ClickHouse Column Store

FrOSCon 21. 8. 2016, St. Augustin

Oli Sennhauser

Senior MySQL Consultant, FromDual GmbH

oli.sennhauser@fromdual.com

Über FromDual GmbH

www.fromdual.com

Support

Beratung

remote-DBA

Schulung

Inhalt

ClickHouse Column Store

- Womit fing alles an?
- ClickHouse
- Etwas Marketing
- Performance Vergleiche
- Das richtige Tool?
- Column Store vs. Row Store
- Welche Abfragen
- Star Model
- Installation
- Daten abfragen und einfügen
- OLTP, ETL, OLAP
- Tabellen Typen
- PoC
- Mikro-Benchmarks
- ClickHouser Cluster

Spezialist vs. Generalist

In my opinion the best way to organize data management is to run a specialized OLTP engine on current data. Then, send transaction history data, perhaps including an ETL component, to a companion data warehouse (DWH).

Specialised transactional data management systems are a factor of 50 or so faster than legacy RDBMSs on the transaction piece, while column stores, are a similar amount faster on historical analytics.

In other words, specialization allows each component to run dramatically faster than a 'one size fits all' solution.

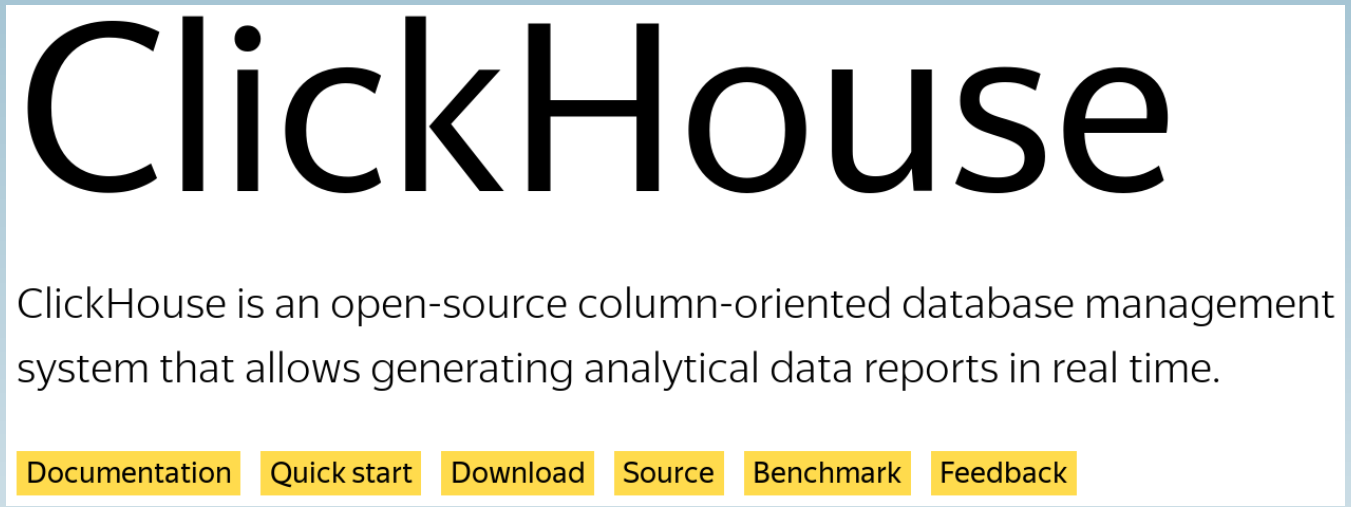
Michael Ralph Stonebraker (*1943), computer scientist specialized in database research.

Womit fing alles an?

