DataBlick Handbuch

Stand: 0.99b, Juni 2026

DataBlick ist ein Lernwerkzeug für relationale Datenbanken im Informatikunterricht. Es arbeitet mit SQLite-Datenbanken und verbindet eine übersichtliche grafische Bedienung mit sichtbarem SQL, Import und Export, Schema-Diagramm, grafischem Abfrageentwurf und didaktischen Fehlermeldungen.

Dieses Handbuch beschreibt die wichtigsten Arbeitsabläufe für Lehrkräfte und Lernende. Es ist bewusst nicht nur als technische Bedienungsanleitung geschrieben. Die einzelnen Funktionen werden so erklärt, dass deutlich wird, welches Datenbankkonzept jeweils dahintersteht. DataBlick soll nicht nur dabei helfen, eine Datenbank zu bedienen, sondern auch zu verstehen, wie Tabellen, Schlüssel, Beziehungen und Abfragen zusammenwirken.

Wichtig ist die Unterscheidung zwischen Programm und Datenbank: DataBlick ist nicht die Datenbank selbst. DataBlick ist das Programm, mit dem eine Datenbank geöffnet, betrachtet, verändert und abgefragt wird. Die eigentliche Datenbank ist eine SQLite-Datei, meist mit der Endung `.db`. Diese Datei enthält die Tabellen, Beziehungen und weitere Datenbankobjekte. DataBlick speichert eigene Abfragen zusätzlich in internen Metadaten.

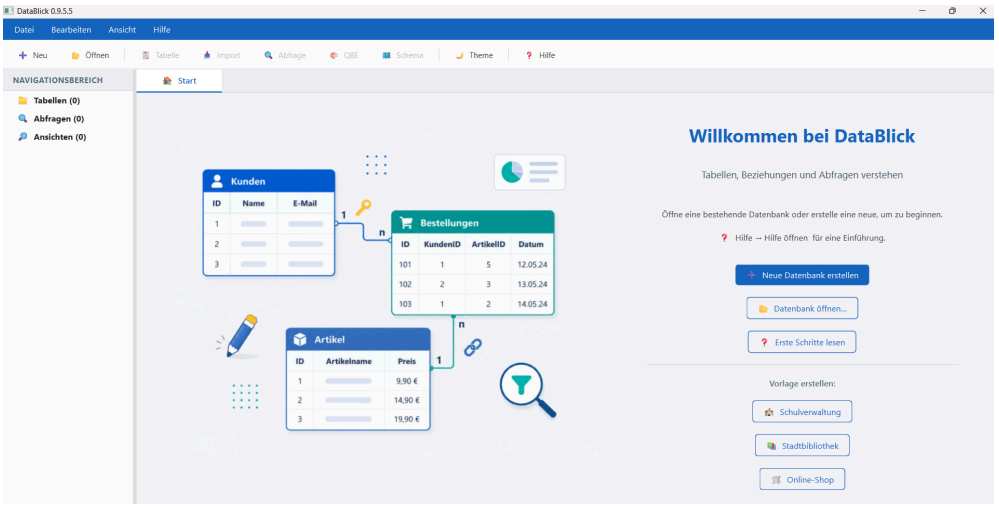
1. Installation und Start

Für den normalen Unterrichtseinsatz ist DataBlick als ausführbares Programm vorgesehen. Auf Windows wird DataBlick z.B. über die Datei `DataBlick.exe` gestartet. Lernende müssen dafür keine Python-Befehle eingeben und benötigen keine eigene Python-Installation.

Der typische Ablauf im Unterricht ist einfach:

1. Den DataBlick-Ordner öffnen.
2. `DataBlick.exe` starten.
3. Eine neue Datenbank anlegen oder eine vorhandene `.db`-Datei öffnen.

Nach dem Start erscheint die Willkommensansicht. Von dort aus kann eine neue Datenbank angelegt, eine bestehende Datenbank geöffnet oder die Hilfe aufgerufen werden. Im Abschnitt **Vorlagen** stehen drei fertige Beispieldatenbanken bereit: **Schulverwaltung**, **Stadtbibliothek** und **Online-Shop**. Ein Klick auf eine Vorlage erzeugt sofort eine neue, vollständig befüllte Datenbank — praktisch für schnelle Unterrichtseinstiege ohne eigenes Datenmodell.



DataBlick Startseite mit Vorlagen-Datenbanken

Für den schulischen Einsatz sollte vorab geprüft werden, ob DataBlick auf den Zielrechnern gestartet werden darf. Manche Schulumgebungen blockieren unbekannte ausführbare Dateien oder erlauben das Starten nur aus bestimmten Verzeichnissen. In diesem Fall muss DataBlick durch die schulische IT freigegeben oder in einem erlaubten Programmordner bereitgestellt werden.

DataBlick speichert Datenbanken nicht in einem proprietären Sonderformat. Eine Datenbank ist eine SQLite-Datei. Wer eine Datenbank weitergeben, sichern oder auf einem anderen Rechner öffnen möchte, gibt daher normalerweise die `.db`-Datei weiter. Das Programm DataBlick und die Datenbankdatei sind zwei verschiedene Dinge.

2. Grundbegriffe

DataBlick verwendet normale SQLite-Datenbanken. Eine Datenbank besteht aus Tabellen. Tabellen bestehen aus Spalten und Zeilen. Jede Zeile beschreibt einen Datensatz, jede Spalte beschreibt eine Eigenschaft dieser Datensätze.

Ein einfaches Beispiel ist eine Tabelle `kunden`. Jede Zeile steht für einen Kunden. Die Spalten können zum Beispiel `id`, `name`, `ort` und `email` heißen. Eine zweite Tabelle `bestellungen` könnte Bestellungen speichern und über eine Kundennummer auf die Tabelle `kunden` verweisen. Genau darin liegt die Grundidee relationaler Datenbanken: Informationen werden nicht beliebig in einer einzigen großen Tabelle gesammelt, sondern sinnvoll auf mehrere Tabellen verteilt und über Schlüssel miteinander verbunden.

Wichtige Begriffe:

- **Datenbank**: Eine strukturierte Sammlung von Daten. In DataBlick ist dies technisch eine SQLite-Datei, zum Beispiel `schulbibliothek.db`.
- **Datenbankverwaltungssystem**: Die Softwarekomponente, die Daten speichert, Abfragen ausführt und Regeln der Datenbank durchsetzt. Bei DataBlick übernimmt SQLite diese Aufgabe.
- **DataBlick**: Das grafische Lernwerkzeug, mit dem die SQLite-Datenbank im Unterricht bedient wird.
- **Tabelle**: Sammlung gleichartiger Datensätze, zum Beispiel `kunden`, `artikel` oder `ausleihen`.
- **Datensatz**: Eine einzelne Zeile in einer Tabelle. Ein Datensatz in `kunden` beschreibt also genau einen Kunden.
- **Feld / Spalte**: Eigenschaft eines Datensatzes, zum Beispiel `name`, `email` oder `geburtsdatum`.
- **Datentyp**: Art der gespeicherten Werte, zum Beispiel Text, ganze Zahl oder Dezimalzahl.

- **Primärschlüssel**: Eindeutige Kennung eines Datensatzes, meist ein Feld wie `id`.
- **Fremdschlüssel**: Verweis auf einen Datensatz in einer anderen Tabelle.
- **Abfrage**: Eine Frage an die Datenbank, meist als SQL oder grafisch als QBE.
- **Gespeicherte Abfrage**: Eine in DataBlick abgelegte SQL- oder QBE-Abfrage. Sie kann später erneut geöffnet, kopiert, umbenannt oder exportiert werden.
- **View**: Eine SQLite-Ansicht innerhalb der Datenbank. DataBlick zeigt Views im Navigationsbereich unter **Ansichten** und öffnet sie als schreibgeschützte Datenblattansicht. Im QBE-Designer stehen Views als Abfragequelle zur Verfügung. Über den Button **Als View** im SQL-Editor kann ein `SELECT`-Statement direkt als View in der Datenbank gespeichert werden. Die entstandene View erscheint sofort in der Seitenleiste und im Schema-Diagramm.

Diese Begriffe beschreiben unterschiedliche Ebenen einer Datenbank. Die Tabelle ist die sichtbare Struktur, der Datensatz ist ein einzelnes Objekt innerhalb dieser Struktur, der Primärschlüssel macht Datensätze eindeutig unterscheidbar, und Fremdschlüssel verbinden Datensätze aus verschiedenen Tabellen. DataBlick zeigt diese Ebenen nicht nur als Begriffe, sondern macht sie in den verschiedenen Ansichten praktisch erfahrbar.

3. Eine Datenbank anlegen oder öffnen

Eine neue Datenbank wird über **Datei → Neue Datenbank** oder `Strg+N` angelegt. DataBlick erstellt dabei eine SQLite-Datei. Diese Datei kann später auch in anderen SQLite-Programmen geöffnet werden, zum Beispiel in DB Browser for SQLite, SQLiteStudio oder DBeaver.

Eine bestehende Datenbank wird über **Datei → Öffnen** oder `Strg+O` geladen. Nach dem Öffnen erscheinen Tabellen und gespeicherte Abfragen links im Navigationsbereich.

Im Navigationsbereich stehen wichtige Objektaktionen über Rechtsklick zur Verfügung. Tabellen können dort in der Datenblatt- oder Entwurfsansicht geöffnet, umbenannt, kopiert, exportiert oder gelöscht werden. Gespeicherte Abfragen können geöffnet, umbenannt, kopiert, als SQL-Code exportiert oder mit aktuellem Ergebnis nach CSV beziehungsweise Excel exportiert werden. Beim Kopieren vergibt DataBlick automatisch einen freien Namen wie `kunden_1`, `kunden_2` oder `abfrage_1`.

DataBlick legt eigene interne Tabellen an. Diese speichern zum Beispiel gespeicherte Abfragen, QBE-Entwürfe oder Import-Spezifikationen. Sie werden im Navigationsbereich ausgeblendet, damit Lernende sich auf die fachlich relevanten Tabellen konzentrieren können.

Interne Tabellen sollten nicht manuell gelöscht oder verändert werden. Sie gehören nicht zum fachlichen Datenmodell der Lernenden, sondern dienen DataBlick zur Verwaltung eigener Zusatzinformationen.

Die Datenbankdatei kann grundsätzlich unabhängig von DataBlick gespeichert, kopiert oder gesichert werden. Wer eine Datenbank an eine andere Person weitergibt, muss daher normalerweise die `.db`-Datei weitergeben. Screenshots oder Exportdateien ersetzen die eigentliche Datenbankdatei nicht.

4. Tabellen erstellen und bearbeiten

Eine neue Tabelle wird über **Bearbeiten → Neue Tabelle** oder den Toolbar-Button **Tabelle** erstellt. Dabei werden Feldnamen und Datentypen festgelegt. Außerdem kann bestimmt werden, ob ein Feld Primärschlüssel ist, ob es leer bleiben darf, ob Werte eindeutig sein müssen und ob ein Standardwert verwendet werden soll.

