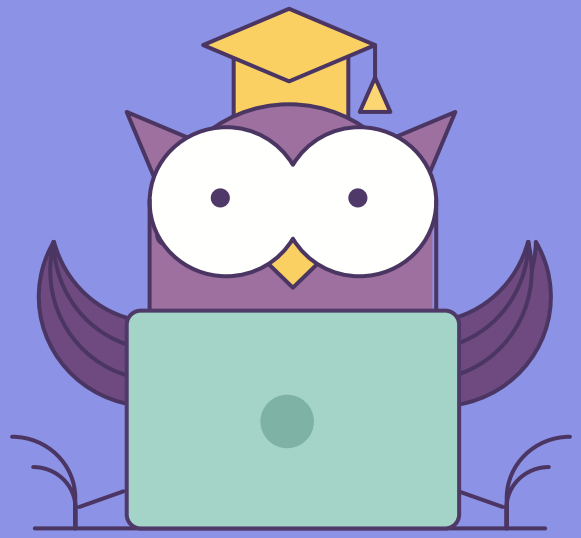
ОНЛАЙН-ОБРАЗОВАНИЕ

# Меня хорошо слышно && видно?



Напишите в чат, если есть проблемы!

Ставьте + если все хорошо

OTUS

# MPP DB

Большие базы для больших данных

Егор Матешук

# Цели сегодняшнего занятия

OTUS

После этого занятия вы будете знать

- Что такое MPP DB
- В каких случаях они нужны, в каких нет
- Что общего у разных MPP DB
- Как подключаться к DB Vertica
- Как создавать таблицы и проекции
- Как выполнять запросы к DB Vertica и анализировать их производительность

# 01

# Что такое MPP DB

Напишите, пожалуйста, в чат, какие базы используются в аналитике в вашей компании или те, с которыми вы сталкивались в каких-либо проектах

## Неофициальное определение

**MPP (massively parallel processing) database** - класс СУБД, распределяющих хранимые данные на множество вычислительных узлов для параллельной обработки больших объемов данных. Каждый узел имеет собственное хранилище и вычислительные ресурсы, позволяющие выполнять часть общего запроса к базе.

# Для чего они нужны

MPP DB хорошо подходят в качестве аналитического хранилища данных благодаря тому что

- масштабируются горизонтально
- позволяют выполнять тяжелые запросы
- позволяют загружать данные на большей скорости
- не имеют единой точки отказа

# Особенности MPP

Если всё так здорово и мощно, то почему не использовать MPP вместо всех остальных баз?

- Один большой запрос - хорошо, много мелких - плохо
- Распределенная архитектура - накладные расходы
- Только аналитические SQL-запросы (скажи нет OLTP, Key-Value и т д)

# Особенности MPP

Что это значит на практике?

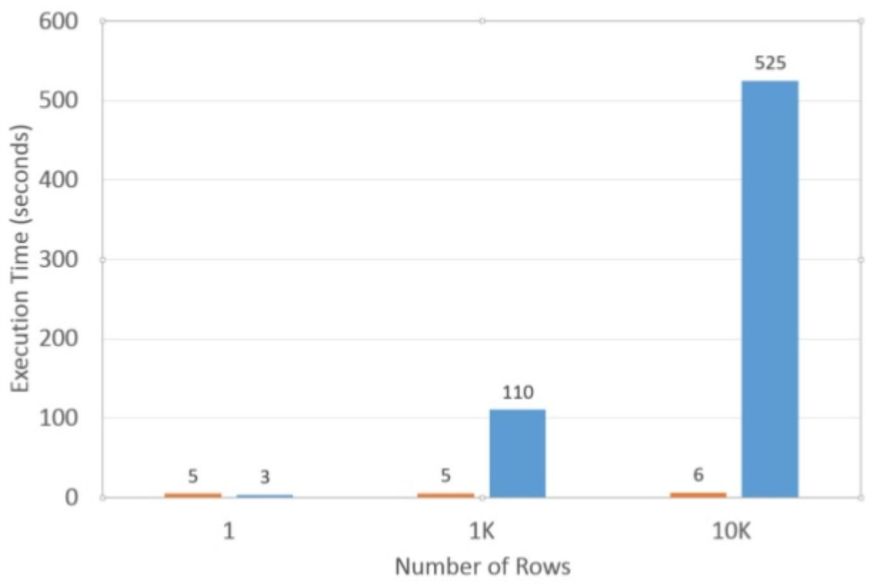
- Быстрая параллельная загрузка данных
- Очень быстрые и дешевые SELECT, INSERT
- Очень медленные и дорогие DELETE, UPDATE
- Ограниченная поддержка индексов и транзакций

# Быстрая загрузка данных

Все ноды равноправны и у вас нет “бутылочного горлышка”

На графике:

сравнение параллельной и обычной загрузки в DB Vertica (оранжевый - параллельная, синяя - JDBC)



# Дешевые SELECT, INSERT

OTUS

Почему быстро выполняются SELECT запросы?

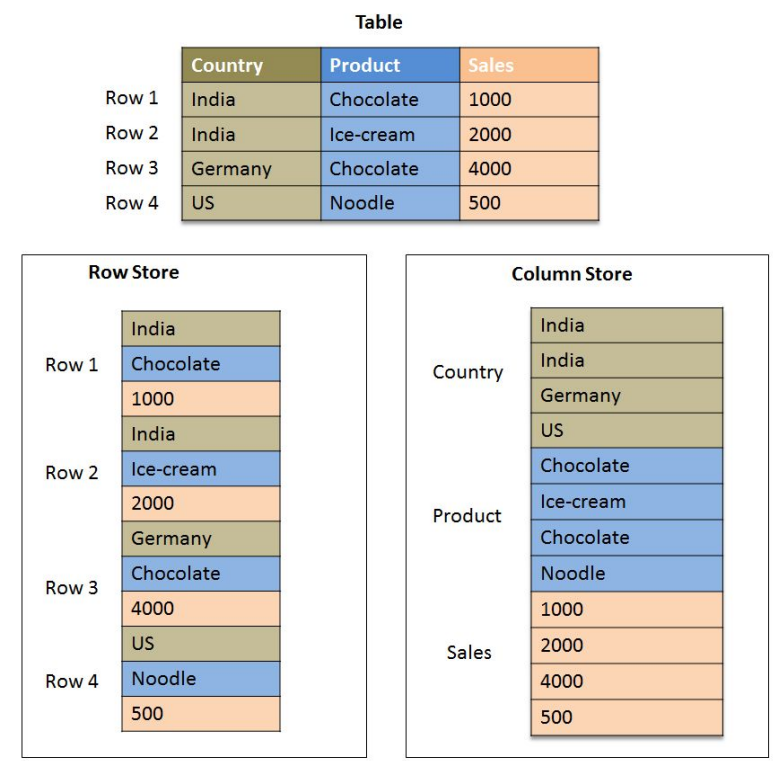
- Параллельное чтение
- Колоночный формат
- Сжатие данных
- Оптимизация данных на этапе записи