- Wie bin ich überhaupt drauf gekommen?
- Twitter: Jonathan Levin am 15. Juni 2016

- Warum interessiert mich das?
  - Wir haben zur Zeit keinen brauchbaren Column Store im MySQL Eco-System! (Infobright, InfiniDB -> MariaDB Column Store)

ClickHouse

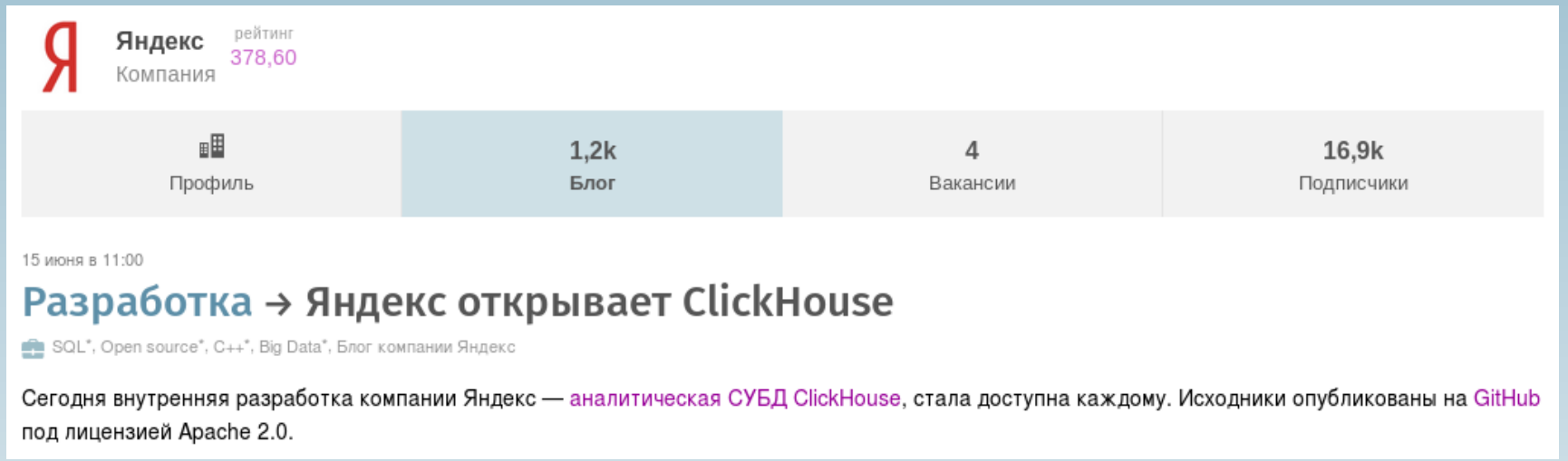


- **Sofort geklickt...**
- **Erfrischend NICHT Marketing lastig...**
- **https://clickhouse.yandex/**

Nachforschungen

- Hadoop Weekly Issue #175, 19 June 2016
- Yandex has open sourced ClickHouse, a columnar analytics database. The system is built for both horizontal and vertical scaling. It supports complex data types (e.g. arrays) and can do approximate queries. The team has also published benchmark results in comparison to several other databases. (June 15)
- https://www.hadoopweekly.com/Hadoop-Weekly-175.html

Original Ankündigung



Яндекс рейтинг 378,60
Компания

Профиль	1,2k Блог	4 Вакансии	16,9k Подписчики

15 июня в 11:00

Разработка → Яндекс открывает ClickHouse

SQL*, Open source*, C++*, Big Data*, Блог компании Яндекс

Сегодня внутренняя разработка компании Яндекс — аналитическая СУБД ClickHouse, стала доступна каждому. Исходники опубликованы на GitHub под лицензией Apache 2.0.

- **Heute (15. Juni 2016) hat das Unternehmens Yandex seine analytische Datenbank ClickHouse, frei zugänglich gemacht. Die Quellen wurden auf GitHub unter der Apache-2.0-Lizenz veröffentlicht.**
- **https://habrahabr.ru/company/yandex/blog/303282/**
- **https://github.com/yandex/ClickHouse**
- **Es war nicht ganz einfach an die ursprüngliche Quelle zu gelangen! (zu neu? Filter Bubble, oder schlimmer?)**