Für Zahlenfelder (`INTEGER`, `REAL`, `NUMERIC`) kann zusätzlich ein reines Anzeigeformat hinterlegt werden. Diese Einstellung verändert die gespeicherten Werte nicht, sondern nur die Darstellung in der Datenblattansicht und im QBE-Abfrageergebnis — auch bei Aggregatfunktionen wie Summe oder Mittelwert. **Anzeige runden auf** akzeptiert Werte von `-3` bis `6`: `-3` rundet auf Tausender, `0` auf ganze Zahlen, `1` bis `6` auf Nachkommastellen. **Anzeige-Symbol** kann bis zu drei Zeichen enthalten, zum Beispiel `€` oder `US$`.

Für Felder mit Fremdschlüssel steht zusätzlich das Feld **FK-Anzeigefelder** zur Verfügung. Hier werden kommagetrennt die Spalten der referenzierten Tabelle angegeben, die in der Datenblattansicht als lesbarer Bezeichner angezeigt werden sollen — zum Beispiel `Vorname, Name`. Ohne Eintrag wird automatisch die erste Textspalte der referenzierten Tabelle verwendet.

Vorhandene Tabellen können im Navigationsbereich per Rechtsklick umbenannt oder kopiert werden. Eine Kopie übernimmt die Tabellenstruktur und die vorhandenen Daten. DataBlick wählt automatisch einen freien Namen mit Zähl-Suffix, zum Beispiel `kunden_1`. Diese Funktion ist nützlich, wenn Lernende gefahrlos mit einer Variante einer Tabelle experimentieren sollen.

Beim Kopieren werden auch die Fremdschlüssel mitkopiert, die von dieser Tabelle ausgehen. Wird zum Beispiel `Bestellungen` kopiert, verweist `Bestellungen_1.KundenID` weiterhin auf `Kunden.KundenID`. Beziehungen anderer Tabellen, die auf die ursprüngliche Tabelle zeigen, werden dagegen nicht automatisch auf die Kopie umgebogen. Eine Tabellenkopie ist also eine Struktur- und Datenkopie dieser einen Tabelle, keine automatische Kopie des ganzen Beziehungsausschnitts.

Unterstützte Datentypen im Unterrichtskontext:

- `TEXT`: Text, Namen, Orte, E-Mail-Adressen
- `INTEGER`: Ganze Zahlen
- `REAL`: Dezimalzahlen
- `NUMERIC`: Zahlen mit flexibler Speicherung
- `BLOB`: Binäre Daten

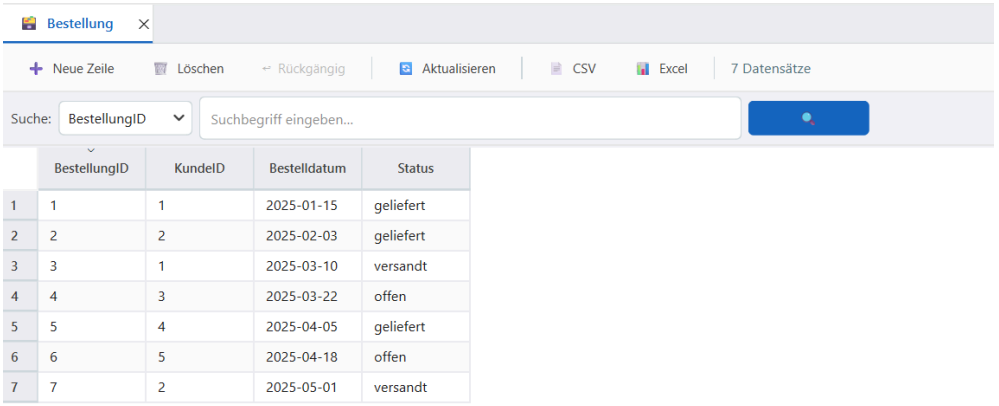
DataBlick nutzt SQLite. SQLite ist bei Datentypen flexibler als viele Serverdatenbanken. Datumswerte werden in DataBlick sinnvoll als ISO-Text gespeichert, zum Beispiel `2026-06-08`. Diese Schreibweise hat den Vorteil, dass Datumswerte auch als Text in der richtigen zeitlichen Reihenfolge sortiert werden können.

Beim Tabellenentwurf sollte man nicht nur überlegen, welche Daten gespeichert werden sollen, sondern auch, welche Daten eindeutig sein müssen und welche Beziehungen zu anderen Tabellen bestehen. Eine Tabelle `kunden` benötigt zum Beispiel einen Primärschlüssel. Eine Tabelle `bestellungen` benötigt neben einem eigenen Primärschlüssel häufig einen Fremdschlüssel, der auf den zugehörigen Kunden verweist.

Im Unterricht ist es sinnvoll, den Tabellenentwurf zunächst fachlich zu begründen und erst danach technisch umzusetzen. Die entscheidende Frage lautet nicht: „Welche Spalten kann ich schnell anlegen?“, sondern: „Welche Informationen gehören zusammen, und wie vermeide ich unnötige Dopplungen?“

5. Datenblattansicht

In der Datenblattansicht werden Daten wie in einer Tabelle angezeigt. Datensätze können hinzugefügt, geändert, gelöscht, gefiltert und exportiert werden.



	BestellungID	KundeID	Bestelldatum	Status
1	1	1	2025-01-15	geliefert
2	2	2	2025-02-03	geliefert
3	3	1	2025-03-10	versandt
4	4	3	2025-03-22	offen
5	5	4	2025-04-05	geliefert
6	6	5	2025-04-18	offen
7	7	2	2025-05-01	versandt

Datenblattansicht mit Datensätzen

Werte werden direkt in den Zellen bearbeitet. Eine Zelle kann per Doppelklick, durch direktes Tippen in eine markierte Zelle oder mit der Editier-Taste geöffnet werden. Sobald die Bearbeitung abgeschlossen ist, zum Beispiel beim Verlassen der Zelle oder beim Wechsel in eine andere Zeile, speichert DataBlick die Änderung sofort in der Datenbank. Es gibt in der Datenblattansicht keinen separaten Speichern-Button für einzelne Zellwerte.

Eine neue Zeile wird am Ende der Tabelle eingefügt. Je nach Ansicht ist dort eine leere Eingabezeile sichtbar oder sie wird über **Neue Zeile** beziehungsweise `Alt+N` erzeugt. In diese Zeile werden die Werte des neuen Datensatzes eingetragen. Nach dem Verlassen der Zeile wird der Datensatz gespeichert, sofern keine Datenbankregel verletzt wird.

Ein leer eingegebener Wert wird als `NULL` gespeichert. `NULL` bedeutet: kein Wert vorhanden. Das ist nicht dasselbe wie die Zahl 0 und auch nicht dasselbe wie ein leerer Text. Diese Unterscheidung ist in Datenbanken wichtig. Eine fehlende Telefonnummer bedeutet zum Beispiel nicht, dass die Telefonnummer `0` lautet.

Dieses automatische Speichern gilt für Datenwerte in der Datenblattansicht. Entwurfsansichten, SQL-Abfragen und QBE-Abfragen haben dagegen eigene Speichern-Schritte. Wenn beim Schließen eines Tabs oder beim Beenden von DataBlick dort noch ungespeicherte Änderungen vorhanden sind, fragt DataBlick nach, ob gespeichert, verworfen oder abgebrochen werden soll.

DataBlick verwendet in der Datenblattansicht ein unmittelbares Speichermodell. Änderungen werden also direkt an die Datenbank übergeben. Für die zuletzt geänderte Zelle steht ein einmaliger Rückgängig-Schritt zur Verfügung: `Strg+Z` oder der Button ↩ **Rückgängig** in der Toolbar macht die letzte Zelländerung wieder rückgängig, solange kein weiterer Datensatz bearbeitet oder die Ansicht neu geladen wurde. Dieses Ein-Schritt-Undo ist kein Ersatz für echte Transaktionen. Wenn mehrere Änderungen rückgängig gemacht werden sollen, empfiehlt es sich für Unterrichtsphasen mit experimentellem Arbeiten, mit Kopien der Datenbankdatei zu arbeiten.

Typische Aktionen:

- **Neue Zeile**: Fügt am Ende der Tabelle einen neuen Datensatz ein.
- **Löschen**: Entfernt ausgewählte Datensätze.
- **Aktualisieren**: Lädt die Tabelle neu.
- **CSV / Excel**: Exportiert die Tabellenzeilen.
- **Filter**: Begrenzt die sichtbaren Datensätze.

Tastenkombinationen in der Datenblattansicht:

- `Alt+N`: Neue Zeile
- `Entf`: Ausgewählte Zeile löschen
- `Strg+Z`: Letzte Zelländerung rückgängig machen (ein Schritt)

- `F5`: Aktualisieren
- `Alt+Links` / `Alt+Rechts`: Seite wechseln
- `Esc`: Neu angelegte, noch nicht gespeicherte Zeile verwerfen

Fremdschlüsselfelder sind in der Spaltenüberschrift mit einem 🔗 **-Symbol** gekennzeichnet. Ein Tooltip zeigt die Ziel-Tabelle und -Spalte sowie die konfigurierten Anzeigefelder. In der Zelle wird nicht der rohe Schlüsselwert angezeigt, sondern ein lesbarer Bezeichner — zum Beispiel `1 – Anna Müller` statt nur `1`. Welche Spalten der referenzierten Tabelle dabei kombiniert werden, lässt sich in der Entwurfsansicht unter **FK-Anzeigefelder** festlegen (z. B. `Vorname, Name`). Ohne Konfiguration wird automatisch die erste Textspalte der referenzierten Tabelle verwendet. Beim Bearbeiten öffnet sich ein Dropdown-Menü mit allen gültigen Werten — kein Rohwert muss von Hand eingetippt werden.

Neue Zeilen und Pflichtfelder: Eine neu angelegte Zeile wird erst beim Verlassen der Zeile (Wechsel zu einem anderen Datensatz) in der Datenbank gespeichert. Felder, die als Pflichtfeld (`NOT NULL`) definiert sind, müssen dabei ausgefüllt sein. Noch nicht ausgefüllte Pflichtfelder erscheinen rot hinterlegt, und eine Meldung in der Statuszeile nennt die betroffenen Felder beim Namen. Mit `Esc` kann eine neu angelegte, noch nicht gespeicherte Zeile verworfen werden. `Esc` wirkt dabei ausschließlich auf die gerade neu eingetragene Zeile — Änderungen an bereits gespeicherten Datensätzen können damit nicht rückgängig gemacht werden.

Beim Löschen von Datensätzen ist besondere Vorsicht nötig, wenn andere Tabellen über Fremdschlüssel auf diese Datensätze verweisen. Je nach eingestellter Regel kann das Löschen verhindert werden oder abhängige Datensätze können automatisch mitgelöscht werden. Deshalb ist es sinnvoll, Löschoperationen immer zuerst fachlich zu überlegen: Welche weiteren Daten hängen an diesem Datensatz?