# Параллельное чтение

- Каждая таблица физически “размазана” по узлам
- Запрос к базе превращается в несколько параллельных запросов к узлам базы
- ...
- PROFIT x N

# Колоночный формат

- Позволяет читать только нужные колонки
- Увеличивает долю последовательного чтения
- Позволяет применять различные техники сжатия



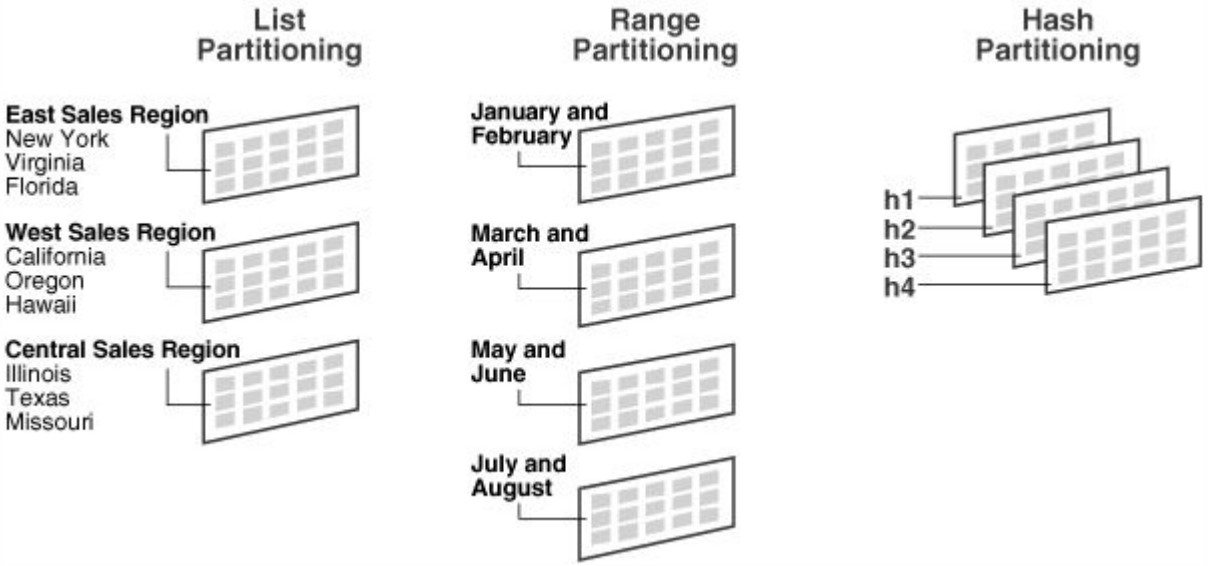
# Сжатие данных

- IO дисков часто становится узким местом в аналитических запросах
- Место тоже не резиновое
- MPP поддерживают сжатие методами (ZLIB, LZ4, LZO, … - конкретный набор зависит от базы)
- Позволяет снизить объем занимаемого места в 3-5 раз

# Оптимизация на этапе записи

Физическое партицирование (набор полей, диапазон дат, выражения)

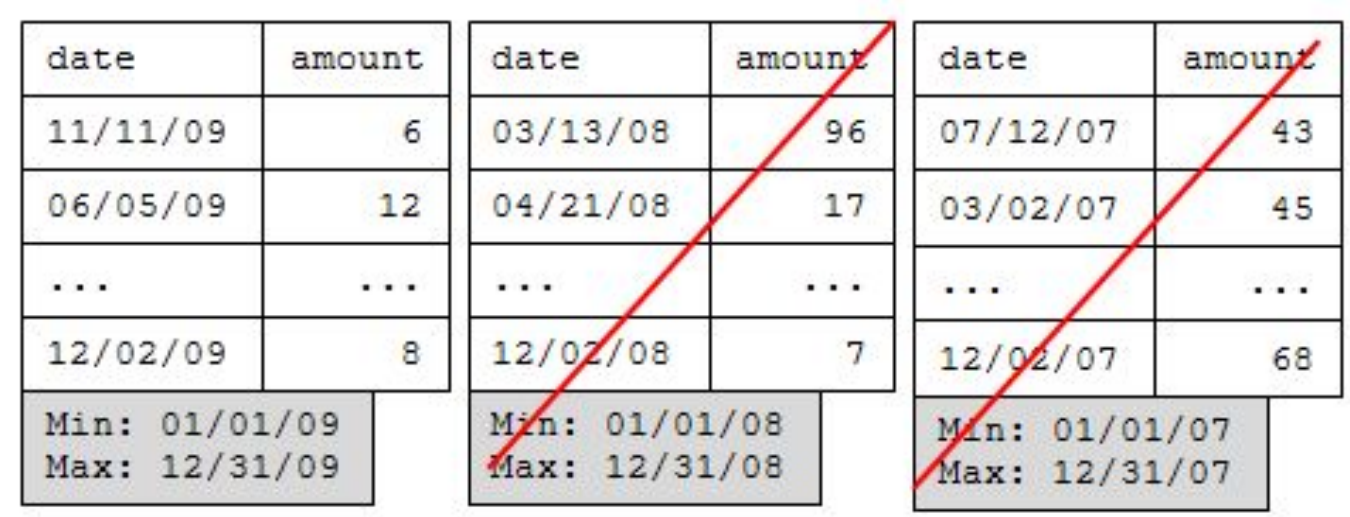
```
CREATE TABLE ...
PARTITION BY EXTRACT(year FROM date);
```



# Оптимизация на этапе записи

Как результат - ускорение чтения с помощью partition pruning

SELECT ... WHERE date = '12-2-2009';



# Дешевые SELECT, INSERT

MPP поддерживают потоковую запись

- Write ahead log
- Write-optimized containers
- Streaming buffers

Поэтому частые одиночные INSERT продолжают быть оптимальными

# Дорогие DELETE, UPDATE

Почему дорого удалять/изменять отдельные записи?

