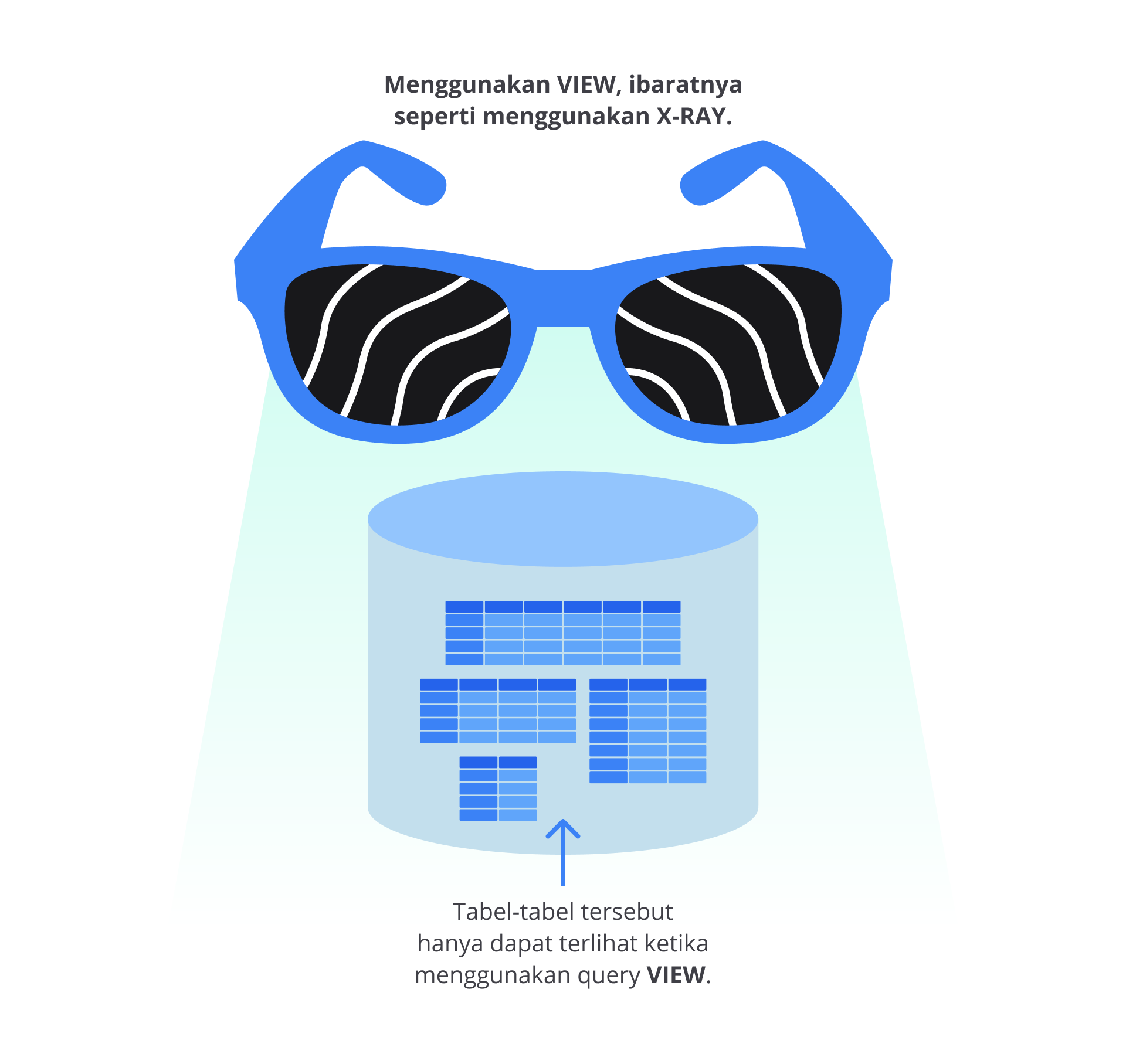
* DROP TABLE dapat digunakan pada *multi table* (banyak tabel) tetapi tidak akan menghapus seluruh tabel tersebut. Jadi, Anda cukup memanggil tabel yang akan dihapus, maka tabel tersebut tidak akan tersedia lagi.

**VIEW**

**Mengenal VIEW**

VIEWatau tabel virtual adalah tabel yang hanya ada saat Anda menggunakan *view*dalam query. Tujuan dibuatnya VIEWadalah untuk mempermudah penulisan query dan juga alasan keamanan karena dapat menyembunyikan beberapa kolom yang bersifat rahasia, atau dalam beberapa kasus digunakan untuk mempercepat proses menampilkan data. Ini juga dianggap sebagai tabel virtual karena berfungsi seperti tabel dan operasi yang sama.

Namun, tabel virtual tidak tinggal di basis data. Itu dibuat saat Anda menggunakan *view*dan kemudian dihapus. Simak analogi VIEW pada gambar di bawah ini.



Penggunaan VIEW ibarat Anda dapat melihat sebuah objek dengan hanya menggunakan x-ray, misalnya organ dalam anggota tubuh. Organ-organ tersebut hanya akan terlihat apabila menggunakan x-ray. Nah,fungsi dari VIEW juga untuk melihat sesuatu yang tidak bisa terlihat secara langsung.

Mengapa Menggunakan VIEW?

1. **Anda dapat menyimpan perubahan pada struktur basis data agar tidak mengganggu aplikasi yang bergantung pada tabel Anda.**  
   Sederhananya, Anda akan membuat tampilan data serta dapat mengubah struktur tabel yang mendasarinya. Nah, tetapi Anda sebelumnya membuat tampilan yang meniru struktur tabel terlebih dahulu sehingga tidak perlu mengubah aplikasi menggunakan data.
2. **VIEW membuat pekerjaan Anda lebih mudah dengan menyederhanakan query kompleks menjadi perintah sederhana.**  
   Anda tidak perlu membuat gabungan dan subquery yang rumit berulang kali saat membuat VIEW. Misalnya ketika Anda menghubungkan SQL ke PHP atau bahasa pemrograman lain, VIEW akan menyembunyikan queryyang rumit, lalu menyederhanakannya. Sungguh menarik, bukan?
3. **Anda dapat menyembunyikan informasi yang tidak diperlukan oleh pengguna.**  
   Akan ada kondisi ketika Anda membuat tabel yang berisi informasi-informasi yang rahasia, tetapi aplikasi tersebut tetap dapat diakses oleh pengguna. Solusinya adalah menggunakan VIEW. Anda dapat mengizinkan pengguna hanya mengakses informasi yang mereka butuhkan, sambil tetap menyembunyikan informasi sensitif.

**Data Manipulation Language**

Data Manipulation Language (DML) merupakan sub-perintah pada SQL yang dimanfaatkan dalam manipulasi basis data yang sudah dibuat. Nah, pada modul ini, Anda akan mengenali, mengidentifikasi penggunaan query, serta latihan untuk meningkatkan kemampuan dalam SQL. Penasaran? Yuk, simak penjelasannya.

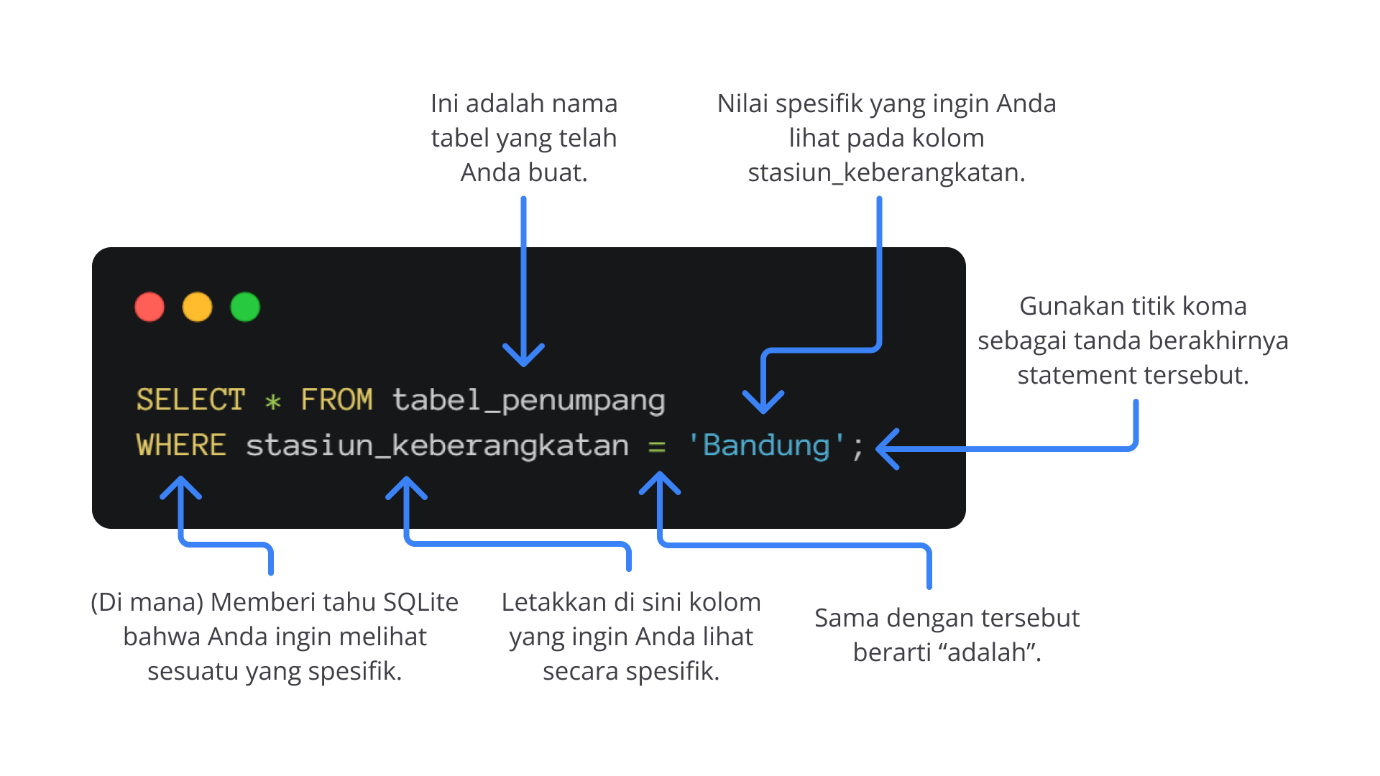
**SELECT**

Sebelum menjelaskannya secara teori, mari kita analogikan terlebih dahulu agar Anda paham cara kerja dari SELECT.

