

# View erstellen

In einigen Fällen ist es hilfreich sich die Abfragen als Views zu speichern, z. B.:

- Abfragen nicht immer wieder neu eingeben zu müssen
- Datenzugriffe beschränken (Ein Katalog mit den Abfragen wird freigegeben, statt die originalen Kataloge)
- Vereinfachung von Zugriffen für Anwender (verstecken der Komplexität)

Eine View wird mit dem Statement CREATE VIEW erstellt. Dafür am besten einen eigenen Katalog erstellen wie in der Installation gezeigt: <https://jaeckel.one/link/230#bkmrk-dummy.properties%3A-di>

Bevor die View als Abfrage erstellt werden kann sind noch der Katalog und das Schema auszuwählen, in dem die View erstellt werden soll. Für dieses Beispiel wird der Katalog *dummy* und das Schema *default* ausgewählt.

Nun kann in dem integrierte SQL Client einfach die CREATE VIEW Abfrage erstellt werden. Dafür natürlich vorher erstmal ein SELECT ausprobieren, welches dann als View erstellt werden soll. Dem gewünschten SELECT werden dann einfach die Schlüsselworte `CREATE VIEW view_name AS` vorangestellt.

Es folgt ein Beispiel.

```
CREATE VIEW diesIstEineView
AS
SELECT
    person.persontype
    ,person.firstname
    ,person.lastname
    ,person.modifieddate
    ,NewTable.column1
    ,NewTable.column2
FROM sqlserver.person.person AS person
JOIN mysql.Test.NewTable AS NewTable
    ON person.businessentityid = NewTable.ID;
```

Sollte die View aufgrund eines Fehler nicht erstellt werden können, wird die Fehlermeldung ausgegeben. Sofern alles klappt, gibt es nur folgendes Feedback.

The screenshot shows a SQL IDE interface with a dark theme. At the top, there are tabs for 'SQL' and 'Session Properties'. Below the tabs are dropdown menus for 'Catalog' and 'Schema', and a green 'Run' button. The main area contains the following SQL code:

```
CREATE VIEW diesIstEineView
AS
SELECT
  person.persontype
  ,person.firstname
  ,person.lastname
  ,person.modifieddate
  ,NewTable.column1
  ,NewTable.column2
FROM sqlserver.person.person AS person
JOIN mysql.Test.NewTable AS NewTable
ON person.businessentityid = NewTable.ID limit 100
```

Below the code, the 'Results' section is visible, showing the session ID '20240724\_182247\_00409\_t9nfc'. The results area is currently empty, with a 'result' label and a dark background. At the bottom right, there are navigation controls: 'Rows per page: 10', '1-1 of 1', and navigation arrows.

Von nun an kann die View wie folgt aufgerufen werden.

```
SELECT * FROM dummy.default.diesIstEineView;
```

The screenshot shows the same SQL IDE interface. The 'SQL' tab is active, and the 'Run' button has been clicked. The main area now contains the query:

```
SELECT * FROM dummy.default.diesIstEineView;
```

The 'Results' section shows the session ID '20240724\_182402\_00410\_t9nfc'. Below the session ID, a table of results is displayed with the following columns: 'persontype', 'firstname', 'lastname', 'modifieddate', 'column1', and 'column2'. The table contains three rows of data:

persontype	firstname	lastname	modifieddate	column1	column2
EM	Roberto	Tamburello	2007-11-04 00:00:00.000	Einkauf	Unangenehmer Genosse
EM	Rob	Walters	2007-11-28 00:00:00.000	Vertrieb	Sehr aufdringlich
EM	Gail	Erickson	2007-12-30 00:00:00.000	IT	Sehr freundlich

At the bottom right, the navigation controls show 'Rows per page: 10', '1-3 of 3', and navigation arrows.

Mit Hilfe des Dummy Kataloges kann nun eine Datenquelle für Metabase erstellt werden. Metabase selbst kann keine Abfragen aus mehreren Datenquellen erzeugen. Doch mit Presto als Mittelsmann und mit dem eigenen Katalog können nun die gewünschten Abfragen als Views gespeichert werden. In Metabase wird dann der Katalog als Quelle eingerichtet und danach können wie gewohnt mit der Metabase UI Abfragen gebastelt werden. Die Anwender sehen gar nicht, das darunter mehrere Quellen zusammengeführt wurden.

Version #1

Erstellt: 2024-07-24 20:11:03 CEST von Marcel

Zuletzt aktualisiert: 2024-07-24 20:27:27 CEST von Marcel