- Эту запись нужно найти
- При удалении/изменении одной строки приходится целиком перезаписывать большой кусок

# Дорогие DELETE, UPDATE

Можно ли с этим что-нибудь сделать?

- Delete vectors
- Удаление партициями
- Compaction

# Индексы и транзакции

- Индексы и транзакции становятся слишком дорогими при таких объемах данных и формате хранения
- Индексы и транзакции требуют синхронизации между нодами
- В результате индексы и транзакции обычно поддерживаются слабо в MPP DB

Напишите, пожалуйста, в чат 1 особенность MPP DB, которая вам запомнилась

# Общие признаки MPP

- Поддержка параллельной загрузки
- Колоночный формат хранения
- Оптимизация данных на этапе записи
- Дорогие DELETE, UPDATE
- Ограниченная поддержка индексов и транзакций

# 02

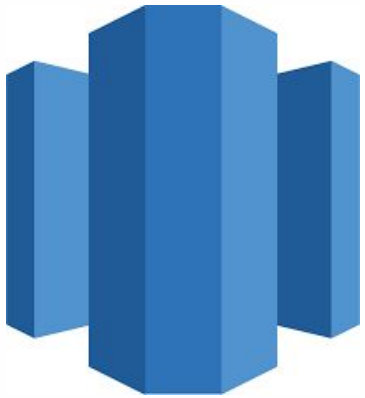
# Существующие решения

# On-premise MPP

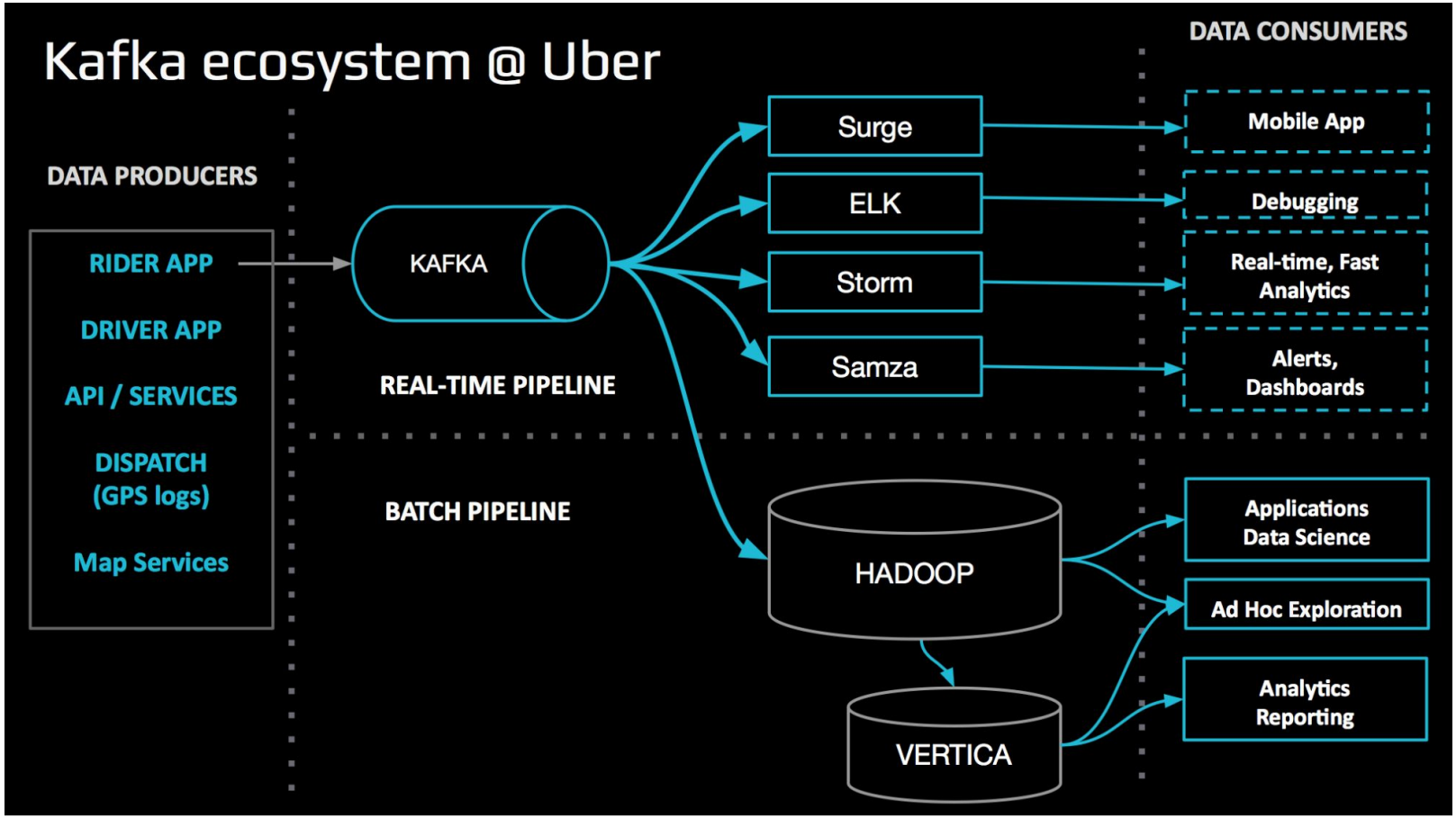
- Vertica
- Greenplum
- Teradata
- Kudu
- ClickHouse