Anda merupakan seorang *train attendant*yang memiliki sejumlah data penumpang untuk keberangkatan Bandung–Jakarta. Satu waktu, kondektur Anda meminta data penumpang dengan pemberangkatan di stasiun Bandung, sedangkan tabel data tercampur antara penumpang yang berasal dari Stasiun Bandung, Stasiun Cimahi, Stasiun Purwakarta, Stasiun Bekasi, dan Stasiun Gambir.

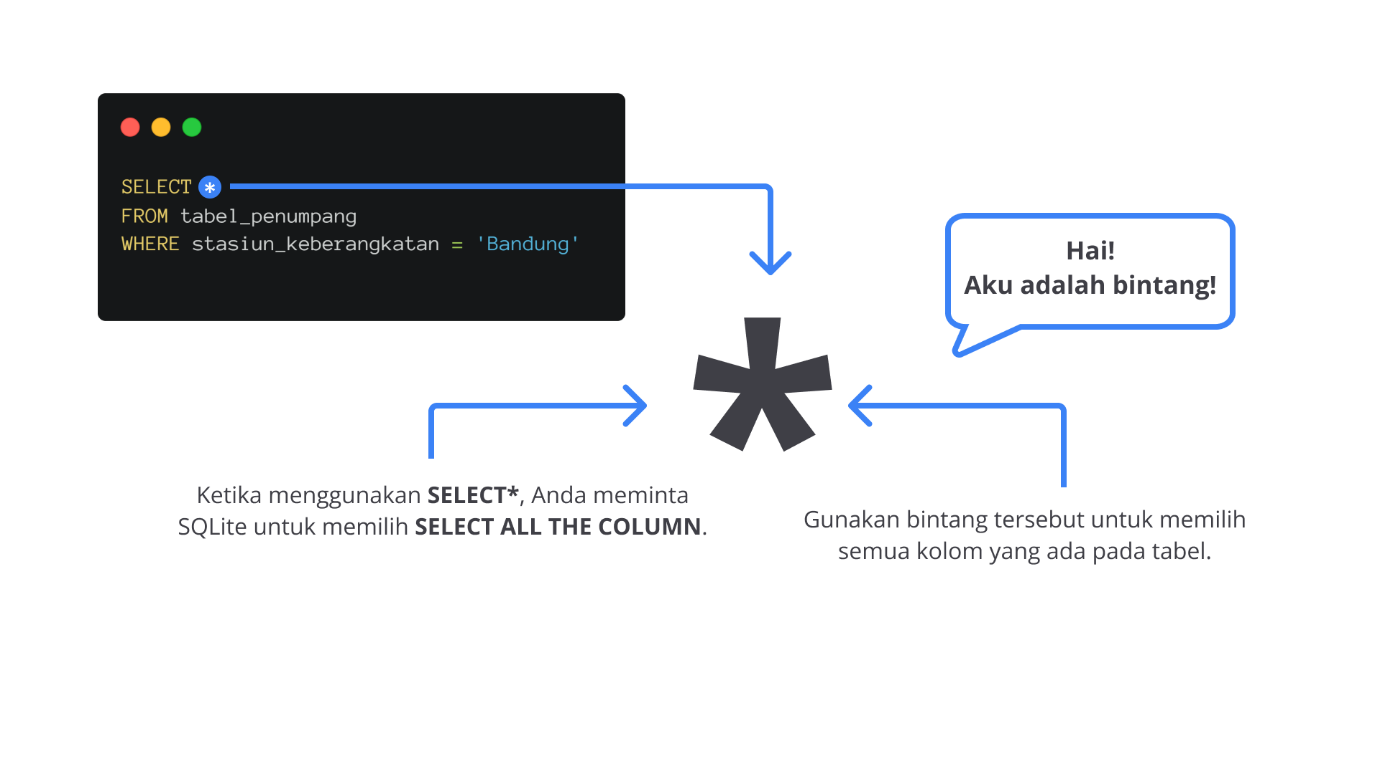
Pernyataan SELECT akan membantu Anda menemukan semua penumpang yang terdapat dalam tabel menggunakan queryseperti berikut.

Dengan SELECT, Anda juga dapat memilih penumpang yang keberangkatannya berasal dari stasiun tertentu, contohnya keberangkatan Stasiun Bandung. Anda dapat menggunakan queryyang dibantu oleh pernyataan WHERE, seperti tertera pada penjelasan di bawah ini.



**Apa arti dari \***

Bintang tersebut memberi tahu RDBMS (Relational Database Management System) untuk memilih semua kolom pada tabel Anda.

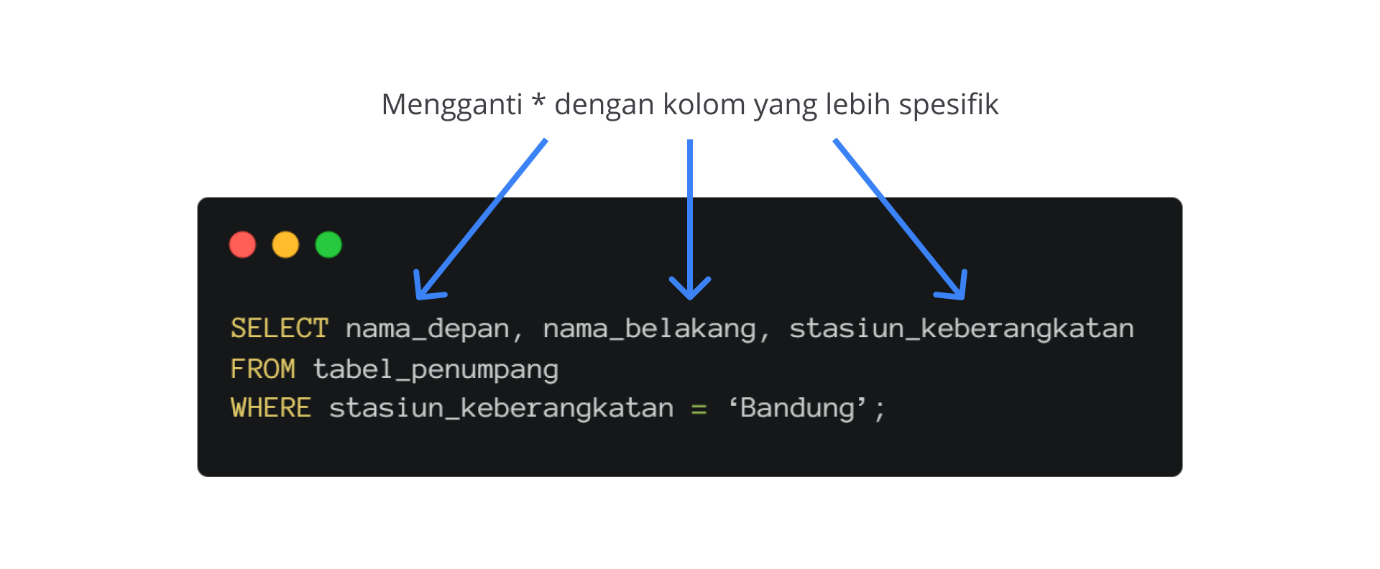


**Cara Menggunakan SELECT**

Anda telah paham cara kerja SELECT menggunakan tanda bintang (\*), lalu bagaimana caranya agar kita dapat menggunakan pernyataan tersebut untuk item-item tertentu saja?

Sebelumnya tanda bintang digunakan untuk **memilih semua kolom**. Nah, jika ingin melihat kolom tertentu, Anda bisa menuliskan kolom-kolom yang akan ditampilkan pada konsol. Simak penjelasannya di bawah ini.

