

Presto

Presto kann von mehreren Datenquellen (SQL, MySQL, Redis, Mongo, ...) Daten sammeln, diese zusammenführen und ggf. aggregieren und anderen Tools bereitstellen. So wird Presto z. B. von Metabase und Qlik unterstützt, sodass es in diesen Tools als Datenquelle verwendet werden kann.

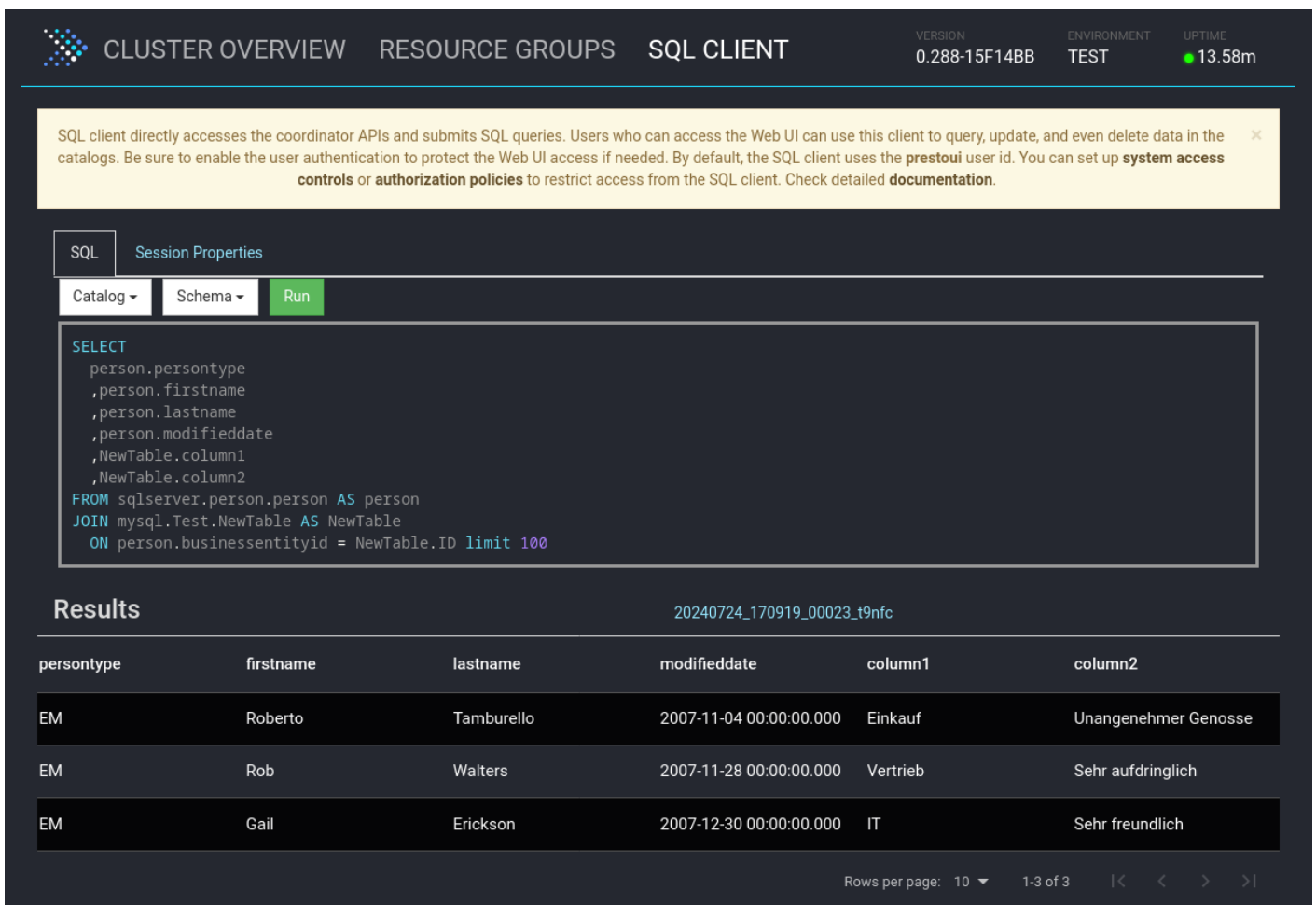
- [Übersicht](#)
- [Installation](#)
- [Eine erste Abfrage](#)
- [View erstellen](#)

Übersicht

First of all: Presto ist **keine** Datenbank. Es speichert keine Daten, sondern ruft diese in Echtzeit von Datenbanken ab.