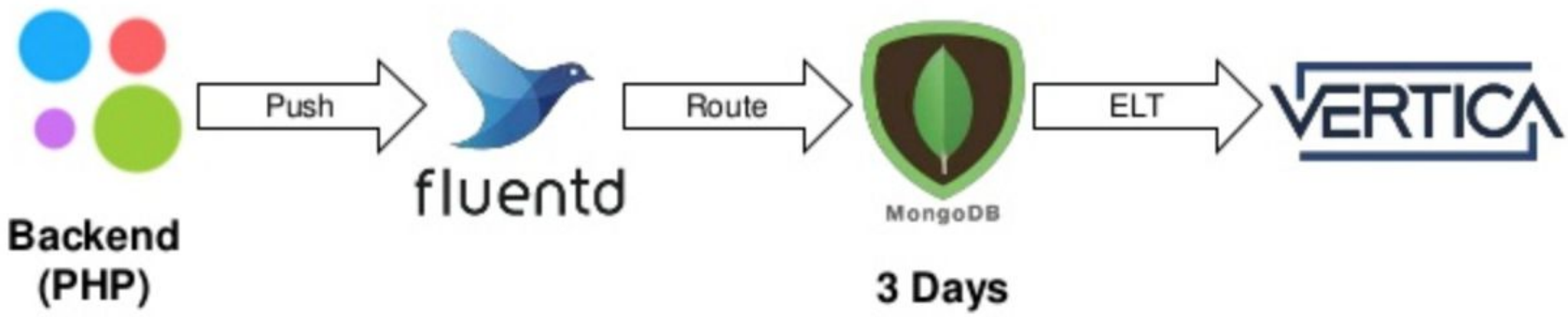
- BigQuery (Google Cloud)
- Redshift (Amazon Cloud)
- SQL Data Warehouse (Azure Cloud)



# Uber


Kafka ecosystem @ Uber
DATA CONSUMERS
DATA PRODUCERS
RIDER APP
DRIVER APP
API / SERVICES
DISPATCH
(GPS logs)
Map Services
KAFKA
REAL-TIME PIPELINE
Surge
ELK
Storm
Samza
Mobile App
Debugging
Real-time, Fast
Analytics
Alerts,
Dashboards
BATCH PIPELINE
HADOOP
VERTICA
Applications
Data Science
Ad Hoc Exploration
Analytics
Reporting

# Архитектура обработки ClickStream в Avito




https://tech.avito.ru/tags/video/anchor-modeling

# 03

# Vertica

# Что такое Vertica

OTUS

База данных

- Аналитическая
- Колоночная
- Распределенная (shared-nothing)
- С поддержкой основных механизмов хранения данных MPP - сжатие, партицирование, кодирование
- Продвинутый оптимизатор запросов

# История Vertica

В начале 2000-х группой ученых была разработана колоночная база C-Store, которая использовала принципиально новый подход к хранению данных.

В 2005 году эти же люди основали компанию Vertica, которая начала развивать одноименную базу в качестве коммерческого форка C-Store.

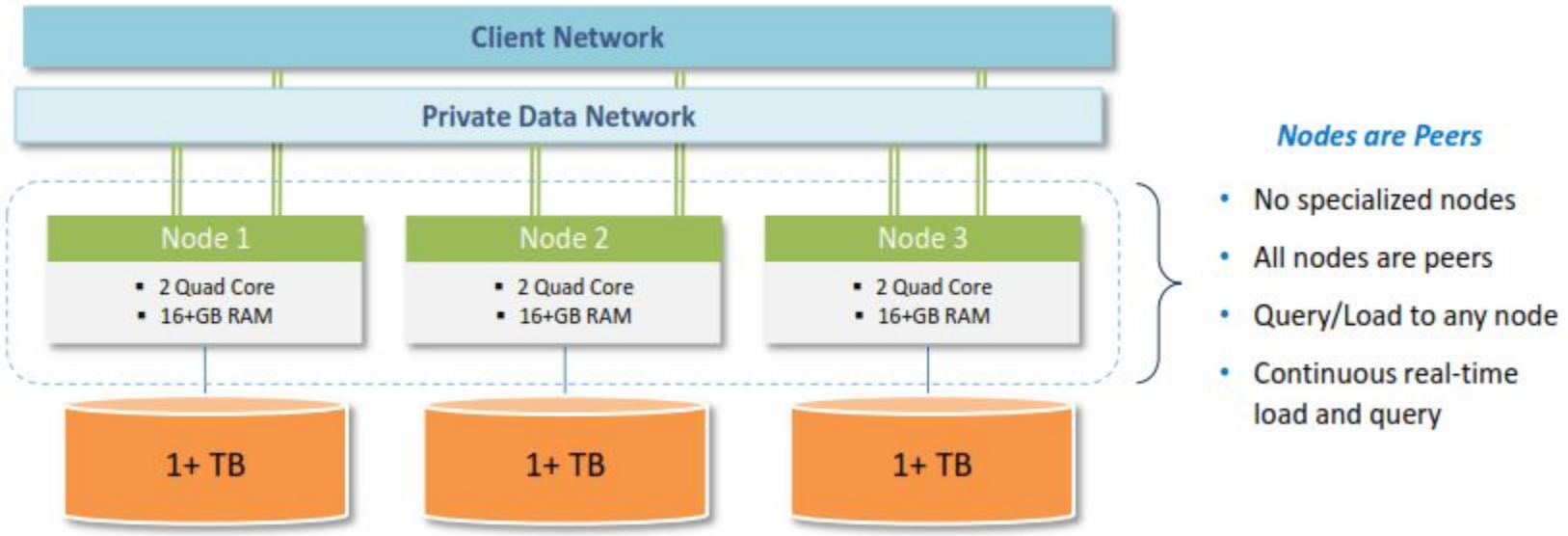
Сейчас Vertica принадлежит компании Micro Focus.

# Архитектура Vertica

Несколько нод, объединенных в общий кластер.

