Tabellum for mobile

User Guide

Version 1.0 · 08.11.2026

Contents

1 Getting started ... 4
1.1 The screen ... 4
1.2 The demo database ... 5
1.3 Your workspace ... 5
2 Connections ... 6
2.1 Adding a connection ... 6
2.2 Cloud databases ... 7
2.3 SQLite files ... 8
2.4 Editing a connection ... 8
2.5 SSH and WireGuard tunnels ... 9
2.6 Sharing connections ... 10
3 Browsing and editing data ... 12
3.1 The schema tree ... 12
3.2 A table’s rows ... 12
3.3 Editing cells ... 14
3.4 The inspector ... 14
4 Writing SQL ... 15
4.1 The query tab ... 15
4.2 Completion ... 15
4.3 Parameters ... 15
4.4 The query plan ... 15
4.5 The command palette ... 16
4.6 The approval dialog ... 17
5 Charts, dashboards and export ... 19
5.1 Charts ... 19
5.2 Dashboards ... 19
5.3 Exporting results ... 20
6 Diagrams, snapshots and comparing schemas ... 21
6.1 Schema diagrams ... 21
6.2 Snapshots ... 21
6.3 Comparing two schemas ... 21
7 The schema designer ... 24
7.1 Starting a design ... 24
7.2 The canvas ... 25
7.3 Reviewing and applying ... 26
7.4 When the database has changed ... 27
7.5 Full-screen editors ... 27
7.6 Creating a database ... 29
8 Views, users and roles ... 30
8.1 Making a view without writing SQL ... 30
8.2 Users and roles ... 30
9 The AI agent ... 31
9.1 Setting it up ... 31
9.2 Asking ... 32
9.3 What the agent can and cannot do ... 32
10 Backup, restore and sync ... 34
10.1 Backing up ... 34
10.2 Restoring ... 34

10.3 Syncing the workspace with git 35
11 Reference 37
11.1 Databases 37
11.2 This guide in Tabellum 37
11.3 On a phone 37
11.4 If something goes wrong 37

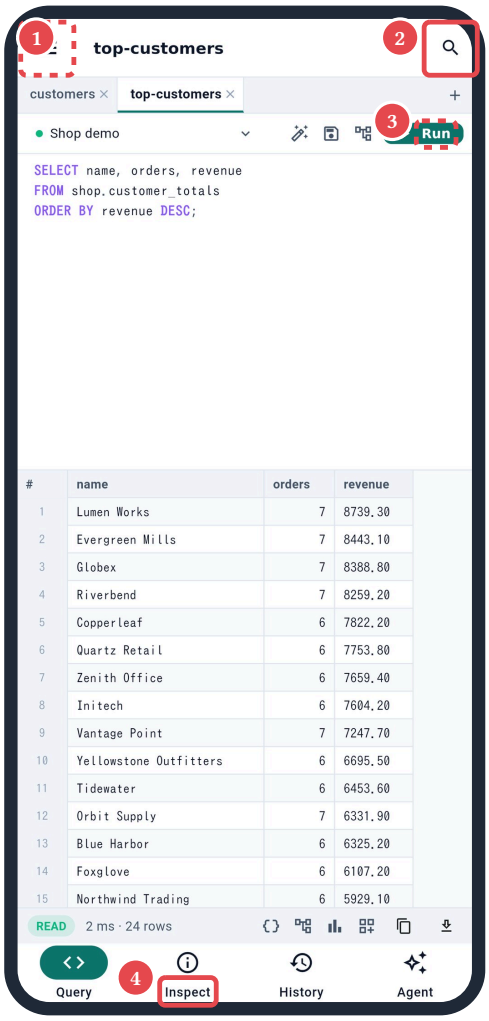
1 Getting started

Tabellum is a database client for PostgreSQL, MySQL and MariaDB, SQL Server, BigQuery, SQLite and MongoDB. On a phone it browses and edits data, runs SQL, draws charts and dashboards, compares and designs schemas, and has an AI assistant that never writes to a database without your approval.

Everything you create — connections, saved queries, dashboards, designs, snapshots — is a plain file in the app's workspace. Passwords and API keys are not: they stay in the phone's secure storage (the Keychain on iPhone, the Keystore on Android).

1.1 The screen

The screen has three parts: the title bar at the top, the page in the middle, and four buttons at the bottom.



1. The menu: connections, queries, dashboards, designs
2. Search everything
3. Run the query
4. Inspector, history and agent pages

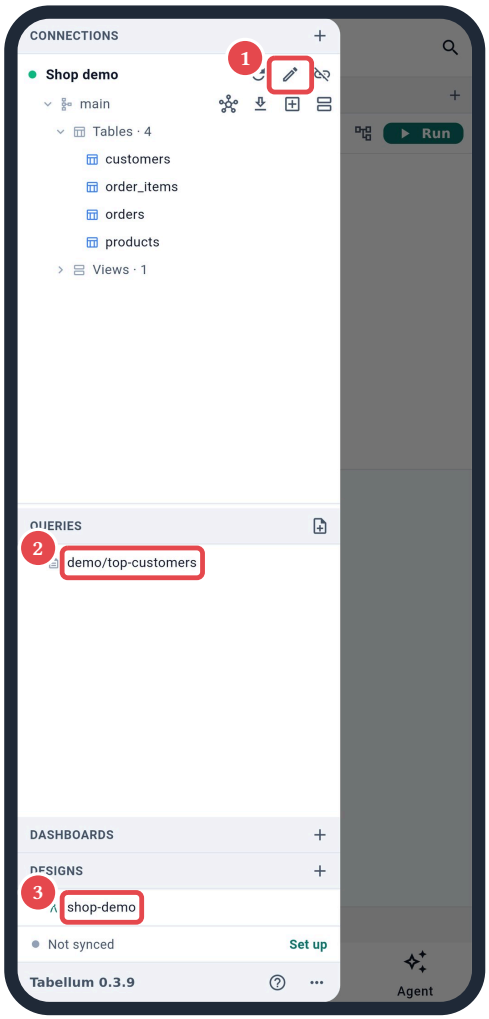
Figure 1: A saved query and its rows

- The menu button 1 opens the side menu with your connections, saved queries, dashboards and designs. Whatever you open there opens as a tab.
- The search button 2 opens the command palette: connections, tables, saved queries and commands, found by a few letters.
- **Run** 3 runs the query in the open tab.
- The buttons at the bottom switch the page: **Query** shows the open tabs, **Inspect** 4 the selected object's columns, indexes and constraints, **History** what ran, and **Agent** the AI agent.

Open tabs are listed in a strip above the page; tap one to switch, tap its `×` to close it, and tap `+` for a new query tab.

1.2 The demo database

The first time Tabellum starts, its workspace already holds a small demo: an invented shop in an SQLite file inside the app, a saved query, and a design of its schema. Nothing needs a network or a server, so it is the quickest way to try every feature. Deleted demo files stay deleted; to get them back, choose **Open the demo database** in the command palette.



1. Edit the connection
2. An example saved query
3. A design of the demo schema

Figure 2: The side menu with the demo connection, its example query and its design

1. Open the menu and tap **Shop demo** to connect. The pencil (1) edits the connection.
2. Tap the example query (2), then **Run**.
3. Tap the design (3) to see the schema designer (see Section 7).

1.3 Your workspace

The workspace is a folder inside the app. In it, `connections/` holds the connections, `queries/` the saved queries, `dashboards/`, `designs/` and `snapshots/` the rest, and `sqlite/` the SQLite databases that belong to the workspace. The files are readable YAML, SQL and JSON, so the workspace can be kept in a git repository and shared with your other devices (see Section 10.3).

> **Note**
> Removing the app removes its workspace. Sync it to a repository, or export what you want to keep, first.

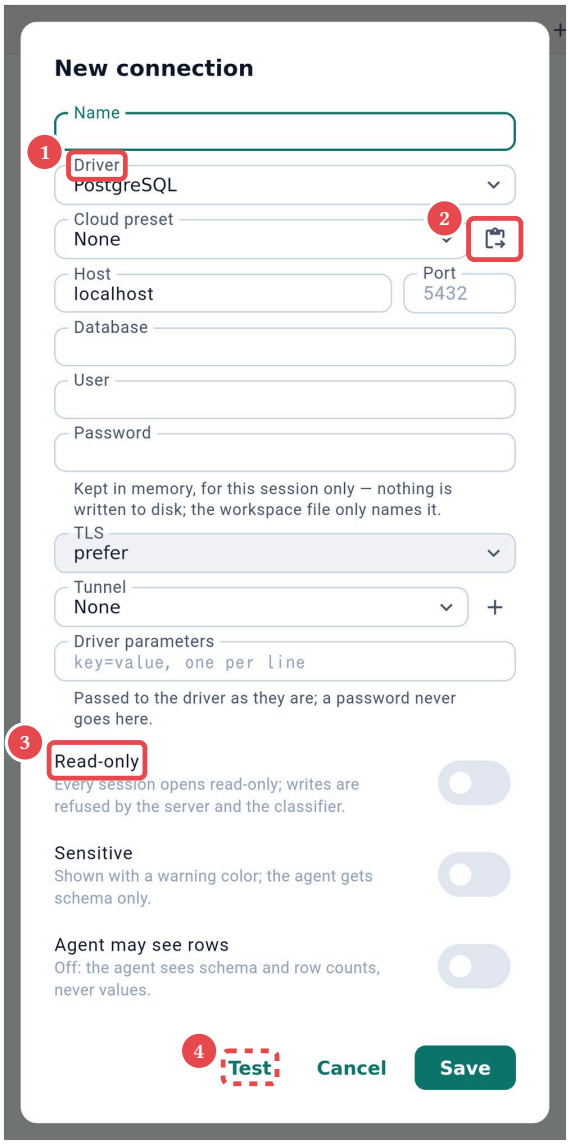
A connection file never contains a password. The password is in the secure storage of the device it was typed on; on another device you type it once.

2 Connections

A connection is the address of one database and how to log in to it. Tabellum keeps as many as you like and can have several open at once.

2.1 Adding a connection

1. Open the menu and tap `+` next to `CONNECTIONS`.
2. Pick the **Driver** (1): PostgreSQL, MySQL / MariaDB, SQL Server, BigQuery, SQLite or MongoDB.
3. Fill in the host, port, database, user and password.
4. Tap **Test** (4). When the test passes, tap **Save**.



1. The database type
2. Fill the form from a connection string
3. Refuse every write on this connection
4. Try the connection before saving

Figure 3: A new connection

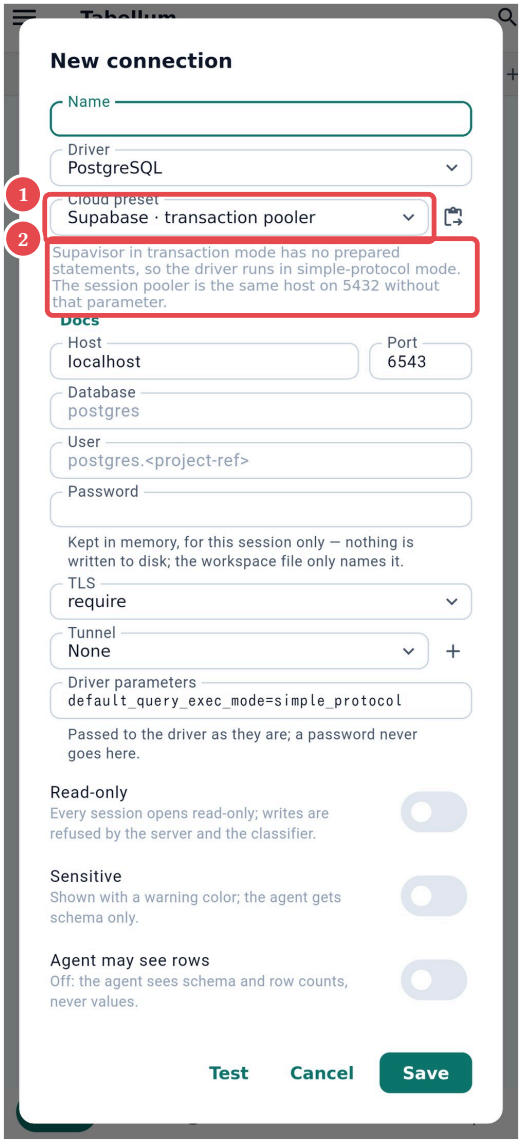
Three switches decide what may happen on the connection:

- **Read-only** 3: every session opens read-only. The server refuses writes, and Tabellum refuses them before they reach it.
- **Sensitive**: the connection is drawn in a warning color, and the AI agent gets only its schema.
- **Agent may see rows**: off by default. While it is off, the agent sees tables, columns and row counts, never values.