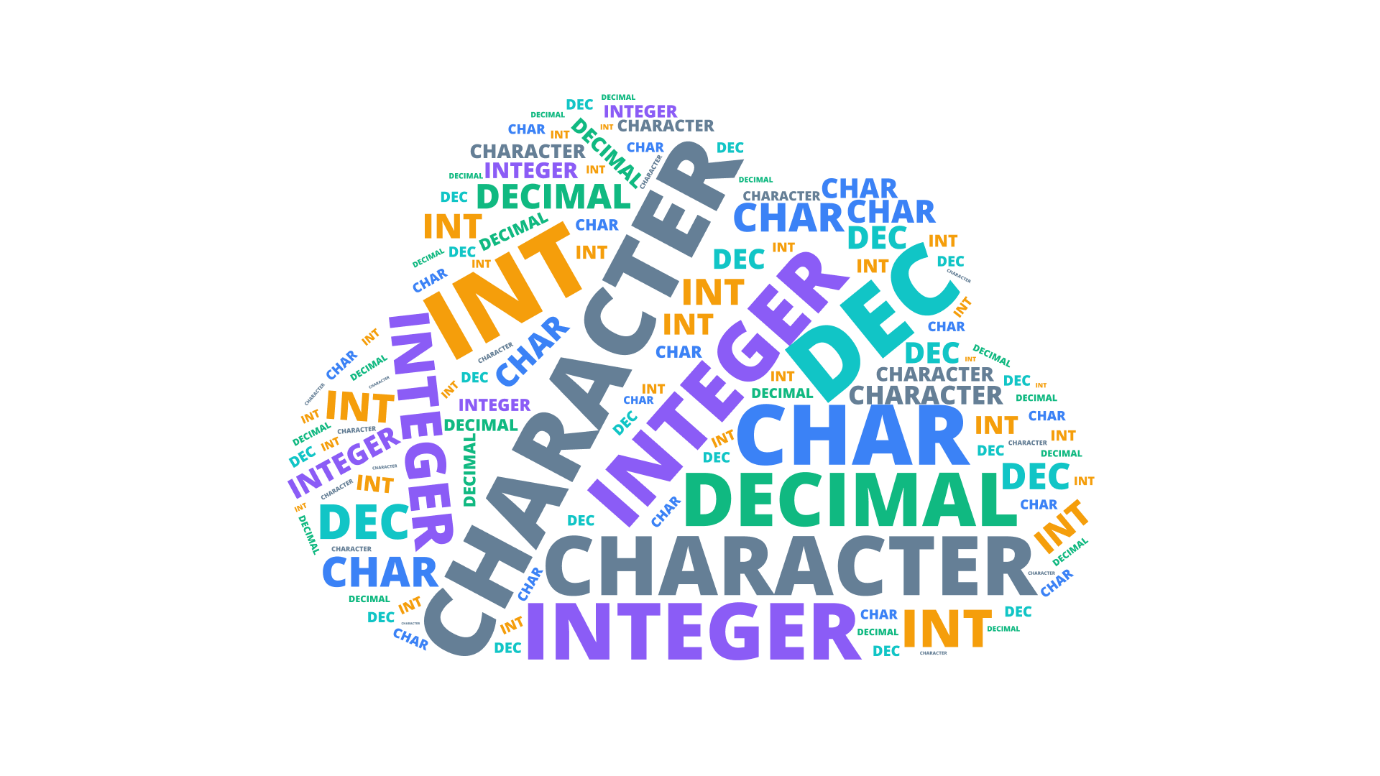
**SELECT untuk Data Spesifik**



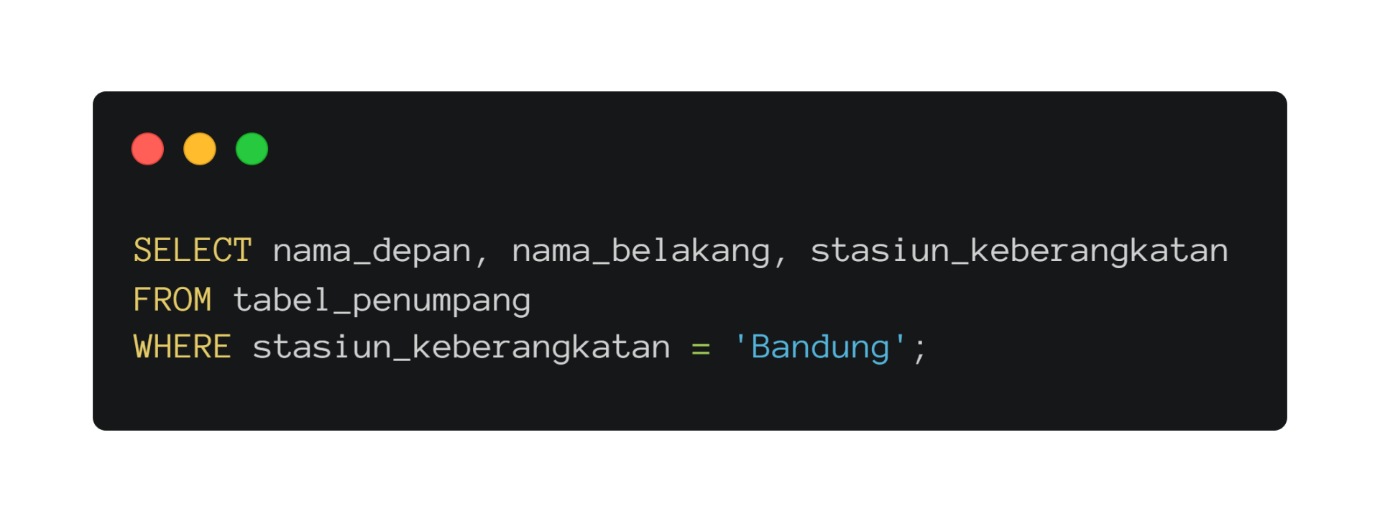
Setelah menulis query SELECT diikuti nama kolom yang akan Anda tampilkan, contohnya nama\_depan, nama\_belakang, stasiun\_keberangkatan. Lalu, tuliskan FROM, yaitu asal tabel Anda didapatkan, dan WHERE, yaitu spesifikasi kata yang Anda cari di konsol tersebut. Selanjutnya, tutup dengan tanda titik koma (;) yang mengartikan query tersebut sudah selesai.

**Cara Menggunakan WHERE**

Untuk menulis pernyataan WHERE yang valid, Anda perlu memastikan setiap tipe data yang Anda sertakan adalah benar. Berikut adalah tipe-tipe data yang dapat Anda gunakan.



Sederhananya, ketika Anda ingin menuliskan query dengan tipe data CHAR, INT, dan BLOB, perhatikan contohnya di bawah ini.



Anda memerlukan tanda petik tunggal (‘    ‘) sesudah sama dengan (=) ketika menjalankan program tersebut. Untuk tipe data yang termasuk dalam NUMERIC, yaitu DEC atau DECIMAL, Anda tidak perlu menambahkan tanda petik sesudah tanda sama dengan (=).



Nah, setelah Anda mempelajari SELECT, bagian kedua dari DML adalah INSERT. Yuk, simak penjelasannya di bawah ini.

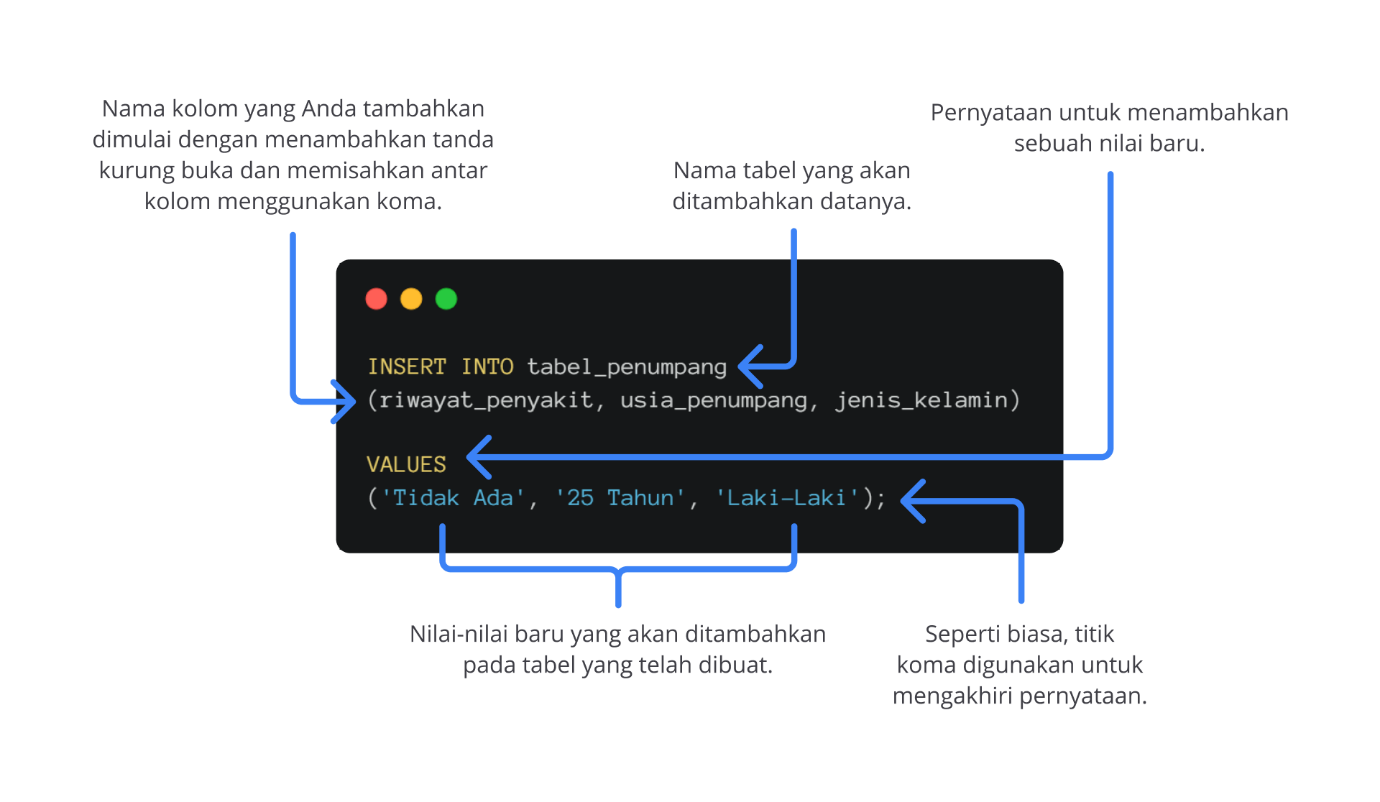
**INSERT**

Anda dapat lihat query di bawah ini yang merupakan contoh dari penggunaan INSERT. INSERT berfungsi untuk memasukkan nilai *record*baru dalam sebuah tabel pada basis data. Di bawah ini merupakan queryapabila Anda menggunakan pernyataan INSERT.



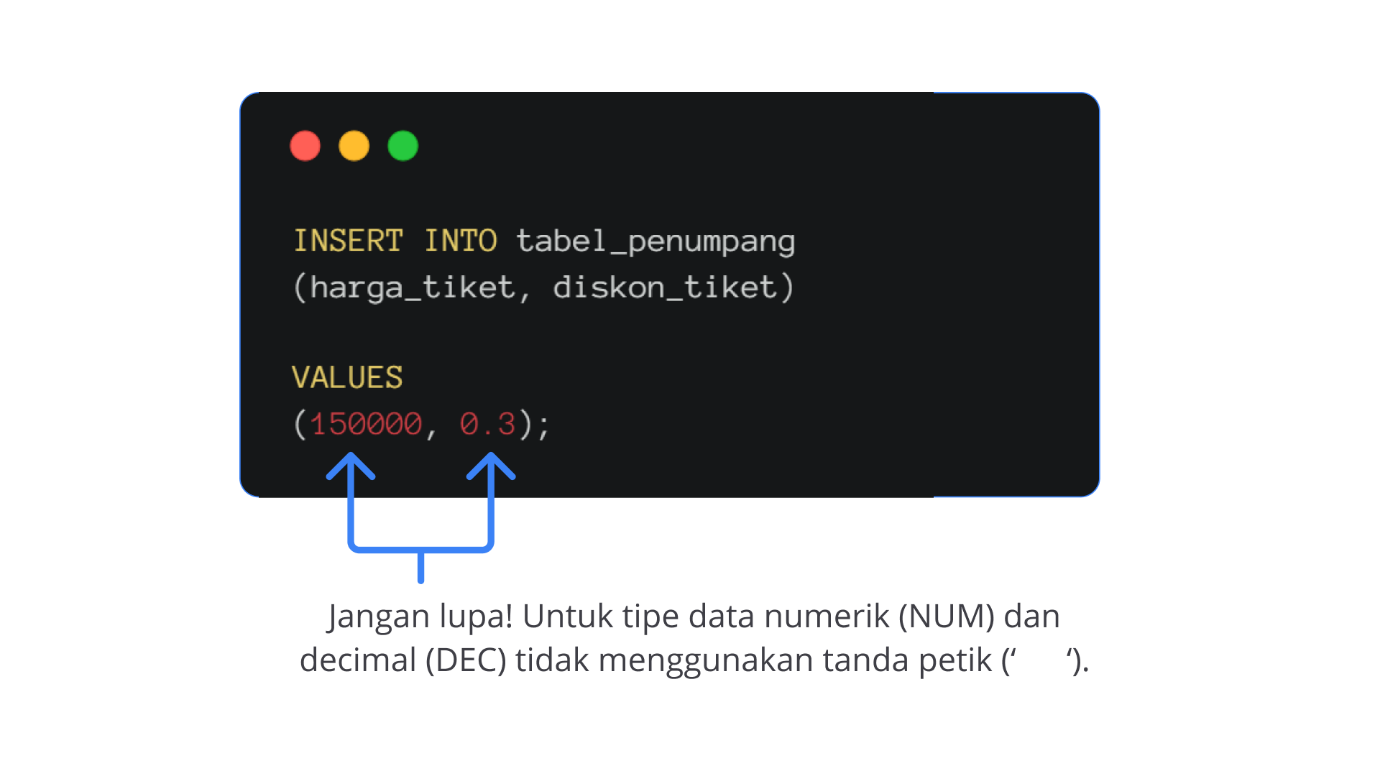
**Cara Menggunakan INSERT**

Setiap nilai dalam tanda kurung kedua harus dalam urutan yang sama dengan nama kolom.