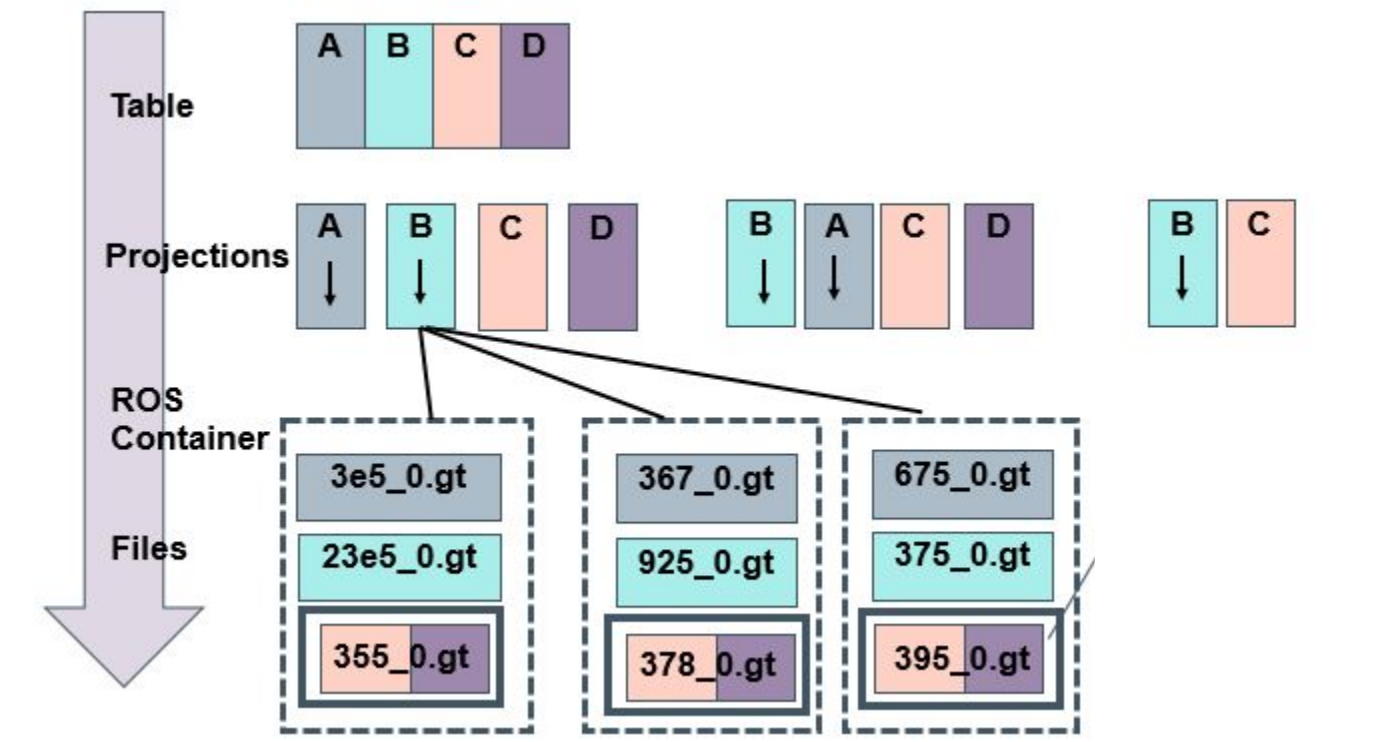
Любая из нод может использоваться для запросов.

Кластер может пережить потерю любой ноды (с учетом репликации).



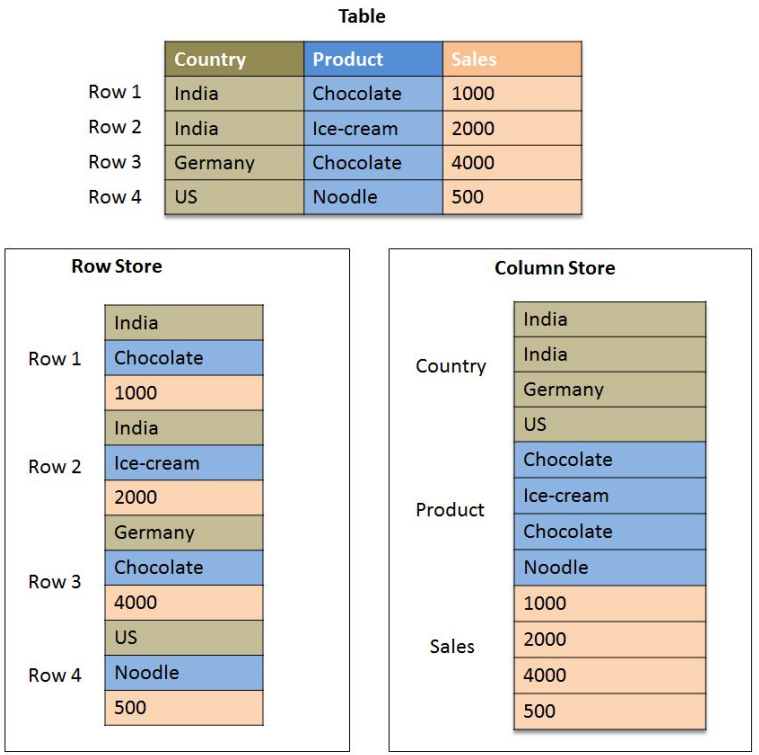
# Иерархия хранения

Таблица - логическое представление, проекции - физическое, состоящее из ROS-контейнеров, соответствующих одному или нескольким файлам.