If you have a connection string (`postgres://…`, `mysql://…`, `mongodb+srv://…`, a JDBC URL), copy it, tap the clipboard button 2 and paste it. Tabellum fills in the form, puts the password into secure storage and lists what it set and what it could not.

2.2 Cloud databases

For a hosted database, pick its provider under **Cloud preset** 1. The preset fills in the port, TLS mode and driver settings the provider needs and explains what is special about it 2 — for example, that a transaction pooler has no prepared statements.



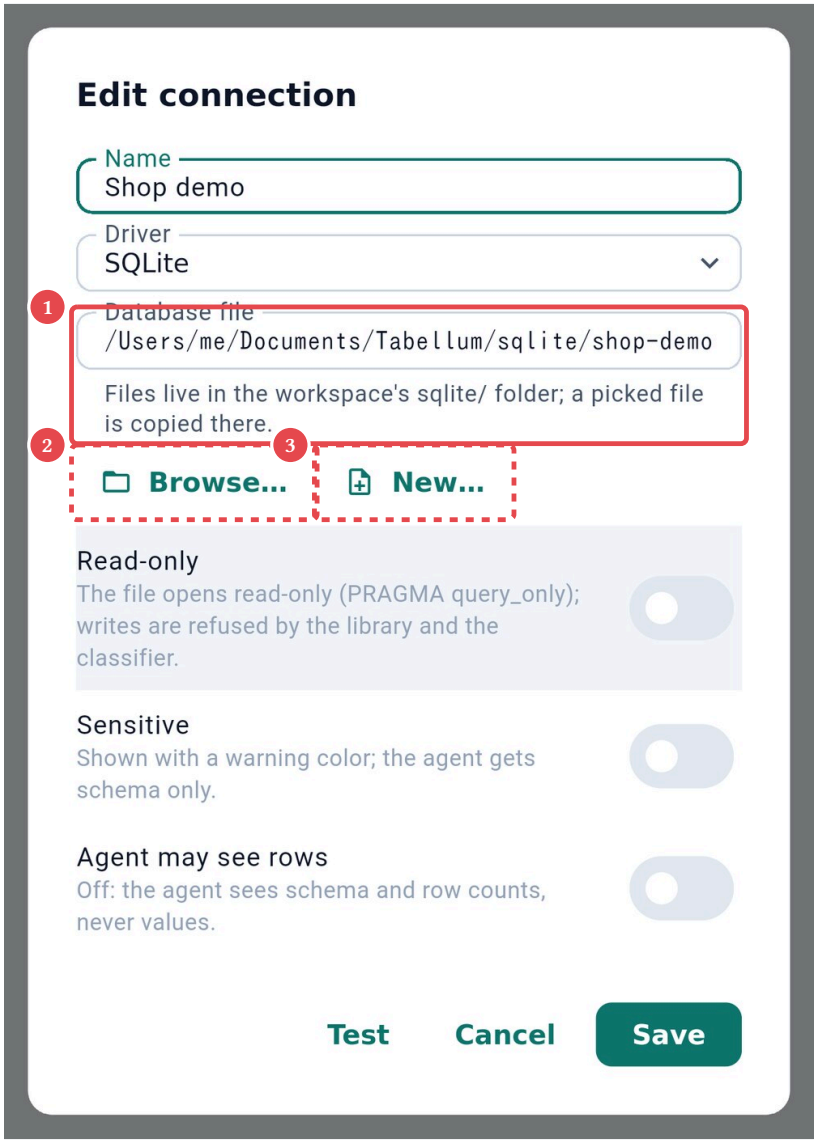
1 A cloud provider's settings

2 What the provider needs

Figure 4: A cloud preset

2.3 SQLite files

An SQLite database is a file, so its connection has no host or login. **Browse…** (2) picks a file and **New…** (3) creates an empty one. A picked file is copied into the workspace's `sqlite/` folder, because the app may only keep access to its own folder.



1. The database file
2. Pick a file
3. Create a new database file

Figure 5: An SQLite connection

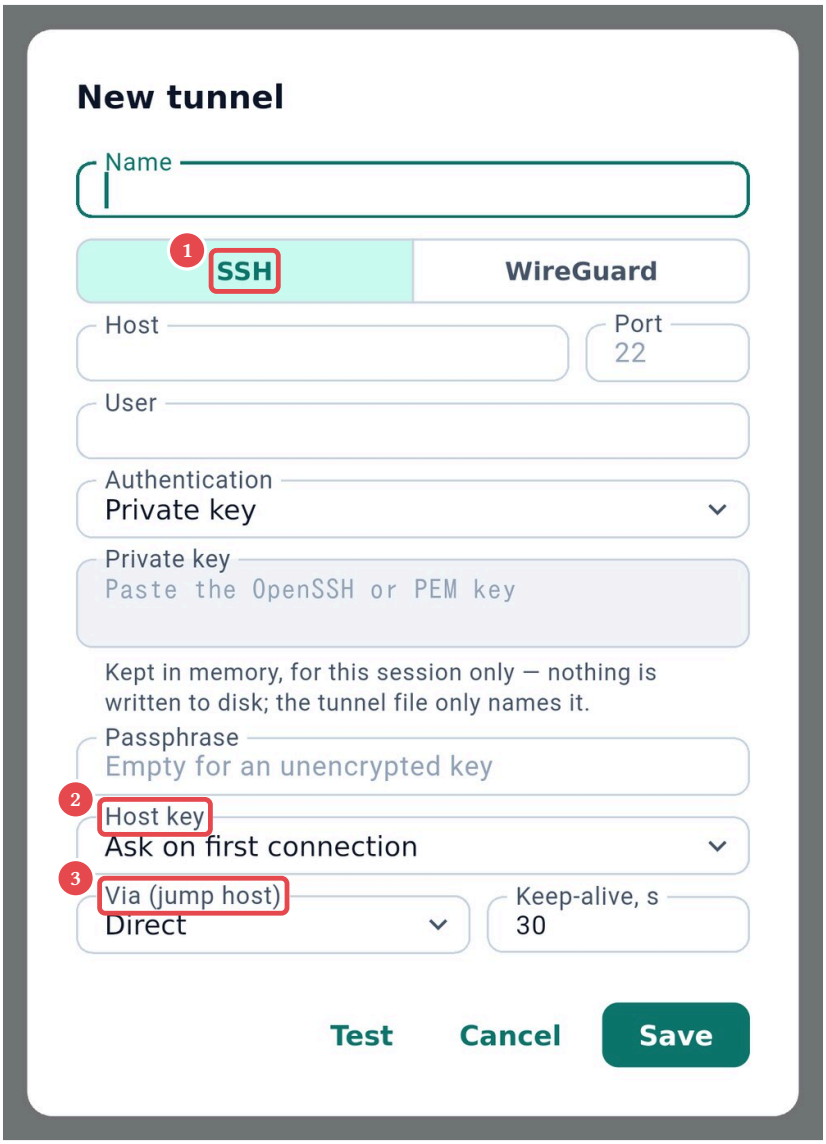
2.4 Editing a connection

Each connection in the menu has a pencil button, **Edit**, that opens the same form; a long press on the connection opens its menu with everything else. A connection that cannot connect shows one line saying what failed and what to check — a refused port, a timeout, a wrong password or a TLS problem.

When you come back to the app after a while, Tabellum reconnects the open connections by itself. A read that failed because the connection dropped is retried once; anything else offers **Run again**.

2.5 SSH and WireGuard tunnels

When a database is only reachable through a server or a VPN, add a tunnel and choose it in the connection's **Tunnel** field (the **+** next to it creates one). Tunnels run inside the app: they need no VPN profile and change nothing else on the phone.



1. SSH, or a WireGuard peer
2. How the server's key is checked
3. Reach it through another tunnel

Figure 6: A new tunnel

- **SSH** (1) logs in with a private key (and its passphrase) or a password. **WireGuard** connects as a peer; paste a `wg-quick` configuration and the fields are filled in.
- **Host key** (2) decides how the server's key is checked. With **Ask on first connection**, Tabellum shows the key's fingerprint the first time and remembers it.
- **Via (jump host)** (3) reaches the server through another tunnel.

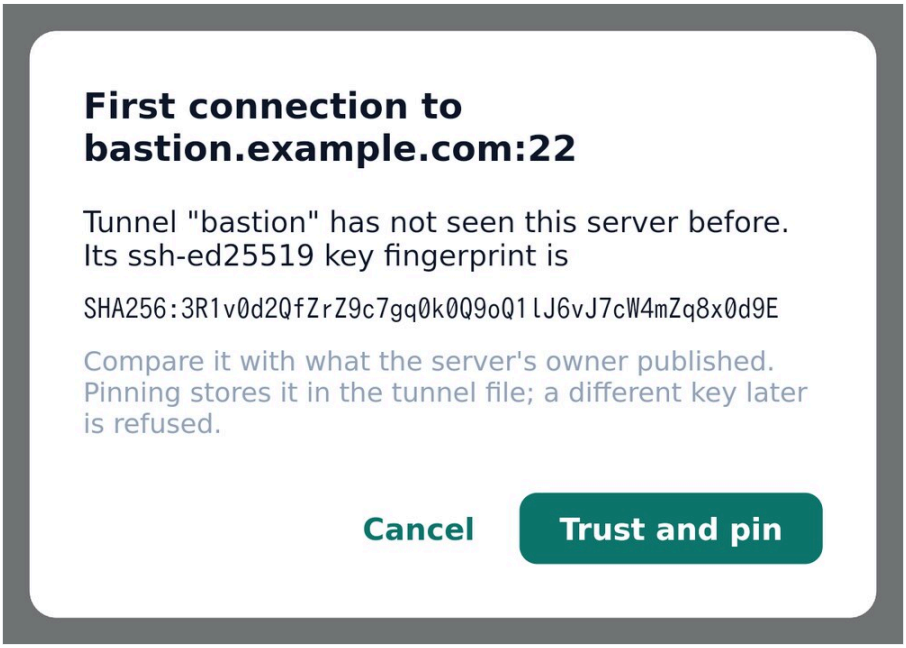
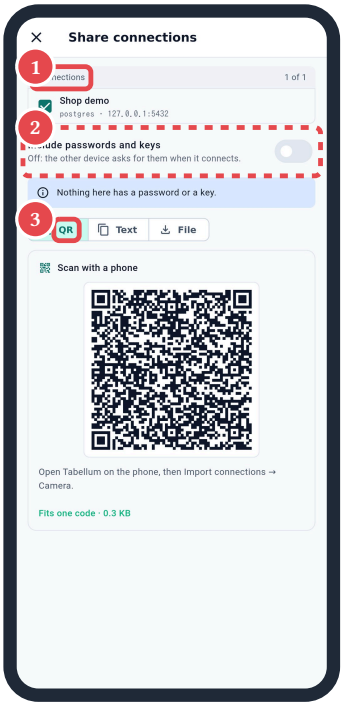


Figure 7: The first connection to an SSH server: compare the fingerprint, then trust it

> **Note**
> A phone has no SSH agent: paste the private key into the tunnel. It is kept in secure storage.

2.6 Sharing connections

Connections and tunnels move between devices with a code: set them up on the computer, then scan them into the phone. Choose **Share connections…** in the Tabellum menu (⋯), or **Share…** in a connection's menu.



