DB CONSOLE

DATENBANK-KONSOLE

Deine Datenbanken — durchgesehen, abgefragt und nur verändert, wenn du es sagst, mit einem Backup darunter.

SO FUNKTIONIERT ES

BENUTZERHANDBUCH · OKTOBER 2026

777 SOFTWARE

777 ADMIN TOOLS

1 — WAS DB CONSOLE IST

Eine Datenbank ist der Ort, an dem ein Programm seine Fakten aufbewahrt: die Benutzer, die Bestellungen, die Einstellungen, die Geschichte. Eine Datenbank-Admin-Konsole ist die Werkbank, um in eine hineinzusehen — ihre Tabellen zu sehen, ihre Zeilen zu lesen, ihr Fragen in ihrer eigenen Sprache (SQL) zu stellen und, wenn der Admin es entscheidet, zu verändern, was darin ist.

DB Console ist diese Werkbank als ein Programm, das auf deinem eigenen Computer lebt. Es begann als ein kleines Werkzeug, gebaut für die eigene Admin-Arbeit dieses Hauses; die eigenständige Version behält die Seele dieses Werkzeugs genau — sie findet deine Datenbanken für dich, sie sichert, bevor sie schreibt, und sie blockiert jede Veränderung, bis der Admin es sagt — und wächst sie zu einem Desktop-Programm, das jeder Admin auf seine eigenen SQLite- und PostgreSQL-Datenbanken zeigen kann.

> *Die ganze Persönlichkeit von DB Console ist es, den Admin vor dem dicken Finger zu schützen. Jedes Datenbank-Werkzeug der Erde führt fröhlich die schlimmste Abfrage deines Lebens um 3 Uhr morgens aus. Dieses hier behandelt deinen eigenen schlimmsten Moment als einen Design-Fall: Es startet nur-lesend, es lässt dich jede Veränderung prüfen, bevor sie läuft, es kopiert die Datenbank, bevor irgendein Schreiben landet, und es macht die wirklich gefährlichen Anweisungen schwer, aus Versehen abzufeuern. Die Abschnitte 5–7 sind dieses System, Stück für Stück.*

Was es nicht ist: Es ist kein Server, und es ist kein Dienst, der läuft, wenn du nicht hinsiehst. Es öffnet sich, wenn du es öffnest, es verbindet sich nur dorthin, wohin du es zeigst, und wenn du es schließt, ist es geschlossen. Es gibt keine Konten, nichts meldet sich bei irgendjemandem zurück, und das Programm wird ausgeliefert mit dem Wissen, wie man mit Datenbanken spricht — niemals damit, wo deine leben (§10).

2 — SICH MIT DEINEN DATENBANKEN VERBINDEN

DB Console spricht heute mit zwei Arten von Datenbank. Sie verbinden sich unterschiedlich, denn sie leben unterschiedlich:

Art	So verbindest du dich
SQLite — eine Datenbank, die eine Datei ist	Eine SQLite-Datenbank ist eine Datei auf einer Platte. Öffne eine direkt, oder lasse die Konsole sie für dich finden: Du genehmigst die Ordner, in denen sie suchen darf, sie prüft den Identitäts-Kopf jeder Kandidaten-Datei, um sicher zu sein, dass sie wirklich eine SQLite-Datenbank ist, und listet jede, die sie findet — mit ihrer Größe und Tabellenzahl — einmal, selbst wenn eine Datei an zwei Orten auftaucht. Sie sucht nur in Ordnern, die du genehmigt hast. Es gibt keine eingebaute Haus-Liste: Das Programm kommt an, ohne irgendeinen Ordner zu kennen.
PostgreSQL — eine Datenbank, die ein Server ist	Eine PostgreSQL-Datenbank lebt auf einem Server, also füllst du ein kurzes Formular aus: die Adresse des Servers, seinen Port, den Datenbanknamen und deinen Benutzernamen — eine Adresse wie db.example.local. Das Adressfeld wird leer ausgeliefert: Die erste Adresse, die es jemals enthält, ist eine, die du getippt hast. Die Verbindung, die es herstellt, ist eine echte, zu dem Server, den du benannt hast — die Konsole tut niemals so, als wäre sie mit etwas verbunden, das sie nicht wirklich erreicht hat.