# Сжатие

- BLOCK_DICT, BLOCKDICT_COMP
- BZIP_COMP, GZIP_COMP, Zstandard Compression
- COMMONDELTA_COMP, DELTARANGE_COMP
- DELTAVAL, GCDDELTA
- RLE

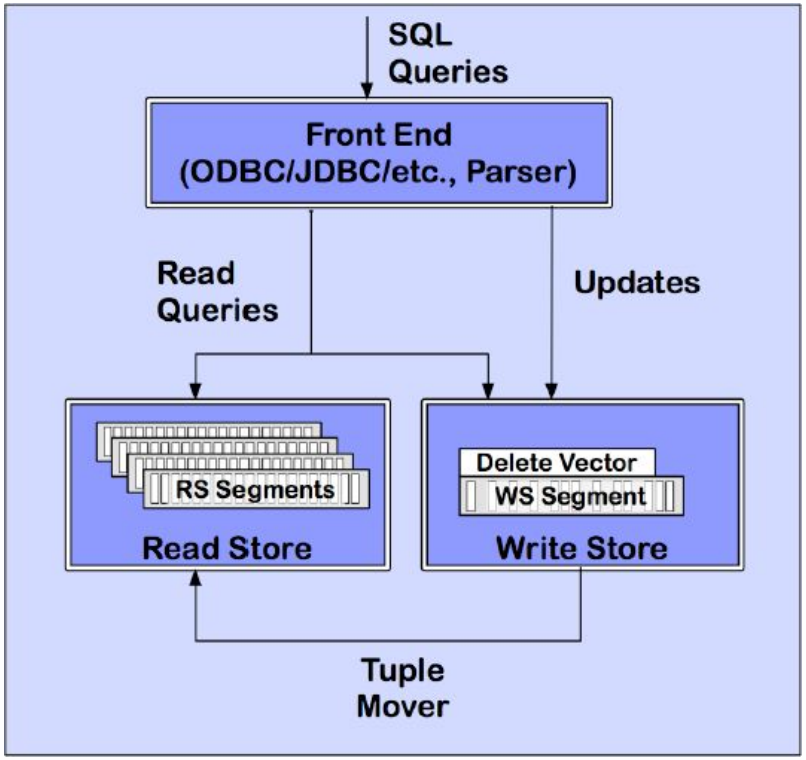


При обычной записи (trickle load) данные загружаются в WOS, которое хранит их в памяти в неоптимизированном виде.

При прямой записи (direct load) данные пишутся сразу в ROS - append-only формат со сжатием, кодированием, сортировкой.

# Tuple Mover

Задача TM - оптимизировать хранилище путем перемещения данных из WOS в ROS (Moveout), склейкой ROS контейнеров (Mergeout) и удалением удаленных записей (O_o)?



# Delete vectors

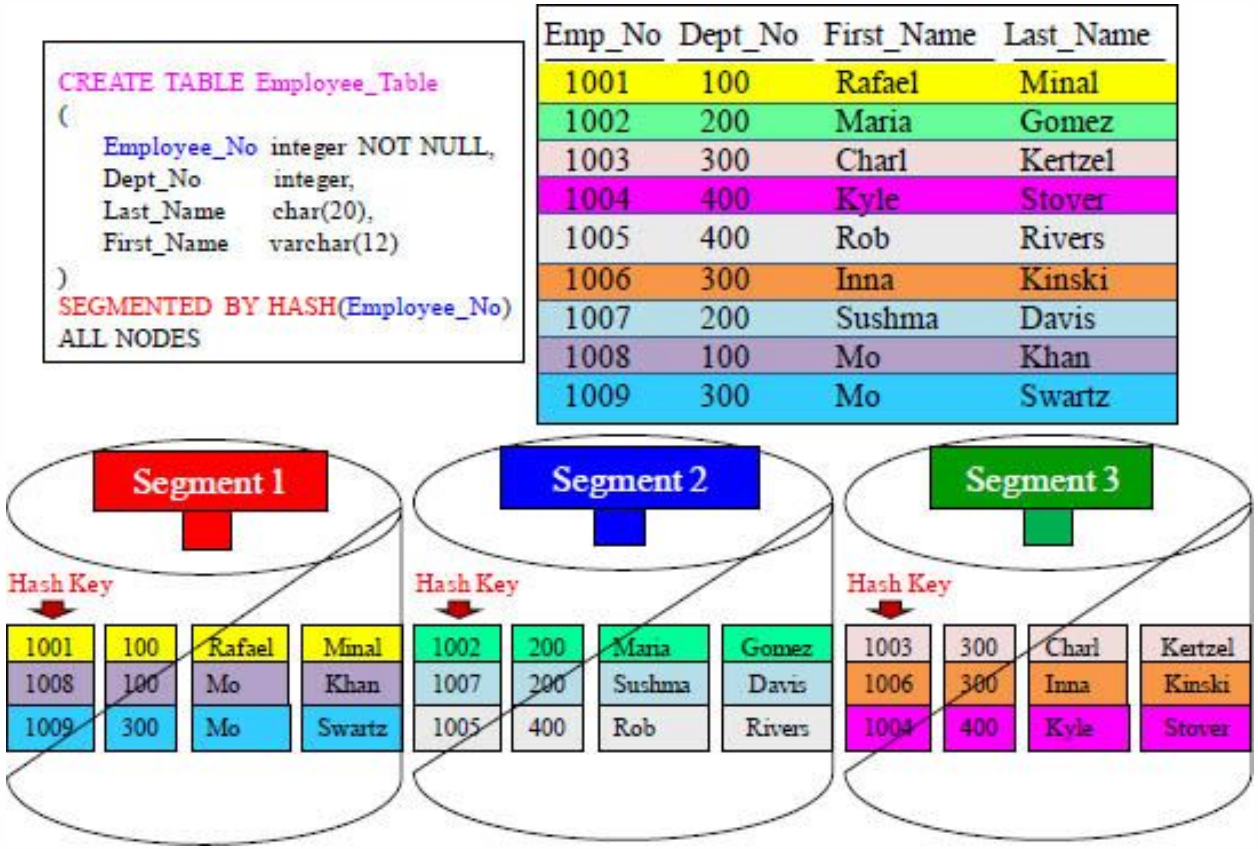
DELETE ничего на самом деле не удаляет!

На самом деле запись просто помечается как “удаленная” и будет действительно удалена только при перезаписи контейнера.

Индексы удаленных записей и называются delete vectors.