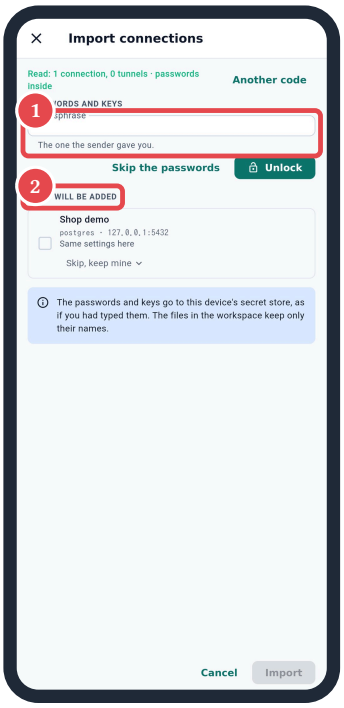
1. Pick the connections; their tunnels come along
2. Passwords and keys, encrypted, only if you ask
3. Show it as a QR code, a text, or a file

Figure 8: A connection shared as a QR code

Tick what to send (1) a connection's tunnel comes along by itself. Passwords and keys stay on this device unless you turn on **Include passwords and keys** (2) and type a passphrase twice, at least 6 characters. Then show it as a QR code, a text to copy, or a file (3).

> **Warning**
> Anyone who has the code and the passphrase can reach those databases. Send the passphrase another way than the code.

To receive, choose **Import connections…**. On **QR code**, point the camera at the code, or choose a photo or screenshot of it with **Choose a photo instead**. **Paste** and **File** take the code as text. Tabellum asks for the camera and the photos only here.



1. The passphrase you were given
2. What is added, and what to do with a name taken here

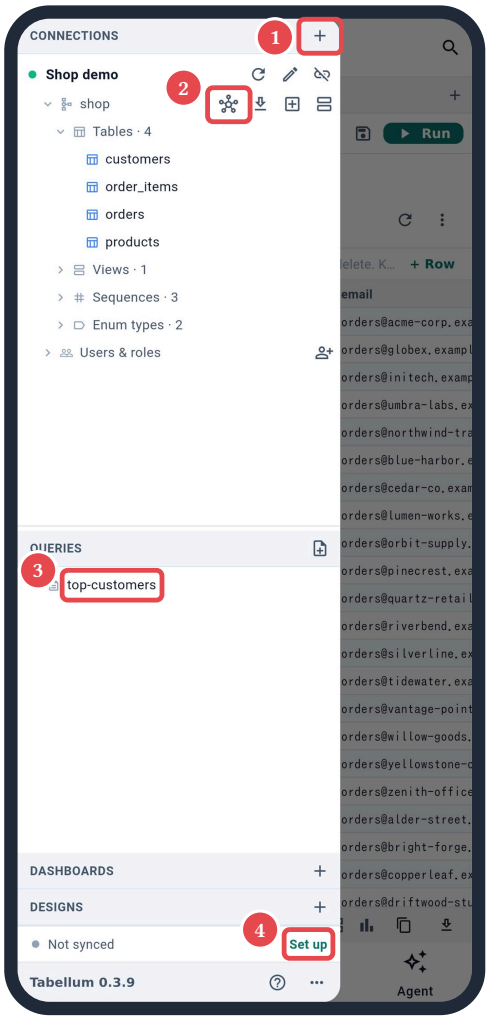
Figure 9: A code that carries passwords

If the code carries passwords, type the passphrase (1) and choose **Unlock**. **Skip the passwords** leaves them to type later. The list (2) shows what will be added, and asks what to do when a connection of that name is already on the phone. Nothing is written until **Import**.

3 Browsing and editing data

3.1 The schema tree

Open the menu and tap a connection to see its schemas. Each schema groups its tables, views, functions, sequences and enum types; **Users & roles** lists the logins.



1. Add a connection
2. The schema as a diagram
3. Saved queries
4. Sync with a git repository

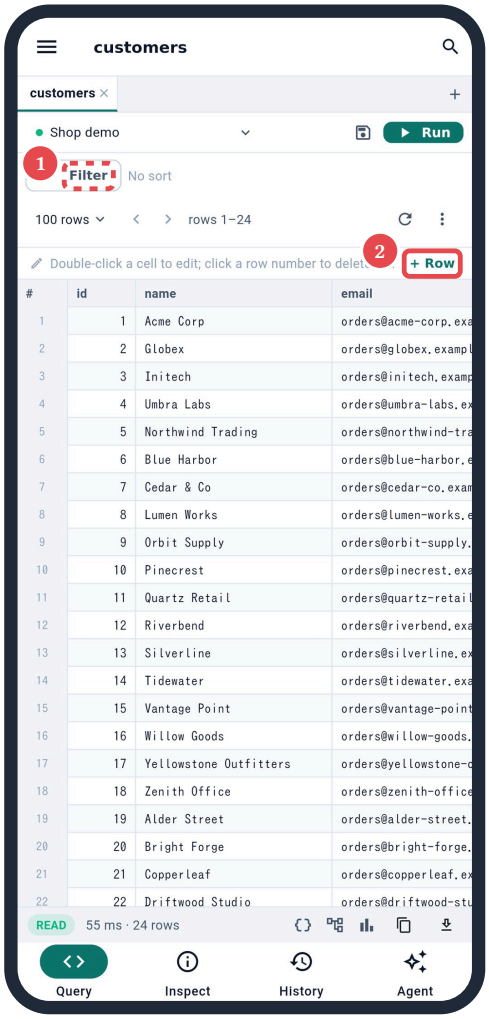
Figure 10: The side menu with the demo shop open

- 1 adds a connection.
- 2 draws the schema as a diagram (see Section 6.1).
- Saved queries 3 are files in `queries/`; a tap opens one in a tab.
- 4 sets up sync (see Section 10.3).

A long press on an object opens its menu: export, import into a table, design, edit, and more.

3.2 A table's rows

A tap on a table opens its rows in a tab, 100 at a time. Use the arrows to page, and tap a column header to sort by it.



1. Filter the rows
2. Add a row

Figure 11: A table's rows

Filter 1 adds a condition; the value is passed to the database as a parameter, never pasted into the SQL. The ⋮ menu counts every row or opens the same view as a query you can edit.

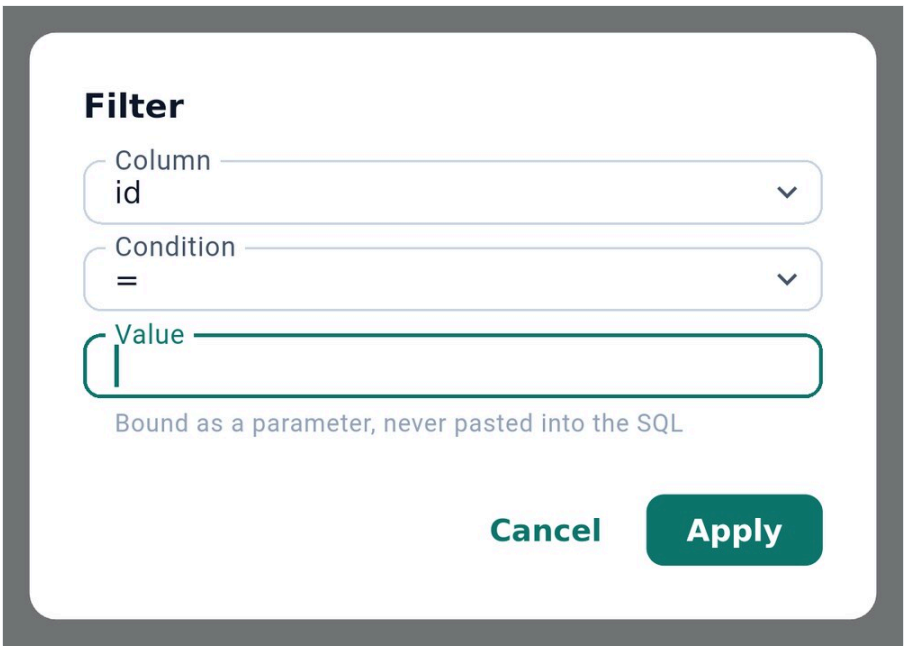
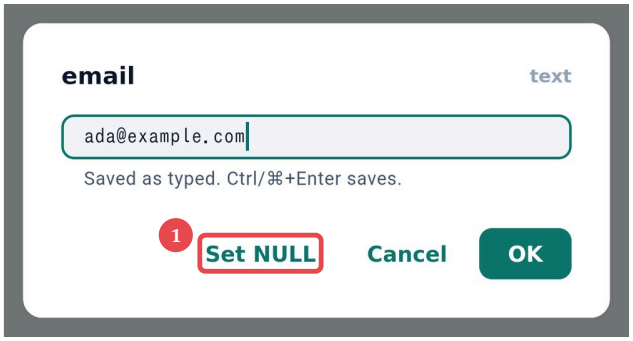


Figure 12: A filter on one column

The bar under the rows shows how long the read took and how many rows came back. Its buttons switch the result between a table, JSON and a chart, copy it, and export it as CSV, JSON, SQL `INSERT` statements or an Excel workbook.

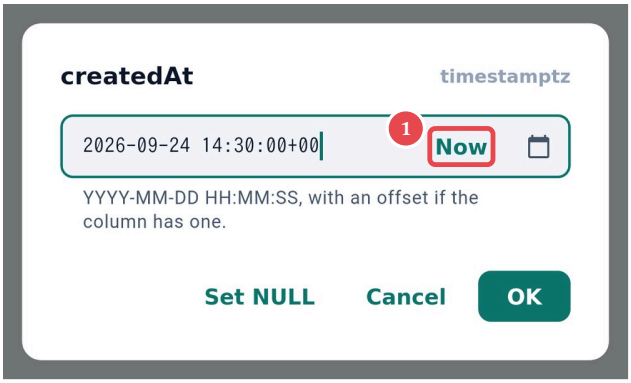
3.3 Editing cells

Double-tap a cell to edit it. The editor fits the column's type: a date and time field with **Now**, a list for an enum, a check box for a boolean, a JSON editor, numbers and UUIDs checked as you type. **Set NULL** (1) stores NULL.



1 Store NULL instead of a value

Figure 13: Text



1 The current time

Figure 14: Date and time

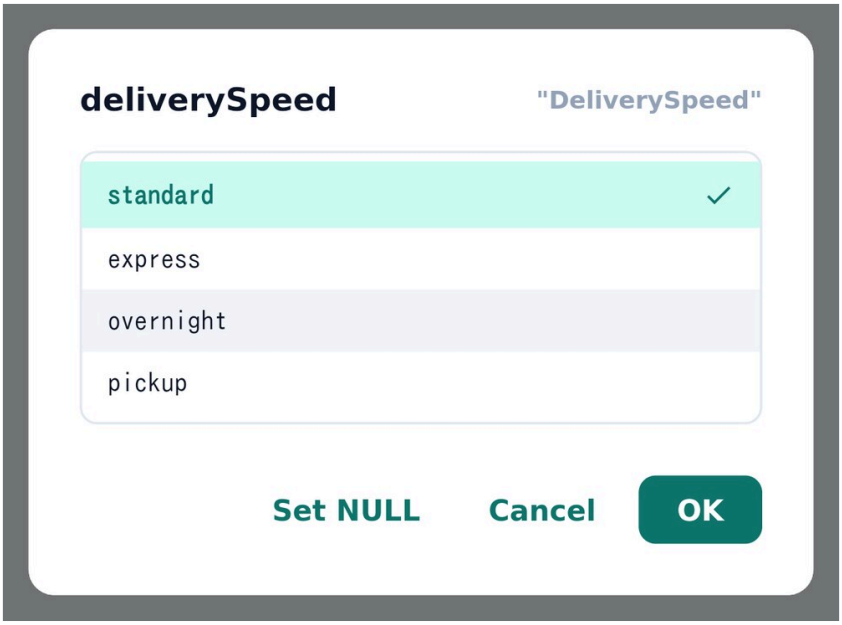


Figure 15: An enum column offers its values

Tap a row number to mark the row for deletion, or **+ Row** (2) in the rows picture to add one. Nothing is written until you save: the edits become SQL statements, and they run through the approval dialog (see Section 4.6) so you see exactly what changes.