Lalu, bagaimanakah apabila Anda ingin memasukkan data harga tiket kereta dan diskon tiket?

Anda dapat menambahkan tipe data berupa numerik (NUM) atau desimal (DEC) seperti yang telah dicontohkan pada gambar berikut.



**DELETE**

Setelah mempelajari SELECT dan INSERT, DELETE adalah pernyataan untuk menghapus nilai yang tidak diperlukan lagi.

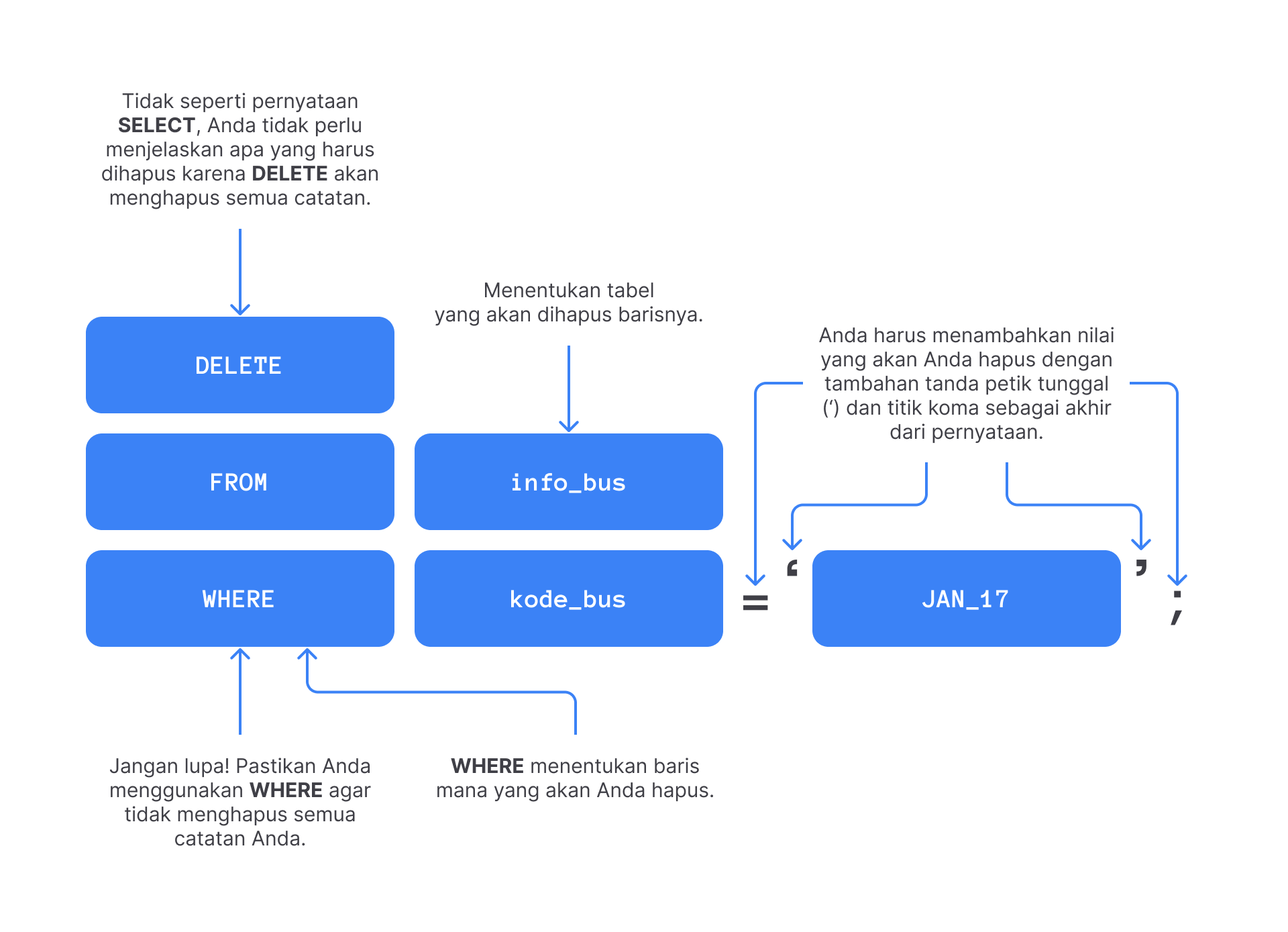
Sebelum mempelajari DELETE secara teori, perhatikan studi kasus di bawah ini agar Anda paham cara kerja dari pernyataan DELETE.

Mari kita ambil contoh, Anda seorang penumpang yang meminta bantuan kepada kondektur bus untuk melacak keberadaan bus Primajasa jurusan Bandung–Bogor. Sang kondektur membuat tabel info\_bus untuk melacaknya, serta bisa menggunakan kolom terakhir\_dilihat untuk melacak keberadaan bus tersebut.

dua data yang saling bertumpuk/*double* harus dipastikan untuk menghapus data tersebut. Sebab, apabila terjadi data *double*akan memperberat cara kerja *software*SQL. Coba bayangkan, bagaimana apabila terjadi data *double* pada seribu tabel? Sangat menyusahkan, bukan?

Lalu, bagaimana cara menghapus data-data duplikat tersebut?

**Cara Menggunakan DELETE - WHERE**

Pada materi ini, Anda akan mempelajari DELETE bersama WHERE dan DELETE bersama INSERT. Pernyataan ini bisa jadi merupakan sebuah kebalikan dari pernyataan SELECT. Ketika SELECT dan WHERE adalah kegiatan untuk mencari dengan kalimat tertentu, sedangkan DELETE dan WHERE adalah kegiatan untuk menghapus kalimat tertentu (latihan akan diberikan pada materi selanjutnya).

Perlu digarisbawahi, perbedaan antara DROP\_TABLE dengan DELETE adalah DROP\_TABLE bertugas untuk menghapus seluruh baris dan kolom pada tabel, sedangkan DELETE bertugas untuk menghapus beberapa baris yang Anda tuju. Lebih lengkapnya, mari kita simak penjelasannya di bawah ini.

**Ketentuan Menggunakan DELETE**

1. Anda tidak dapat menggunakan pernyataan DELETE untuk menghapus baris atau kolom pada sebuah tabel.
2. Anda dapat menggunakan DELETE untuk menghapus satu atau beberapa baris, bergantung pada pernyataan WHERE yang mengeliminasi data.
3. WHERE digunakan untuk memilih data yang akan dihapus.

**Catatan:** Hati-hati dengan menggunakan DELETE!

Setiap kali menghapus catatan, Anda berisiko menghapus catatan yang tidak ingin dihapus secara sengaja. Pastikan Anda menyertakan pernyataan WHERE yang tepat untuk menargetkan baris-baris yang akan dihapus.



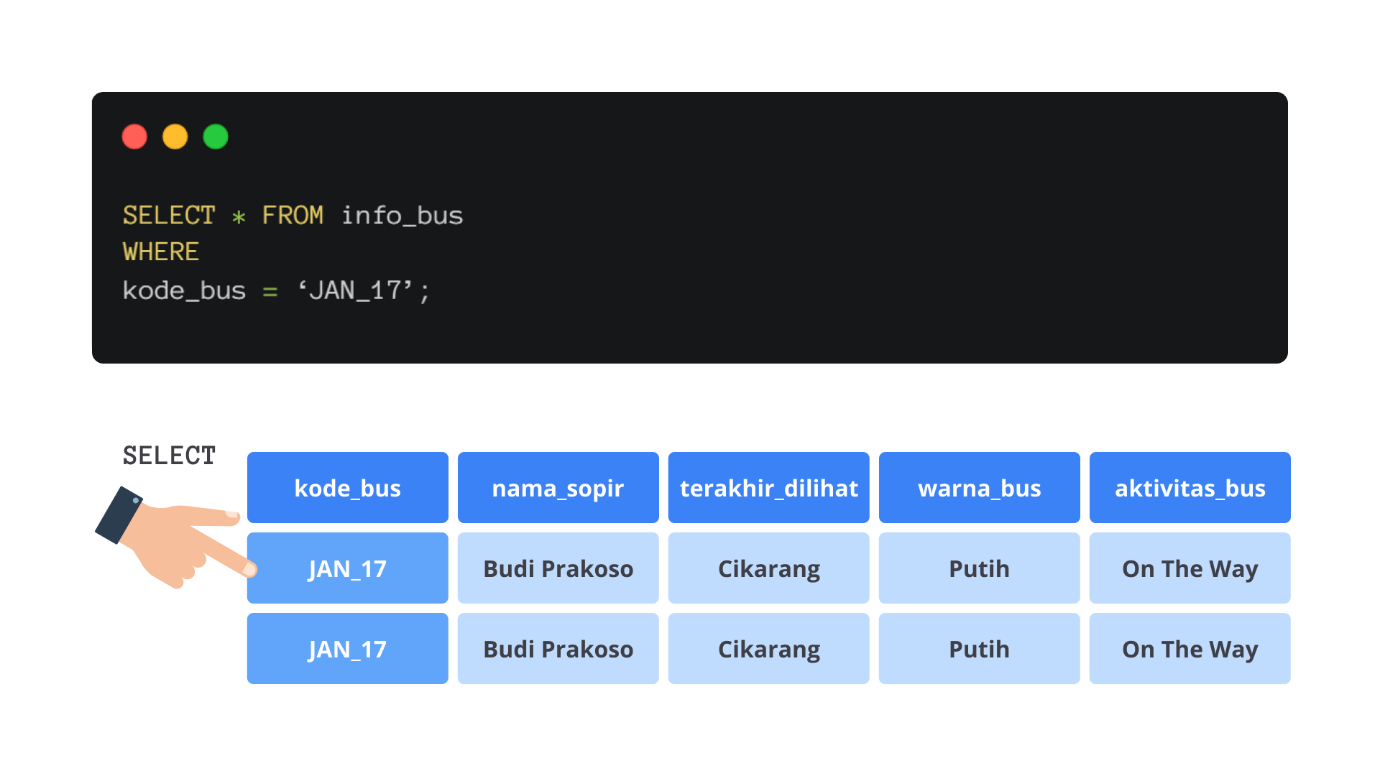
Ketika Anda ingin menghapus baris tersebut, pastikan pernyataan WHERE Anda langsung merujuk pada baris tersebut. Apabila tidak, ada kemungkinan Anda akan kehilangan data lain dengan spesifikasi yang hampir mirip tetapi tidak serupa.