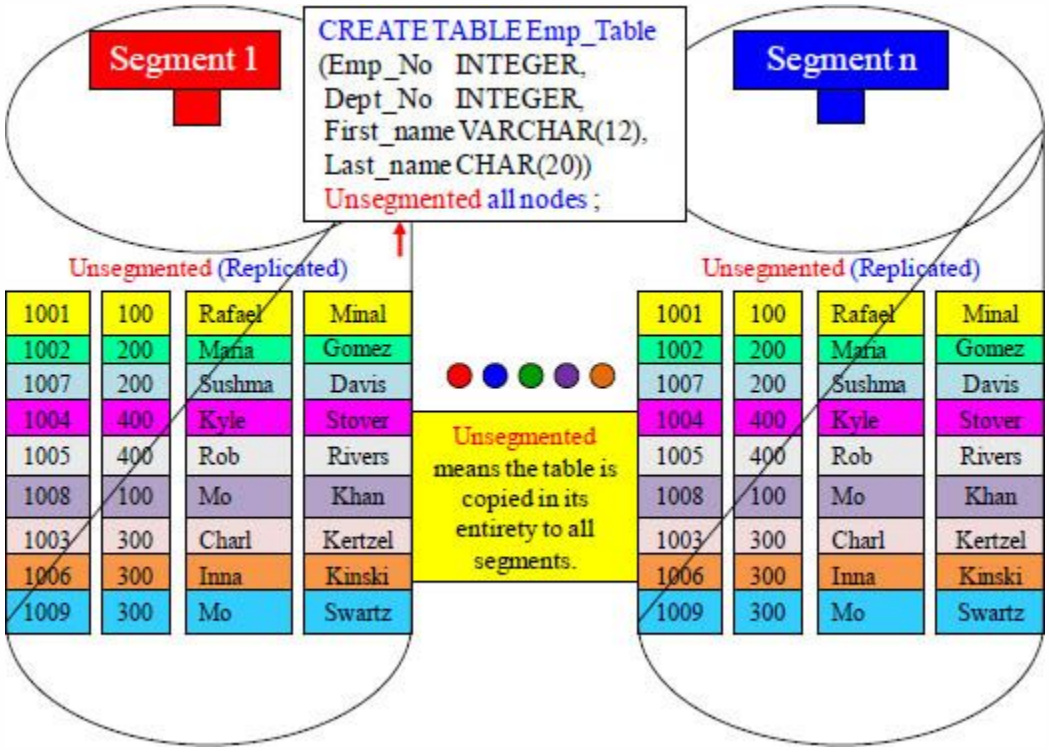
# Сегментация таблиц

Сегментация - способ распределить одну таблицу (а точнее её проекцию) по нескольким узлам ноды.



# Сегментация таблиц

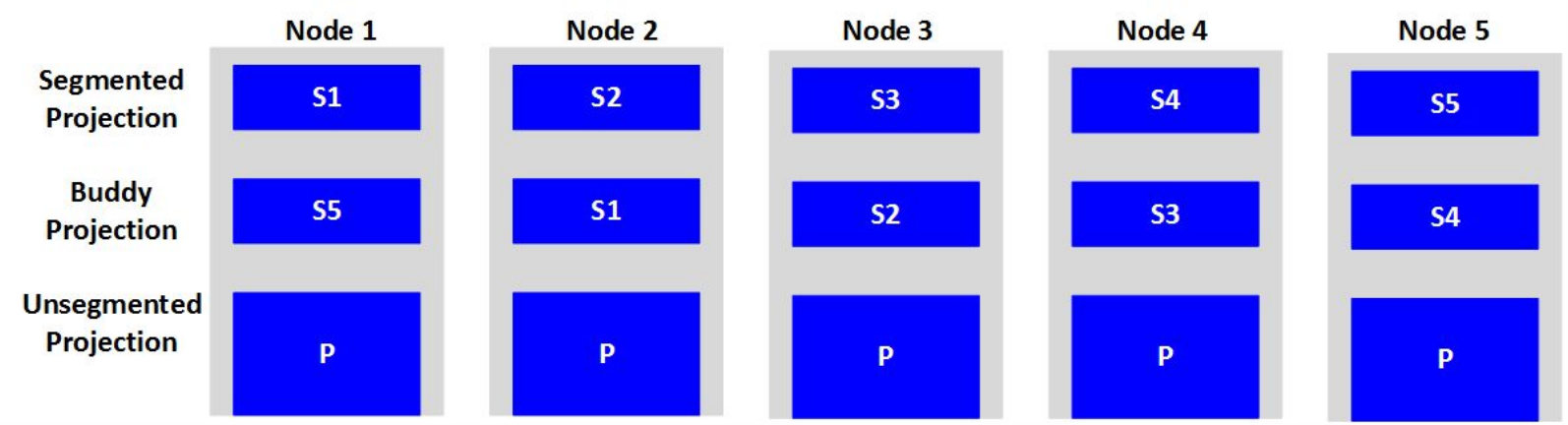
Можно создавать unsegmented таблицы, что полезно для небольших таблиц или справочников



# Репликация (aka K-Safety)

K-Safety - репликация данных на соседние ноды для обеспечения отказоустойчивости.

“K-Safety = 2” значит, что кластер хранит 3 копии данных и может легко пережить потерю 2 нод.



# Проекции

“Проекции” таблиц с разным партицированием.

Как результат - оптимальный доступ к одним и тем же данным при разных запросах.

```
CREATE PROJECTION person_age

AS SELECT * FROM person ORDER BY age;
```

# Проекции

Проекции основываются на таблице, но могут

- содержать подмножество колонок таблицы
- иметь другое сегментирование
- иметь другой порядок сортировки данных

# Оптимизация Join'ов

Есть два типа - Merge Join и Hash Join.

HJ - стандартный вариант, при котором находятся общие для двух таблиц значения хэшей от полей, по которым происходит join.