3.4 The inspector

With an object selected, the **Inspect** page lists its columns with their types, its indexes and its constraints. **Copy DDL** copies the `CREATE` statement; **Browse rows**, **Design table** and **Edit definition** open the object in a tab.

4 Writing SQL

4.1 The query tab

Tap `+` above the tabs (or **New query tab** in the palette) for an editor on the connection chosen at its top. Type SQL and tap **Run**: Tabellum runs the selection if there is one, otherwise the statement under the cursor. The buttons next to **Run** format the SQL, save the tab as a file in `queries/`, and show the query plan.

While a query runs, **Run** becomes a red **Stop** button that counts the seconds.

4.2 Completion

The editor completes as you type, and it knows both the database's dialect and your schema: tables, columns of the tables the statement names (with their aliases), functions with their arguments, and whole join conditions along foreign keys. After `ON`, the joins the foreign keys allow come first. Tap a suggestion to insert it.

4.3 Parameters

A query can have named parameters — `:since`, `$1` or `@name`, as the database writes them. Before it runs, Tabellum asks for their values and passes them to the database separately, so they are never pasted into the SQL.

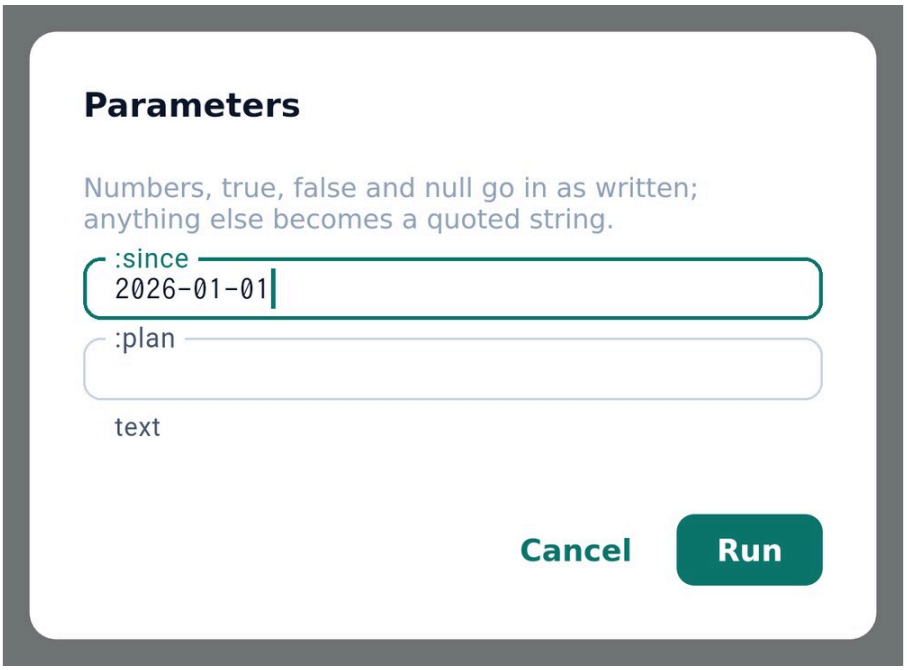


Figure 16: Values for a query's parameters

4.4 The query plan

The plan button asks the database how it would run the statement and shows the answer as a tree. Each step has a bar for its share of the cost, the costliest steps are highlighted, and the database's own output is a tap away. Tap it again to go back to the rows without running anything.

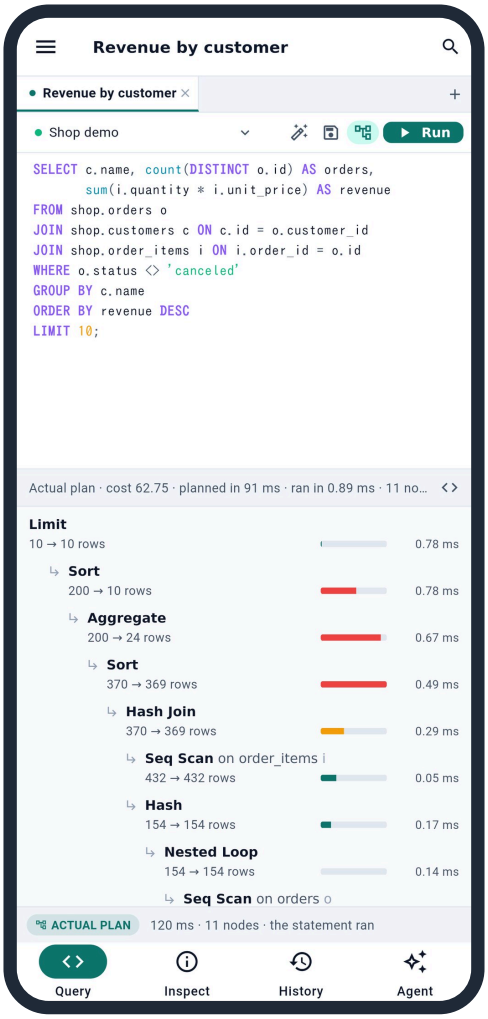


Figure 17: A plan: each step with its rows and share of the time

4.5 The command palette

The search button in the title bar opens the palette: type a few letters of a connection, table, saved query or command and tap the result.

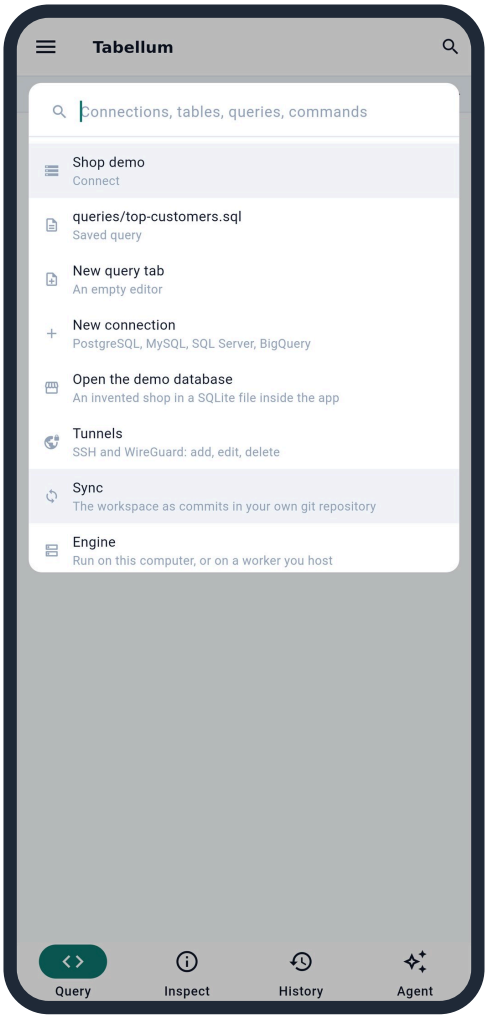
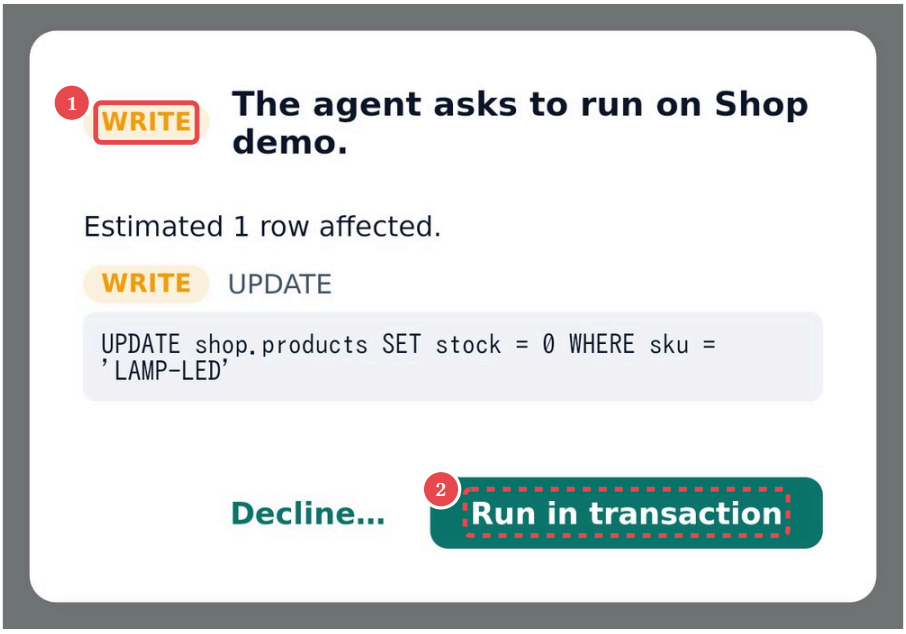


Figure 18: The command palette

4.6 The approval dialog

Tabellum reads every statement before it runs. Anything it cannot prove to be a read — `UPDATE`, `DELETE`, `INSERT`, `CREATE`, `ALTER`, `DROP` — stops at an approval dialog that shows the statements, their kind [1] and, where the database can tell, how many rows they would change.



1. What kind of statement it is
2. Run it, all or nothing

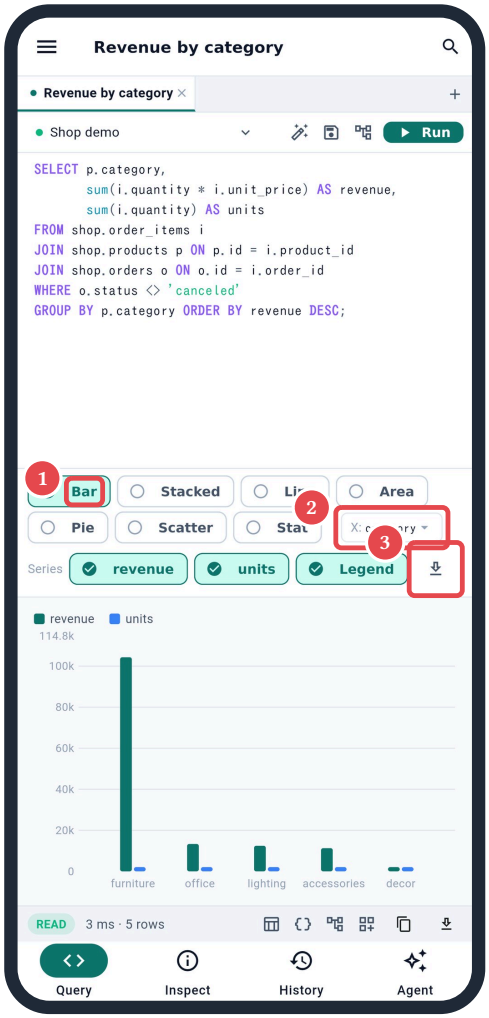
Figure 19: A write waits for your tap

Run in transaction (2) runs them all or none. Before a statement that changes the schema, the dialog offers to back up first. This dialog is the same whether you typed the SQL, edited cells, applied a design or the AI agent proposed it.

5 Charts, dashboards and export

5.1 Charts

Any result can be a chart: tap the chart button in the bar under the result. Tabellum suggests one from the columns; change the kind (1), the column along x (2) and the series to taste. (3) saves the chart as PNG or SVG.