Misalnya, Anda ingin menghapus nama sopir Budi Prakoso dalam tabel tersebut karena terjadi duplikasi, dengan posisi terakhir dilihat berada di Cikarang, warna bus putih, dan aktivitas bus sedang *On The Way.*

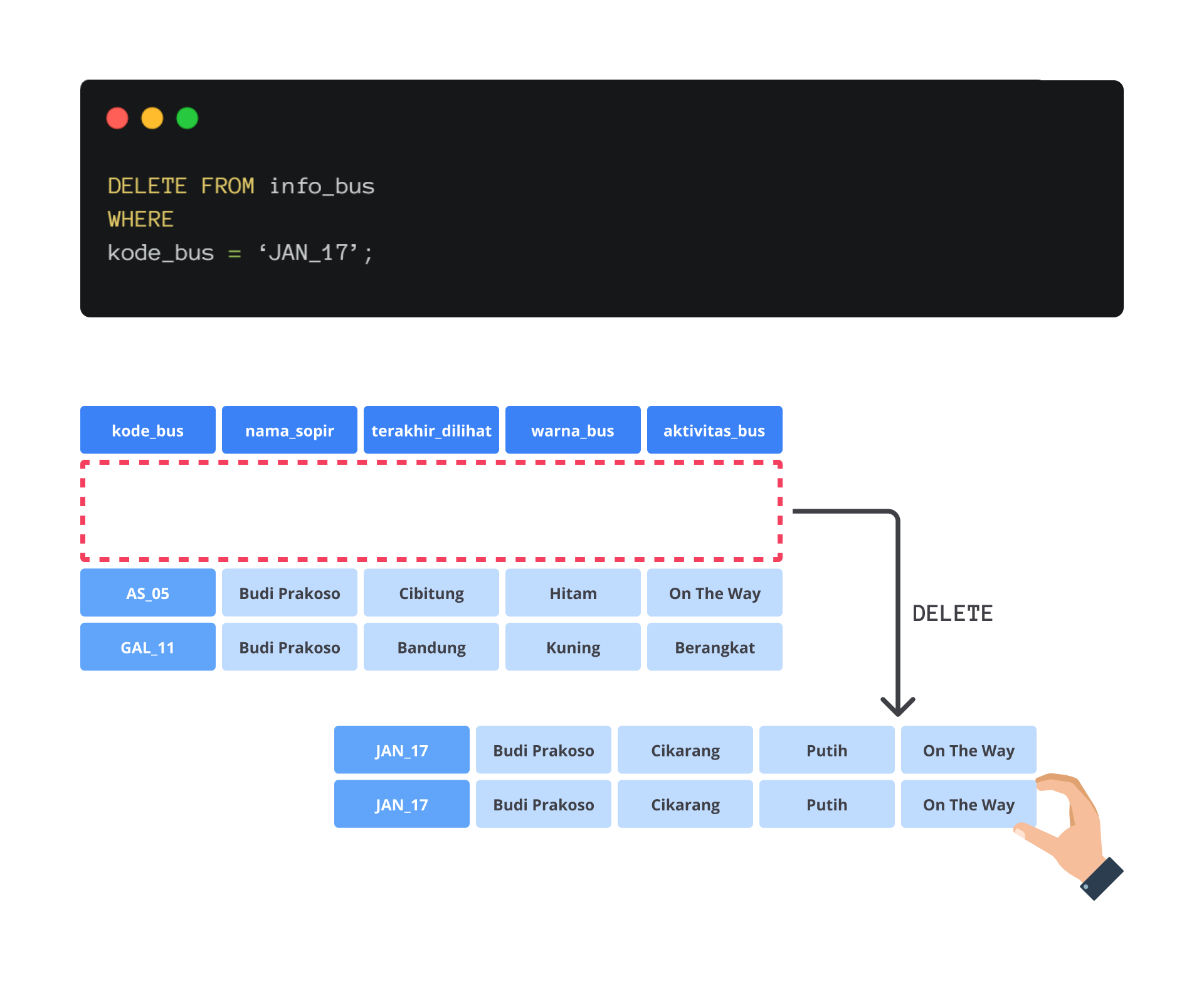
Apabila tidak seperti itu, misalnya hanya mencari nama sopir Budi Prakoso, atau hanya atributnya saja, atau aktivitasnya saja, kemungkinan besar Anda akan kehilangan data lain yang seharusnya tidak hilang. Sebab, konsol akan menyeleksi sesuai perintah saja.

Lalu, bagaimana caranya menggunakan DELETE dengan WHERE? Simak penjelasannya di bawah ini, ya.

1. Pilih catatan yang akan Anda hapus menggunakan SELECT.



1. Kemudian, gunakan DELETE untuk data yang lama dengan pernyataan WHERE yang sama dengan SELECT di atas.



**UPDATE**

Setelah mempelajari DELETE, WHERE, dan INSERT dalam materi sebelumnya, setidaknya Anda sudah paham cara kerja dari DELETE. Selain menggunakan tiga pernyataan tersebut, Anda bisa menggunakan kembali baris yang sudah ada pada tabel dan hanya mengubah nilai kolom yang ingin diubah.

Pernyataan tersebut bernama UPDATE. Pernyataan tersebut berfungsi memperbarui kolom ke nilai yang baru, seperti namanya. Layaknya SELECT dan DELETE, Anda harus memberinya pernyataan WHERE untuk menunjukkan baris yang ingin diterapkan UPDATE.

**Syarat Menggunakan UPDATE**

1. Anda dapat menggunakan UPDATE untuk mengubah nilai satu kolom atau tabel kolom. Tambahkan *column = value*ke pernyataan SET dan beri tanda titik koma di akhir.
2. Anda dapat menggunakan UPDATE untuk memperbarui satu baris atau beberapa baris, tergantung pada pernyataan WHERE.

Bagaimana apabila kita gabungkan antara UPDATE, INSERT, dan DELETE?

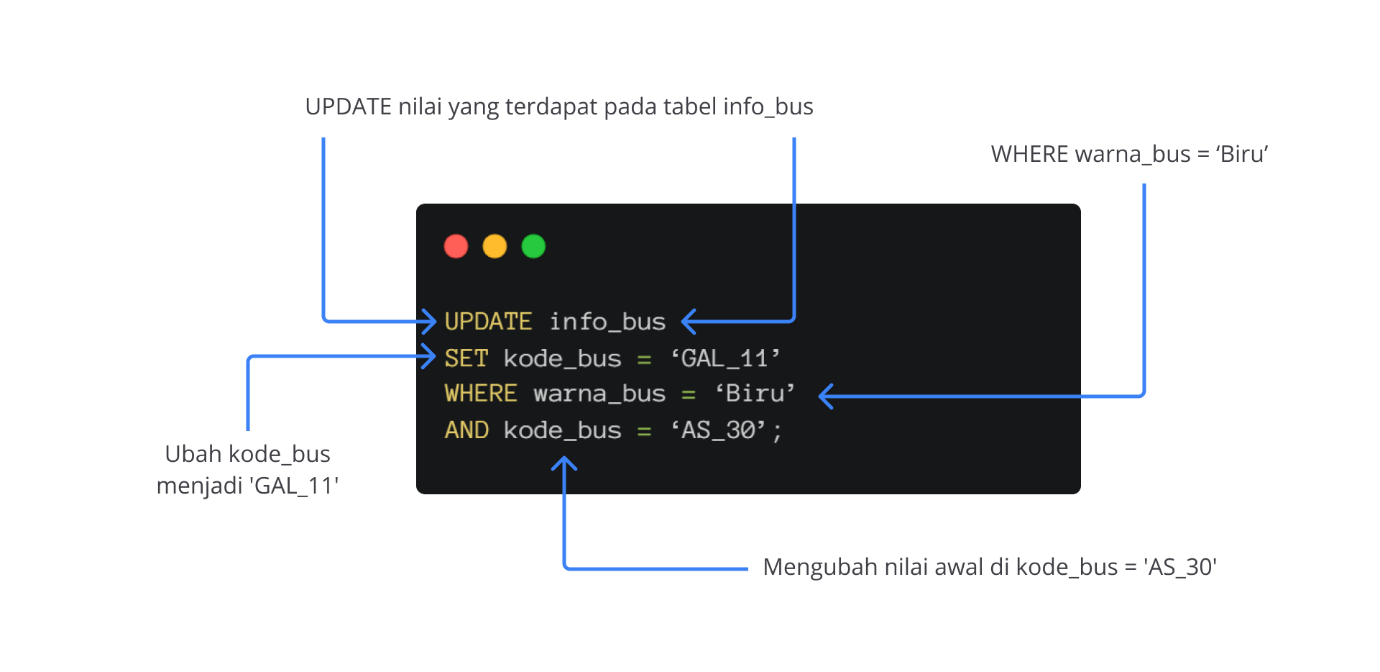
Nah, saat memutuskan untuk menggunakan UPDATE, Anda tidak menghapus apa pun. Alih-alih, Anda mendaur ulang data menjadi yang baru. Contohnya seperti berikut.



Kata kunci SET tersebut memberi tahu RDBMS bahwa ia perlu mengubah kolom sebelum tanda sama dengan dengan pernyataan WHERE untuk memuat nilai setelah tanda sama dengan.

Sebab, pada dasarnya pernyataan UPDATE dapat menggantikan kombinasi antara INSERT dan DELETE.

Perhatikan contoh di bawah ini yang akan digunakan pada tabel info\_bus.



**Constraints**

Di antara banyak hal yang perlu dipelajari dalam SQL, salah satunya adalah *constraints.*Pada modul ini, Anda akan mengidentifikasi cara kerja dari komponen-komponen penyusun constraints,di antaranya NOT NULL, Unique, Primary Key, dan Index. Penasaran? Yuk, simak penjelasannya!

**Mengenal Constraints**

Saat mempelajari basis data, pada umumnya kita harus memahami fungsi dan cara kerja dari *constraint*.Sebab, saat membuat tabel kita harus mendeklarasikan atribut-atributnya, tipe data masing-masing atribut, dan *constraint*-nya apabila ada. Lalu, apa itu *constraint*?