6. Entwurfsansicht

Die Entwurfsansicht bearbeitet die Struktur einer Tabelle. Hier werden Feldnamen, Datentypen, Primärschlüssel, Pflichtfelder, eindeutige Werte und Standardwerte verwaltet.

In der Feldtabelle zeigt die Spalte **PK** an, ob ein Feld zum Primärschlüssel gehört. Ein Schlüssel-Symbol in dieser Spalte kennzeichnet jedes Primärschlüsselfeld. Werden mehrere Felder als **PK** markiert, erzeugt DataBlick einen zusammengesetzten Primärschlüssel, zum Beispiel `PRIMARY KEY (kurs_id, schueler_id)`. Ein automatisch fortlaufender `INTEGER PRIMARY KEY` ist dagegen nur beim einzelnen Integer-Primärschlüsselfeld möglich.

Die Entwurfsansicht verändert nicht einzelne Datenwerte, sondern den Aufbau der Tabelle. Das ist ein wesentlicher Unterschied zur Datenblattansicht. Wenn in der Datenblattansicht ein Name geändert wird, betrifft das einen Datensatz. Wenn in der Entwurfsansicht ein Feld entfernt oder umbenannt wird, betrifft das die Struktur der gesamten Tabelle. Beim Umbenennen eines Feldes übernimmt DataBlick die vorhandenen Werte in die umbenannte Spalte. Beim Löschen eines Feldes gehen die Werte dieser Spalte verloren.

Wichtige Hinweise:

- Eine Tabelle muss mindestens ein Feld besitzen.
- Ein `INTEGER PRIMARY KEY` wird von SQLite automatisch fortgezählt.
- Pflichtfelder dürfen nicht leer bleiben.
- `UNIQUE` verhindert doppelte Werte in einem Feld.
- Ein Standardwert wird verwendet, wenn beim Einfügen kein Wert angegeben wird.

- Beim Entfernen bestehender Felder gehen die Daten dieser Spalten verloren.
- Beim Umbenennen bestehender Felder bleiben die Werte erhalten.
- DataBlick zeigt deshalb vor riskanten Schemaänderungen eine Warnung an.

Tipp vor riskanten Strukturänderungen: Wer auf Nummer sicher gehen möchte, speichert die Datenbank vorher unter einem neuen Namen (**Datei → Speichern unter…**, `Strg+Shift+S`). Das erzeugt eine vollständige Kopie der aktuellen `.db`-Datei — ein einfaches, zuverlässiges Backup ohne zusätzliche Werkzeuge. Im Unterricht ist das gleichzeitig ein sinnvoller Anlass, das Konzept der Datensicherung praktisch zu besprechen.

DataBlick legt bei QBE-Aktionsabfragen (`UPDATE` und `DELETE` aus dem grafischen Abfrageentwurf) automatisch Backups an. Im freien SQL-Editor entscheidet die Nutzerin oder der Nutzer bewusst selbst: Vor schreibenden Einzelanweisungen erscheint ein Bestätigungsdialog, bei einfachen `UPDATE`- und `DELETE`-Anweisungen zusätzlich mit Vorschau der betroffenen Datensätze. Dort kann optional **Vorher Backup anlegen** aktiviert werden. Sicherungsdateien liegen im gleichen Ordner wie die Datenbank und tragen einen Zeitstempel im Namen (z. B. `schulbibliothek.db.20260612_130902.bak`). Über **Datei → Backup wiederherstellen…** lässt sich eine dieser Sicherungskopien auswählen und als aktuelle Datenbank wiederherstellen. **Achtung:** Die Wiederherstellung ersetzt die aktuelle Datenbankdatei vollständig durch die ausgewählte Sicherung. Dieser Vorgang ist nicht umkehrbar, solange keine eigene Kopie der aktuellen Datei vorhanden ist. DataBlick fragt vor der Wiederherstellung zur Bestätigung.

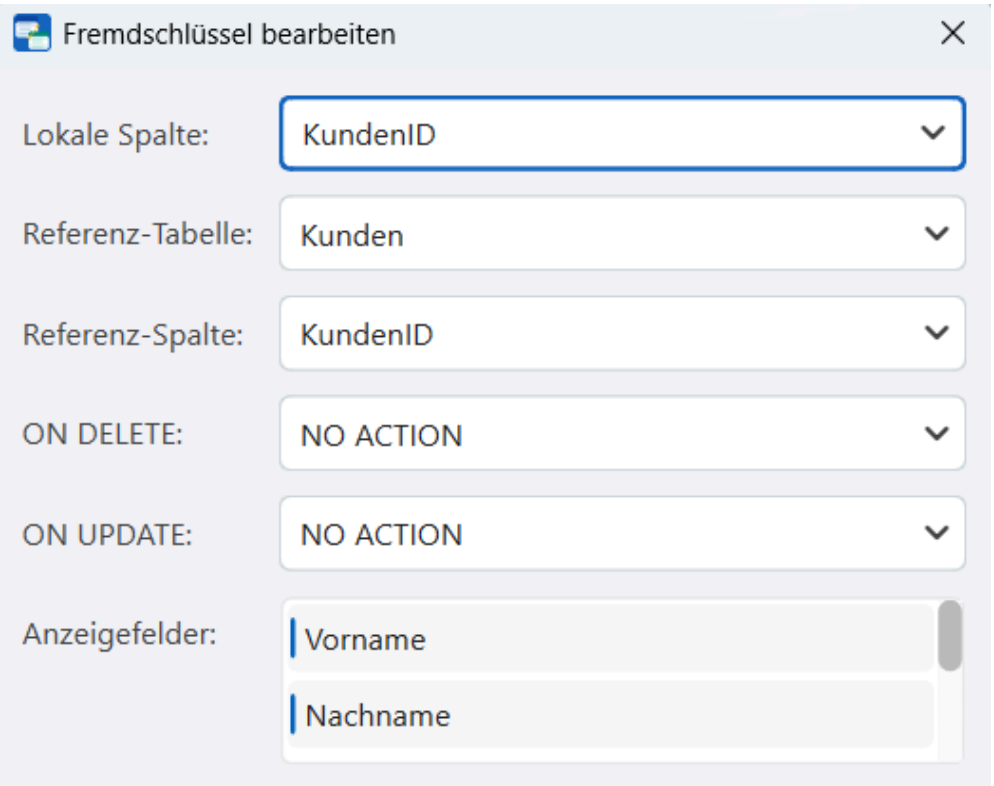
Für Primärschlüsselfelder zeigt DataBlick die Checkboxen **NOT NULL – Pflichtfeld** und **UNIQUE – Eindeutig** automatisch als aktiviert und ausgegraut. Ein Primärschlüssel ist von SQLite immer eindeutig und darf nicht leer sein. Diese Einschränkungen können daher nicht einzeln abgewählt werden — sie gelten immer.

Indizes zur Performance-Optimierung stehen nicht im Zentrum des schulischen Einstiegs. DataBlick verwaltet sie in der Entwurfsansicht daher nicht als eigenes Unterrichtselement. Fortgeschrittene Nutzerinnen und Nutzer können Indizes aber über den SQL-Editor mit `CREATE INDEX` anlegen. Für den Grundkurs reicht es normalerweise, Primärschlüssel, Fremdschlüssel und eindeutige Werte zu verstehen.

Für den Unterricht ist die Entwurfsansicht besonders wichtig, weil hier sichtbar wird, dass Tabellen nicht nur aus Daten bestehen. Tabellen haben Regeln. Diese Regeln legen fest, welche Werte erlaubt sind, welche Werte fehlen dürfen und wie Datensätze eindeutig unterschieden werden.

7. Fremdschlüssel

Im unteren Bereich der Entwurfsansicht gibt es den Tab **Fremdschlüssel**. Dort können Beziehungen zwischen Tabellen angelegt, bearbeitet und gelöscht werden.



Fremdschlüssel-Tab in der Entwurfsansicht

Fremdschlüssel sichern Beziehungen zwischen Tabellen. Eine Bestellung kann zum Beispiel nur auf einen Kunden verweisen, der wirklich existiert. Die Fremdschlüssel-Prüfung ist in DataBlick automatisch aktiv und muss von Lernenden normalerweise nicht selbst eingestellt werden.

Ein Fremdschlüssel besteht aus:

- Lokaler Spalte, zum Beispiel `kunde_id`
- Referenz-Tabelle, zum Beispiel `kunden`
- Referenz-Spalte, meist `id`
- Regel für `ON DELETE`
- Regel für `ON UPDATE`

Wichtige Regeln:

- **NO ACTION / RESTRICT**: Löschen oder Ändern wird verhindert, wenn abhängige Daten existieren.
- **CASCADE**: Abhängige Datensätze werden mitgelöscht oder mitgeändert.
- **SET NULL**: Der Fremdschlüssel wird auf `NULL` gesetzt.
- **1:n** entsteht durch eine normale Fremdschlüsselspalte.
- **1:1** entsteht, wenn die Fremdschlüsselspalte zusätzlich `UNIQUE` ist.

DataBlick prüft beim Speichern, ob bestehende Daten die neuen Fremdschlüssel verletzen. Wenn zum Beispiel `kunde_id = 999` eingetragen ist, aber kein Kunde mit `id = 999` existiert, wird die Schemaänderung abgelehnt.

Didaktisch ist dieser Punkt zentral: Fremdschlüssel sind nicht nur Linien in einem Diagramm. Sie sind Regeln, die die Datenbank aktiv durchsetzt. Dadurch wird verhindert, dass in einer Tabelle Verweise auf Datensätze stehen, die es gar nicht gibt.

8. Import

Der Import-Assistent importiert CSV- und Excel-Dateien. Er führt durch vier Schritte:

1. Datei auswählen und Vorschau prüfen
2. Import-Optionen festlegen
3. Spalten und Datentypen festlegen
4. Ziel festlegen und importieren

Importe können in eine neue Tabelle gehen oder an eine bestehende Tabelle angehängt werden.

Nützliche Funktionen:

- Trennzeichen und Kodierung wählen
- Kopfzeile ein- oder ausschalten
- Dezimaltrennzeichen festlegen
- Datumsformat angeben
- Import-Spezifikation in Schritt 1 laden
- Import-Spezifikation im letzten Schritt speichern

Bei Excel-Dateien sind Trennzeichen und Zeichenkodierung nicht relevant. Schritt 2 zeigt dann nur die passende Excel-Option zur Kopfzeile und eine kurze Vorschau. Bei CSV-/TXT-Dateien stehen dort zusätzlich Trennzeichen und Kodierung zur Verfügung.