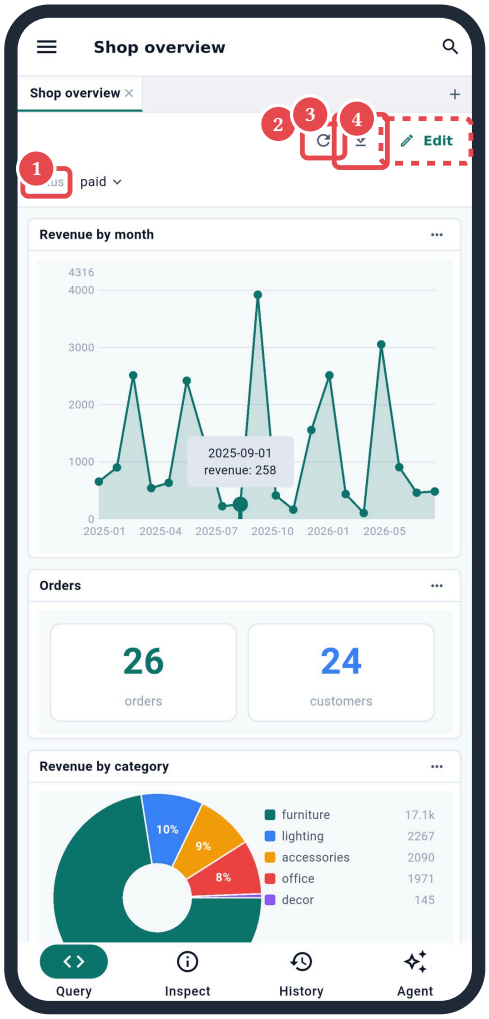
1. The kind of chart
2. The column along x
3. Save as PNG or SVG

Figure 20: Revenue and units by category

The kinds are bar, stacked bar, line, area, pie, scatter and stat (one big number). A line or area over dates or numbers is drawn in their order, not the rows' order.

5.2 Dashboards

A dashboard is a page of panels — charts and tables — each with its own query, on any connection. It lives as a file in `dashboards/`; on a phone its panels stack in one column.



1. A parameter every panel uses
2. Refresh every panel
3. Export as PNG or PDF
4. Edit panels and parameters

Figure 21: A dashboard with a parameter and its panels

- Parameters 1 are shared by every panel: change one and they all run again.
- 2 refreshes every panel; a panel can also rerun on its own timer while the dashboard is open.
- 3 exports the dashboard as a file, a PNG or a PDF.
- 4 edits the panels: their queries, their charts and their place on the grid.

To add a result you are looking at, use **Add to dashboard…** in the bar under it.

> **Note**
> Dashboards only read. A panel's query is checked like any other, and a write is refused.

5.3 Exporting results

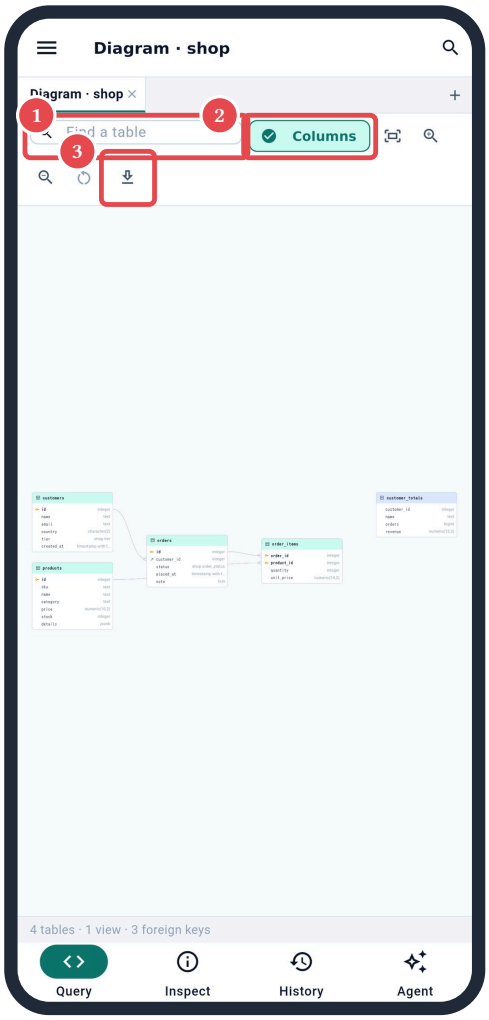
The export button under a result saves it as CSV, JSON, SQL `INSERT` statements or an Excel workbook (a large result is split across sheets). The copy menu copies a cell, a row as JSON, a column, the column names, or the whole result as CSV, JSON or SQL.

To export a whole table, schema or database, long-press it in the menu; **Import into table…** in a table's menu loads a CSV or JSON Lines file into it, in one transaction.

6 Diagrams, snapshots and comparing schemas

6.1 Schema diagrams

The **Diagram** button next to a schema in the menu draws its tables and the foreign keys between them.



1. Find a table
2. Columns or names only
3. Save as SVG or PNG

Figure 22: The demo shop as a diagram

- 1 finds a table by name and centers it.
- 2 switches between full columns and table names only — the better view for a large schema.
- 3 saves the diagram as SVG, or PNG while it is small enough.

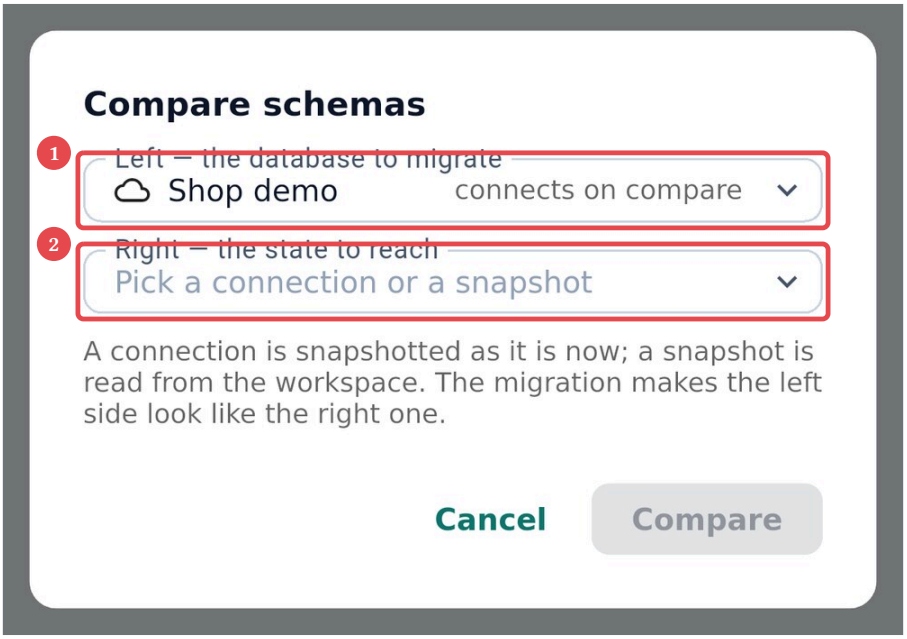
Drag the canvas to pan and pinch to zoom. Drag a table by its card to move it; Tabellum remembers where you put it. A tap on a table selects it and lights up its foreign keys, and the **Inspect** page shows it; a tap on empty canvas clears the selection.

6.2 Snapshots

A snapshot is a schema's structure — tables, columns, keys, indexes, views, functions — saved as a file in `snapshots/`, without any rows. Take one with **Save schema snapshot** in the palette or the connection's menu — before a risky change, for example — then compare against it later, or bring a database back to it.

6.3 Comparing two schemas

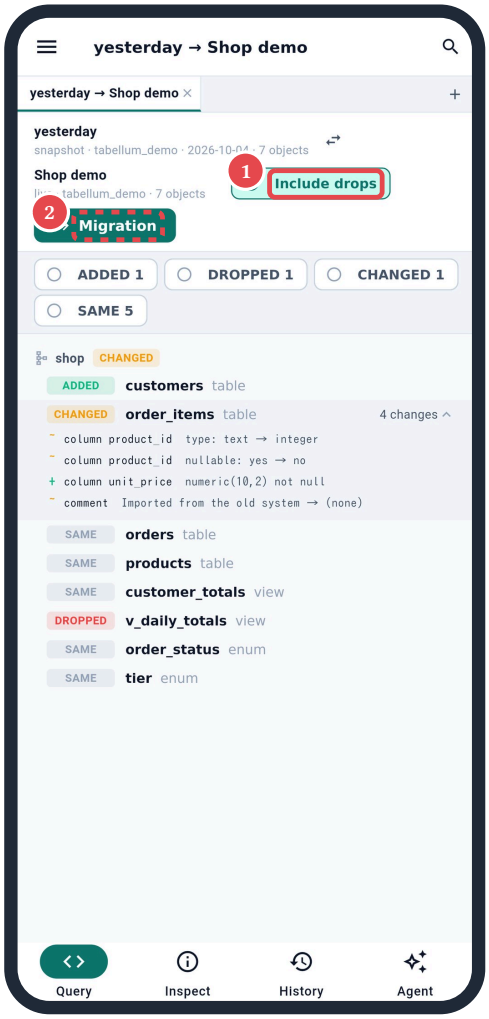
Compare schemas (in the palette or a connection's menu) compares two sides: the one to migrate 1 and the one to reach 2. Each side is a connection as it is now or a saved snapshot.



1. The database to migrate
2. The state to reach

Figure 23: Choosing the two sides

The result lists every object as added, dropped, changed or the same, with what changed in each.



1. Also drop what the target lacks
2. The migration as SQL

Figure 24: What differs between yesterday's snapshot and the database

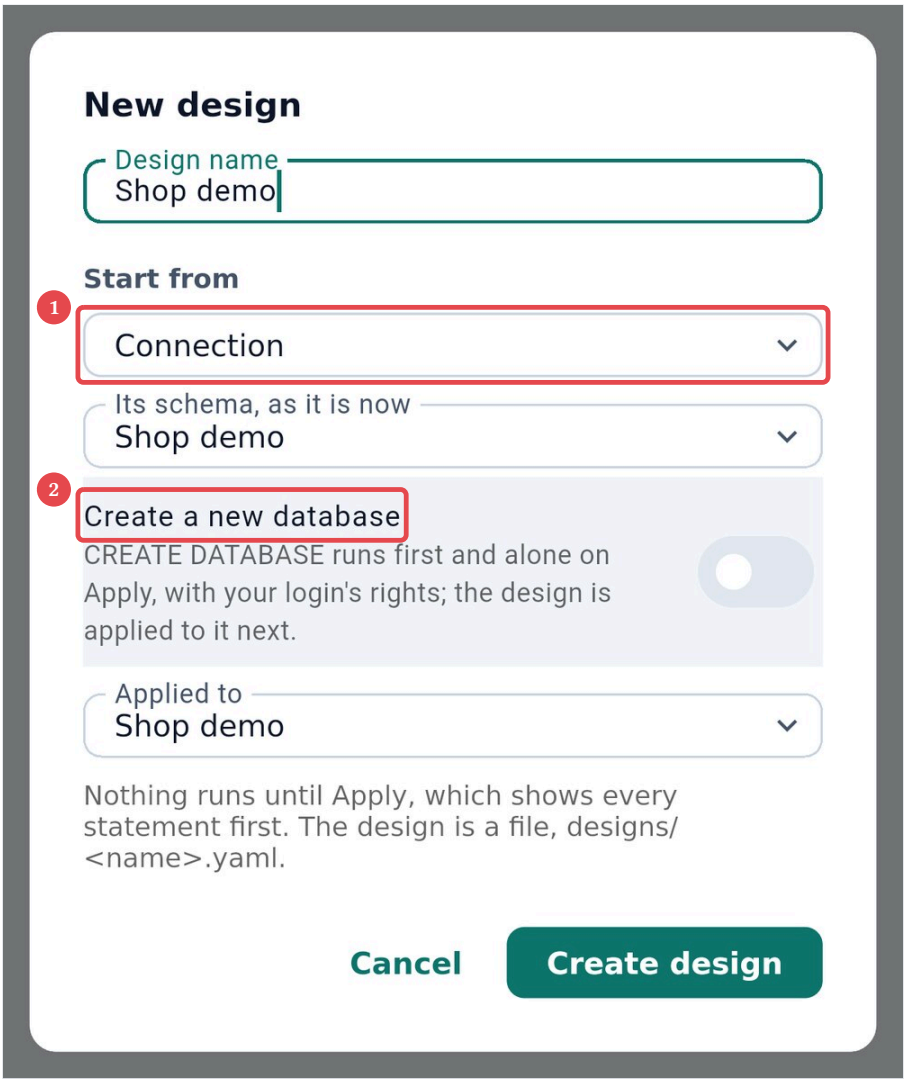
Migration (2) opens the SQL that makes the left side match the right in a new tab. It runs only when you run it, through the approval dialog. Drops are left out unless **Include drops** (1) is on.