*Constraint*adalah batasan yang ditempatkan dalam satu atau lebih kolom tabel. Ada beberapa jenis *constraint*,di antaranya *primary key, foreign key, unique,*dan *check.*Pada kelas Belajar Dasar Structured Query Language (SQL) ini, Anda hanya akan mempelajari *primary key*dan *unique*karena hanya akan fokus terhadap tabel tunggal, sedangkan untuk *foreign key*dan *check*bisa Anda dapatkan di kelas tingkat lanjutan.

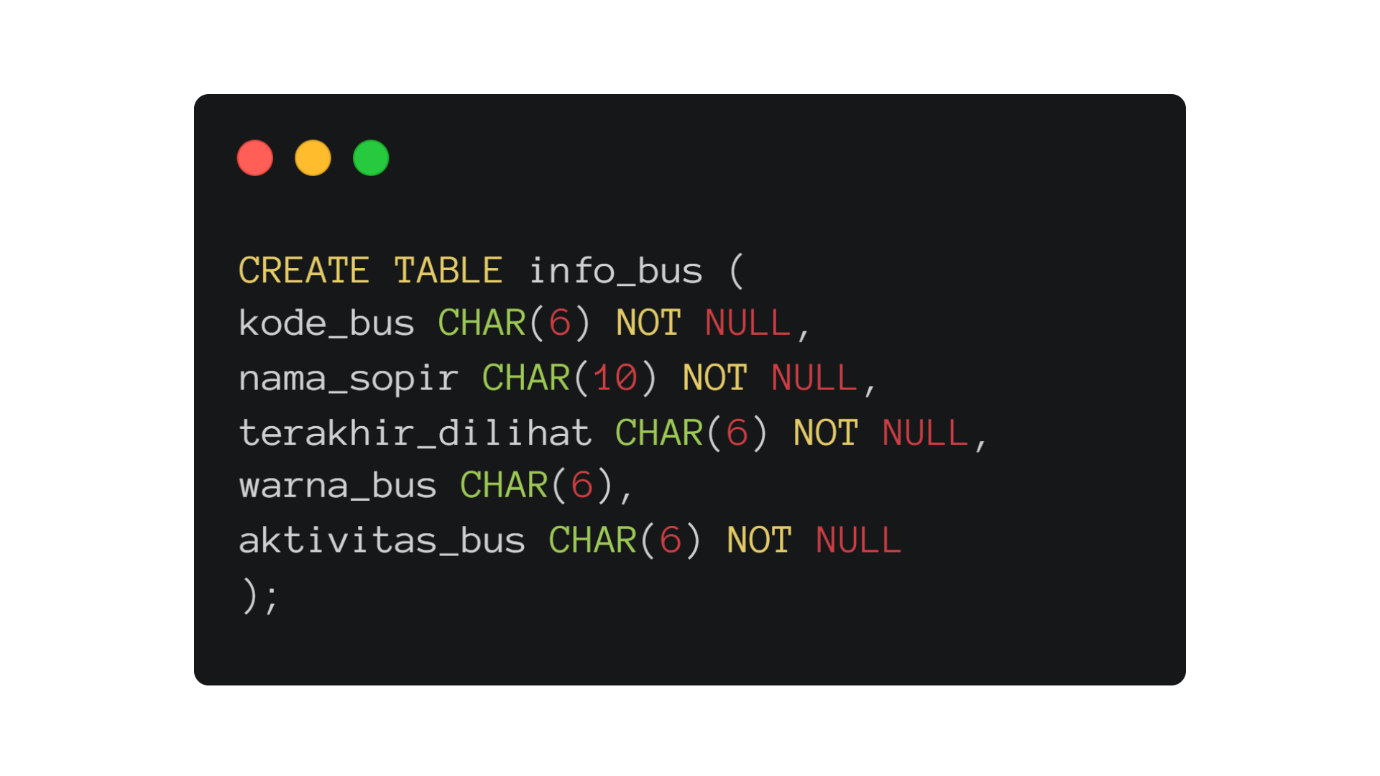
Sebelum Anda mengenal lebih jauh mengenai teori dari *primary key*dan *unique*,perhatikan terlebih dahulu analogi yang telah disusun agar Anda lebih paham terkait cara kerja dari komponen-komponen penyusun *constraint.*



Anda merupakan seorang kondektur bus Bahagia dengan tujuan Bandung–Surabaya dan titik pemberhentian Bandung, Jogja, Malang, dan Surabaya. Anda memiliki sejumlah data penumpang yang memiliki spesifikasi nomor id, nama lengkap, asal keberangkatan, dan kelas bus. Untuk menyusun nama-nama tersebut di dalam tabel, Anda membutuhkan bantuan NOT NULL, *unique,*dan *primary key.*Perhatikan implementasinya di bawah ini.

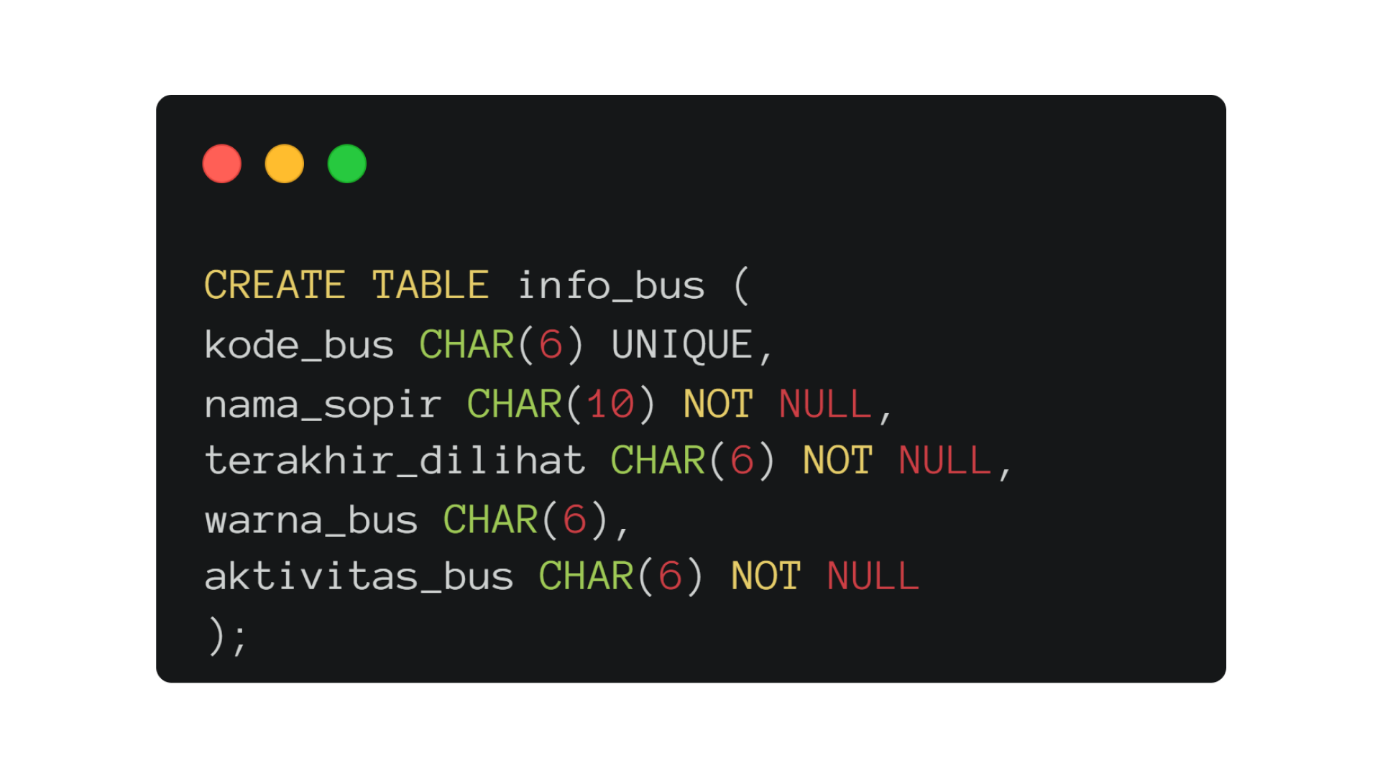
**NOT NULL**

Berfungsi untuk memastikan bahwa setiap kolom tabel memiliki nilai dan tidak kosong. Perlu digarisbawahi bahwa kosong dan nol (0) adalah berbeda. Kosong berarti tidak memiliki nilai, sedangkan nol (0) adalah sebuah nilai.