Eine Excel-Arbeitsmappe kann mehrere Arbeitsblätter enthalten. DataBlick importiert immer genau ein Blatt. Sind mehrere Blätter vorhanden, erscheint beim Öffnen der Datei ein Auswahldialog mit Radiobuttons; voreingestellt ist das in Excel zuletzt aktive Blatt. Über die Schaltfläche **Blatt wechseln …** in Schritt 2 lässt sich die Auswahl jederzeit ändern. Das gewählte Blatt wird in der Importspezifikation mitgespeichert.

DataBlick importiert aus Excel-Dateien die gespeicherten **Werte**, nicht die **Formeln**. Eine Zelle mit `=SUMME(A1:A10)` wird also mit ihrem zuletzt von Excel berechneten Ergebnis übernommen, nicht mit der Formel selbst. Wurde eine Formelzelle noch nie berechnet und gespeichert, liegt kein Wert vor und die Zelle wird leer importiert; der Assistent weist in der Vorschau darauf hin. Abhilfe schafft das einmalige Öffnen und Speichern der Datei in Excel.

Eine Importspezifikation ist ein gespeichertes Rezept für wiederkehrende Importe. Sie enthält zum Beispiel Dateioptionen, Kopfzeile, Datumsformat, erkannte oder korrigierte Spaltentypen, Zieltabellenmodus und Primärschlüsselwahl. Beim nächsten Import kann sie im ersten Schritt geladen werden. Speichern ist erst im letzten Schritt sinnvoll, wenn alle Einstellungen geprüft wurden.

Wenn die erste Zeile keine Feldnamen enthält, erzeugt DataBlick automatisch Feldnamen wie `Spalte_1`, `Spalte_2` und so weiter. Fehlt nur eine einzelne Überschrift, wird nur diese Spalte entsprechend ersetzt. Sonderzeichen, Leerzeichen und Zeilenumbrüche in Überschriften werden für den Datenbankgebrauch bereinigt, zum Beispiel wird aus `Transaction ID` der Feldname `Transaction_ID`. Doppelte Feldnamen erhalten eine laufende Ergänzung wie `_2`.

Wenn einzelne Zeilen fehlschlagen, kann DataBlick eine Fehlertabelle anlegen, in der die problematischen Zeilen gesammelt werden.

Nach einem abgeschlossenen Import zeigt DataBlick einen Ergebnis-Dialog. Ein **Details**-Button öffnet das vollständige **Import-Protokoll** mit:

- Quelldatei und Zieltabelle
- Modus (Neue Tabelle oder An bestehende Tabelle anhängen)
- Anzahl der gelesenen, importierten und fehlerhaften Zeilen
- Hinweis auf die Fehlertabelle oder den Vermerk “Keine Fehlertabelle angelegt”
- **Import-Einstellungen**: Trennzeichen, Kodierung, Dezimaltrennzeichen, Datumsformat, Primärschlüsselmodus

Gerade bei CSV-Problemen ist der Einstellungsabschnitt hilfreich: Trennzeichen, Kodierung und Dezimaltrennzeichen sind häufige Fehlerquellen. Das Protokoll dokumentiert alle gewählten Einstellungen, damit Diagnose und Fehlersuche einfacher werden.

Beim Import ist zu beachten, dass externe Daten selten perfekt zur Datenbankstruktur passen. Spaltennamen können ungünstig sein, Zahlen können als Text vorliegen, Datumsformate können uneinheitlich sein, und einzelne Zeilen können fehlerhafte Werte enthalten. Der Import-Assistent hilft dabei, solche Probleme früh zu erkennen.

In Schritt 3 warnt DataBlick, wenn eine gewählte Typzuordnung Werte verlieren würde. Wird eine Spalte zum Beispiel als `REAL` markiert, obwohl einzelne Zellen Text enthalten, würden diese Werte zu `NULL`. Wird eine Spalte als `DATE` markiert, obwohl einzelne Werte nicht zum gewählten Datumsformat passen, würden auch diese Werte zu `NULL`. Bei `INTEGER` weist DataBlick außerdem auf Dezimalwerte oder nicht lesbare ganze Zahlen hin. In solchen Fällen sollte der Datentyp, das Dezimaltrennzeichen oder das Datumsformat geprüft werden, bevor der Import fortgesetzt wird.

Für den Unterricht ist der Import besonders nützlich, wenn reale oder halb reale Datensätze verwendet werden sollen. Lernende sehen dabei, dass Datenbanken nicht nur künstlich erzeugte Beispieldaten speichern, sondern auch aus Tabellenkalkulationen oder CSV-Dateien gefüllt werden können. Gleichzeitig wird deutlich, dass gute Datenqualität Voraussetzung für sinnvolle Abfragen ist.

Tabelle erneut importieren (überschreiben): Über den Navigationsbereich lässt sich ein bereits abgeschlossener Import wiederholen. Ein Rechtsklick auf eine Tabelle öffnet das Kontextmenü; dort steht **Neu importieren (überschreiben)**. Diese Aktion öffnet den Import-Assistenten mit der zuletzt für diese Tabelle gespeicherten Importspezifikation vor. Am Ende des Assistenten wird die Zieltabelle vollständig durch die neuen Daten ersetzt. Das ist praktisch, wenn eine Quelldatei regelmäßig aktualisiert wird und die Tabelle in der Datenbank immer den aktuellen Stand widerspiegeln soll.

9. SQL-Editor

Der SQL-Editor dient zum Schreiben und Ausführen eigener SQL-Abfragen.

Öffnen:

- **Bearbeiten → Neue Abfrage**
- `Strg+Q`
- Toolbar-Button **SQL-Abfrage**

Funktionen:

- SQL schreiben und ausführen (Strg+Enter) oder als Skript (Strg+Shift+Enter)
- Ergebnisse als Tabelle anzeigen, nach Spalten sortierbar (Klick auf Spaltenkopf)
- Ergebnis als CSV oder Excel exportieren
- aus geladenen SQL-Ergebnissen einfache Balkendiagramme erzeugen und als PNG exportieren
- Abfrage speichern (Strg+S)
- Ergebnis einer Abfrage als neue Tabelle speichern (**Als Tabelle**)
- Aktuelles SELECT-Statement als View in der Datenbank speichern (**Als View**)
- Gespeicherte Abfrage über den Navigationsbereich als SQL-Code exportieren

Gespeicherte Abfragen erscheinen links im Navigationsbereich. Ein kleines Symbol vor dem Namen zeigt den Typ: ⊞ steht für eine QBE-Abfrage (mit grafischem Entwurf), ✎ steht für eine reine SQL-Abfrage. Per Rechtsklick stehen folgende Aktionen zur

Verfügung: Abfrage öffnen, Umbenennen, Kopieren, Exportieren (CSV, Excel, SQL) und Löschen. Für QBE-Abfragen gibt es zusätzlich **Als SQL-Abfrage kopieren**: Dabei wird der generierte SQL-Text als neue, reine SQL-Abfrage gespeichert — ohne den grafischen Entwurf. Der Namensvorschlag lautet `sql_[Abfragename]` und kann im Eingabedialog angepasst werden.

Mit **Als View** wird das aktuelle SELECT-Statement als SQLite-View in der Datenbank gespeichert. Der eingegebene View-Name erscheint dann automatisch im Navigationsbereich unter **Ansichten**, kann im QBE-Designer als Quelle verwendet werden und ist auch in anderen SQLite-Programmen sichtbar. **Als Tabelle** speichert dagegen das Abfrageergebnis als echte Tabelle — eine Momentaufnahme der aktuellen Daten.

Bei sehr großen freien `SELECT`-Abfragen lädt der SQL-Editor das Ergebnis blockweise. Zunächst werden bis zu 5000 Zeilen angezeigt. Wenn weitere Zeilen vorhanden sind, erscheint im Ergebnisbereich der Button **Weitere 5000 laden**. Jeder Klick lädt den nächsten Block derselben laufenden Abfrage nach. Dadurch bleibt die Oberfläche auch bei großen Ergebnismengen bedienbar. Die Datenblattansicht arbeitet davon unabhängig weiterhin mit ihrer eigenen Seitenanzeige.

Wichtig für Export und Weiterverarbeitung: **CSV**, **Excel**, **Diagramm** und **Als Tabelle** verwenden im SQL-Editor die aktuell geladenen Ergebniszeilen. Wer ein sehr großes Ergebnis vollständig exportieren, visualisieren oder als Tabelle speichern möchte, muss die weiteren Blöcke vorher nachladen oder die Abfrage fachlich mit `WHERE`, `LIMIT` oder einer geeigneten Sortierung eingrenzen. Eine mit **Als View** gespeicherte Abfrage ist dagegen keine Kopie der angezeigten Zeilen, sondern eine gespeicherte SELECT-Definition. Wird diese View später als Quelle für weitere Abfragen verwendet, wertet SQLite die vollständige View-Definition neu aus.

Mit **Diagramm** kann aus dem aktuell geladenen SQL-Ergebnis ein einfaches Balkendiagramm erzeugt werden. Ein Dialog fragt Beschriftungsspalte, numerische Wertespalte, Titel, Achsenbeschriftungen und Sortierung ab. Gibt es im geladenen Ergebnis keine numerische Spalte, erscheint direkt ein Hinweis. Optional können die Zahlenwerte direkt an den Balken angezeigt werden. Die SQL-Abfrage muss die Daten bereits passend vorbereiten, zum Beispiel mit `GROUP BY` und `SUM`, `AVG` oder `COUNT`; DataBlick aggregiert Rohdaten im Diagramm nicht automatisch. Das Diagramm erscheint in einem eigenen Tab, kann über das × im Tab geschlossen und als PNG exportiert werden. Negative Werte werden mit einer Nulllinie dargestellt. Wenn weitere Ergebniszeilen vorhanden sind, weist DataBlick darauf hin, dass nur die geladenen Zeilen visualisiert werden.

Der Tab **Python** ist standardmäßig ausgeblendet, damit der SQL-Editor für Anfänger übersichtlich bleibt. Mit `Strg+P` wird er eingeblendet; dieselbe Tastenkombination blendet ihn wieder aus. Der Tab erzeugt passend zum SQL-Text ein kleines Python-Beispiel für Thonny. Der Code verwendet bewusst zuerst `sqlite3` und den Pfad der aktuell in DataBlick geöffneten Datenbank. Das passt zum Unterrichtseinstieg, weil die Verbindung zur Datenbank, die SQL-Abfrage, die Parameterübergabe und die Ergebniszeilen direkt sichtbar bleiben.

Benannte SQL-Parameter erscheinen als `PARAMS`-Dictionary mit `TODO`-Werten. Dadurch lässt sich auch der Schutz vor SQL-Injection gut zeigen: Werte werden getrennt vom SQL-Text übergeben und nicht per Zeichenkettenverkettung oder f-String in die Abfrage eingebaut. Der Python-Code wird nicht in DataBlick ausgeführt, sondern dient zum Kopieren, Nachvollziehen und Weiterarbeiten im Python-Unterricht.

