

SQL-Tabellen

- [SQL Tabelle anlegen](#)
- [SQL Tabelle ändern](#)
- [Mehr → Alle SQL-Tabellen exportieren / SQL-Tabellen importieren](#)

SQL Tabelle anlegen

SQL Tabelle anlegen ist der Schritt **vor Neu**: die Tabelle gibt es noch nicht. Der Dialog fragt Verbindung, Tabellename und Spalten ab und lässt das Portalskript ein `CREATE TABLE` ausführen – auf MySQL oder MS SQL, je nachdem, was in der `dbConn.json` zu dieser Verbindung steht.

Wo die Tabelle entsteht

Feld	Bedeutung
Verbindung	Die DEXPRO-Datenbank, in der die Tabelle angelegt wird. Workflow, DOCUMENTS und Squeeze stehen nie zur Wahl – siehe Geschützte Datenbanken
Tabellename	Buchstaben, Ziffern und <code>_</code> , beginnend mit einem Buchstaben oder <code>_</code> , höchstens 63 Zeichen. Ein Name, den es in dieser Datenbank schon gibt, wird abgewiesen

ID und Licence sind immer dabei

Jede so angelegte Tabelle beginnt mit denselben zwei Spalten wie jede ausgelieferte DEX-Tabelle:

Spalte	Form
ID	<code>BIGINT</code> , Primärschlüssel, von der Datenbank hochgezählt
Licence	<code>varchar(50)</code> , <code>NOT NULL</code> , Standardwert ist der Mandant dieser Anmeldung

Beide zeigt der Dialog nur an. Anlegen, ändern oder weglassen kann sie niemand – das entscheidet das Portalskript.

Damit ist der Primärschlüssel erschöpft: eine weitere Spaltenspalte bietet der Dialog nicht an, und ohne Schlüssel gibt es auch keinen Auto-Wert, denn eine hochzählende Spalte muss der Schlüssel sein. In den Spaltenzeilen stehen **PK** und **Auto** deshalb nur als Strich.

Die eigenen Spalten

Je Zeile: Name, Typ, Länge, Standardwert und der Haken **NULL**.

Typ	MS SQL / MySQL	Länge
Text	NVARCHAR(n) / VARCHAR(n)	Zeichenzahl, Vorgabe 255
Langtext	NVARCHAR(MAX) / LONGTEXT	-
Zeichen (feste Länge)	NCHAR(n) / CHAR(n)	Vorgabe 10
Ganzzahl	INT	-
Ganzzahl (groß)	BIGINT	-
Ganzzahl (klein)	SMALLINT	-
Dezimalzahl	DECIMAL(p, s)	Stellen, Nachkommastellen - z. B. 18,2
Gleitkommazahl	FLOAT / DOUBLE	-
Ja/Nein	BIT / BIT(1)	-
Datum	DATE	-
Datum und Zeit	DATETIME	-
Binärdaten	VARBINARY(MAX) / LONGBLOB	-

- **NULL** - ohne diesen Haken muss jede neue Zeile einen Wert mitbringen, oder die Spalte braucht einen Standardwert.
- **Standardwert** - bei Datum und Zeit steht `now` für die aktuelle Zeit (`GETDATE()` bzw. `CURRENT_TIMESTAMP`).
- Spaltennamen folgen derselben Regel wie der Tabellename. Doppelte Namen und die beiden festen Namen `ID` und `Licence` werden abgewiesen.

SQL anzeigen

`SQL anzeigen` holt das fertige Statement vom Portalskript und zeigt es in der Statuszeile - **ausgeführt ist dabei noch nichts**. Das Statement wird nicht in der Seite zusammengebaut, sondern im Backend aus geprüften Bausteinen; die Vorschau ist also genau das, was später läuft.

Anlegen

`Anlegen` führt das `CREATE TABLE` aus. Der Fortschrittskasten bleibt stehen, bis die Datenbank geantwortet hat.

Ist **Definition dazu anlegen** angehakt, steht anschließend gleich eine neue Definition offen - mit den Spalten der frisch angelegten Tabelle, noch ungespeichert. Der Name wird aus dem Tabellennamen abgeleitet und ist frei wählbar.

