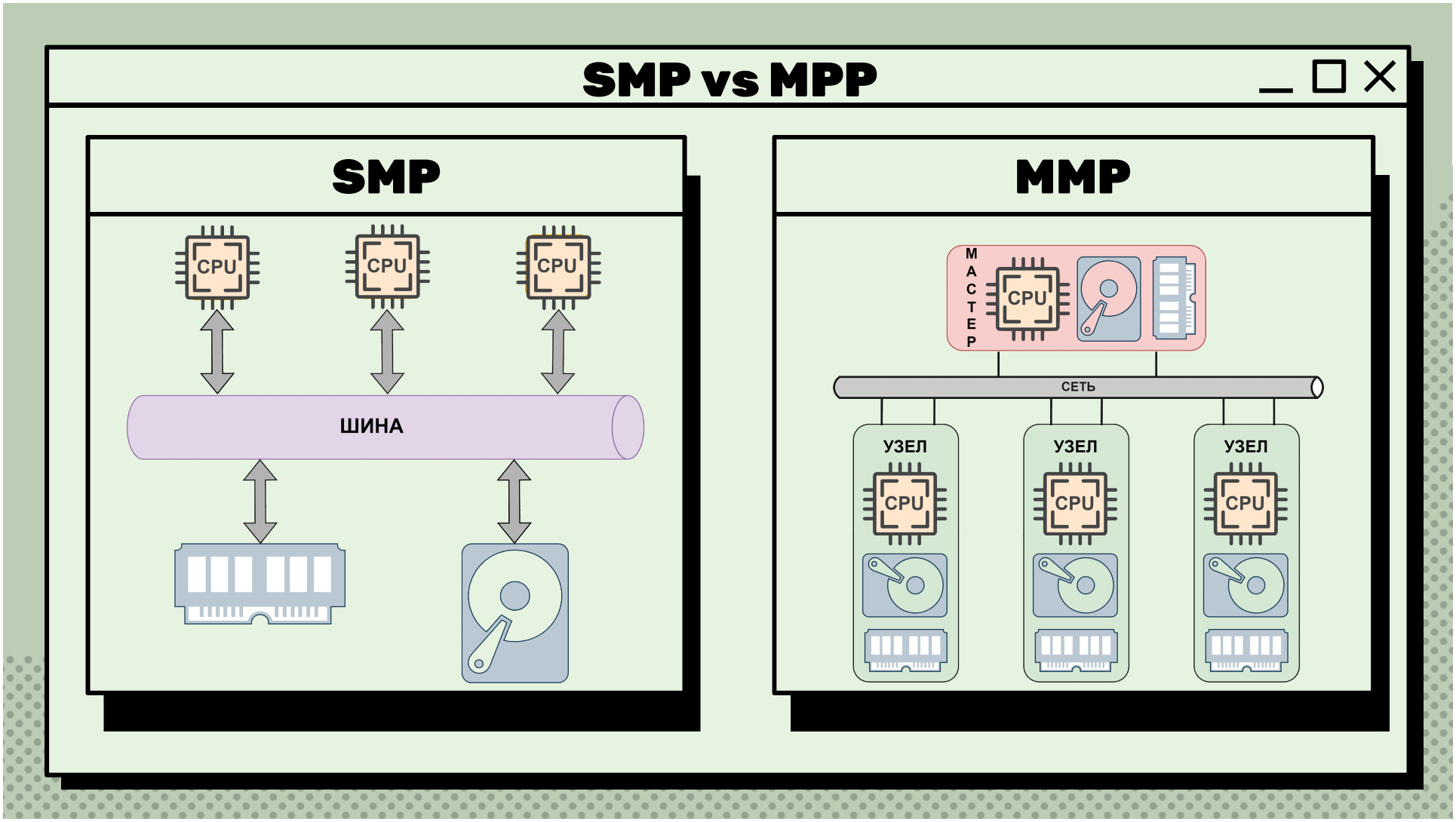