7 The schema designer

The designer is a canvas for changing a schema the way you would draw it: add tables, drag foreign keys between columns, edit types and constraints. Nothing touches the database until you tap **Apply**, and then you see every statement first. A design is a file in `designs/`.

7.1 Starting a design

Open the menu and tap **+** next to **DESIGNS**.



1. Where the design starts from
2. Create a new database instead

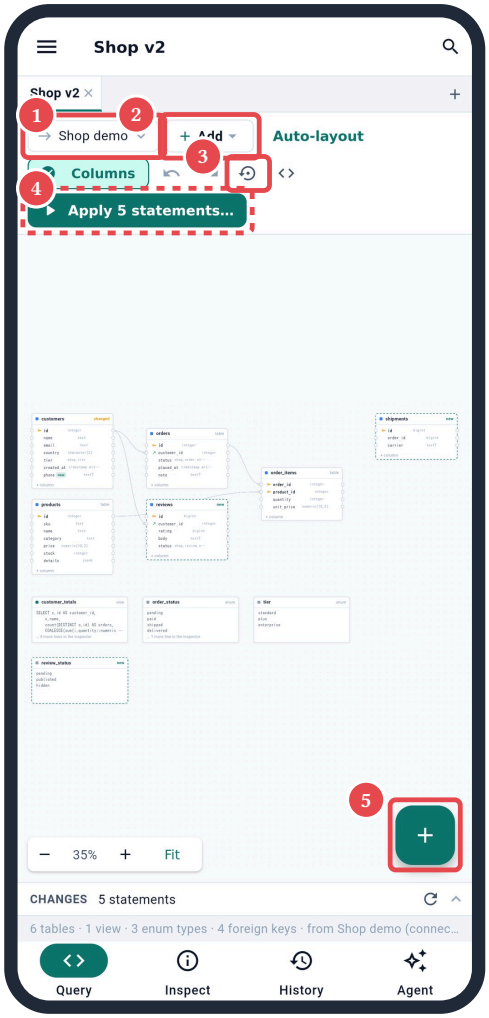
Figure 25: A new design

Under **Start from** (1), choose:

- **Connection** — the database's schemas as they are now. Applied back to the same database, nothing changes until you edit something.
- **Snapshot** — a saved snapshot, to build on it or bring a database back to it.
- **Design** — a copy of another design.
- **Empty** — a blank schema.

Create a new database (2) makes the design create its database first; you choose its name, the server it is created on and its options.

7.2 The canvas



1. The connection Apply runs on
2. Add a table, view, function…
3. Reload the design from the database
4. Apply the changes
5. Add an object

Figure 26: A design on the canvas

- 1 is the connection **Apply** runs on.
- 2 and the round **+** button 5 add a table, view, materialized view, function, procedure, sequence or enum type — what the database supports.
- Drag a table to move it. Drag from a column's port (the ring at its edge) onto another table's column to create a foreign key.
- The arrows just before 3 undo and redo any change, including moves.
- **Auto-layout** arranges the canvas; **Columns** shows or hides the columns.

Tables marked **new** or **changed** differ from the database; a red dashed table is one the database has and the design does not, which **Apply** would drop.

A tap on a table opens its inspector in a sheet from the bottom: name, comment, columns with their types and defaults, keys, foreign keys, checks and indexes, and the SQL for that object. Drag the sheet down to close it; its expand button opens the object full-screen.

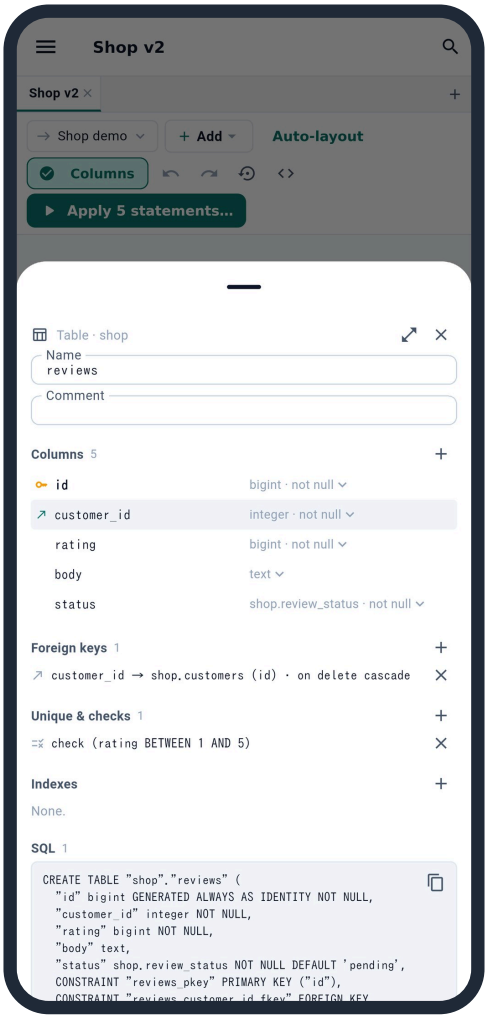


Figure 27: The inspector of the selected table

7.3 Reviewing and applying

The **Changes** bar under the canvas is always up to date: how many statements **Apply** would run, what it would drop, any warnings, and the statements themselves. Tap it to open it.

Tap **Apply** 4 in the canvas picture. The statements open in the approval dialog and run in one transaction where the database allows it.

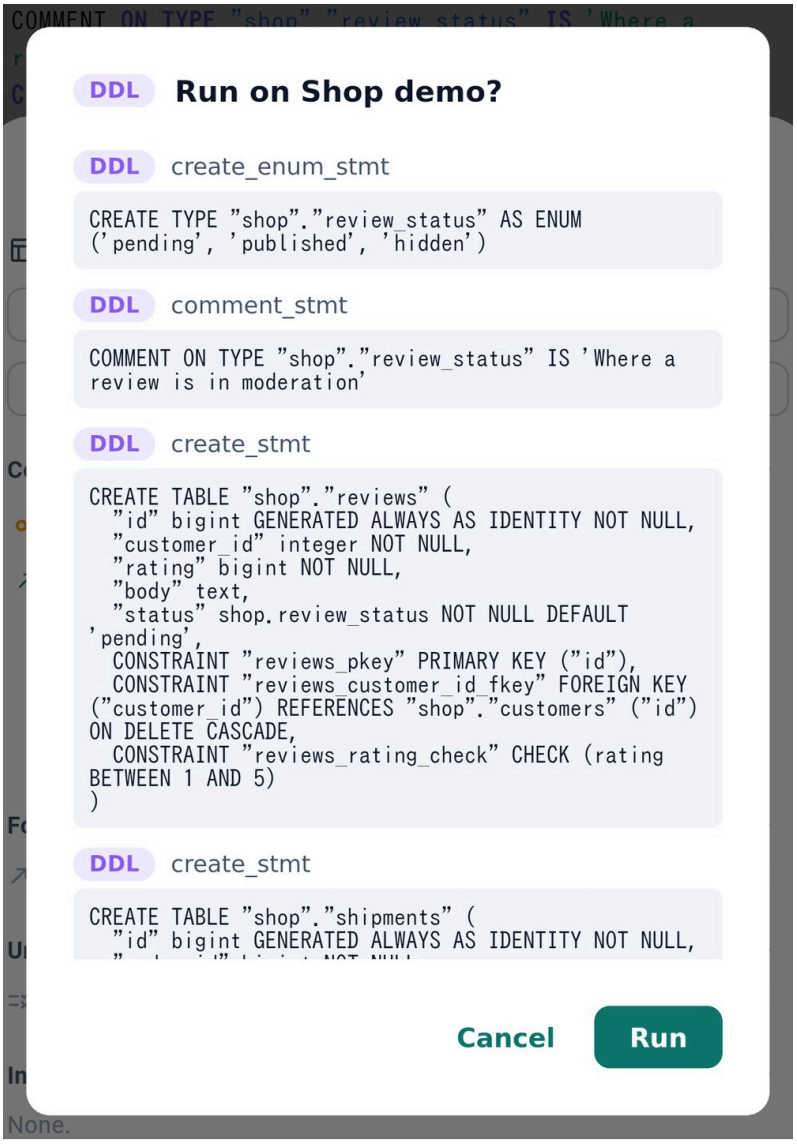


Figure 28: Applying a design

The code button next to **Apply** opens the same script in a query tab instead, to run it yourself or keep it.

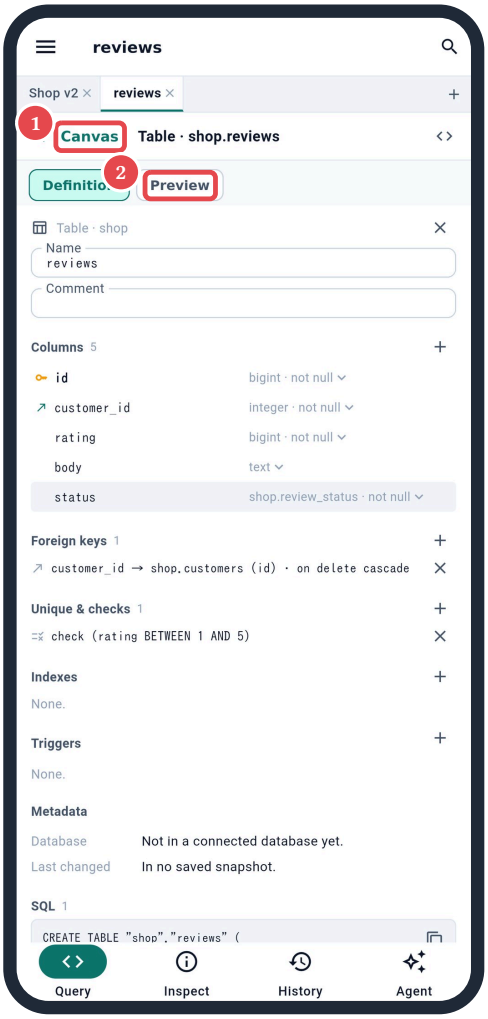
7.4 When the database has changed

If someone changed the database after you made the design — a migration, a new table — **Apply** would undo their change. Two buttons bring the design up to date:

- **Reload from the database** 3 reads the database again, lists what your design would lose, and replaces the design with the database's schema. Table positions are kept, and undo brings your version back.
- **Take from the database**, in the inspector of a table marked **changed**, replaces that one table with the database's version and keeps the rest of your design.

7.5 Full-screen editors

The expand button in the inspector opens an object in its own tab, with its definition and a preview: sample rows, the query plan, or the SQL that would run.



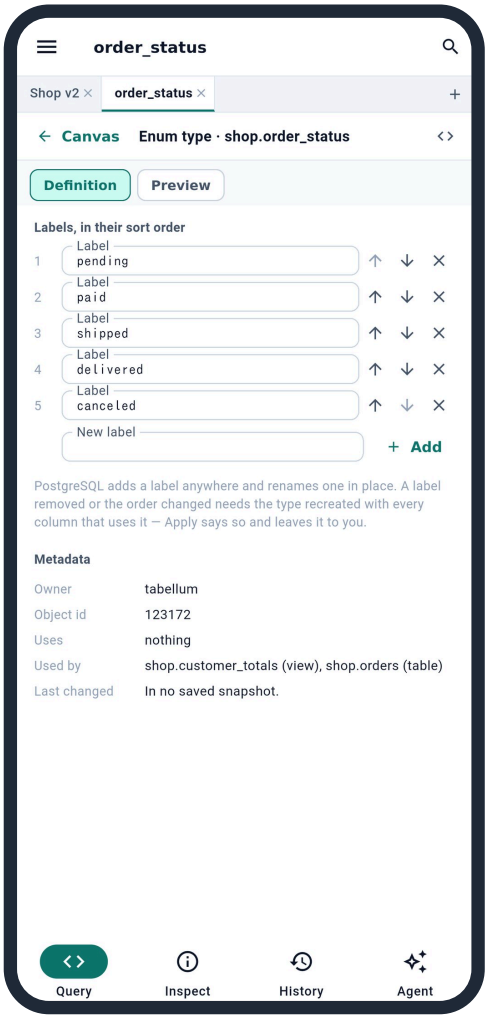


Figure 30: An enum type and its values

1. Back to the canvas
2. The form, or its preview

Figure 29: A table full-screen

Canvas 1 goes back to the design. **Definition** and **Preview** 2 switch between the form and the preview.