Etwas Marketing

- Verwaltet extrem grosse Volumen an Daten (Big Data?)
- Die weltweit zweitgrösste Web Analytics Platform läuft darauf.
- Mit über 13 Billionen Datenbank-Records (was ist eine Billion? (10^{12}))
- Über 20 Milliarden Events pro Tag (20×10^9)
- Erfolgreich eingeführt bei CERN's LHCb Experiment (10 Mia Events mit über 1000 Attributen (= Spalten) pro Event)
- Cluster bestehend aus 394 Knoten in 6 Datazentern
- Bis zu 2 Billion Rows pro Knoten
- Bis zu 100 Tbyte pro Knoten
- Arbeitet 100 – 1000 mal schneller als herkömmliche Ansätze
- Verwaltet Petabytes (1000 Tbyte) von Daten

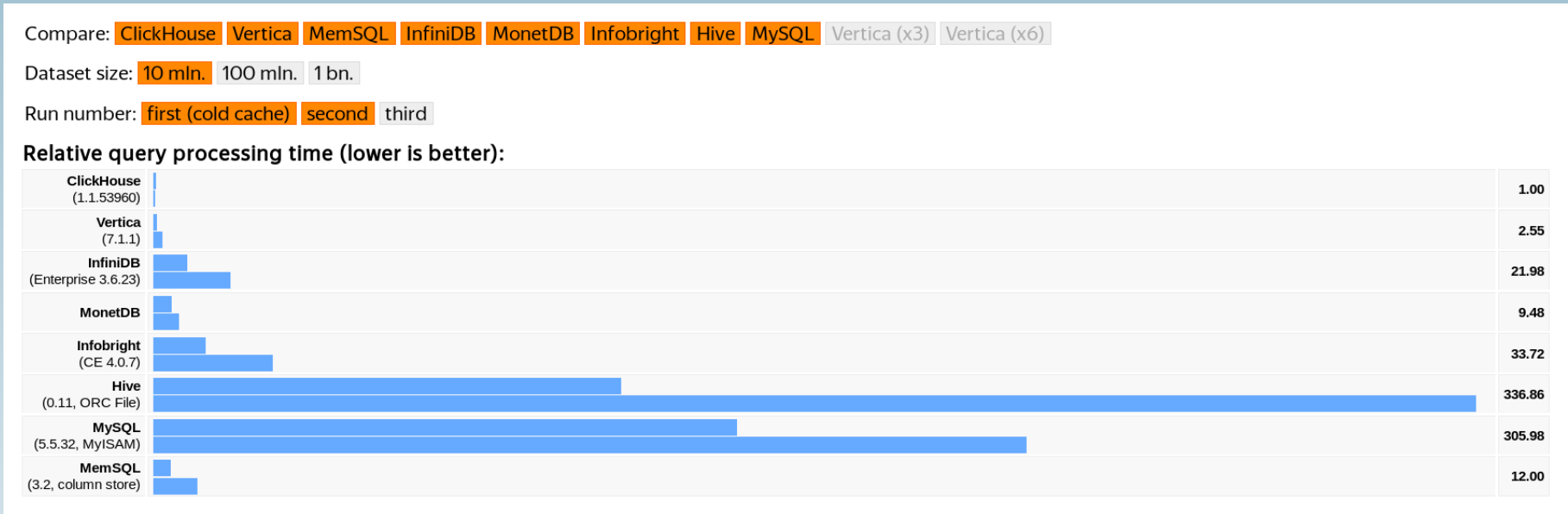
Funktionalität

- SQL Support
- Horizontal und vertikal linear skalierbar
- Performt auf 100-Knoten Cluster sowie auf Einzelservern (und sogar VM's)
- Faktor 100 - 1000 x schneller als traditionelle Zeilen orientierte Systeme
- Minimalisiert Datentransfer
- Schnell: Bis zu 2 Tbyte/s!
- Spalten orientiertes RDBMS (Column Store)
- Vektorisierte Abfrageausführung (Vectore Engine, muss mal gucken, was das ist!)
- Daten-Komprimierung
- Parallele verteilte Abfrageausführung
- Echtzeit Laden und Abfragen der Daten
- Hochverfügbarkeit durch Datenreplikation
- Lokale und verteilte Joins
- Verteile Leseanfragen werden automatisch balanciert.