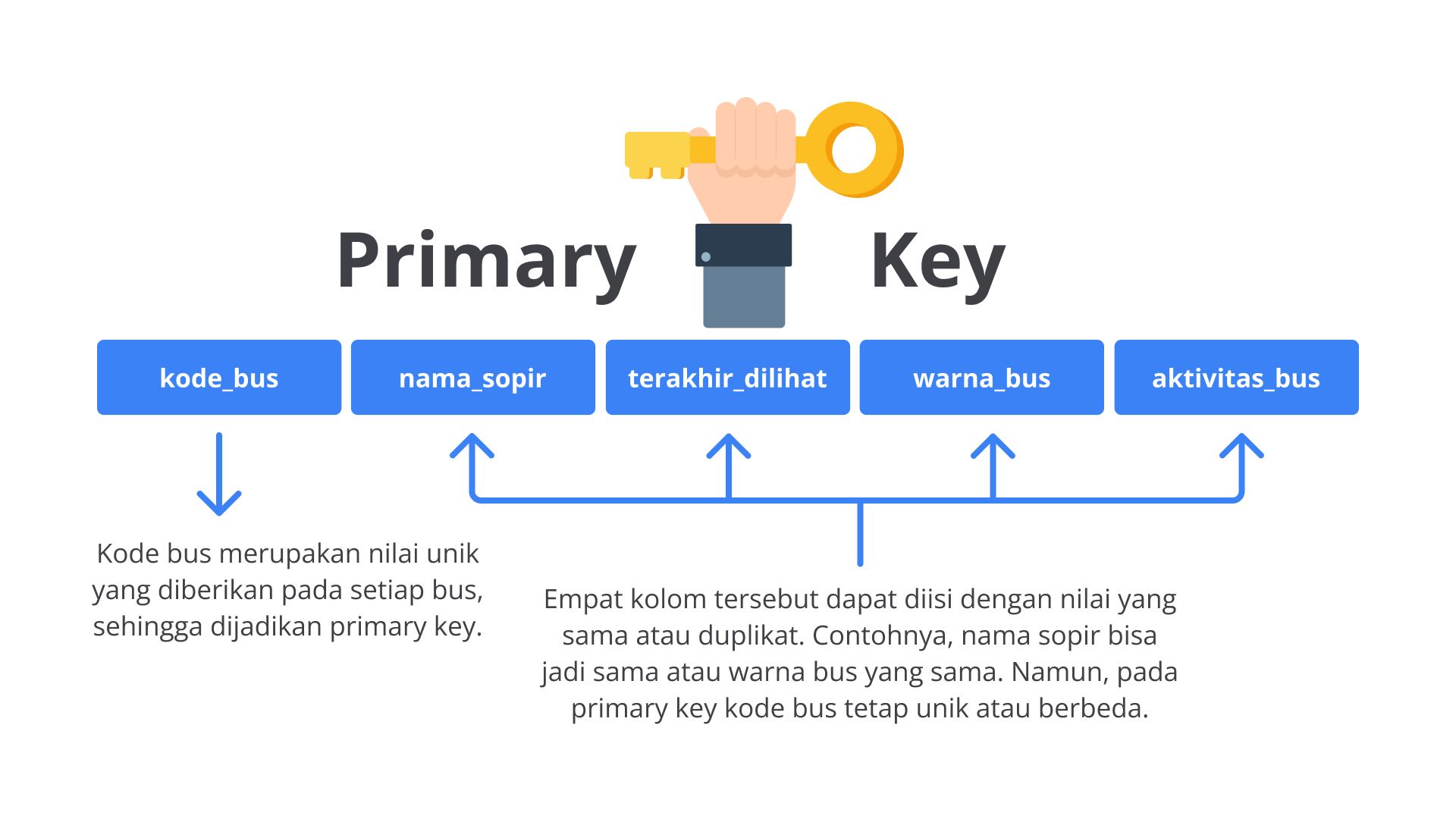


**Unique**

Ini memastikan setiap *record*pada tabel memiliki nilai yang unik atau berbeda.



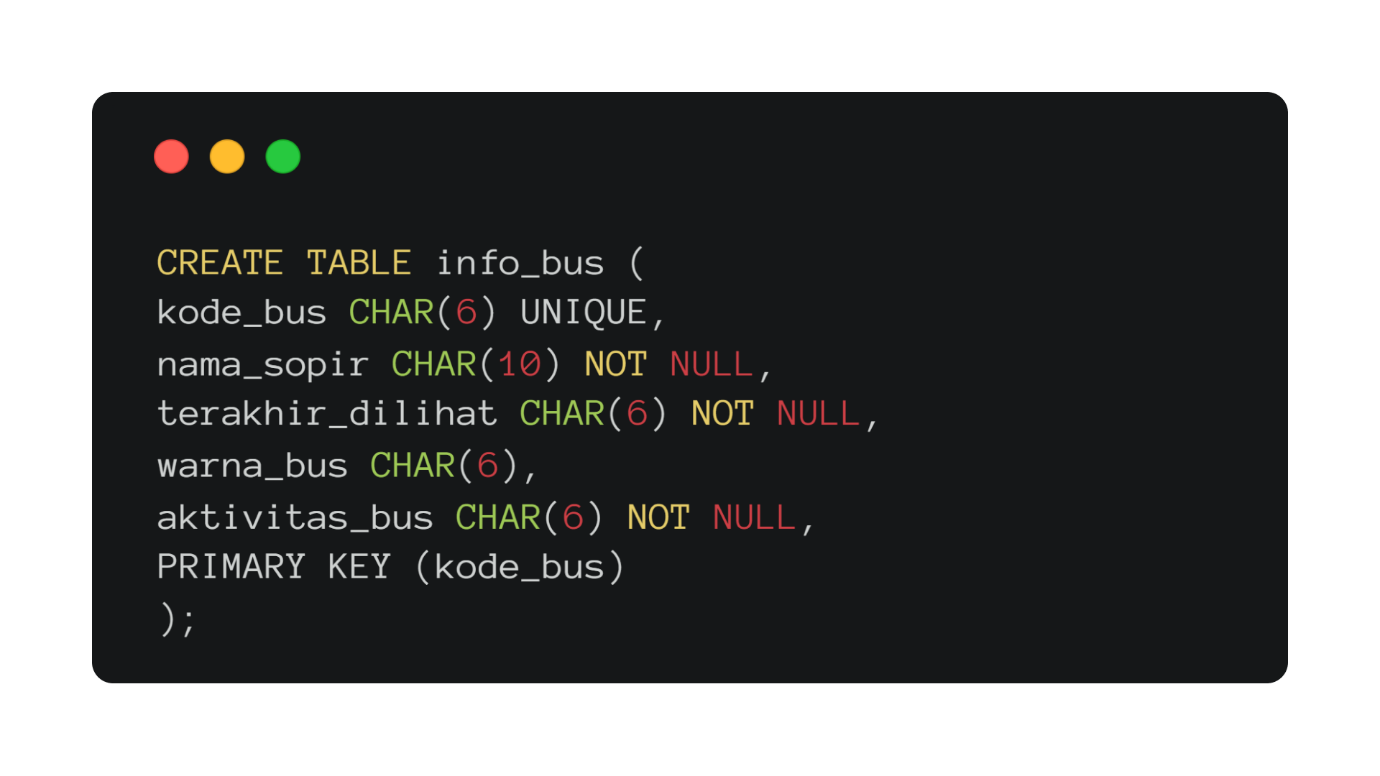
**Primary Key**



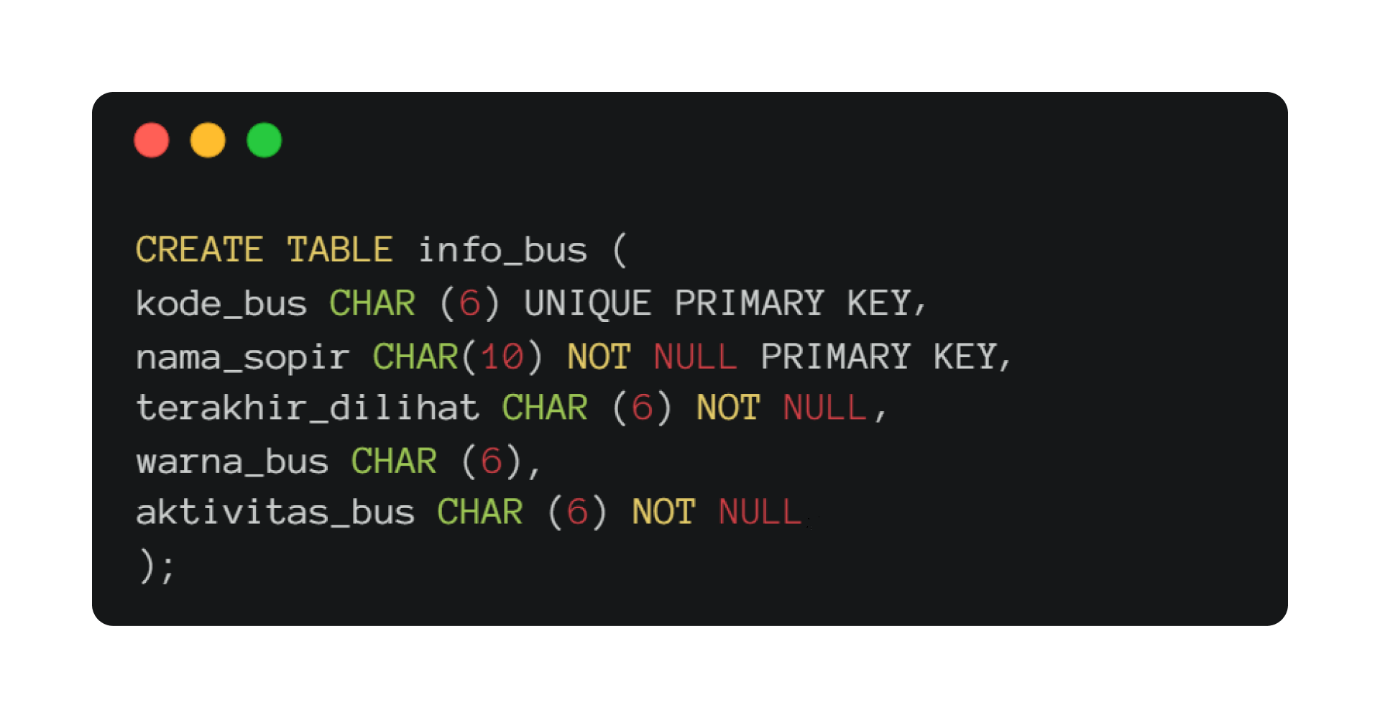
Setelah Anda mempelajari NOT NULL dan *unique*, *primary key*digunakan untuk mengidentifikasi setiap barispada suatu tabel yang bersifat NOT NULL(memastikan suatu kolom berisi nilai dan tidak kosong)danunique*.* Data pada kolom primary key tidak dapat diulang.

Catatan lainnya adalah primary key dapat dibuat pada kolom tunggal dan kolom jamak (banyak).

Berikut adalah contoh pada kolom tunggal.



Berikut adalah contoh pada kolom jamak (banyak).



**Index**

Pada bagian ini, Anda akan dikenalkan dengan *index.*Menurut KBBI, *index*atau indeks adalah kata atau istilah penting yang terdapat dalam buku cetakan (biasanya pada akhir buku) dan tersusun menurut abjad untuk memberikan informasi mengenai halaman tempat kata atau istilah itu ditemukan.