Die Aufgabe von Presto im Umfeld eines Datawarehouses ist das Ausführen von Abfragen in verschiedenen Datenbanken, die Daten zu verbinden und ggf. zu aggregieren (Summe, Durchschnitt, ...) und/oder zu filtern. Das Ergebnis wird dann an die anfragende Anwendung zurückgegeben.

Es folgt ein Beispiel. In diesem wird über den integrierten Web-SQL-Client eine Abfrage ausgeführt. Dabei werden zum einen Daten von einem Microsoft SQL Server abgefragt und mit Daten von einem MySQL Server kombiniert.



The screenshot shows the Presto SQL Client interface. At the top, there are navigation tabs: CLUSTER OVERVIEW, RESOURCE GROUPS, and SQL CLIENT. The SQL CLIENT tab is active, showing the version (0.288-15F14BB), environment (TEST), and uptime (13.58m). A yellow warning box at the top states: "SQL client directly accesses the coordinator APIs and submits SQL queries. Users who can access the Web UI can use this client to query, update, and even delete data in the catalogs. Be sure to enable the user authentication to protect the Web UI access if needed. By default, the SQL client uses the prestouid user id. You can set up system access controls or authorization policies to restrict access from the SQL client. Check detailed documentation." Below the warning box, there are tabs for SQL and Session Properties. The SQL tab is active, showing a query editor with a "Run" button. The query is:

```
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 query editor, there is a "Results" section showing the execution ID (20240724_170919_00023_t9nfc) and a table of results. The table has six columns: persontype, firstname, lastname, modifieddate, column1, and column2. There are three rows of data. At the bottom right, there is a pagination control showing "Rows per page: 10" and "1-3 of 3".

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

Der Web Client eignet sich sehr gut zum Testen von Ad-Hoc Abfragen und um neue Views zu erstellen.

Die selbe Abfrage kann dann in der BI Software wie z. B. Metabase ausgeführt werden.



Suche...

+ Neu



Startseite

SAMMLUNGEN

Unsere Analysen

Deine persönliche Sammlung

DATEN

Daten durchsuchen

Neue Frage

Speichern

Dummy

```
1 SELECT
2   person.persontype
3   ,person.firstname
4   ,person.lastname
5   ,person.modifieddate
6   ,NewTable.column1
7   ,NewTable.column2
8 FROM sqlserver.person.person AS person
9 JOIN mysql.Test.NewTable AS NewTable
10  ON person.businessentityid = NewTable.ID
```

persontype

firstname

lastname

modifieddate

column1

column2

EM	Roberto	Tamburello	November 4, 2007, 12:00 AM	Einkauf	Unangenehmer Genosse
EM	Rob	Walters	November 28, 2007, 12:00 AM	Vertrieb	Sehr aufdringlich
EM	Gail	Erickson	Dezember 30, 2007, 12:00 AM	IT	Sehr freundlich

Visualisierung

Zeige 3 Zeilen

Installation

Presto steht als kostenlose Installation zur Verfügung. Im folgenden wird die Installation mittels Docker gezeigt.

Die offizielle Dokumentation kann hier nachgelesen werden: [Deploy Presto From a Docker Image](#)

Für Presto werden neben der Docker Compose Konfigurationsdatei noch weitere Konfigurationen benötigt, deswegen am besten wie immer einen eigenen Ordner erstellen, in den alle Konfigdateien kopiert werden.

Zuerst die folgende docker-compose.yml