Anwendungsfälle

- Web- und App-Datenanalyse
- Werbe-Netzwerke und Real-Time-Bidding (RTB)
- Telekommunikation
- E-Commerce
- Information Security Analysen
- Monitoring und Telemetrie (Zeitreihen)
- Business Intelligence (BI)
- Online Spiele
- Internet of Things (IoT)

Performance Vergleiche

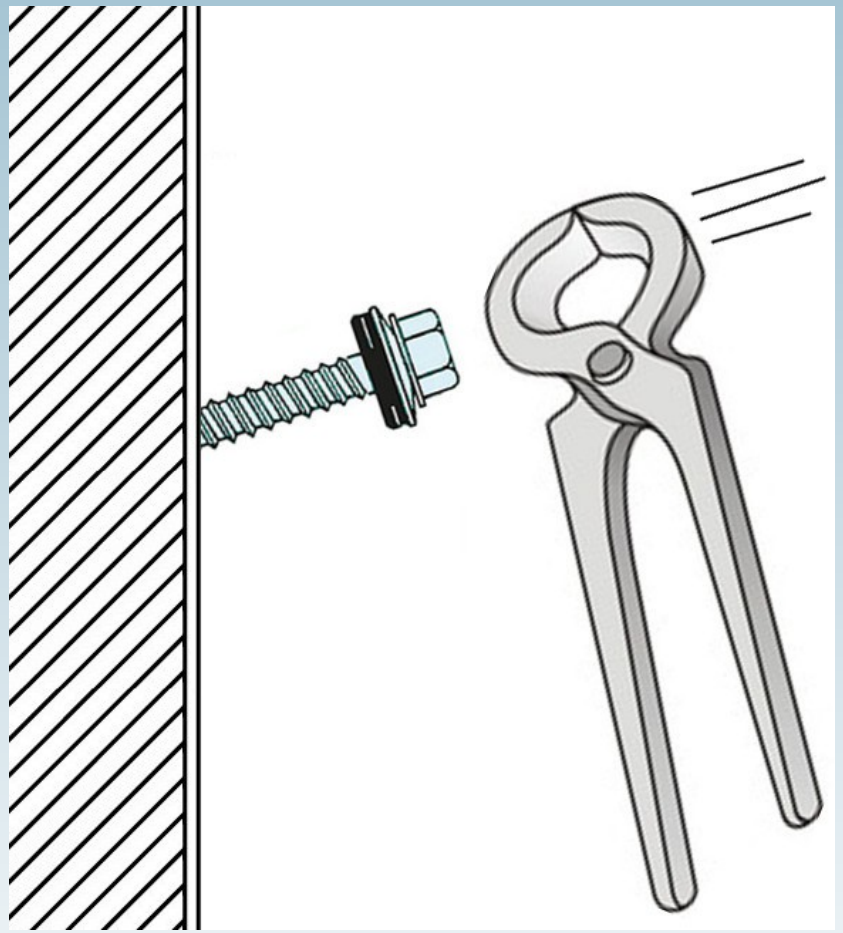


Kleinere Werte sind besser!

- **https://clickhouse.yandex/benchmark.html**

Appetit geweckt?

Das richtige Tool?



- **Das richtige Werkzeug für die entsprechende Aufgabe!**

Wann braucht man so was?