Gespeicherte Verbindungen

- Verbindungen, die du behältst, werden unter Namen gespeichert, in deinen eigenen Worten — „Shop Postgres", „Laptop data store". Eine gespeicherte Verbindung erinnert sich an die Adresse, den Port, den Benutzernamen und die Dateipfade — die Wegbeschreibung, nicht die Schlüssel.
- Passwörter werden niemals in diese gespeicherte Liste geschrieben. Der speichernde Code selbst streift alles heraus, das wie ein Passwort geformt ist, bevor die Liste die Platte berührt. Passwörter leben nur an einem Ort: dem verschlüsselten Tresor (§8).
- Jede neue Verbindung startet nur-lesend (§5) — einschließlich solcher, die du vor langer Zeit gespeichert und einfach noch nicht umgelegt hast. Lesen ist immer erlaubt; irgendetwas zu verändern, braucht eine Entscheidung, die du absichtlich triffst.

3 — SICH UMSEHEN: BAUM, RASTER, STRUKTUR

Sobald du verbunden bist, gibt dir das Fenster drei Arten, auf dieselbe Datenbank zu sehen. Zusammen beantworten sie: „Was ist hier drin, und wie sieht es aus?"

Ansicht	Was sie dir zeigt
Der Schema-Baum	Die Datenbank als Gliederung: ihre Tabellen und Sichten, jede öffnet sich, um ihre Spalten zu zeigen. Dies ist die Karte — was hier ist, bevor du irgendetwas berührst.
Das Daten-Raster	Die Zeilen einer Tabelle, 50 pro Seite, mit Seiten-Steuerungen. Klicke auf eine Spaltenüberschrift, um nach dieser Spalte zu sortieren. Ein Suchfeld jagt deine Worte durch jede Spalte auf einmal. Leere Zellen zeigen sich als NULL — das Wort der Datenbank für „hier ist nichts gespeichert" — und binäre Daten zeigen sich als ein klares Etikett wie [blob 12B]: gezeigt als das, was sie sind, niemals geraten, niemals ein Absturz.
Die Struktur-Ansicht	Eine Tabelle, auseinandergenommen: jede Spalte mit ihrem Typ und Standard, der Primärschlüssel, die Fremdschlüssel, die sie mit anderen Tabellen verbinden (in beiden Richtungen), ihre Indexe und die echte CREATE-Anweisung der Tabelle — die genauen Worte, die sie gebaut haben, lesbar und kopierbar.

Hinsehen verändert nichts. Du kannst den ganzen Tag in diesem Fenster eine Produktions-Datenbank durchsehen, und die Datenbank wird niemals wissen, dass du da warst — das ist, was nur-lesend bedeutet, und es ist der Zustand, in dem du startest (§5).

4 — DER SQL-EDITOR: FRAGE SIE SELBST

Der Editor ist der Ort, an dem du SQL schreibst — die eigene Sprache der Datenbank — wenn Baum und Raster nicht die Frage sind, die du beantwortet brauchst. Tippe eine Anweisung, führe das Ganze aus oder nur den Teil, den du ausgewählt hast, und die Antwort landet in einem Ergebnis-Raster unter dem Editor, in derselben sauberen Tabelle wie der Browser.

- **Die 500-Zeilen-Decke, laut ausgesprochen.** Eine Frage kann eine Million Zeilen treffen; das Raster zeigt die ersten 500 und sagt es. Eine gedeckelte Antwort ist als gedeckelt beschriftet — niemals still beschnitten und als die ganze Wahrheit vorgelegt.
- **Nimm die Antwort mit.** Ergebnisse exportieren sich in eine CSV- oder JSON-Datei auf deiner eigenen Platte — deine, in offenen Formaten, zu öffnen in allem.
- **Lesende Anweisungen laufen.** Eine Frage (ein SELECT) läuft einfach — Lesen ist, wofür die Konsole da ist.
- **Verändernde Anweisungen laufen nicht — noch nicht.** Wenn deine Anweisung etwas verändern würde (INSERT, UPDATE, DELETE, DROP und ihre Verwandten), wird sie nicht ausgeführt, wenn du auf Ausführen drückst. Sie geht in die Ablage für ausstehende

Änderungen (§6), und das Sicherheitssystem übernimmt sie von dort. Diese Umleitung ist keine Ablehnung; sie ist das Produkt, das wie entworfen arbeitet.

> *Ein Satz zum Merken: In DB Console sind eine Änderung auszuführen und eine Änderung landen zu lassen zwei verschiedene Handlungen. Du kannst alles schreiben. Nichts landet, bis es die Ablage, die Warnungen und dein Commit („Festschreiben") passiert hat.*

5 — STANDARDMÄSSIG NUR LESEN: DER GOLDENE RAHMEN

Jede Verbindung — eine Datei, die du gerade geöffnet hast, ein Server, den du gerade gespeichert hast, einer, den du vor einem Jahr gespeichert hast — startet nur-lesend und bleibt es, bis du persönlich diese Verbindung auf schreibbar umlegst. Das Umlegen gilt pro Verbindung und wird erinnert: Deine Spiel-Datenbank kann schreibbar leben, während die Produktions-Datenbank niemals das Gold verlässt.