```
presto:
  image: prestodb/presto
  container_name: presto
  volumes:
    - /pfad/zu/presto/config.properties:/opt/presto-server/etc/config.properties
    - /pfad/zu/presto/jvm.config:/opt/presto-server/etc/jvm.config
    - /pfad/zu/presto/catalog:/opt/presto-server/etc/catalog
  ports:
    - 8080:8080
  restart: unless-stopped
```

In der Konfiguration sind bereits 2 Dateien und ein Ordner angegeben. In den Konfigurationsdateien sind einige Einstellungen hinterlegt, wie z. B. der Port und über welchen Link Presto erreichbar sein soll. Für ein produktives Setup müssten noch einige Einstellungen mehr vorgenommen werden, aber zum Testen reichen diese bereits aus.

config.properties

```
coordinator=true
node-scheduler.include-coordinator=true
http-server.http.port=8080
discovery-server.enabled=true
discovery.uri=http://localhost:8080
```

jvm.config

```
-server
-Xmx2G
-XX:+UseG1GC
-XX:G1HeapRegionSize=32M
-XX:+UseGCOverheadLimit
-XX:+ExplicitGCInvokesConcurrent
-XX:+HeapDumpOnOutOfMemoryError
-XX:+ExitOnOutOfMemoryError
-Djdk.attach.allowAttachSelf=true
```

Nachdem die beiden Konfigdateien im selben Ordner erstellt wurden wie die Compose Konfiguration, ist noch der Ordner *catalog* anzulegen. In diesen Ordner wird nun für jede Datenquelle eine eigene Konfiguration hinterlegt.

Folgend ein paar Beispiele (die Dateien bitte in dem Ordner *catalog* speichern).

In den folgenden Konfigurationen bitte unbedingt auf Leerzeichen achten. Diese müssen entfernt werden, denn sonst wird z. B. der Connector nicht erkannt, wenn sich hinter dem Namen ein Leerzeichen befindet.

dummy.properties: Diese Konfiguration ist besonders, da sie keine Datenquelle definiert. Aber es können z. B. Views in diesem Katalog erstellt werden, sodass sich Views über mehrere Datenquellen hinterlegen lassen und dann z. B. in Metabase geladen werden können. In Metabase wird dann einfach dieser Dummy Katalog als Datenquelle hinterlegt und Metabase merkt gar nicht, dass die Daten aus verschiedenen Datenbanken stammen.

```
connector.name=memory
```

mysql.properties: Eine Konfiguration für einen MySQL Server, bitte die Daten an die eigene Umgebung anpassen.

```
connector.name=mysql
connection-url=jdbc:mysql://mysql:3306
connection-user=presto_user
connection-password=HierDasPasswort
case-insensitive-name-matching=true
```

sqlserver.properties: Eine Konfiguration für einen Microsoft SQL Server, bitte die Daten an die eigene Umgebung anpassen.