- **Strukturierte Daten!**

ClickHouse
Column Store

Konventionelles
RDBMS

Column Store vs. Row Store

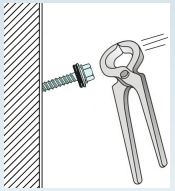
- **SAP HANA Tutorial:**
  http://saphanatutorial.com/column-data-storage-and-row-data-storage-sap-hana/

Welche Abfragen?


```
SELECT sum(AdvEngineID), count(), avg(ResolutionWidth)
  FROM hits

SELECT AdvEngineID, count() FROM hits WHERE AdvEngineID != 0
 GROUP BY AdvEngineID ORDER BY count() DESC LIMIT 10

SELECT UserID FROM hits WHERE UserID = 12345678901234567890

SELECT CounterID, avg(length(URL)) AS l, count() AS c
  FROM hits WHERE URL != '' GROUP BY CounterID
HAVING c > 100000 ORDER BY l DESC LIMIT 25

SELECT Title, count() AS PageViews FROM hits
 WHERE CounterID = 34
   AND EventDate >= toDate('2013-07-01')
   AND EventDate <= toDate('2013-07-31')
   AND NOT DontCountHits AND NOT Refresh AND notEmpty(Title)
 GROUP BY Title ORDER BY PageViews DESC LIMIT 10
```

Böse

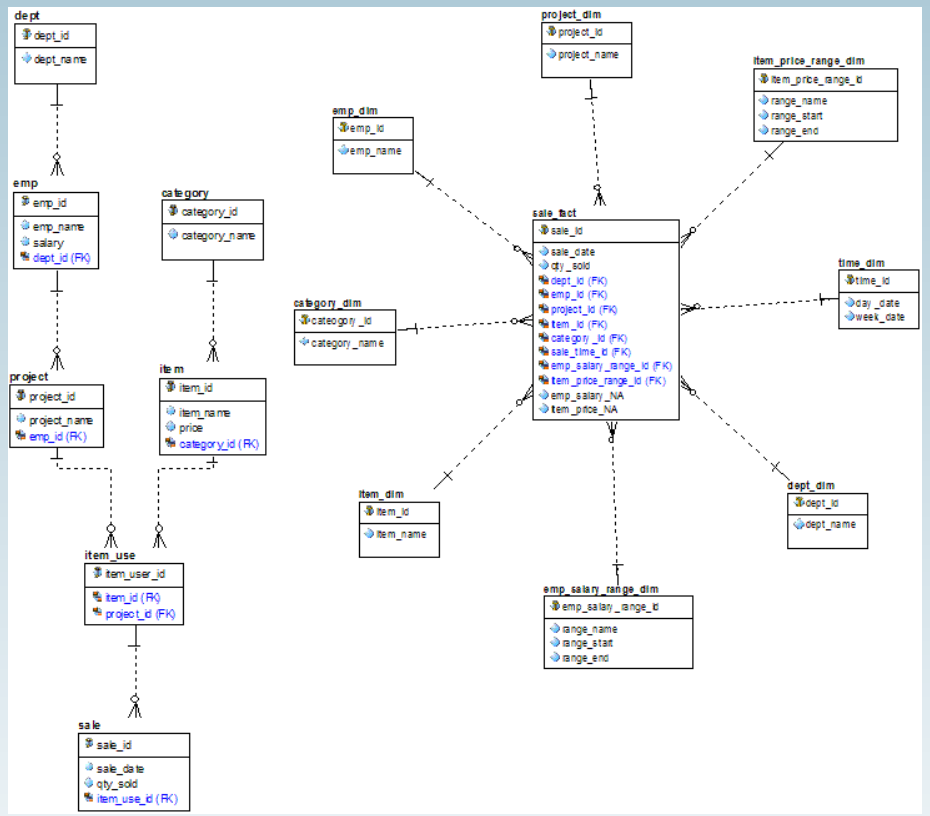

```
SELECT *
FROM table
WHERE ...
```



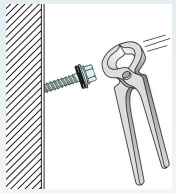
ClickHouse ist ein Column Store NICHT ein Row Store!