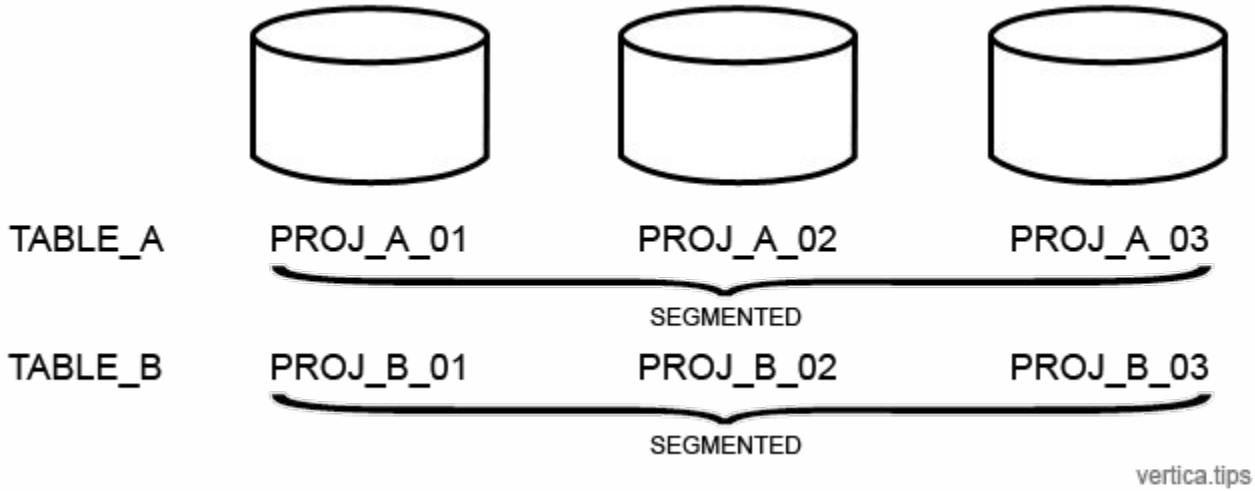
MJ - более быстрый вариант, но требующий совпадения порядков сортировки для двух проекций.

Подробное описание алгоритма MJ.

# Оптимизация Join'ов

Есть еще два типа - Local Join и Network Join.

Если две проекции имеют одинаковые сегменты на всех нодах, то join можно сделать прямо на них, не перекладывая данные между нодами.

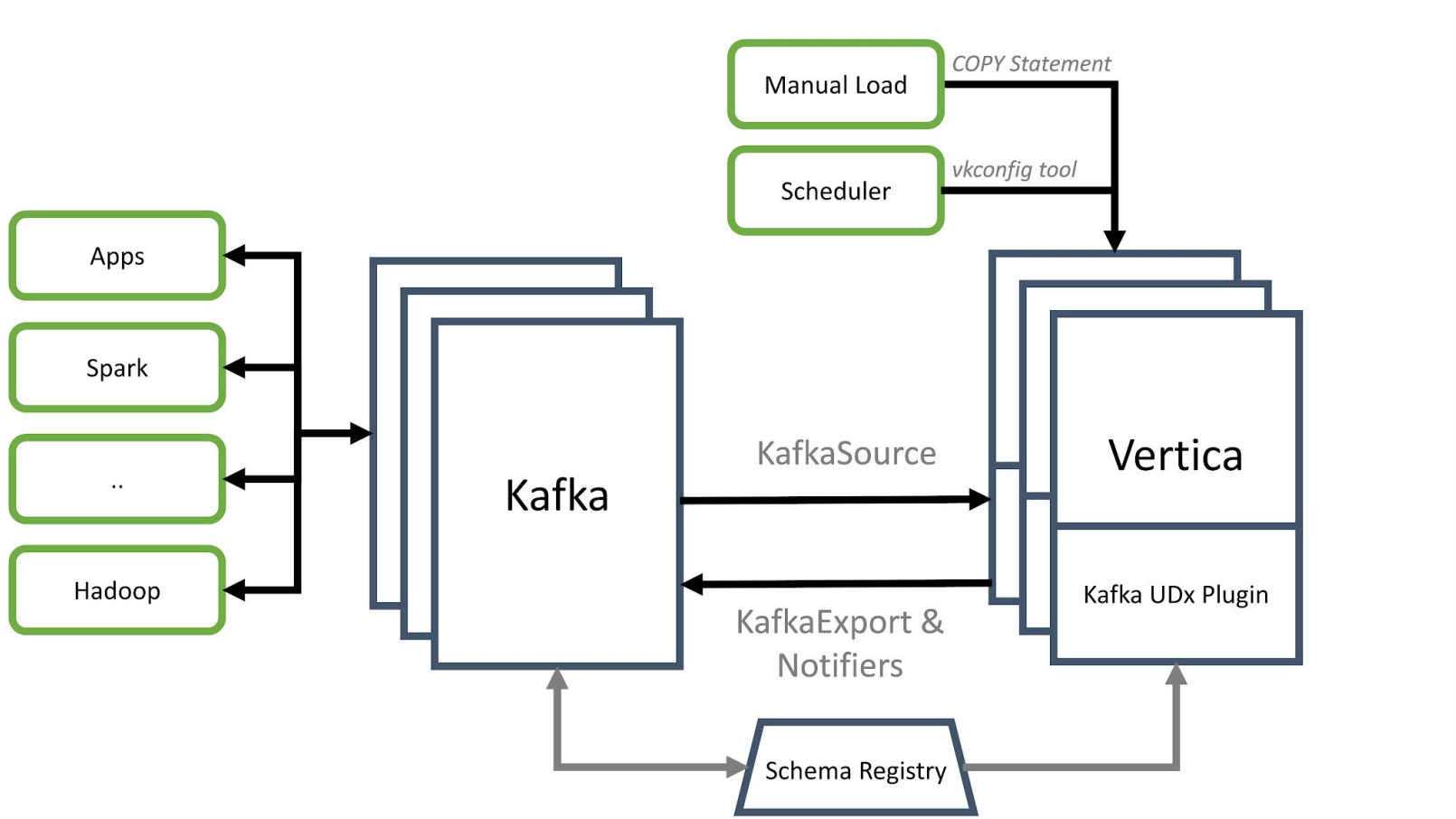


# Интеграция с другими системами

OTUS

- External tables
- Apache Spark
- Apache Kafka
- Apache Hive

# Интеграция с Apache Kafka



Заполните, пожалуйста, опрос в личном кабинете!

# Полезные ссылки

- http://vertica.tips/2014/03/16/join-operator-overview/
- http://db.lcs.mit.edu/projects/cstore/vldb.pdf

# Егор Матешук

egor@mateshuk.com

# Спасибо
# за внимание!