```
connector.name=sqlserver
connection-
```

```
url=jdbc:sqlserver://sql:1433;databaseName=AdventureWorks2022;trustServerCertificate=true;
connection-user=presto_user
connection-password=HierDasPasswort
```

Presto unterstützt natürlich noch viele weitere Datenbanken.

Hinweis: Wenn neue Kataloge hinzugefügt werden, muss Presto neugestartet werden.

Für eine Übersicht sowie Einrichtungshinweise am besten in die offizielle Dokumentation schauen:

[Connectors](#)

Sobald alles eingestellt ist, kann der Docker Container erstellt und gestartet werden.

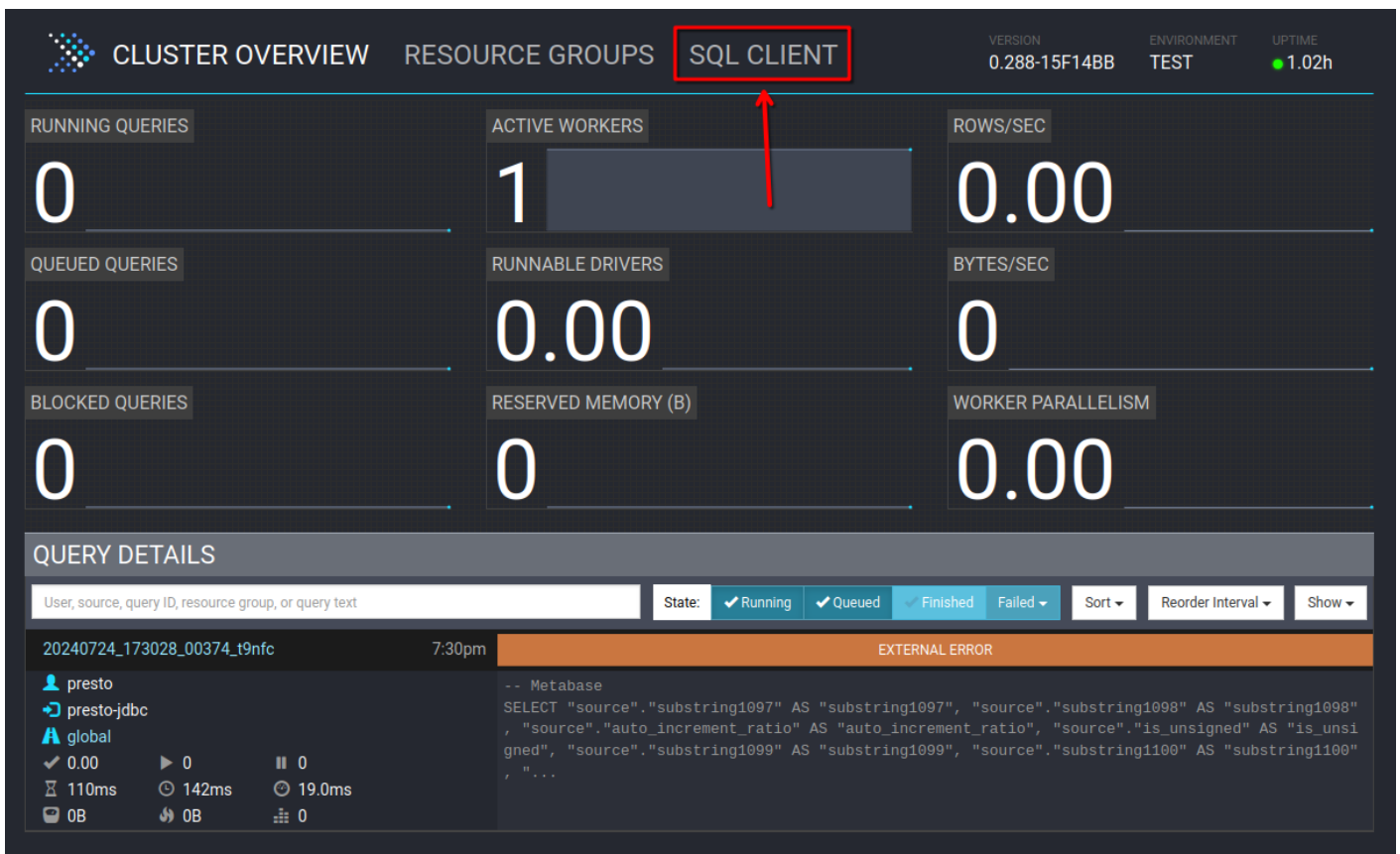
Um die Docker Compose Konfiguration auszuführen, kann am besten in das Verzeichnis der YAML Datei gewechselt werden. Danach wird je nach gewählter Installation `sudo docker-compose up -d` oder `sudo docker compose up -d` (keine Bindestrich zwischen docker und compose) eingegeben, um die Standard Konfiguration `docker-compose.yml` zu starten. Compose erstellt dann die gewünschten Container mit den angegebenen Optionen. Sollten die Container bereits mit dieser Compose Konfiguration erstellt worden sein, so werden die Container in dieser neu erstellt, dessen Konfiguration geändert wurde.