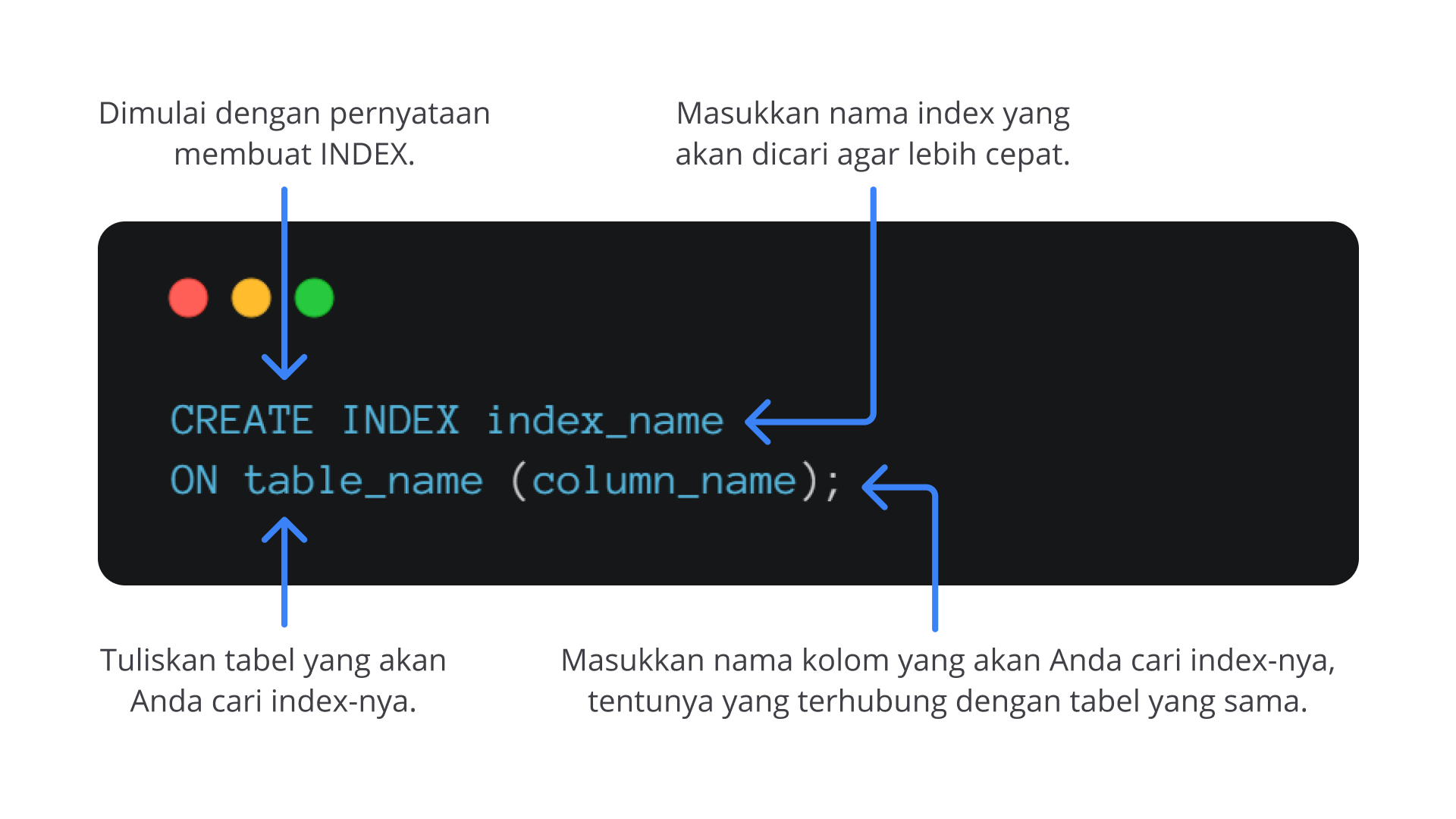
Indeks pada SQL adalah sebuah mekanisme untuk menemukan item tertentu dalam sumber daya. Seumpama indeks dalam buku, indeks pada SQL jugamencantumkan kata dan frasa dalam urutan abjad, memungkinkan pembaca untuk berpindah dengan cepat ke huruf tertentu,menemukan entri yang diinginkan, dan kemudian menemukan halaman tempat kata atau frasa tersebut dapat ditemukan.

Masih sama seperti analogi *constraints*sebelumnya, Anda merupakan seorang kondektur bus Bahagia dengan tujuan Bandung–Surabaya dengan titik pemberhentian Bandung, Jogja, Malang, dan Surabaya. Anda memiliki sejumlah data penumpang yang memiliki spesifikasi kode bis, nama supir, terakhir dilihat, warna bis, dan aktivitas bis.

Anda ingin mencari warna bus hitam dari banyaknya data, bisa saja Anda mencari satu per satu baris dan menyeleksi warna bus apabila hanya terdapat sedikit baris, tetapi bagaimana apabila terdapat jutaan baris? Apakah Anda akan menyeleksinya satu per satu?

Jawabannya adalah tentu tidak, bukan? Maka dari itu, Anda memerlukan indeksuntuk dapat menyeleksi atau mencari hal secara cepat pada sebuah tabel yang terdapat baris dan kolom yang sangat banyak. Lalu, bagaimana cara kerja dari indeks? Simak penjelasannya di bawah ini.

Ketika Anda membuat indeks, basis data akan menghasilkan metode untuk secara cepat mencari data berdasarkan satu atau banyak kolom. Di bawah ini merupakan pernyataan yang digunakan untuk membuat indeks pada DB Browser SQLite.



Pada dasarnya, Anda bisa mencari sebuah data dalam tabel tanpa menggunakan indeks, tetapi tentu memerlukan waktu dalam mencarinya. Dengan bantuan CREATE INDEX terlebih dahulu, Anda dibantu dalam mencari sebuah data yang banyak dengan waktu lebih cepat dari biasanya.

Berikut adalah contoh penggunaan indeks yang dapat Anda coba pada DB Browser SQLite.

Anda akan menggunakan nama tabel, diikuti dengan nilai yang akan dicari, kemudian tambahkan “idx” untuk mengindikasikan bahwa querytersebut merupakan indeks. Sebagaimana yang Anda lihat pada gambar di atas, ketika Anda *run*,basis data akan cukup membaca sebatas tabel info\_bus dan kolom terakhir\_dilihat.

Di bawah ini adalah langkah kedua yang dapat Anda lakukan setelah CREATE INDEX. Ketika Anda mencari sebuah nilai pada kolom terakhir\_dilihat pada tabel info\_bus, kecepatannya akan sangat berbeda dengan tidak membuat indeks terlebih dahulu.

**Pengenalan Diagram**

Selain mempelajari query, Anda juga akan diperkenalkan pada jenis-jenis diagram yang biasa digunakan dalam SQL seperti tabel tunggal (*entity*) dan tabel jamak (*Entity Relational Diagram*). Penasaran? Yuk, simak penjelasannya.

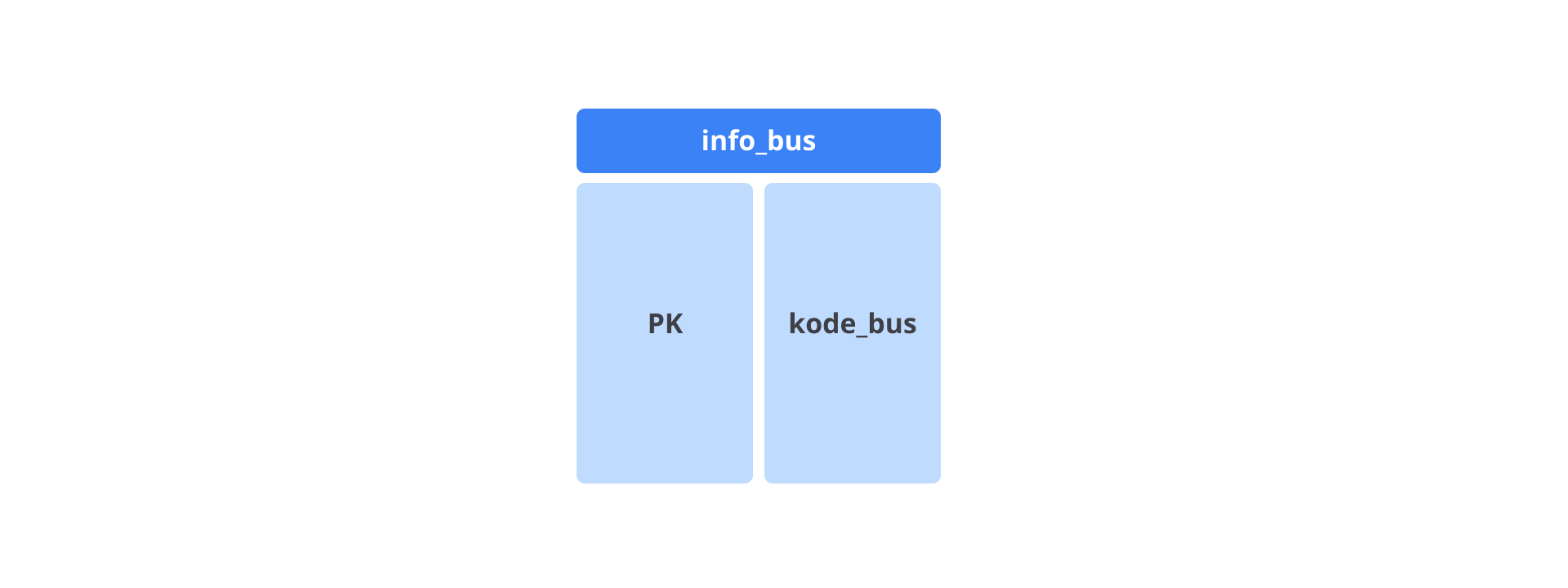
**Mengenal Diagram**

Satu hal terakhir yang perlu Anda ketahui dalam kelas Belajar Dasar Structured Query Language (SQL) ini adalah terkait diagram. Sebelum ke bahasan inti, apakah Anda telah mengetahui definisi diagram? Menurut KBBI, diagram adalah gambaran (sketsa) untuk memperlihatkan atau menerangkan sesuatu.

Nah,pada submodul ini, Anda akan dikenalkan kepada diagram hubungan entitas atau *Entity Relationship Diagram*(ERD). ERD adalah cara umum untuk melihat data dalam basis data. Setelah sebelumnya mempelajari mengenai primary key*,*di sini Anda akan melihat tata cara implementasinya. Maka dari itu, simak penjelasannya dengan saksama, ya!

**Jenis-Jenis Diagram**

Pada SQL, umumnya diagram terbagi menjadi dua, yaitu **tabel tunggal (*entity*)**dan **tabel jamak (Entity Relational Diagram)**. Pada dasarnya, ERD merupakan gabungan dari beberapa entity yang saling memiliki keterkaitan dan hubungan satu sama lain. Pada kelas ini, Anda hanya akan mengenal dan mempelajari tabel tunggal (*entity*) karena hanya akan fokus pada satu tabel saja, sedangkan materi ERD akan Anda dapatkan di kelas tingkat selanjutnya.



Sebagaimana yang Anda lihat pada gambar tersebut terdapat primary key pada entityyang ada dalam tabel info\_bus dan kolom kode\_bus*.* Perlu dicatat bahwa primary key bisa terdapat pada banyak kolom yang memang nilainya unik serta terdapat pada banyak tabel dan saling terhubung, yaitu Entity Relational Database*.*

Pada materi ini, ketika Anda memasukkan nilai unik yang tidak terdapat pada kolom dan tidak terhubung dengan tabel lain, secara otomatis nilai tersebut akan menjadi sebuah primary key.