Und du kannst immer sehen, in welcher Welt du bist. Der Fensterrahmen selbst trägt den Modus, in Farbe, zu jeder Zeit:

Rahmen	Was es bedeutet
GOLD — READ-ONLY (Gold — nur lesen)	Du kannst alles ansehen und nichts verändern. Verändernde Anweisungen wird ein Landeplatz verweigert — sie können nicht einmal in die Ablage gelangen. Dies ist der Startzustand jeder Verbindung.
RED — WRITABLE (Rot — schreibbar)	Du hast diese Verbindung absichtlich umgelegt. Veränderungen sind möglich — durch die Ablage, das Backup und die Wächter (§6–§7), niemals um sie herum. Die rote Kante ist da, damit du deine Gefahrenstufe quer durch den Raum sehen kannst.

Warum standardmäßig Gold? Weil die meiste Admin-Arbeit Lesen ist — einen Wert prüfen, eine Zeile jagen, eine Frage beantworten — und das eine Mal von hundert, bei dem du etwas veränderst, genau das Mal ist, bei dem du es zuerst laut sagen müssen solltest. Standardmäßig nur-lesend bedeutet, dass der Unfall absichtlich sein muss, bevor er echt werden kann.

6 — DIE ABLAGE FÜR AUSSTEHENDE ÄNDERUNGEN: PRÜFEN, BEVOR ES LÄUFT

Auf einer schreibbaren Verbindung beginnt eine verändernde Anweisung eine Reise mit vier Stationen — und sie kann an jeder von ihnen weggeworfen werden:

1. **Sie wird geschrieben.** Du tippst die Anweisung in den Editor und führst sie aus. Der Datenbank ist nichts geschehen.

2. **Sie wartet in der Ablage.** Die Anweisung sammelt sich in der Ablage für ausstehende Änderungen, die dir — in klarem SQL, genau so, wie sie laufen wird — jede Anweisung zeigt, die gleich ausgeführt wird. Ein Zähler im Fensterrahmen hält die Anzahl vor deinen Augen: *3 changes pending* („3 Änderungen ausstehend").
3. **Sie wird gewarnt, wenn sie eine Warnung verdient.** Manche Anweisungen lösen die Wächter aus (§7) und halten hier für ein Gespräch in klaren Worten an, bevor sie weiter können.
4. **Du entscheidest.** *Commit* („Festschreiben") führt die ganze Ablage aus — nachdem das Backup (§7) sicher gemacht ist. *Discard* („Verwerfen") wirft sie weg, und die Datenbank erfährt niemals davon.

Die Ablage ist der Unterschied zwischen einer Konsole, die tut, was du getippt hast, und einer, die tut, was du gemeint hast. Ein Tippfehler in einer Anweisung ist eine Kuriosität, während er in der Ablage sitzt, lesbar in deiner Mußе. Derselbe Tippfehler, auf Klick ausgeführt, ist ein Vorfall. DB Console hält jede Änderung eine Kuriosität, bis du sie persönlich beförderst.

> *Schließe das Fenster mit ausstehenden Änderungen, und die Konsole fragt dich namentlich, was mit ihnen zu tun ist. Nichts Ausstehendes verdunstet jemals still — und nichts Ausstehendes schleicht sich jemals durch, auch nicht.*

7 — BACKUP VOR DEM SCHREIBEN, UND DIE WÄCHTER

Das Backup

Wenn du *Commit* drückst, ist die erste Handlung der Konsole nicht die Änderung — sie ist die Kopie. Für eine SQLite-Datenbank wird die ganze Datei in ein Backup mit Zeitstempel kopiert (derselbe Name, mit .bak und der hinzugefügten Zeit), und erst dann läuft die Ablage. Die neuesten zehn Backups werden behalten. Zwei Details zählen:

- Die Kopie wird gemacht, bevor irgendetwas ausgeführt wird — also existiert das Backup noch, wenn ein Schreiben dann auf halbem Weg fehlschlägt. Das Netz wird vor dem Sprung gewoben, nicht nach dem Fall.
- Ein schlechtes Schreiben ist daher immer rückgängig zu machen: Lege die Backup-Datei zurück an den Platz der Datenbank, und du stehst dort, wo du vor der Änderung standest — jede Zeile, jede Tabelle.