Anschließend ist Presto unter <http://localhost:8080/> erreichbar.

Eine erste Abfrage

Eine erste Abfrage ist mit dem integrierten Web Client schnell erstellt.

Zuerst wird die Weboberfläche von Presto geöffnet, z. B: unter <http://localhost:8080/> Presto öffnen und *SQL Client* auswählen.



The screenshot displays the Presto SQL Client interface. At the top, there are navigation tabs: 'CLUSTER OVERVIEW', 'RESOURCE GROUPS', and 'SQL CLIENT' (highlighted with a red box and a red arrow). The top right corner shows system information: 'VERSION 0.288-15F14BB', 'ENVIRONMENT TEST', and 'UPTIME 1.02h'.

The main dashboard is divided into several sections:

- RUNNING QUERIES:** 0
- ACTIVE WORKERS:** 1
- ROWS/SEC:** 0.00
- QUEUED QUERIES:** 0
- RUNNABLE DRIVERS:** 0.00
- BYTES/SEC:** 0
- BLOCKED QUERIES:** 0
- RESERVED MEMORY (B):** 0
- WORKER PARALLELISM:** 0.00

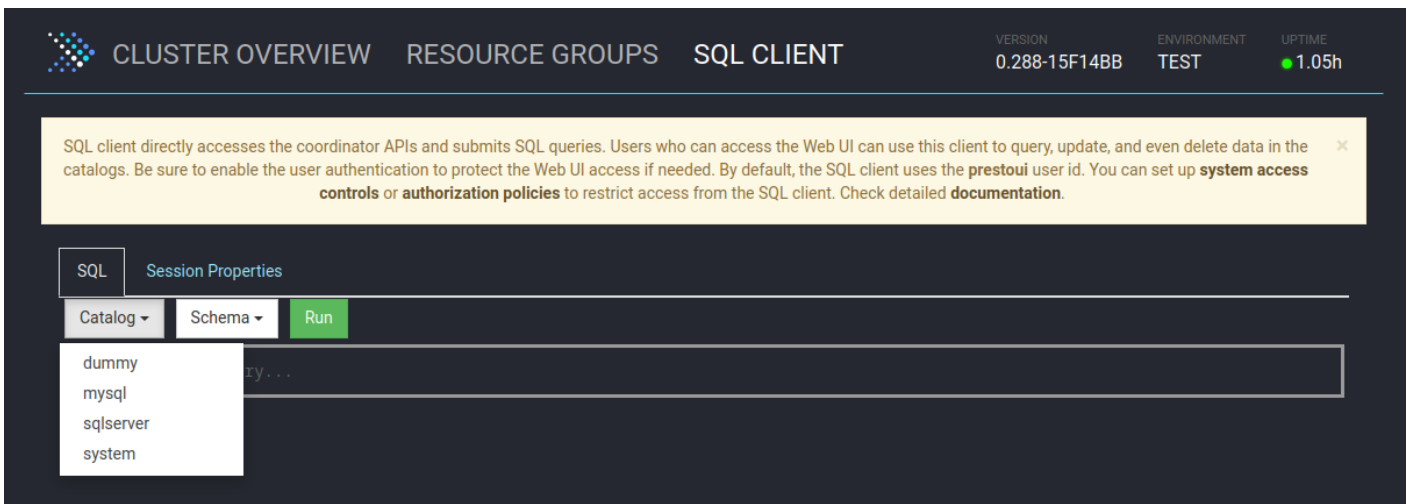
The 'QUERY DETAILS' section at the bottom shows a search bar and filters. The current query is identified by ID '20240724_173028_00374_t9nfc' at '7:30pm'. The state is 'EXTERNAL ERROR'. The query text is:

```
-- Metabase
SELECT "source"."substring1097" AS "substring1097", "source"."substring1098" AS "substring1098"
, "source"."auto_increment_ratio" AS "auto_increment_ratio", "source"."is_unsigned" AS "is_unsi
gned", "source"."substring1099" AS "substring1099", "source"."substring1100" AS "substring1100"
, "..."
```

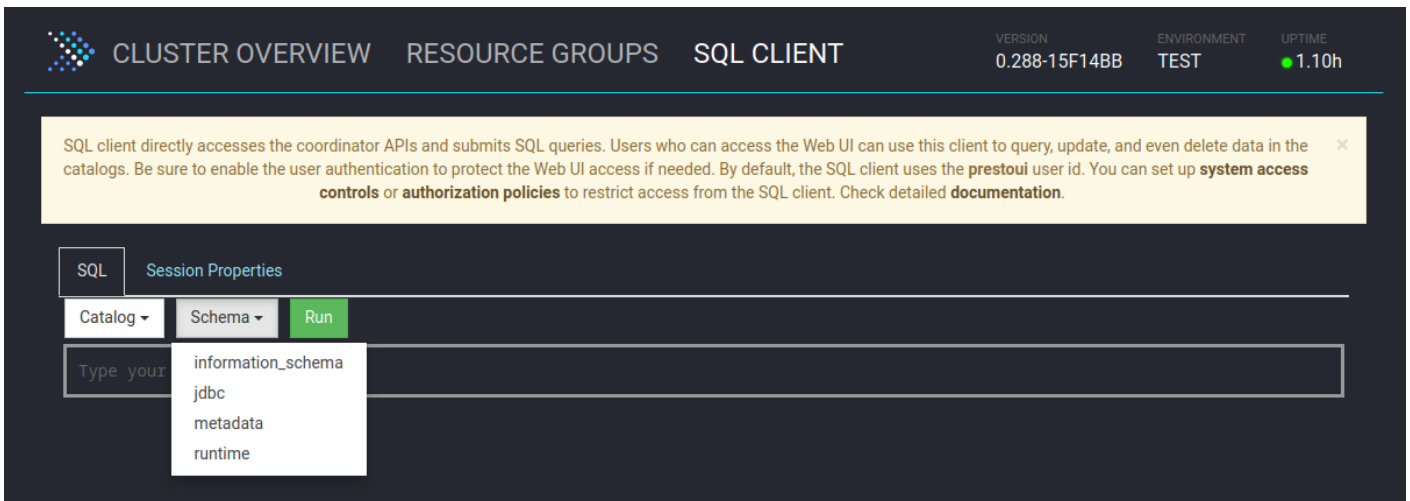
Metadata for the query is shown on the left:

- Users: presto, presto-jdbc, global
- Progress: 0.00, 0, 0
- Timings: 110ms, 142ms, 19.0ms
- Resources: 0B, 0B, 0

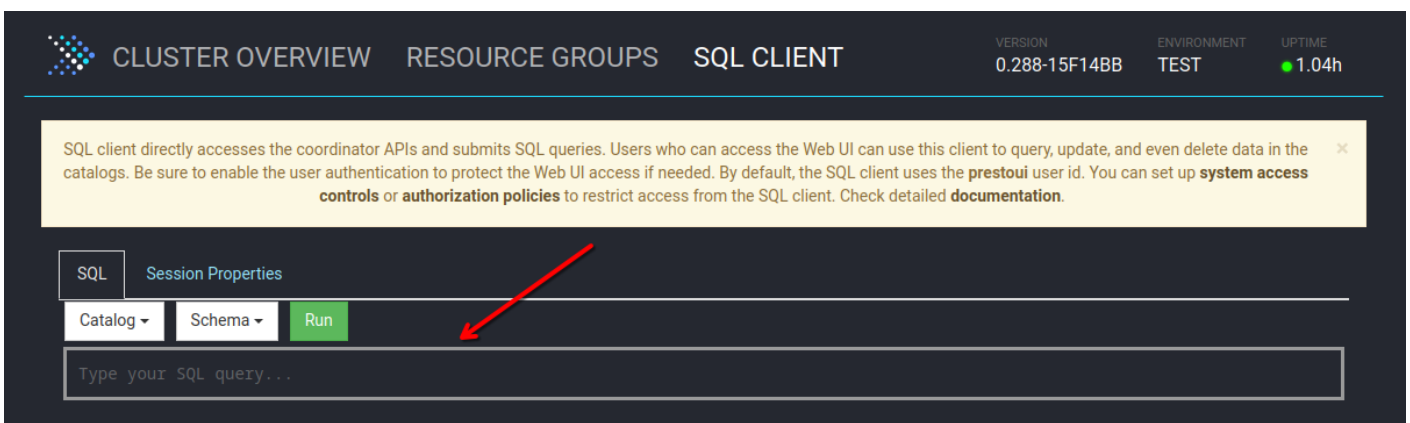
Dann kann über die das Dropdown *Catalog* geprüft werden, welche Kataloge also Datenquellen in Presto vorhanden sind. Hierzu einfach das Dropdown anklicken und es sollten alle Kataloge aus dem Ordner *catalog* angezeigt werden. Zusätzlich wird immer der Katalog *system* angezeigt, welcher die Systeminformationen von Presto selbst bereitstellt. Wird einer der Kataloge angeklickt, so wird dieser ausgewählt und alle Abfragen haben automatisch diesen Katalog vorbelegt, sodass dieser nicht mehr explizit in den Abfragen anzugeben ist.



Wird einer der Kataloge ausgewählt, z. B. *system*, so werden die Schemas von diesem in dem 2. Dropdown *Schema* angezeigt. Auch das Schema kann ausgewählt werden, damit auch dieses nicht in den Abfragen angegeben werden muss.



In das Textfeld unter SQL kann die Abfrage eingegeben werden.



Im folgenden werden einige wichtige Abfragen gezeigt, um Informationen über die verschiedenen Kataloge und ihre Inhalte zu erhalten.

Alle Kataloge anzeigen