7.6 Creating a database

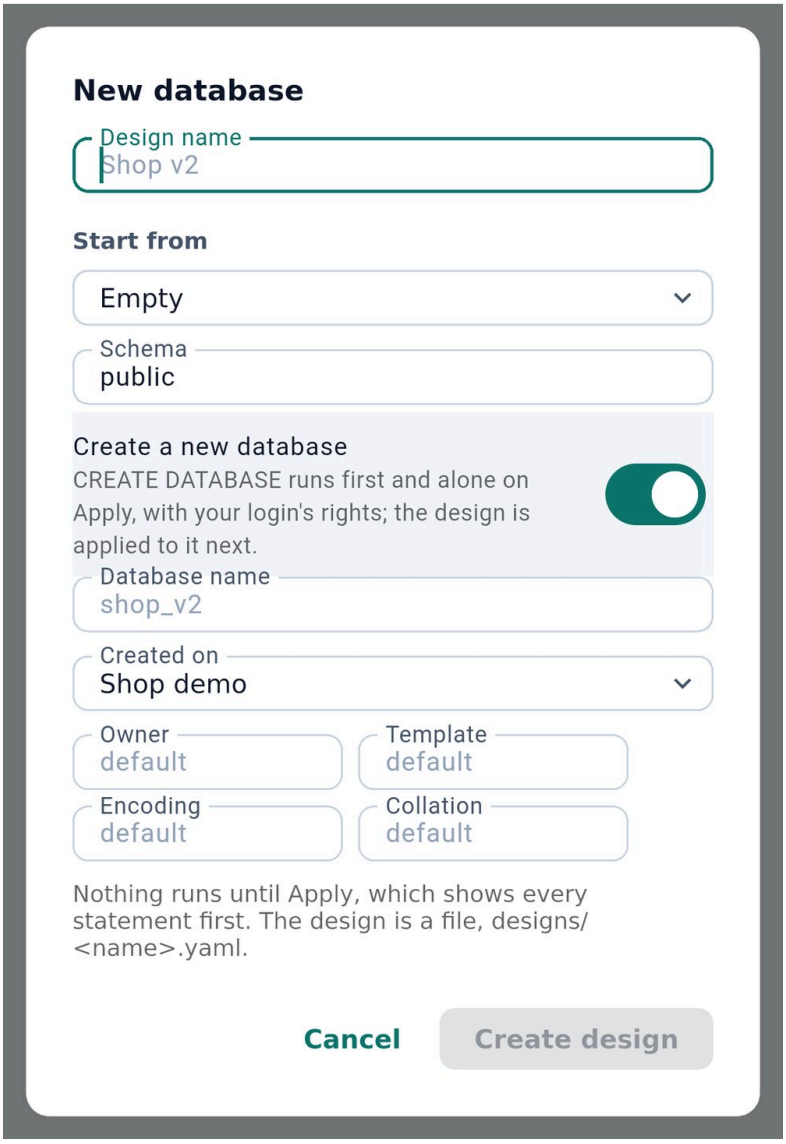


Figure 31: A design that creates its database

On PostgreSQL you choose the owner, template, encoding and collation; on MySQL the character set and collation; on SQL Server the owner and collation. Without a server, the new database is an SQLite file in the workspace.

8 Views, users and roles

8.1 Making a view without writing SQL

New view next to a schema in the menu (or **Save as view…** above a table's rows) opens the view designer: pick the source table, the columns to keep, conditions and a sort order, and whether the view is materialized. **Preview SQL** shows the `CREATE VIEW` statement, which runs through the approval dialog.

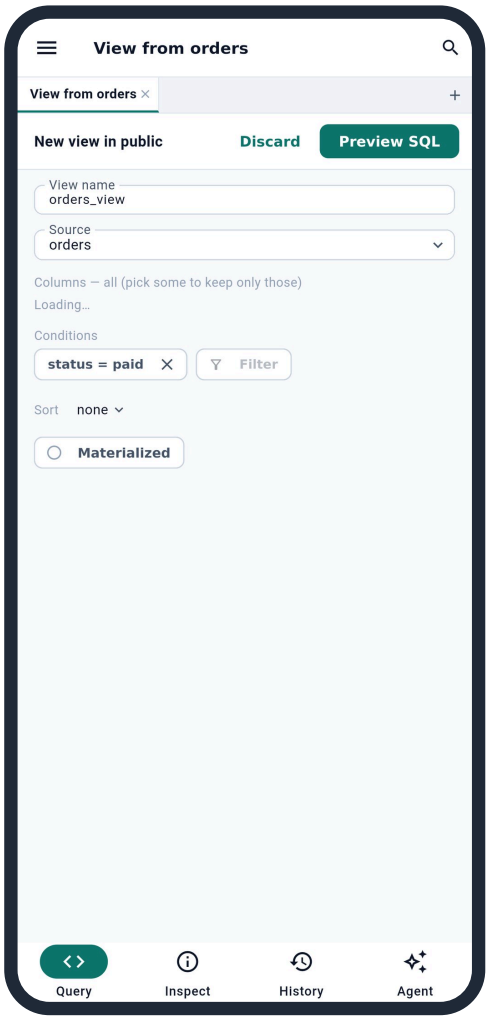


Figure 32: A view of the paid orders

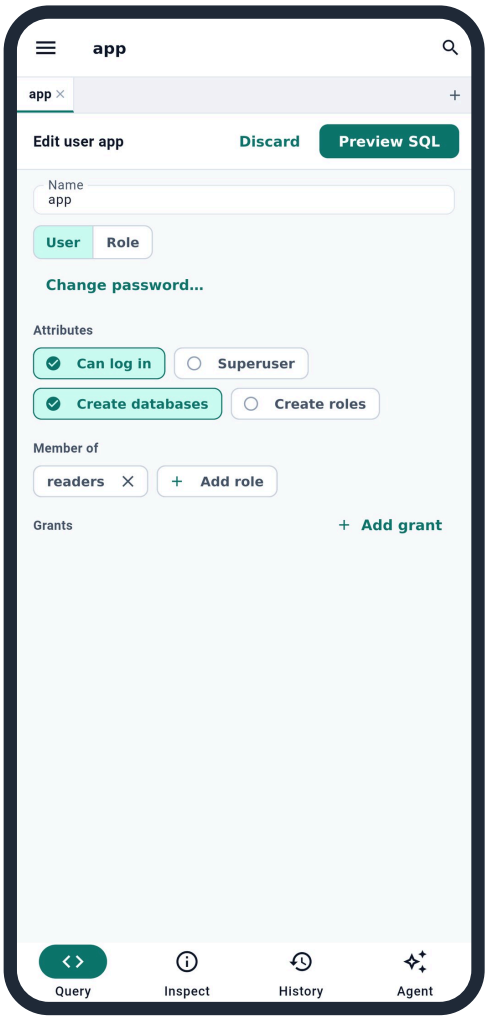


Figure 33: A user, its flags and its roles

8.2 Users and roles

Users & roles under a connection lists the logins and roles the connected user can see; a superuser is drawn in the warning color. The **+** next to it creates a user, a tap on an existing one opens it.

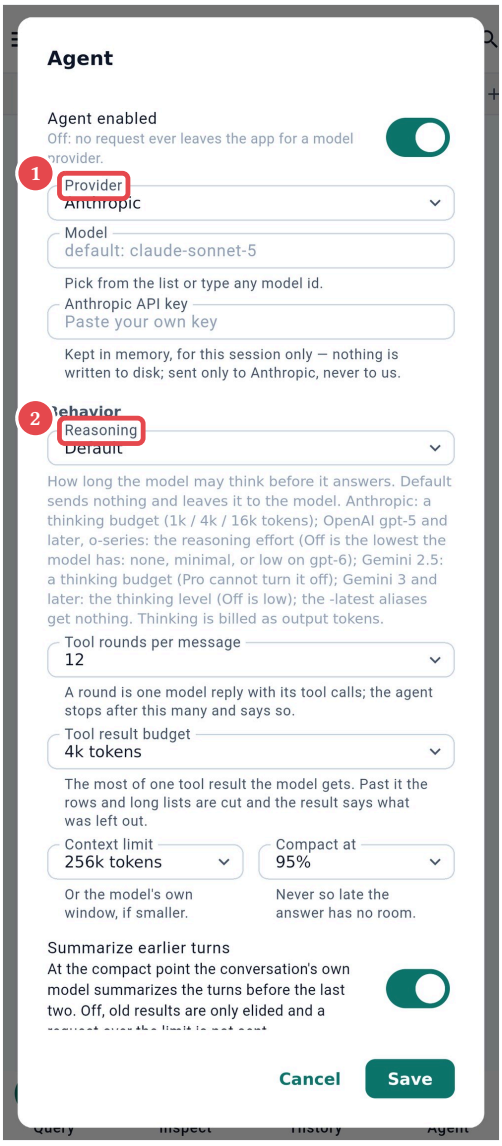
You can set the name, the password (**Change password…** on an existing user), whether it can log in, the database's own flags (on PostgreSQL **Superuser**, **Create databases** and **Create roles**), the roles it is a member of, and its grants. **Preview SQL** shows the statements; they run through the approval dialog. BigQuery has no users of its own — its access is managed in Google Cloud.

9 The AI agent

The agent answers questions about your databases in plain language: it reads the schema, runs read-only queries, explains plans, drafts SQL into the editor, builds dashboards and designs. It uses your own account with OpenAI, Anthropic or Google Gemini, and every write it proposes waits for your approval.

9.1 Setting it up

Tap **Agent** at the bottom of the screen, then the settings button.



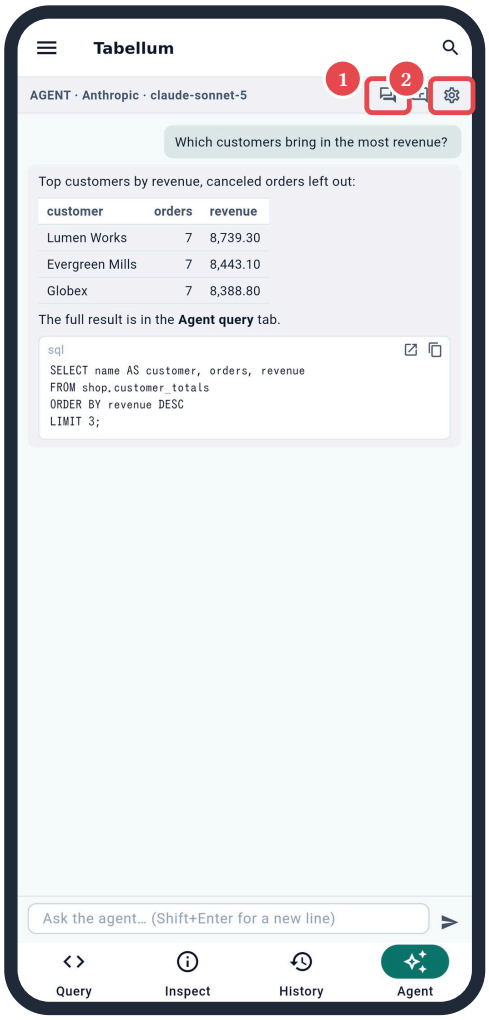
1. OpenAI, Anthropic or Gemini
2. How long the model may think

Figure 34: Agent settings

1. Turn on **Agent enabled**.
2. Choose the **Provider** 1 and a model, and paste your API key. The key is kept in secure storage and sent only to that provider.
3. Optionally change **Reasoning** 2 (how long the model may think), the number of tool rounds per message and the context limit.