Relation vs. Star Model

- **`JOINs` sind teuer, daher neues Datenmodel:**



- **http://www.orafaq.com**



Allgemeines Vorgehen

- Design DWH
  - Ziele (KPI) definieren
  - Datenmodell erstellen
  - Datenquellen eruieren
  - ETL Prozess festlegen

- Praxis: Wir sammeln erst mal und gucken dann wozu man die Daten brauchen kann... :-(

Download und Installation

- Anforderungen: Ubuntu, x86_64 mit SSE 4.2
  - Pakete für Ubuntu 12.04, 14.04, 16.04
  - `cat /proc/cpuinfo | grep --color sse4_2`
  - RedHat Pakete sind im Bau...

```
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv E0C56BD4

sudo mkdir -p /etc/apt/sources.list.d
echo "deb http://repo.yandex.ru/clickhouse/trusty stable main" |
  sudo tee /etc/apt/sources.list.d/clickhouse.list
sudo apt-get update

sudo apt-get install clickhouse-server-common clickhouse-client

sudo service clickhouse-server start
clickhouse-client
```

Interfaces und Server

- `clickhouse-server` lauscht auf
  - Port 9000 (nativ)
  - Port 9009 (Komm für Server Replicas) und
  - Port 8123 (http)
- Interfaces:
  - HTTP (Port 8123), nutzbar von allen Plattformen
  - Natives Interface (TCP) C++-API (Port 9000)
  - Command-line Client (`clickhouse-client`)

Daten abfragen über http

- http (GET, read-only)

```
curl 'http://127.0.0.1:8123/?database=foodmart&query=sel
ect%20sum(store_sales)%20from%20sales_fact'

wget -O- -q  'http://127.0.0.1:8123/?database=food
mart&query=select sum(store_sales) from sales_fact'

GET 'http://127.0.0.1:8123/?database=foodmart&query=se
lect sum(store_sales) from sales_fact'

echo "select sum(store_sales) from sales_fact" | \
curl 'http://localhost:8123/?database=foodmart' \
--data-binary @-
```

- JDBC/ODBC?

Daten einfügen über http

- http (POST)

```
echo 'CREATE TABLE t (a UInt8) ENGINE = Me
mory' | POST 'http://localhost:8123/'

echo 'INSERT INTO t VALUES (1),(2),(3)' | \
POST 'http://localhost:8123/'

echo '(4),(5),(6)' | POST 'http://local
host:8123/?query=INSERT INTO t VALUES'

echo -ne '10\n11\n12\n' | POST 'http://local
host:8123/?query=INSERT INTO t FORMAT TabSeparated'
```

- JDBC/ODBC?

Befehle

- Command-line Client (CLI)
- Ähnlich wie MySQL

```
shell> clickhouse-client

client> SHOW DATABASES
clinet> SHOW TABLES
clinet> SHOW PROCESSLIST
clinet> SHOW CREATE TABLE
clinet> DESCRIBE TABLE
clinet> EXISTS TABLE
clinet> USE db
clinet> SET param = value
clinet> OPTIMIZE TABLE
```

Daten abfragen über CLI


```
:) SELECT sum(store_sales) FROM sales_fact

SELECT sum(store_sales)
FROM sales_fact

┌─sum(store_sales)─┐
│ 11160893.72000474 │
└──────────────────┘

1 rows in set. Elapsed: 0.016 sec.
Processed 1.08 million rows, 8.63 MB
(68.67 million rows/s., 549.40 MB/s.)

shell> clickhouse-client --database=foodmart \
--query="SELECT sum(store_sales) FROM sales_fact"
```

Daten laden über CLI


```
echo -ne "1, 'some text', '2016-08-14 00:00:00'
2, 'some more text', '2016-08-14 00:00:01'" | \
clickhouse-client --database=test \
--query="INSERT INTO test FORMAT CSV";

cat <<_EOF | \
clickhouse-client --database=test \
--query="INSERT INTO test FORMAT CSV";
3, 'some text', '2016-08-14 00:00:00'
4, 'some more text', '2016-08-14 00:00:01'
_EOF
```