```
SHOW CATALOGS;
```

The screenshot shows the SQL Server Enterprise Manager interface. At the top, there is a 'SQL' tab and a 'Session Properties' button. Below that, there are dropdown menus for 'Catalog' and 'Schema', and a green 'Run' button. The query 'SHOW CATALOGS;' is entered in the text area. The 'Results' pane shows the following output:

Catalog
dummy
mysql
sqlserver
system

At the bottom right, there is a pagination control showing 'Rows per page: 10' and '1-4 of 4'.

Alle Schema eines Kataloges anzeigen

```
SHOW SCHEMAS FROM sqlserver;
```

The screenshot shows the SQL Server Enterprise Manager interface. At the top, there is a 'SQL' tab and a 'Session Properties' button. Below that, there are dropdown menus for 'Catalog' and 'Schema', and a green 'Run' button. The query 'SHOW SCHEMAS FROM sqlserver;' is entered in the text area. The 'Results' pane shows the following output:

Schema
db_accessadmin
db_backupoperator
db_datareader
db_datawriter
db_ddladmin
db_denydatareader
db_denydatawriter
db_owner
db_securityadmin
dbo

At the bottom right, there is a pagination control showing 'Rows per page: 10' and '1-10 of 18'.

Alle Tabellen eines Schemas anzeigen

```
SHOW TABLES FROM sqlserver.person;
```

SQL Session Properties

Catalog Schema Run

```
SHOW TABLES FROM sqlserver.person
```

Results

20240724_180717_00390_t9nfc

Table

- address
- addresstype
- businessentity
- businessentityaddress
- businessentitycontact
- contacttype
- countryregion
- emailaddress
- password
- person

Rows per page: 10 1-10 of 15

Weitere Statements

Für weitere interessante Statements am besten in die Dokumentation schauen: [SQL Statement Syntax](#)

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 a session ID '20240724_182247_00409_t9nfc'. The results area is currently empty, with a 'result' label and a dark box. 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 is highlighted. The main area contains the following SQL code:

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

The 'Results' section shows a 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, there are navigation controls: '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.