9.2 Asking

Type a question and tap the send button. The agent reads what it needs and answers with tables and SQL; a longer result opens in a query tab.



1. Earlier conversations
2. Provider, model and limits

Figure 35: An answer, with the query it ran

- 1 lists your earlier conversations; **New conversation** starts a fresh one.
- 2 opens the settings.
- Once a request has gone out, **What was sent** appears next to them and shows exactly what was sent to the model, request by request.

When a conversation gets long, the agent summarizes the earlier turns so it can keep going; the summary is marked in the conversation.

9.3 What the agent can and cannot do

- It reads freely: schemas, row counts, plans, and the results of read-only queries.
- It never runs a write itself. An `UPDATE`, a `DELETE` or a schema change goes to the approval dialog (see Section 4.6), exactly as if you had typed it.
- It sees row values only on connections with **Agent may see rows** turned on. On every other connection it gets tables, columns and counts, never data. A **Sensitive** connection gives it the schema only.
- Your files can steer it: an `agent.md` in the workspace adds your own instructions to every conversation.

> **Note**
> Requests go straight from your phone to the provider you chose. Tabellum has no server of its own in between.

10 Backup, restore and sync

10.1 Backing up

Long-press a connection in the menu and choose **Back up…**.

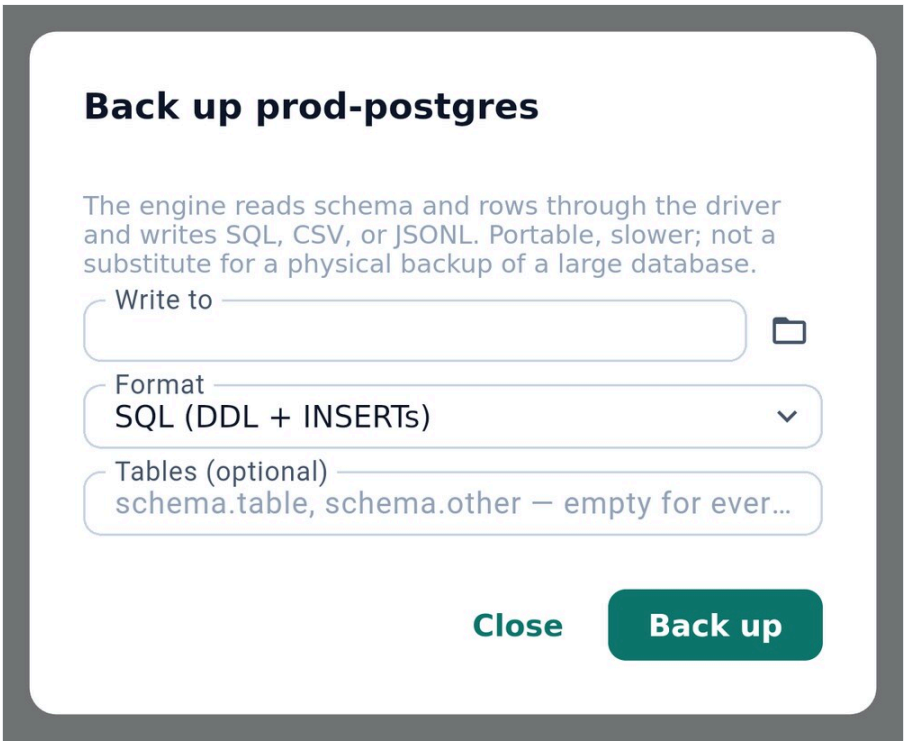


Figure 36: Backing up a database

On a phone, Tabellum backs up through the database connection itself: the schema and the rows are written as one SQL file (statements and `INSERT`s), as CSV or JSON Lines with one file per table, or as an Excel workbook with one sheet per table. **Tables** limits it to some tables. This is portable and works everywhere, but it is slower than the database's own backup tools and no substitute for them on a very large database.

10.2 Restoring

Restore…, in the same menu, runs an SQL file statement by statement, or loads CSV or JSON Lines rows into a table, in one transaction where the database supports it.

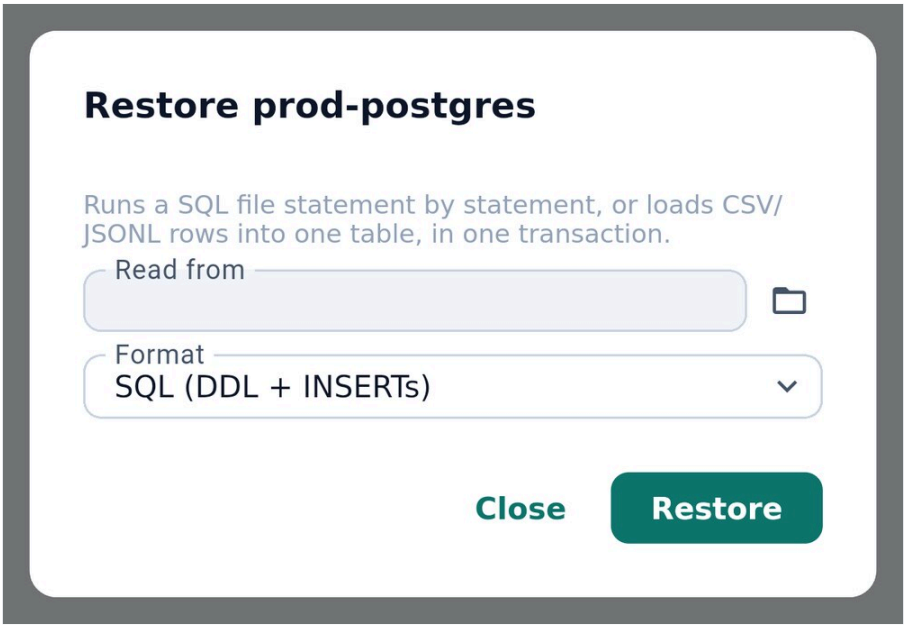
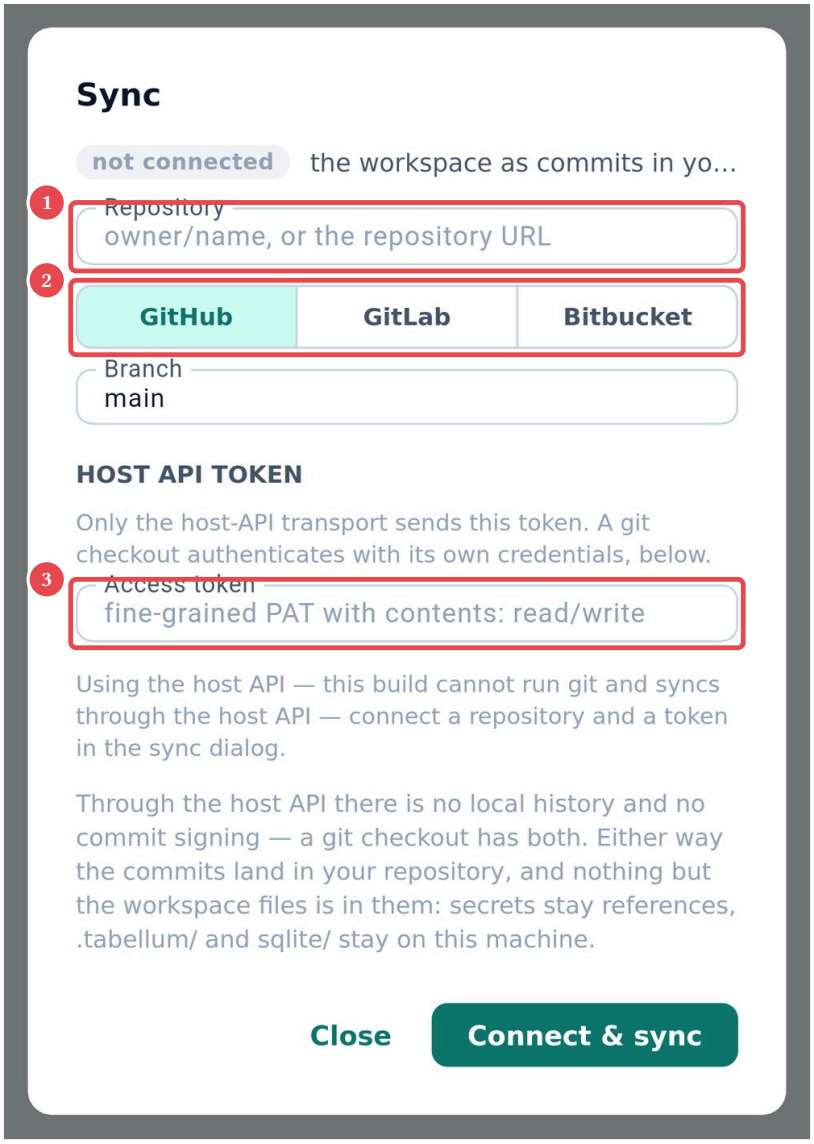


Figure 37: Restoring from a file

10.3 Syncing the workspace with git

Sync keeps the workspace in your own git repository on GitHub, GitLab or Bitbucket, so your computer — or another phone — sees the same connections, queries and dashboards.



1. Your repository
2. GitHub, GitLab or Bitbucket
3. An access token for that repository

Figure 38: Connecting the workspace to a repository

1. Choose **Sync** in the palette, or **Set up** at the bottom of the menu.
2. Enter the repository 1 and pick its host 2.
3. Paste an access token 3 that may read and write that repository.
4. Tap **Connect & sync**.

The phone syncs through the host's API, so it needs no git of its own. Each sync commits your changes and brings in everyone else's. When two people edit the same connection, dashboard or tunnel, Tabellum merges them field by field; a real conflict is shown side by side for you to choose.

> **Note**
> Passwords, keys and tokens never go into the repository — only their names. On another device you type them once.

11 Reference

11.1 Databases

Database	Notes
PostgreSQL	Everything in this guide, including enum types, extensions, materialized views and plans
MySQL, MariaDB	Schemas are databases; users and roles with the server's own flags
SQL Server	Schemas, procedures, table-valued functions, users and roles
BigQuery	A dry run tells how many bytes a statement would process before it runs; access is managed in Google Cloud
SQLite	A file in the workspace; nothing to install
MongoDB	Collections as tables; find and aggregate pipelines in the editor

11.2 This guide in Tabellum

The same guide is inside the app, screenshots and marks included, and needs no network: tap **User guide** in the command palette (the search button in the title bar), or the **User guide** button in Help (the **?** at the bottom of the menu). The chapter list switches to the desktop guide. On the website it is at tabellum.dudko.dev/manual, a page per chapter, with this PDF.

11.3 On a phone

The phone app is the same Tabellum as on a computer, with these differences:

- The workspace is inside the app, not a folder you choose.
- Backups go through the database connection, not tools such as `pg_dump`.
- There is no SSH agent; paste private keys into the tunnel instead.
- Sync goes through the GitHub, GitLab or Bitbucket API.
- There is no MCP server for external AI tools.
- Updates come from the App Store or Google Play.

The engine that talks to the databases runs inside the app. **Engine** in the command palette can point the app at a Tabellum worker you host instead, for databases only that machine can reach.

11.4 If something goes wrong

- **A connection fails.** The message under it says what failed and what to check. "Connection refused" means nothing listens on that host and port; a timeout usually means a firewall or a VPN — or a server that only your office network can reach.
- **A query takes too long.** Tap the red **Stop** button.
- **The agent does not answer.** Check its API key and model in the agent's settings; **What was sent** in the agent's page shows each request.
- **A design wants to undo someone else's change.** Use **Reload from the database** (see Section 7).

Support: tabellum.dudko.dev/support.