Fallen lassen kann der Editor eine Tabelle nicht. Ein `DROP TABLE` gibt es hier nicht – das bleibt Sache der Datenbank. Einzelne selbst angelegte Spalten lassen sich über [SQL Tabelle ändern](#) wieder löschen.

SQL Tabelle ändern

SQL Tabelle ändern ist der Schritt **danach**: die Tabelle steht, es fehlt eine Spalte. Der Dialog kann absichtlich nur den Teil von `ALTER TABLE`, bei dem nichts unabsichtlich verlorengeht.

Möglich	Nicht möglich
Spalte hinzufügen – immer mit NULL erlaubt, ohne Schlüssel, ohne Auto-Wert	Spalte umbenennen
Zeichenspalte verlängern – nur größer	Typ wechseln
Selbst angelegte Spalte löschen	Länge verkleinern, fremde Spalten löschen, Tabelle löschen

Die vorhandenen Spalten stehen deshalb als Text da, mit genau einem Eingabefeld dort, wo die Länge steht. Die Entscheidung darüber fällt im Portalskript, nicht im Formular.

Welche Tabelle geändert wird

Verbindung und Tabelle werden oben gewählt; Views stehen nicht zur Wahl, ihre Spalten kommen aus ihrer Abfrage. Sobald eine Tabelle feststeht, stehen ihre Spalten darunter, so wie die Datenbank sie meldet.

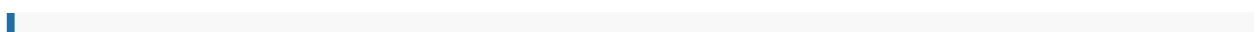
Spalte verlängern

Bei einer Zeichenspalte lässt sich die Länge **nur nach oben** ändern; der Tooltip nennt den aktuellen Wert und die Obergrenze des Typs. Was beim Verkürzen abgeschnitten würde, käme nicht zurück – deshalb gibt es diesen Weg nicht. Reicht die Obergrenze nicht, ist eine neue Spalte vom Typ *Langtext* der Weg.

Spalte löschen

Ein **x** hat nur eine Spalte, die **dieser Editor selbst angelegt** hat. Er merkt sich das in der Custom Property `TS_DdlAddedColumns`. Was aus der Auslieferung kommt oder von Hand entstanden ist, hat keinen Löschknopf und wird auch dann abgelehnt, wenn die Anfrage es verlangt.

Ein Klick markiert die Spalte nur. Gelöscht wird sie erst mit **Ändern** und einer Rückfrage – ein zweiter Klick auf das **x** nimmt die Markierung wieder weg.



Mit der Spalte fällt ihr Inhalt in jeder Zeile. Das lässt sich nicht zurückholen.

ID und **Licence** bleiben in jedem Fall, wie sie sind: eine Tabelle, in der sie anders aussehen, passt nicht mehr zu den anderen.

Änderungen in die Definitionen überführen

Die dritte Karte des Dialogs zieht die Definitionen nach – denn eine Spalte in der Datenbank ist erst zu gebrauchen, wenn eine Definition sie kennt.

Angeboten wird, was auf **genau diese Tabelle** zeigt (dieselbe Verbindung, derselbe Tabellenname), voreingestellt alles davon:

- Angehakte Definitionen bekommen die **neuen** Spalten eingetragen, **gelöschte** fallen dort heraus. Was schon dasteht, bleibt unangetastet.
- Die gerade **geöffnete** Definition wird dabei nicht im Hintergrund gespeichert: sie bekommt die Spalten im Arbeitsspeicher und gilt als geändert. So verliert niemand offene Änderungen dadurch, dass eine Tabelle erweitert wurde.
- Eine Definition, deren JSON nicht lesbar ist, wird nicht angefasst – sie muss erst von Hand geradegezogen werden.

Zeigt keine Definition auf die Tabelle, sagt der Dialog das: die Spalten entstehen dann nur in der Datenbank, und **Neu** macht daraus eine Definition.

SQL anzeigen und Ändern

Wie beim Anlegen zeigt **SQL anzeigen** das Statement, das später läuft, ohne etwas auszuführen. **Ändern** führt es aus; bei markierten Löschungen heißt der Knopf **Ändern und löschen** und fragt vorher nach.