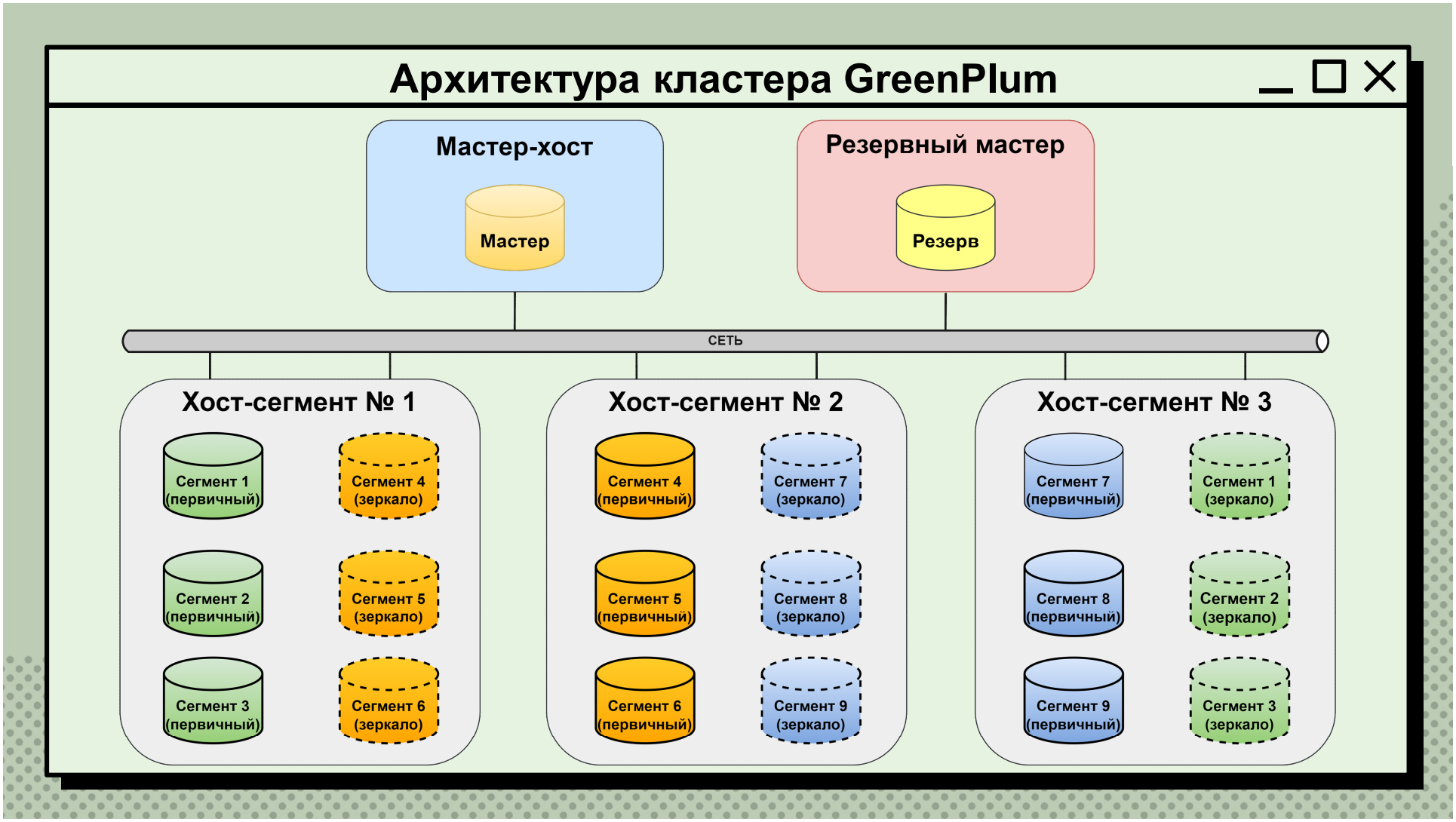

GreenPlum
Просто о сложном!