Für einen PostgreSQL-Server gibt es keine einzelne Datei zu kopieren, und dieses Handbuch wird nichts anderes vortäuschen. Dort ist dein Netz der Rest des Systems, und er ist echt: standardmäßig nur-lesend (§5), jede Änderung in der Ablage geprüft (§6) und die Wächter unten — plus welche Backups der Server selbst behält, die die gute Praxis des Server-Admins selbst bleiben.

Die Wächter — klare Worte vor klarem Schaden

Anweisung	Was die Konsole sagt und tut
DELETE oder UPDATE ohne WHERE	Eine WHERE-Klausel ist der Teil, der sagt, welche Zeilen. Ohne eine bedeutet die Anweisung jede Zeile — also sagt die Konsole genau das, in klaren Worten, bevor die Änderung festgeschrieben werden kann: Dies wird jede Zeile in der Tabelle verändern. Wahrscheinlich nicht, was du gemeint hast — und wenn es ist, was du gemeint hast, kannst du es sagen und fortfahren, mit offenen Augen.
DROP oder TRUNCATE	Diese verändern eine Tabelle nicht; sie beenden eine. Sie bleiben entschärft, bis du den genauen Namen der Tabelle tippst, um das *Commit* zu scharfmachen — kein Kästchen zum Versehentlich-Anhaken, kein Knopf, den ein verirrter Klick abfeuern kann. Und dieser Wächter lebt im Kern der Konsole, nicht nur an ihren Knöpfen: Ein DROP, das nicht mit dem getippten Namen scharfgemacht wurde, kann sich nicht festschreiben, auf keinem Weg.

Alles andere, das Daten verändert, bekommt die Prüfung der Ablage als seine Untersuchung. Nichts Zerstörerisch-Ausgehendes bekommt jemals einen stillen Lauf.

8 — DER TRESOR: WO PASSWÖRTER LEBEN

Eine Server-Verbindung braucht ein Passwort. Die Regel von DB Console ist absolut: Passwörter sind niemals Klartext, nirgendwo — nicht in der Liste der gespeicherten Verbindungen (§2), nicht in einem Protokoll, nicht auf einem Bildschirm, nachdem du sie getippt hast. Sie leben in einem verschlüsselten Tresor: eine Datei, auf deinem eigenen Computer, versiegelt mit einem Hauptpasswort, das du wählst.

- **Das Hauptpasswort wird niemals gespeichert.** Nicht im Tresor, nicht irgendwo. Wenn du es vergisst, bleibt der Tresor versiegelt — das ist kein Fehler im Design; das ist das Design. Wähle eines, an das du dich erinnern wirst.
- **Wie das Versiegeln funktioniert, in klaren Worten:** Dein Hauptpasswort wird durch eine Einweg-Mühle gefahren — eine Ableitung namens PBKDF2 — 600.000 Mal, mit einer zufälligen Zutat (einem Salt) hineingemischt, um den Schlüssel zu machen. Die Geheimnisse selbst werden mit AES-GCM versiegelt, der standardmäßigen authentifizierten Verschlüsselung: Verändere ein Bit der versiegelten Datei, und sie wird sich nicht öffnen, statt sich falsch zu öffnen.
- **Die Tresor-Datei ist allein auf dein Konto gesperrt** (Berechtigungen 0600) — andere Benutzer desselben Rechners können sie überhaupt nicht lesen, und ohne das Hauptpasswort kann es auch sonst niemand.

- **Ein falsches Passwort scheitert laut.** Tippe das falsche Hauptpasswort, und der Tresor verweigert sich, in klaren Worten. Er öffnet sich niemals halb, rät niemals, lässt niemals einen falschen Schlüssel innen herumklappern.
- **Ein Rückfall ist eingebaut und bewährt.** Auf einem System, dem die standardmäßige Verschlüsselungs-Bibliothek fehlt, versiegelt die Konsole den Tresor mit einer eingebauten Methode, gebaut auf denselben Prinzipien — dieselbe Ableitung in 600.000 Runden, dieselbe Nur-Konto-Sperre — statt jemals auf Klartext zurückzufallen. Es gibt keinen Weg in diesem Programm, der damit endet, dass ein Passwort lesbar auf einer Platte sitzt.