Zusätzlich enthält der Tab eine auskommentierte `pandas`-Variante. Sie ist als Ausblick für spätere Datenanalyse mit DataFrames gedacht, aber für das Grundmaterial nicht erforderlich. Für die ersten SQL- und Parameterbeispiele reicht `sqlite3` didaktisch aus.

Eine SQL-Abfrage beschreibt in Textform, welche Daten aus der Datenbank geholt werden sollen. Eine einfache Auswahlabfrage kann zum Beispiel alle Datensätze aus einer Tabelle anzeigen oder nur bestimmte Spalten auswählen. Durch Bedingungen werden die Datensätze eingeschränkt, durch Sortierungen in eine gewünschte Reihenfolge gebracht und durch Joins mit Daten aus anderen Tabellen verbunden.

Auswahlabfragen verändern die gespeicherten Daten nicht. Daneben gibt es aber auch SQL-Anweisungen, die Daten oder Strukturen verändern können, etwa `INSERT`, `UPDATE`, `DELETE` oder `CREATE TABLE`. Deshalb sollte im Unterricht klar zwischen einer reinen Auswahlabfrage und einer verändernden Anweisung unterschieden werden.

Für fehlende Werte ist in SQL die Schreibweise `IS NULL` wichtig. Ein Vergleich mit `= NULL` funktioniert nicht wie erwartet, weil `NULL` kein normaler Wert ist, sondern für „kein Wert vorhanden“ steht.

```sql
SELECT *
FROM kunden
WHERE email IS NULL;
```

Das Gegenstück lautet `IS NOT NULL`.

```sql
SELECT *
FROM kunden
WHERE email IS NOT NULL;
```

Parametrierte Abfragen ermöglichen es, beim Ausführen Werte einzugeben, ohne den SQL-Text jedes Mal zu ändern. Die Schreibweise folgt dem SQLite-Standard: `:Parametername` im SQL-Text. DataBlick erkennt diese Platzhalter automatisch und öffnet vor der Ausführung für jeden Parameter einen Eingabedialog.

```sql
SELECT *
FROM bestellungen
WHERE stadt = :Stadt AND gesamtbetrag > :Mindestumsatz;
```

Beim Ausführen erscheinen zwei Dialoge nacheinander — einer für `:Stadt`, einer für `:Mindestumsatz`. Die eingegebenen Werte werden sicher übergeben; SQL-Injection ist nicht möglich. Das Abbrechen eines Dialogs bricht die Ausführung ab.

Der SQL-Editor führt lesende Einzelanweisungen normalerweise direkt aus. Schreibende Anweisungen wie `INSERT`, `UPDATE`, `DELETE`, `CREATE`, `ALTER`, `DROP` und `REPLACE` müssen vorher bestätigt werden. Bei einfachen `UPDATE`- und `DELETE`-Anweisungen zeigt DataBlick eine Vorschau der betroffenen Datensätze; fehlt eine `WHERE`-Bedingung, wird deutlich gewarnt. Im Dialog kann optional ein Backup angelegt werden. Für fortgeschrittene Experimente können Transaktionen verwendet werden, wenn SQLite dies in der jeweiligen Situation zulässt. Mit `BEGIN`, `COMMIT` und `ROLLBACK` lassen sich mehrere Änderungen als zusammengehöriger Block behandeln. Die Datenblattansicht arbeitet dagegen mit unmittelbarem Speichern einzelner Änderungen.

```sql
BEGIN;
UPDATE kunden
SET ort = 'Karlsruhe'
WHERE id = 1;
ROLLBACK;
```

Dieses Beispiel macht die Änderung wieder rückgängig, solange sie innerhalb der Transaktion noch nicht mit `COMMIT` bestätigt wurde. Für den normalen Einstieg ist dieses Thema nicht nötig, für fortgeschrittene Lernende aber hilfreich, um zu verstehen, dass Datenbanken Änderungen kontrolliert bündeln können.

Skript-Modus

Enthält der SQL-Editor **zwei oder mehr Anweisungen**, wechselt die Toolbar automatisch: Der **Ausführen**-Button verschwindet und an seiner Stelle erscheint — nach bestandener Syntaxprüfung — der Button **Skript ausführen** (`Strg+Shift+Enter`).

Im Skript-Modus führt DataBlick alle Anweisungen der Reihe nach aus:

- Skripte ohne eigene Transaktionsbefehle laufen automatisch in einer atomaren Transaktion: Bei einem Fehler wird alles zurückgerollt.
- Explizite `BEGIN`/`COMMIT`/`ROLLBACK`-Blöcke im Skript werden respektiert.
- Parameter (`:Name`) sind im Skript-Modus nicht erlaubt.

Nach der Ausführung öffnet sich das **Skriptprotokoll** (farbkodierte Übersicht aller Anweisungen mit Status). Jede inhaltliche Anweisung erhält zusätzlich einen eigenen nummerierten Tab **[1]**, **[2]** … links neben dem Protokoll-Tab. Das letzte SELECT-Ergebnis bleibt im Ergebnis-Tab verfügbar.

Markierten Text ausführen: Ist im Editor Text markiert, führt `Strg+Enter` nur den markierten Bereich aus — nützlich, um aus einem längeren Skript einzelne Teile isoliert zu testen.

IntelliSense

Der SQL-Editor bietet Autovervollständigung für Tabellen- und Feldnamen. Nach dem dritten eingetippten Zeichen zeigt DataBlick automatisch passende Vorschläge aus der aktuell geöffneten Datenbank an. Die Vorschläge sind kontextsensitiv: Nach `FROM` und `JOIN` erscheinen Tabellennamen, nach einem Punkt hinter einem Tabellennamen erscheinen die Felder dieser Tabelle, und in `SELECT`- und `WHERE`-Position erscheinen alle verfügbaren Tabellen- und Feldnamen. Mit der Pfeiltaste wird ein Vorschlag markiert, mit `Tab` oder `Return` übernommen. `Escape` schließt die Liste.

IntelliSense hilft besonders beim Tippen langer oder ähnlicher Feldnamen und vermeidet Schreibfehler, die SQLite erst beim Ausführen meldet.

Automatische Syntaxprüfung

Während des Tippens prüft DataBlick den SQL-Text fortlaufend auf offensichtliche Syntaxfehler. Sobald ein Fehler erkannt wird, erscheint ein roter Hinweis unterhalb des Eingabefelds. Solange der Text syntaktisch in Ordnung ist, bleibt dieser Bereich leer. Die Prüfung erfolgt clientseitig durch SQLite selbst und erkennt die meisten Tippfehler, vergessene Anführungszeichen oder fehlende Klammern, bevor die Abfrage ausgeführt wird.

10. QBE-Abfrageentwurf

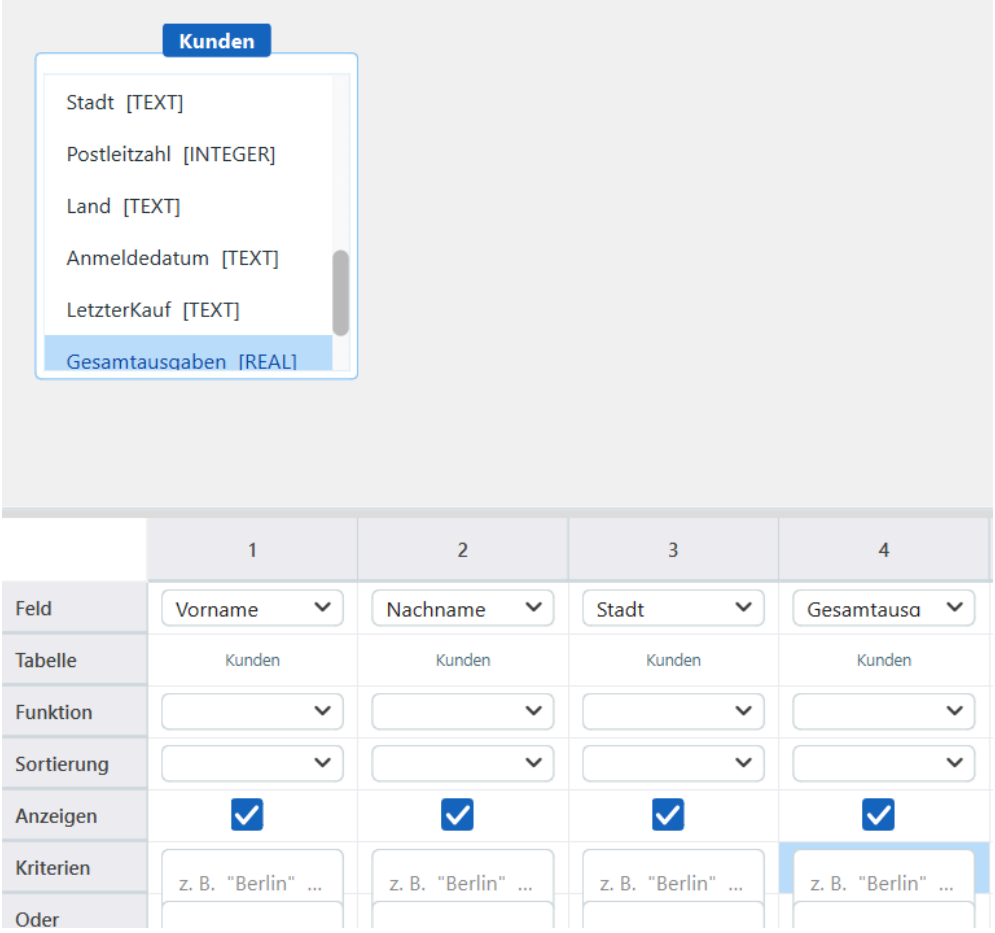
QBE steht für **Query by Example**. Damit können Abfragen grafisch erstellt werden, ohne SQL direkt schreiben zu müssen. QBE ist also kein anderes Datenbanksystem, sondern eine grafische Art, eine Abfrage zu formulieren.

Öffnen:

- **Bearbeiten → Abfrageentwurf**
- `Strg+Shift+Q`
- Toolbar-Button **QBE**

Der QBE-Designer besteht aus vier Ansichten:

- **Entwurf**: Tabellenboxen und Raster für Felder, Funktionen, Sortierung und Kriterien
- **SQL**: Automatisch erzeugter SQL-Text
- **Ergebnis**: Ergebnis der ausgeführten Abfrage
- **Verbal**: Beschreibung der Abfrage in deutschen Sätzen



QBE-Abfrageentwurf mit Tabellenboxen und Raster

Typischer Ablauf:

1. Tabellen hinzufügen.
2. Felder per **Doppelklick** oder Drag & Drop ins Raster übernehmen.
3. Optional in der Zeile **Funktion** Aggregatfunktionen wählen.
4. Kriterien eintragen, zum Beispiel `Berlin`, `>=18`, `M%` oder `IS NULL`.
5. Abfrage ausführen.
6. Optional als DataBlick-Abfrage oder als neue Tabelle speichern.