Ist nichts zu tun – keine neue Spalte, keine gewachsene Länge –, sagt der Dialog das, statt ein leeres Statement zu schicken.

Mehr ? Alle SQL-Tabellen exportieren / SQL-Tabellen importieren

Dieselben beiden Schritte wie *SQL Tabelle anlegen* und *SQL Tabelle ändern*, nur für eine **ganze Datenbank** statt für eine Tabelle. Gemeint ist der Umzug vom Entwicklungssystem zum Kunden: dort stehen zwanzig Tabellen, hier keine davon.

Was in der Datei steht

Die Datei beschreibt den **Aufbau** jeder Tabelle – Spalten, Typ, Länge, Stellen, NULL, Standardwert – und **keine einzige Zeile Daten**.

“ Das ist keine Sicherung. Wer Daten mitnehmen will, nimmt ein Werkzeug der Datenbank.

Sichten, Indizes, `UNIQUE`-Bedingungen, Fremdschlüssel und Trigger gehen nicht mit: was dieser Editor nicht anlegen kann, verspricht er auch nicht. Die Typen in der Datei sind die des Editors und nicht die einer bestimmten Datenbank – nur so lässt sich eine MySQL- in eine MS-SQL-Datenbank spielen. Ein `TINYINT` kommt deshalb als `SMALLINT` wieder an.

Exportieren

Angeboten werden die Datenbanken, in denen dieser Editor Tabellen anlegen darf. Die Workflow-Datenbank, das Repository und Squeeze stehen nicht dabei: ihr Aufbau gehört zum Produkt, kommt aus der Auslieferung und wäre in der Datei nichts als Länge – siehe [Geschützte Datenbanken](#).

Gelesen wird je Verbindung ein Aufruf; zusammengesetzt wird die Datei in der Seite. Eine Datenbank, die nicht antwortet, steht als Fehler in der Rückmeldung, die übrigen kommen trotzdem mit.

Importieren

Angeglichen wird in **eine Richtung**, und zwar genau in den drei Fällen, bei denen nichts verlorengeht:

Fall	Was geschieht
Tabelle fehlt	wird angelegt
Spalte fehlt	wird ergänzt
Zeichenspalte ist kürzer als in der Datei	wird verlängert

Gelöscht wird nichts. Eine Spalte, die in der Datei fehlt, aber in der Zieltabelle steht, bleibt stehen. Verkleinert wird nichts, und kein Typ wird umgedeutet: beides steht als Hinweis in der Planzeile und wird nicht ausgeführt.

Der Fall, um den es vor allem geht, ist der zusammengesetzte: eine Tabelle, die seit dem letzten Abgleich ein Feld dazubekommen hat **und** bei der eine Länge gewachsen ist, kommt in einem Lauf nach.

Schritt 1: Verbindungen zuordnen

Die Verbindungen aus der Datei werden denen dieser Installation zugeordnet, voreingestellt gleicher Name. Zur Wahl stehen die Datenbanken, in denen etwas entstehen darf. Eine Verbindung aus der Datei, zu der hier keine passt, bleibt auf *nicht importieren*.

Die Datenbanken des Produkts stehen auch hier nicht zur Wahl. Eine Zeile, die auf eine von ihnen zeigt, bleibt auf *nicht importieren* – siehe [Geschützte Datenbanken](#).

Schritt 2: „Das würde geschehen“

Prüfen rechnet den Plan; dieser Schritt **liest nur**. Danach steht Tabelle für Tabelle da, was entstünde, mit einem Haken je Zeile, der sich abwählen lässt, und mit Hinweisen zu allem, was übergangen wird (anderer Typ, kürzere Länge, unbekannter Typ, unbrauchbarer Spaltenname, Sicht statt Tabelle, geschützte Datenbank).

Jede neu angelegte Tabelle bekommt auch hier **ID** und **Licence** vorangestellt.

Erst **Importieren** schreibt – über dieselben Schritte wie *SQL Tabelle anlegen* und *SQL Tabelle ändern*, und damit unter denselben Grenzen.

Danach

Eine neue Spalte ist erst zu gebrauchen, wenn eine Definition sie kennt. Der Reiter [Spalten](#) nimmt sie über **+ aus Tabelle ...** in die passende Definition auf.