> *Die ganze Zugangsdaten-Geschichte in einer Zeile: Wegbeschreibungen im Offenen, Passwörter im Tresor, und der Schlüssel des Tresors existiert nur in deinem Kopf.*

9 — DIE EHRLICHEN GRENZEN

Ein Werkzeug, dem du vertrauen kannst, sagt dir, was es nicht kann — und was es einfach noch nicht tut. Dies ist die vollständige Liste für diesen Build:

- **Zwei Engines heute.** Dieser Build spricht SQLite und PostgreSQL. MySQL und MariaDB stehen auf dem Plan für eine spätere Phase — sie sind nicht in diesem Build, und nichts in diesem Handbuch sollte so gelesen werden, als sagte es, dass sie es wären.
- **Änderungen werden geschrieben, nicht geklickt.** In dieser Phase veränderst du Daten, indem du SQL in den Editor schreibst; die Änderungen durchlaufen dann die Ablage, das Backup und die Wächter wie jedes andere Schreiben. Ein Zeilen-Bearbeitungsformular per Zeigen und Klicken ist Arbeit einer späteren Phase — das Sicherheitssystem wurde zuerst gebaut, absichtlich.
- **Die Backup-Kopie ist die Geschichte einer Datei.** Das Backup mit Zeitstempel (§7) schützt SQLite-Datenbanken, die Dateien sind. Die Sicherheit eines PostgreSQL-Servers hier ist die Nur-Lesen-Standardeinstellung, die Prüf-Ablage und die Wächter — klar gesagt, denn vorzutäuschen, eine Dateikopie existierte für einen Server, wäre eine Lüge mit einem Fortschrittsbalken.
- **Zwei kleine Kosmetika, eingestanden.** In einem schmalen Fenster (800 Pixel breit) können die Seiten-Knöpfe des Rasters an der rechten Kante abgeschnitten werden, und in einem kurzen Fenster (600 Pixel hoch) wird der Schema-Baum eingequetscht. Beide sind bekannt, beide sind kosmetisch, und keines berührt ein einziges Byte deiner Daten.
- **Bewährt an Test-Datenbanken; der Test des Gründers ist das Tor.** Jeder Schutz in diesem Handbuch wurde gegen eigens gebaute Test-Datenbanken bewährt — 37 Kernprüfungen und 15 Fensterprüfungen, alle bestanden. Das Programm wurde beim Testen absichtlich niemals auf eine echte Arbeits-Datenbank gezeigt. Es wird erst

veröffentlicht, nachdem der Gründer es mit seinen eigenen Händen benutzt hat.

10 — DEIN RECHNER, DEINE SACHE

Die Privatsphären-Geschichte ist kurz genug, um sie ganz zu drucken:

- **Keine Konten. Keine Telemetrie. Keine Update-Prüfungen.** Die Konsole meldet sich niemals bei irgendjemandem zurück, denn es gibt niemanden, bei dem sie sich zurückmelden könnte.
- **Ihr einziger Netzwerkverkehr sind Verbindungen, die du definierst** — zu deinen eigenen Datenbank-Servern, wenn du dich mit ihnen verbindest. Bevor dieser Build hinausgelassen werden durfte, wurde sein Inneres String für String durchsucht: Er enthält überhaupt keine eingebauten Adressen. Nicht unsere, nicht irgendjemandes. Jede Adresse, die er jemals enthalten wird, ist eine, die du getippt hast.
- **Alles, was er behält, lebt bei dir:** die gespeicherten Verbindungen, der Tresor, deine exportierten CSV- und JSON-Dateien — klare lokale Dateien auf deinem eigenen Rechner, exportierbar und löschbar, deine.
- **Er kann nur verletzen, worauf du ihn zeigst** — und das ganze Sicherheitssystem (§5–§7) existiert, damit selbst dort, etwas zu verletzen, eine geprüfte, gesicherte, absichtliche Handlung braucht. Ein Admin-Werkzeug sollte die sorgfältige Sache zur einfachen Sache machen. Dieses hier wurde dazu gebaut.

> *Das ist das Ganze davon. Öffne deine Datenbanken, lies sie im goldenen Rahmen, und wenn eine Änderung wirklich nötig ist — lege die Verbindung um, prüfe die Ablage, lasse das Backup landen und schreibe mit offenen Augen fest. Das Versprechen der Konsole ist nicht, dass nichts schiefgehen kann. Es ist, dass nichts aus Versehen schiefgeht.*

DB Console — Enclave DB Console · 777 Software · 777 Admin Tools. Dieses Handbuch wird mit dem Programm geliefert. Deine Datenbanken gehören dir, laufen auf deine Art; die Form dieses Werkzeugs — zuerst nur lesen, prüfen vor dem Ausführen, ein Backup unter jedem Schreiben — verändert sich nicht.