Tabellen können über den Button **Tabelle** hinzugefügt oder direkt aus dem Navigationsbereich in den QBE-Entwurf gezogen werden. Drop auf den oberen Entwurfsbereich oder auf das Raster fügt die Tabelle hinzu. Der **Tabelle hinzufügen**-Dialog zeigt drei Tabs: **Tabellen**, **Ansichten** und **Abfragen** — analog zu Microsoft Access. So können nicht nur echte Tabellen, sondern auch Views und gespeicherte SQL-Abfragen als Quelle verwendet werden. Views erscheinen auf dem Canvas mit türkisem Rahmen und einem 👁-Symbol im Titel. Danach lassen sich die Felder aus der Tabellenbox per Doppelklick oder Drag & Drop ins Raster übernehmen.

Ein Doppelklick auf einen Feldnamen in der Tabellenbox übernimmt das Feld in das Raster **und** setzt den Eingabecursor sofort in die Kriterien-Zeile der neuen Spalte. So kann direkt nach dem Doppelklick eine Bedingung eingetippt werden — ohne die Maus erneut zu bewegen.

In der **SQL-Ansicht** zeigt der QBE-Designer das automatisch erzeugte SQL. Diese Ansicht ist schreibgeschützt — ein oranger linker Rand weist darauf hin. Das SQL kann nicht direkt bearbeitet werden, weil Änderungen nicht in den grafischen Entwurf zurückgeführt werden könnten. Stattdessen gibt es per **Rechtsklick** ein Kontextmenü mit folgenden Optionen:

- **Gesamtes SQL kopieren** — überträgt den vollständigen SQL-Text in die Zwischenablage.
- **Auswahl kopieren** — kopiert den markierten Bereich.
- **Als neue Abfrage speichern …** — speichert das angezeigte SQL als eigenständige SQL-Abfrage (ohne QBE-Entwurf).

Im freien SQL-Editor und im SQL-Reiter des QBE-Designers zeigt ein **Tooltip beim Überfahren** von SQL-Schlüsselwörtern (z.B. `SELECT`, `JOIN`, `GROUP BY`, `HAVING`) eine kurze Erklärung an. Das hilft beim Nachvollziehen von selbst geschriebenem und automatisch erzeugtem SQL-Text.

Die Hover-Hilfe deckt alle 147 offiziellen SQLite-Schlüsselwörter und alle von der jeweils verwendeten SQLite-Laufzeit gemeldeten Funktionen ab. Dazu gehören auch Fensterfunktionen wie `ROW_NUMBER() OVER (PARTITION BY ...)`, JSON/JSONB, Datum und Zeit, Mathematik, Aggregation sowie Diagnosefunktionen.

DataBlick speichert QBE-Entwürfe mit ihrem grafischen Zustand. Wird eine QBE-Abfrage später wieder geöffnet, erscheint sie erneut im QBE-Designer.

QBE-Kriterien in DataBlick sind bewusst schülernah formuliert. Sie sind nicht in jedem Fall identisch mit reinem SQL und auch nicht vollständig identisch mit Access. Deshalb zeigt DataBlick den erzeugten SQL-Text an. Lernende können so erkennen, wie aus einer grafischen Eingabe eine SQL-Bedingung entsteht.

Eingabe im QBE-Raster	Bedeutung	Entsprechende SQL-Idee
`Berlin`	Exakt Berlin	`= 'Berlin'`
`>=18`	Mindestens 18	`>= 18`
`M%`	Beginnt mit M	`LIKE 'M%'`
`%burg`	Endet mit burg	`LIKE '%burg'`
`%heim%`	Enthält heim	`LIKE '%heim%'`
`IN ("A", "B")`	A oder B	`IN ('A', 'B')`
`NOT IN ("A", "B")`	Weder A noch B	`NOT IN ('A', 'B')`
`IS NULL`	Kein Wert vorhanden	`IS NULL`
`IS NOT NULL`	Ein Wert ist vorhanden	`IS NOT NULL`
`i:berlin`	Berlin, BERLIN oder berlin	Groß- und Kleinschreibung wird ignoriert
`i:M%`	Beginnt mit M, Groß- und Kleinschreibung egal	`LIKE 'M%'` mit geeigneter Normalisierung

DataBlick unterstützt sowohl `%` als SQLite-Standard-Wildcard als auch `*` als Access-ähnliche Schreibweise. `%` ist in den Beispielen der Standard, weil die erzeugte SQL-Ansicht ebenfalls `%` verwendet. SQL-Schlüsselwörter wie `IN` und `NOT IN` sind in SQLite nicht von Groß-/Kleinschreibung abhängig; DataBlick zeigt sie im Handbuch trotzdem in Großbuchstaben, weil das der üblichen SQL-Schreibweise entspricht.

Mehrere Kriterien in derselben Rasterzeile werden mit `AND` verbunden. Die Zeile **Oder** bildet eine alternative Kriteriengruppe. Über den Button **+ Oder** unter der letzten Oder-Zeile können weitere Oder-Zeilen eingeblendet werden. Jede zusätzliche Oder-Zeile wird

als eigene OR-Gruppe in SQL übersetzt.

Die Zeile **Funktion** ermöglicht Access-ähnliche Aggregatabfragen. Verfügbare Funktionen sind **Gruppierung**, **Summe**, **Anzahl**, **Mittelwert**, **Min**, **Max** und **Bedingung**. **Gruppierung** erzeugt `GROUP BY`, die Rechenfunktionen erzeugen SQL-Aggregate wie `SUM`, `COUNT`, `AVG`, `MIN` und `MAX`. Kriterien auf Aggregatfunktionen werden als `HAVING` übersetzt. **Bedingung** filtert Datensätze vor der Gruppierung und wird nicht im Ergebnis angezeigt.

Abfrage speichern speichert die Frage beziehungsweise den grafischen Entwurf in DataBlick. Beim späteren Öffnen wird sie erneut ausgeführt und beantwortet dann die Frage mit den aktuellen Daten. **Als Tabelle speichern** erzeugt dagegen eine neue echte Tabelle mit den aktuellen Ergebnisdaten. Das ist eine Momentaufnahme und oft praktisch, wenn Lernende ein Abfrageergebnis weiterbearbeiten oder exportieren sollen.

Der Unterschied ist fachlich wichtig: Eine gespeicherte Abfrage ist wie eine gespeicherte Frage. Sie wird immer wieder neu beantwortet. Eine gespeicherte Tabelle ist dagegen eine Kopie des Ergebnisses zu einem bestimmten Zeitpunkt. Wenn sich die Ausgangsdaten später ändern, ändert sich diese Kopie nicht automatisch.

Verbal-Ansicht

Die **Verbal**-Ansicht ist der vierte Tab im QBE-Designer. Der **?**-Button oben rechts in der Toolbar öffnet eine kurze Hilfe zu QBE-Kriterien — mit Beispielen zur `i:`-Schreibweise, Wildcards und `IS NULL`. Die Verbal-Ansicht wird über den **Verbal**-Reiter aufgerufen. Sie zeigt die aktuelle Abfrage als deutschen Satz, zum Beispiel:

> Zeige Vorname, Nachname und Stadt aus der Tabelle Kunde, verbunden über die KundeID (Kunde.KundeID = Bestellung.KundeID) mit der Tabelle Bestellung, aber nur für Datensätze, bei denen Anmeldedatum größer ist als 2023-01-01.

Die Verbal-Ansicht liest den Entwurfszustand direkt aus dem Raster und erzeugt keinen SQL-Text. Sie ist deshalb didaktisch unabhängig von der SQL-Ansicht und kann genutzt werden, bevor Lernende SQL kennen. Die Beschreibung wächst mit der Abfrage: Tabellen, Felder, Bedingungen, Aggregatfunktionen und Sortierungen werden alle in Prosa übersetzt.

Für UPDATE- und DELETE-Entwürfe nennt die Verbal-Ansicht die betroffene Tabelle und die Bedingung, die eingrenzt, welche Datensätze geändert oder gelöscht werden.

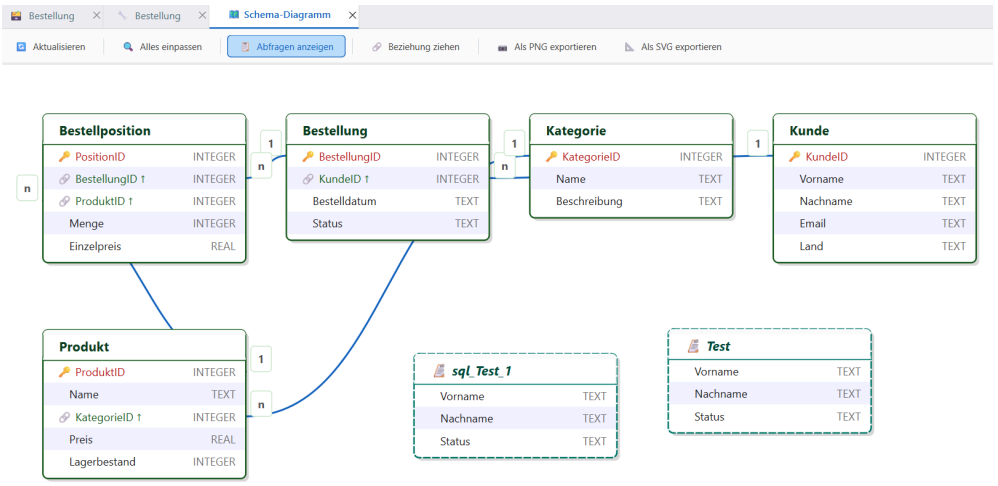
11. Schema-Diagramm

Das Schema-Diagramm zeigt Tabellen und Beziehungen grafisch. Tabellen können verschoben werden. Beziehungslinien zeigen Kardinalitäten:

- `1:n`: Ein Datensatz der Elterntabelle gehört zu mehreren Datensätzen der Kindtabelle.
- `1:1`: Der Fremdschlüssel ist eindeutig oder selbst Primärschlüssel.

Öffnen:

- **Bearbeiten → Schema-Diagramm**
- `Strg+D`
- Toolbar-Button **Schema**



Schema-Diagramm mit Tabellen und Beziehungslinien

Views erscheinen im Schema-Diagramm als gestrichelte teal-Box mit 👁-Präfix im Titel. Fremdschlüssellinien werden für Views nicht gezeichnet, da Views keine eigenen Schlüssel definieren.

Das Diagramm kann als PNG oder SVG exportiert werden. PNG wird in höherer Auflösung erzeugt. SVG ist abhängig von den in der Anwendung gebündelten Komponenten.

Das Schema-Diagramm ist nicht nur eine optische Zusatzfunktion. Es hilft dabei, die Struktur der Datenbank zu überprüfen. Wenn eine Beziehung im Diagramm fehlt, ist häufig auch der entsprechende Fremdschlüssel nicht angelegt. Wenn eine Beziehung unerwartet als `1:1` erscheint, sollte geprüft werden, ob die Fremdschlüsselspalte eindeutig ist.