OLTP -> ETL -> OLAP

- Transaktions-System
- Analyse-System

Extract – Transform – Load

ETL

- Struktur: Datentyp-Namen sind etwas anders als MySQL
  - `UInt64` vs. `BIGINT UNSIGNED`
  - `Float64` vs. `DOUBLE`
  - `String` vs. `VARCHAR`
- https://shinguz.ch/blog/clickhouse-data-types-compared-to-mysql-data-types/
- ClickHouse ist Case sensitiv(er) als MySQL
- Neue Storage Engines (Storage Types)
- Laden von Daten
  - `clickhouse-client` oder http `POST`
- JDBC (auf Github)
- ODBC
  - Noch experimentell
- Talend ETL, Pentaho Kettle, etc.
  - http://www.datasciencecentral.com/profiles/blogs/10-open-source-etl-tools

Tabellen Engine (Typ)

- TinyLog
  - Disk, write-once/read-many, kleine Tabellen bis 1M Rows.
- Log
  - Disk, temp. Daten, write-once/read-many
- Memory
  - RAM, schnell, < 100M Rows, temporäre Daten
- Merge
  - ro, speichert keine Daten, autom. Parallelisieren, mit TinyLog
- Distributed
  - Verteilen auf mehrere Server, speichert keine Daten, parallelisiert
- MergeTree
  - Index auf PK und Date, Vortschrittlichste Engine, parallelisiert, Replikation, grosse Tab
- CollapsingMergeTree
  - MergeTree mit autom. Delete/Collapse
- SummingMergeTree
  - MergeTree mit autom. Sum/Total, Nicht sinnvoll
- AggregatingMergeTree
  - MergeTree mit Aggregation, meinst nicht sinnvoll
- Null
  - Speichert nichts, liefert nichts
- View
  - Für VIEWs
- MaterializedView
  - Für Materialized Views
- Set
  - Pinned im RAM, gespeichert auf Disk für `IN (...)`
- Join
  - Wie Type Set für JOIN
- Buffer
  - Als Cache für Spikes abzufangen

ClickHouse Tabelle


```
:) desc sales_fact;

DESCRIBE TABLE sales_fact

┌─name─────────┬─type────┬─default_type─┬─default_expression─┐
│ product_id   │ Int32   │              │                    │
│ time_id      │ Int32   │              │                    │
│ customer_id  │ Int32   │              │                    │
│ promotion_id │ Int32   │              │                    │
│ store_id     │ Int32   │              │                    │
│ store_sales  │ Float64 │              │                    │
│ store_cost   │ Float64 │              │                    │
│ unit_sales   │ Int32   │              │                    │
└──────────────┴─────────┴──────────────┴────────────────────┘

:) SHOW CREATE TABLE sales_fact

statement: CREATE TABLE foodmart.sales_fact ( product_id Int32
,  time_id Int32,  customer_id Int32,  promotion_id Int32
,  store_id Int32,  store_sales Float64,  store_cost Float64
,  unit_sales Int32) ENGINE = TinyLog
```

PoC Struktur

- MySQL `foodmart` -> ClickHouse

```
mysqldump  --user=root --no-data \
--skip-add-locks --skip-add-drop-table \
foodmart |
grep -v 'some magic here' |
> mysql_structure_dump_foodmart.sql

export DATABASE=foodmart
cat clickhouse_structure_dump_foodmart.sql | \
grep -v ^$ |
while read i ; do
  echo "Creating table ..."
  echo $i | POST "http://localhost:8123/?data
base=$DATABASE"
done
```

PoC Daten


```
mkdir /tmp/foodmart_dump
mysqldump --user=root \
--tab=/tmp/foodmart_dump  foodmart
rm -f /tmp/foodmart_dump/*.sql

export DATABASE=foodmart
for i in `ls *.txt`; do
  t=$(basename $i .txt)
  cat $i | POST "http://local\
host:8123/?database=$DATABASE&que
ry=INSERT INTO $t FORMAT TabSeparated"
done
```

Problem: `NULL`, `\n\r`, etc.