Für den Unterricht eignet sich das Diagramm besonders als Kontrollansicht nach dem Tabellenentwurf. Lernende können vergleichen, ob ihr fachliches Modell, ihr relationales Schema und die tatsächlich angelegte Datenbank übereinstimmen.

12. Datenbank-Info

Die Datenbank-Info zeigt technische Informationen zur aktuell geöffneten Datenbank.

Öffnen:

- **Bearbeiten → Datenbank-Info**
- `Strg+Shift+I`

Angezeigt werden unter anderem:

- Dateipfad und Dateigröße
- SQLite-Version
- Ob die Fremdschlüssel-Prüfung aktiv ist
- Seitengröße und Journal-Modus
- Anzahl von Tabellen, Views, Indizes und gespeicherten Abfragen
- Interne DataBlick-Metadaten
- Hinweise zur Kompatibilität

Die Datenbank-Info ist vor allem dann nützlich, wenn geprüft werden soll, welche Datei gerade geöffnet ist und in welchem Zustand sich die Datenbank befindet. Im Unterricht kann sie auch genutzt werden, um den Unterschied zwischen sichtbaren Unterrichtstabellen und technischen Metadaten zu erklären.

13. Export und Austausch

DataBlick ist kein proprietäres Datenbankformat. Die Datenbankdatei ist eine SQLite-Datei und kann grundsätzlich auch mit anderen SQLite-Programmen geöffnet werden.

Exportmöglichkeiten:

- Tabellen als CSV, Excel-Datei oder tabellenspezifischer SQL-Dump
- SQL-Ergebnisse als CSV oder Excel
- Balkendiagramme aus SQL-Ergebnissen als PNG
- Gespeicherte Abfragen als SQL-Code
- Ergebnisse gespeicherter Abfragen als CSV oder Excel
- Abfrageergebnisse als neue echte Tabelle
- Vollständiger Datenbank-SQL-Dump
- Schema-Diagramm als PNG oder SVG

Kompatibel sind vor allem:

- DB Browser for SQLite
- SQLiteStudio
- DBeaver
- Python `sqlite3`
- sqlite3-Kommandozeile

Nicht direkt kompatibel sind:

- Microsoft Access `.accdb` / `.mdb`
- MySQL-, PostgreSQL- oder SQL-Server-Datenbanken als Serververbindung
- Access-Formulare, Berichte, Makros oder Access-spezifische Abfragen

Viele Exporte sind direkt im Navigationsbereich erreichbar: Rechtsklick auf eine Tabelle exportiert diese Tabelle, Rechtsklick auf eine gespeicherte Abfrage exportiert entweder den SQL-Code oder das aktuelle Ergebnis dieser Abfrage. Der vollständige Datenbank-Dump liegt weiterhin unter **Bearbeiten → SQL-Dump exportieren**.

Beim Export sollte unterschieden werden, was genau ausgegeben wird. Ein CSV-Export enthält nur Tabellen- oder Ergebnisdaten. Ein Export einer Abfrage als `.sql` enthält nur den SQL-Code, nicht das Ergebnis. Ein Schema-Diagramm zeigt die Struktur, enthält aber nicht die Daten selbst. Ein SQL-Dump kann dagegen Tabellenstruktur und Daten in SQL-Form beschreiben. Interne DataBlick-Tabellen und interne `__view_*`-Abfrage-Views werden dabei nicht als ausführbare Objekte exportiert; gespeicherte Abfragen erscheinen aus Sicherheitsgründen als Kommentare. Für Sicherungen ist normalerweise die `.db`-Datei am wichtigsten.

14. Fehler verstehen

DataBlick erklärt wichtige SQLite-Fehler didaktisch. Statt nur eine technische Fehlermeldung zu zeigen, wird erklärt, was vermutlich passiert ist und wie man weiter vorgehen kann.

Beispiele:

- **FOREIGN KEY constraint failed**: Ein Fremdschlüssel verweist auf einen nicht vorhandenen Datensatz.
- **UNIQUE constraint failed**: Ein Wert muss eindeutig sein, existiert aber schon.
- **NOT NULL constraint failed**: Ein Pflichtfeld wurde leer gelassen.
- **no such table**: Eine Tabelle existiert nicht oder wurde falsch geschrieben.

- **syntax error**: Die SQL-Anweisung ist syntaktisch falsch.

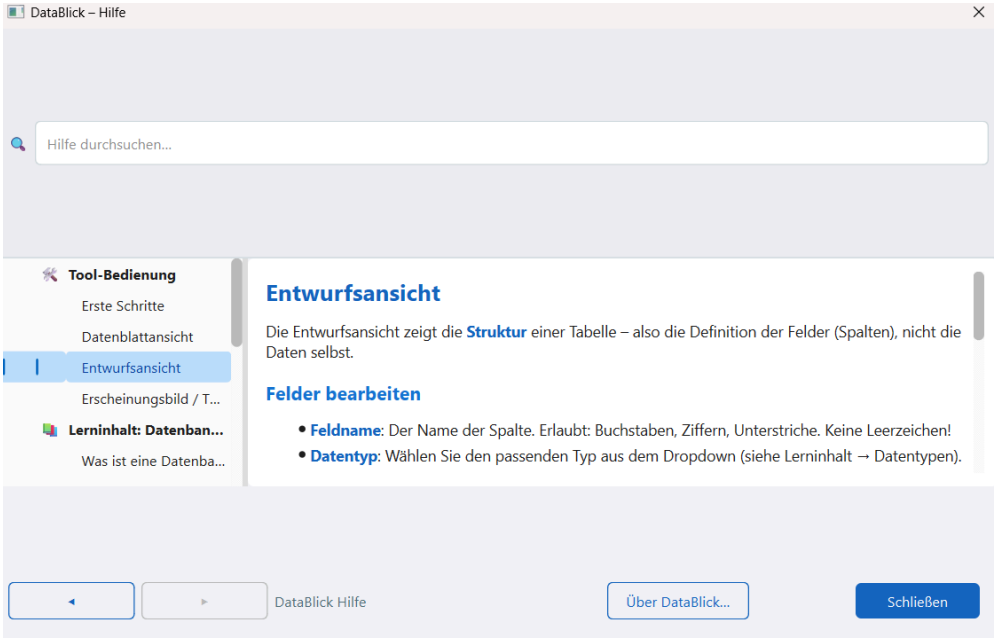
DataBlick extrahiert dabei Tabellen- und Spaltennamen direkt aus der SQLite-Fehlermeldung. Eine NOT NULL-Verletzung zeigt zum Beispiel: “Pflichtfeld-Fehler – Spalte Nachname (Tabelle schüler)”. Eine UNIQUE-Verletzung nennt die betroffene Spalte und Tabelle. Das macht sichtbar, welche Regel und welches Feld betroffen sind.

Im SQL-Editor und im QBE-Ergebnisbereich erscheint die Fehlermeldung in der Ergebnistabelle. In der Entwurfsansicht und in der Datenblattansicht wird beim Speichern eine ausführlichere Erklärung angezeigt.

Fehlermeldungen sind im Unterricht nicht nur Störungen, sondern Lerngelegenheiten. Eine Datenbank meldet Fehler, weil sie Regeln durchsetzt. Wenn ein Pflichtfeld leer bleibt, ein eindeutiger Wert doppelt vorkommt oder ein Fremdschlüssel ins Leere zeigt, schützt die Datenbank ihre eigene Konsistenz. Gerade diese Situationen machen sichtbar, warum Datenbankregeln sinnvoll sind.

15. Hilfe

Die Hilfe ist über **Hilfe → Hilfe öffnen**, `F1` oder den Hilfe-Button erreichbar.



Integriertes Hilfefenster mit Suchfeld

Die Hilfe bietet:

- Themenbaum
- Volltextsuche
- Vor- und Zurück-Pfeile für bereits angezeigte Hilfeseiten innerhalb des Hilfefensters
- **Über DataBlick…** für Version, Lizenz und Programminformationen
- Hilfetexte zu Bedienung und Datenbankgrundlagen
- Referenzthemen, zum Beispiel Datentypen, SQL-Grundlagen und Kompatibilität

Der Dialog **Über DataBlick…** ist über das Hauptmenü **Hilfe → Über DataBlick…** erreichbar. Zusätzlich gibt es im Hilfefenster unten einen eigenen Button **Über DataBlick…**. Die Pfeile im Hilfefenster blättern nicht durch das Handbuch wie Seitenzahlen, sondern springen im Verlauf der Hilfethemen zurück oder vor, die in diesem Hilfefenster bereits geöffnet wurden.

Die Hilfe kann sowohl während freier Arbeitsphasen als auch bei gezielten Aufgaben eingesetzt werden. Für Lernende ist besonders nützlich, dass Begriffe und typische Fehler direkt im Programm nachgeschlagen werden können.

16. Wichtige Tastenkombinationen

Tastenkombination	Funktion
`Strg+N`	Neue Datenbank
`Strg+O`	Datenbank öffnen
`Strg+W`	Datenbank schließen
`Strg+I`	Import-Assistent
`Strg+Q`	Neue SQL-Abfrage
`Strg+Shift+Q`	QBE-Abfrageentwurf
`Strg+D`	Schema-Diagramm
`Strg+Shift+I`	Datenbank-Info
`F5`	Aktualisieren
`F1`	Hilfe
`Alt+F4`	Programm beenden

Im SQL-Editor:

Tastenkombination	Funktion
`Strg+Enter`	Markierten Text ausführen (oder gesamte Abfrage, falls nichts markiert)
`Strg+Shift+Enter`	Gesamtes Skript ausführen (ab 2 Anweisungen)
`Strg+S`	Abfrage speichern
`Strg+Space`	Autovervollständigung manuell auslösen
`Strg+P`	Python-Tab für Thonny ein- oder ausblenden

Tastenkombinationen sind nicht notwendig, können aber bei längeren Arbeitsphasen Zeit sparen. Für den Einstieg reicht die Bedienung über Menü und Toolbar vollständig aus.

17. Beta-Hinweise

DataBlick befindet sich im Beta-Stadium. Der Funktionsumfang ist für Unterrichtstests geeignet, sollte aber vor produktivem Einsatz mit echten Lerngruppen einmal komplett auf dem Zielsystem durchgespielt werden.

Empfohlene Beta-Checkliste:

1. DataBlick auf einem Schulrechner starten.
2. Neue Datenbank anlegen.
3. Tabelle erstellen.
4. CSV importieren.
5. Daten bearbeiten und löschen.
6. Fremdschlüssel anlegen.
7. Schema-Diagramm öffnen.

8. QBE-Abfrage erstellen und speichern.
9. Verbal-Ansicht im QBE-Designer aufrufen.
10. FK-Anzeigefelder in der Entwurfsansicht eines Fremdschlüsselfeldes prüfen.
11. SQL-Abfrage ausführen (Strg+Enter). 11a. Skript-Modus testen: zwei Anweisungen eingeben, Syntaxprüfung abwarten, Skript ausführen (Strg+Shift+Enter), Protokoll-Tab und nummerierte Tabs prüfen.
12. Fehler absichtlich auslösen und Erklärung prüfen.
13. CSV/Excel/SQL-Dump exportieren.
14. Hilfe öffnen und Suche testen.
15. Datenbank schließen und erneut öffnen.

Bekannte Einschränkungen:

- DataBlick arbeitet nur mit SQLite-Dateien.
- `.accdb`-Import aus Microsoft Access ist nicht vorgesehen.
- QBE-Entwürfe können SQL erzeugen, aber beliebiges SQL wird nicht zurück in QBE umgewandelt.
- Bereits gespeicherte Zelländerungen in der Datenblattansicht können nicht allgemein per Undo zurückgenommen werden.
- Interne Tabellen mit dem Präfix `__tabella_` sollten nicht manuell gelöscht werden, wenn gespeicherte Abfragen oder Import-Spezifikationen erhalten bleiben sollen.
- Die SQL-Anweisungen `ATTACH`, `DETACH` und `VACUUM` sind im SQL-Editor aus Sicherheitsgründen gesperrt. Für den normalen Unterricht (SELECT, INSERT, UPDATE, DELETE, CREATE TABLE) hat das keine Auswirkungen.

Für den schulischen Einsatz empfiehlt es sich, vor der ersten Stunde eine kleine Testdatenbank vollständig durchzuspielen. Dabei sollten besonders Import, Fremdschlüssel, QBE, SQL-Editor und Export getestet werden. So lassen sich technische Probleme auf Schulrechnern früh erkennen.

18. Empfohlener Unterrichtseinsatz

DataBlick eignet sich besonders für:

- Einstieg in relationale Tabellen
- Primär- und Fremdschlüssel
- Import realer Daten
- SQL-Grundlagen
- Vergleich von grafischem QBE und SQL
- Diskussion von Datenintegrität
- Export und Austausch mit anderen Werkzeugen

Didaktisch sinnvoll ist ein Wechsel zwischen drei Ebenen:

1. **Daten sehen**: Datenblattansicht
2. **Struktur verstehen**: Entwurfsansicht und Schema-Diagramm
3. **Fragen formulieren**: QBE und SQL

Ein möglicher Unterrichtsgang beginnt mit einer einfachen Tabelle, zum Beispiel einer Liste von Büchern, Kursen oder Kunden. Danach wird gezeigt, warum eine einzige große Tabelle schnell unübersichtlich und fehleranfällig wird. Anschließend werden Informationen auf mehrere Tabellen verteilt und durch Primär- und Fremdschlüssel verbunden. Erst danach folgen Abfragen, zunächst grafisch mit QBE und dann zunehmend in SQL.

DataBlick sollte dabei nicht als Selbstzweck eingeführt werden. Das Programm ist das Werkzeug, mit dem die fachlichen Konzepte sichtbar werden: Tabellen speichern gleichartige Datensätze, Schlüssel machen Datensätze eindeutig, Fremdschlüssel sichern Beziehungen, Abfragen formulieren Fragen an Daten, und gespeicherte Abfragen oder Ergebnistabellen speichern unterschiedliche Arten von Arbeitsergebnissen.

19. Lizenz und Weitergabe

DataBlick ist als Freeware-Werkzeug für den Informatikunterricht angelegt. Der Dialog **Hilfe → Über DataBlick…** weist die Version und die Lizenz als **Freeware** aus. Das bedeutet: DataBlick darf kostenlos genutzt und weitergegeben werden. Das gilt auch für Schülerinnen und Schüler oder zuer Verfügung stellen auf einem Schulserver. Das gilt auch für dieses Handbuch und die bereitgestellten Beispieltabellen.

Der Unterrichtsgang und die Lehrerhinweise dazu unterliegen dem Copywrite. Ein Kopieren im Ganzen oder in Teilen ist nur im genannten Rahmen erlaubt (z.B. Kopieren von Lösungen, falls der Schülerband eingeführt ist).

Idee und Umsetzung: Martin Resch, 2026
Coding: Claude Cowork & ChatGPT Codex

Für Schulen ist vor allem wichtig, dass klar dokumentiert ist, welche Version eingesetzt wird und woher die Programmdatei stammt. Bei Weitergabe an Kolleginnen und Kollegen sollten Programmordner, Handbuch, Beispieldatenbanken und Lizenzhinweis gemeinsam bereitgestellt werden.

Anhang: Vorlagen-Datenbanken

DataBlick enthält drei eingebaute Beispieldatenbanken, die über die Startseite mit einem Klick erzeugt werden können. Sie sind so entworfen, dass sie typische Unterrichtssituationen abdecken und ohne Vorbereitung direkt im Kurs einsetzbar sind.

Schulverwaltung

Die Schulverwaltung modelliert eine vereinfachte Schuldatenbank mit Schülern, Lehrern, Klassen und Fächern. Die Daten zeigen realistische Strukturen: Schüler sind Klassen zugeordnet, Lehrer unterrichten Fächer, und die Stundenplandaten verbinden alle drei Bereiche. Didaktisch eignet sich diese Datenbank besonders für den Einstieg in Fremdschlüssel und Joins, weil die Beziehungen aus dem Schulalltag vertraut sind. Typische Abfragen wie „Welche Schüler hat Klasse 8b?” oder „Welche Fächer unterrichtet Frau Müller?” lassen sich direkt aus dem Kontext entwickeln. Die Datenbank enthält bewusst einige NULL-Werte und unvollständige Einträge, um auch Themen wie IS NULL und LEFT JOIN zu ermöglichen.

Stadtbibliothek

Die Stadtbibliothek bildet einen Buchbestand, Mitglieder und Ausleihen ab. Jedes Buch gehört einer Kategorie an, jede Ausleihe verbindet ein Mitglied mit einem Buch und enthält Ausleihdatum sowie optionales Rückgabedatum. Nicht zurückgegebene Bücher sind an einem NULL-Wert in der Rückgabedatumsspalte erkennbar und eignen sich gut für IS NULL-Abfragen. Die Datenbank enthält außerdem Bücher, die noch nie ausgeliehen wurden, und Mitglieder ohne aktuelle Ausleihe — beides sind klassische

Fälle für LEFT JOIN. Mit Aggregatfunktionen lassen sich Fragen wie „Welche Kategorie ist am häufigsten ausgeliehen?" oder „Wie viele Bücher hat ein bestimmtes Mitglied aktuell?" direkt im QBE-Designer entwickeln.

Online-Shop

Der Online-Shop ist die komplexeste der drei Vorlagen. Er enthält fünf Tabellen: Kategorie, Produkt, Kunde, Bestellung und Bestellposition. Die Daten sind mit gezielten didaktischen Sonderfällen versehen: Ein Produkt hat Lagerbestand null (ausverkauft), ein weiteres ist als inaktiv markiert (abgekündigt, hat aber Verkäufe), ein Produkt wurde noch nie bestellt (LEFT JOIN Demo), und eine Kundin hat noch keine Bestellung aufgegeben (LEFT JOIN Demo). Eine Bestellung ist storniert, zwei Bestellungen sind noch offen. Historische Preise in den Bestellpositionen weichen vom aktuellen Produktpreis ab und zeigen, warum Bestellpositionen einen eigenen Einzelpreis speichern. Die Datumsspalten sind als DATE deklariert und enthalten ISO-Datumsformate (JJJJ-MM-TT), was Datumsvergleiche mit > und < direkt ermöglicht. Fremdschlüssel-Anzeigefelder sind voreingestellt, sodass DataBlick in der Datenblattansicht statt einer nackten ID sofort den Produkt- oder Kundennamen zeigt.

Anhang: Startparameter für Schulumgebungen

DataBlick kann zentral von einem Server oder Netzlaufwerk gestartet werden. Die Programmdatei selbst wird dabei nur gelesen. Datenbanken, Importdateien und Exporte sollten im Unterricht trotzdem in persönlichen oder gruppenbezogenen Arbeitsordnern liegen, damit sich Schülerinnen und Schüler nicht gegenseitig überschreiben.

Für Schulen mit serverbasierten Standardprofilen ist es oft unzuverlässig, sich auf dauerhaft gespeicherte Benutzerprofile zu verlassen. DataBlick verwendet deshalb für Datei-Dialoge folgende Reihenfolge:

1. per Startparameter gesetzter Ordner
2. zuletzt in dieser Sitzung verwendeter Ordner
3. optional in den Benutzereinstellungen gespeicherter Ordner
4. Benutzerordner als Fallback

Folgende Startparameter werden unterstützt:

```
--work-dir      gemeinsamer Standardordner für Datenbanken, Import und
Export
--db-dir        Startordner für Datenbanken öffnen, anlegen und
speichern
--import-dir    Startordner für CSV-, TSV-, TXT- und Excel-Importe
--export-dir    Startordner für CSV-, Excel-, SQL-, PNG- und SVG-
Exporte
--snippet-dir   Ordner für SQL-Bausteindateien nach dem Muster
SQL_Beispiel*.txt
```

`--db-dir`, `--import-dir` und `--export-dir` überschreiben jeweils `--work-dir` für ihren Bereich. Wird `--snippet-dir` mindestens einmal angegeben, sucht DataBlick SQL-Bausteine ausschließlich in diesen angegebenen Ordnern. Ohne `--snippet-dir` gilt die normale Baustein-Suche neben der Programmdatei sowie in `doc`, `sql_snippets` und in der Entwicklungsumgebung.

Beispiel für eine einfache Schulverknüpfung:

```
DataBlick.exe --work-dir "H:\Informatik\DataBlick"
```

Beispiel mit getrennten Ordnern:

```
DataBlick.exe ^
  --db-dir "H:\Informatik\DataBlick\Datenbanken" ^
  --import-dir "\\server\material\datablick\csv" ^
  --export-dir "H:\Informatik\DataBlick\Export" ^
  --snippet-dir "\\server\material\datablick\sql_snippets"
```

Mehrere Bausteinordner sind möglich:

```
DataBlick.exe --snippet-dir "\\server\kurs\teil2" --snippet-dir
"\\server\kurs\teil3"
```

Unter Windows können UNC-Pfade wie `\\server\freigabe\ordner` verwendet werden. Umgebungsvariablen wie `%USERPROFILE%` werden aufgelöst. Unter Linux und macOS funktionieren normale absolute Pfade wie `/srv/datablick` oder `/Volumes/Kurs/DataBlick`; auch `~` wird aufgelöst. Wichtig ist, dass die angegebenen Ordner aus Sicht des gestarteten Benutzerkontos lesbar beziehungsweise für Exporte beschreibbar sind.