Mikro-Benchmark I

- ClickHouse (TinyLog)
- 1 Mio rows
- SELECT COUNT(*) FROM sales_fact;
- 240 ms / 10 ms
- f = 27 mal

- MySQL 5.7 (InnoDB)
- 1 Mio rows
- SELECT COUNT(*) FROM sales_fact;
- 710 ms / 270 ms

Mikro-Benchmark II

MySQL

```
SELECT SUM(store_sales) FROM sales_fact as s
  JOIN time_by_day AS t ON s.time_id = t.time_id
 WHERE t.the_date >= '2009-01-15' AND t.the_date <= '2011-12-15';
```

1990 ms

ClickHouse

Expected UNION

```
SELECT sum(store_sales)
FROM
( SELECT store_sales, time_id FROM sales_fact )
ANY INNER JOIN
( SELECT time_id FROM time_by_day
  WHERE the_date >= '2009-01-15' AND the_date <= '2011-12-15'
) AS t USING (time_id)
```

31 ms (f = 64 mal)

ClickHouse Funktionen

- **Mahr als 300 Funktionen**
- **Strong Typing (keine implizite Konversion)**

- Arithmetische (9)
- Bit (6)
- Vergleich (6)
- Logische (4)
- Typen Konvertierungs (28)
- Datum/Zeit (26)
- String (14)
- String Suche (9)
- String Suchen und Ersetzen (4)
- Array (34)
- Splitten und Mergen von Strings und Arrays (3)
- URL (21)
- IP-Adressen (5)
- Zufallszahlen (2)
- Hash (11)
- Encoding (4)
- Runden (6)
- Mathematische (21)
- Yandex.Metrica Dictionary (16)
- Externe Dictionary (15)
- JSON (7)
- `IN()` (8)
- Andere (13)
- Aggregate Funktionen (36)

System Tabellen


```
:) use system
:) show tables

┌─name──────────────┐
│ clusters          │ → Info über Cluster
│ columns           │ → Spalten in Tabellen
│ databases         │ → Datenbanken
│ dictionaries      │ → Externe Dictionaries
│ events            │ → für Monitoring und Profiling (Memory, IO, etc.)
│ functions         │ → Liste der 372 Funktionen
│ merges            │ → für Monitoring von MergeTree Tabellen
│ metrics           │ → für Monitoring (Memory, IO, etc.)
│ numbers           │ → Zahlen 1 .. oo
│ numbers_mt        │ → Zahlen parallelisiert
│ one               │ → SELECT function() FROM one (dual!)
│ parts             │ → Infos über Teile der MergeTree Tabellen
│ processes         │ → SHOW PROCESSLIST
│ replicas          │ → Infos über Replicas auf lokalem Server
│ replication_queue │ → ??? Doku Bug geöffnet
│ settings          │ → Aktuelle Systemeinstellungen
│ tables            │ → Datenbank, Tabellen und Engines
└───────────────────┘
```

Replizieren und Sharden

- Paralleles Prozessing
  - Query Load Balancing
- Abfragen parallelisieren
- High-Availability

--> ClickHouse Cluster

ClickHouse Cluster

- Zookeper 3.4.5 oder neuer
  - Ubuntu 14.04 ist OK
- `apt-get install zookeeperd`
  - https://zookeeper.apache.org/
  - `echo 'srvr' | nc 127.0.0.1 2181`
- `cat /etc/clickhouse-server/config.xml`
- `cat /etc/zookeeper/conf/zoo.cfg`

ClickHouse KANNNICHT

- Transaktionen (warum?)
- Kein Support für Datenmodifikation: `UPDATE, DELETE, REPLACE, MERGE, UPSERT, INSERT UPDATE.`
  `--> ALTER TABLE ... DROP PARTITION;`
- Set für Aggregation muss in RAM eines Knotens passen.
- Kein `DECIMAL` (€€€) Datentyp -> `Float` oder `Uint/100`
- `NULL`
- „normal“ SQL Joins --> Rewrite the Query!

Q & A

Fragen ?

Diskussion?

Wir haben Zeit für ein persönliches Gespräch...

- **FromDual bietet neutral und unabhängig:**
  - **Beratung**
  - **Remote-DBA**
  - **Support für MySQL, Galera und MariaDB**
  - **Schulung**

www.fromdual.com/presentations