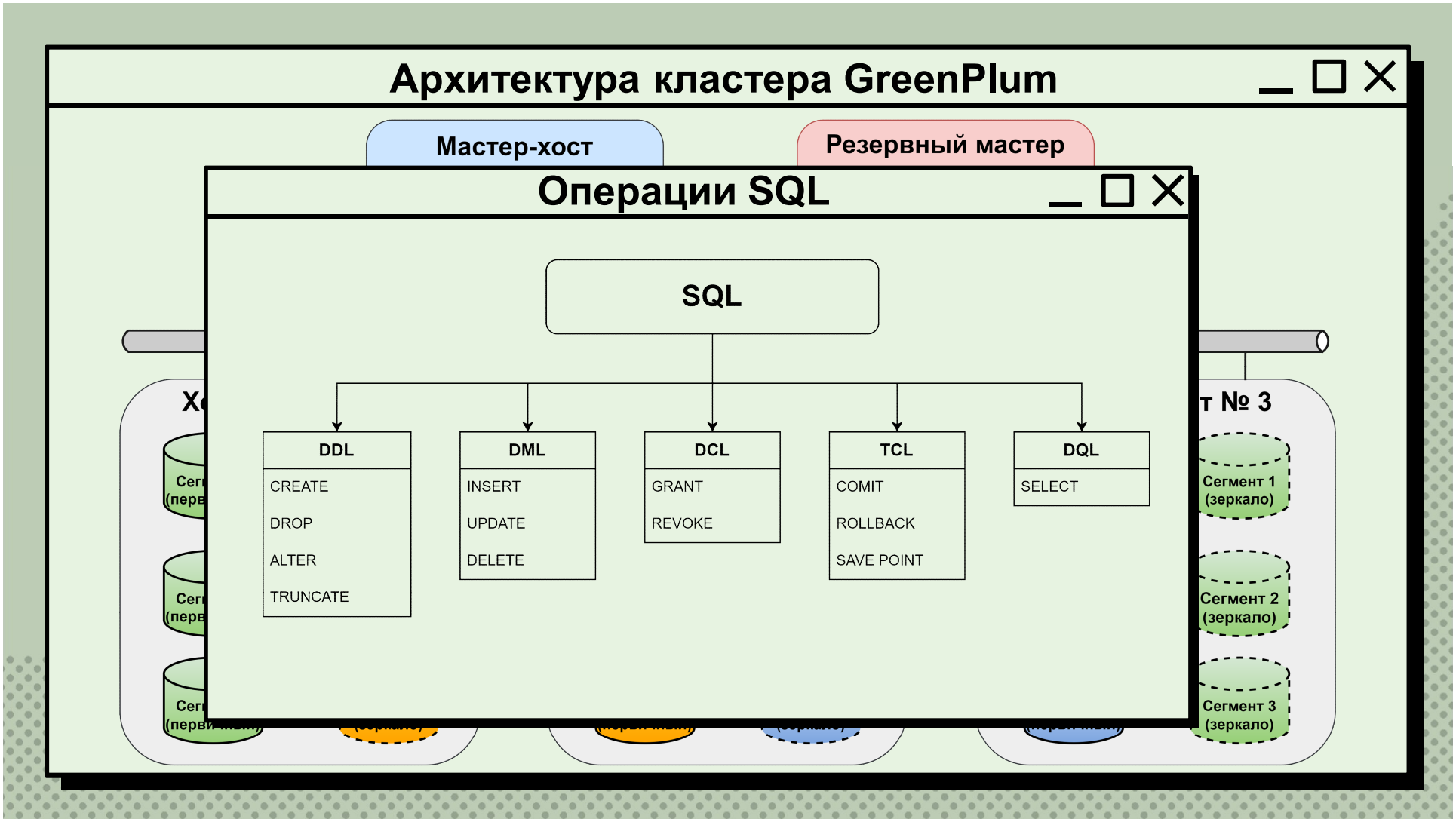
Что это такое?

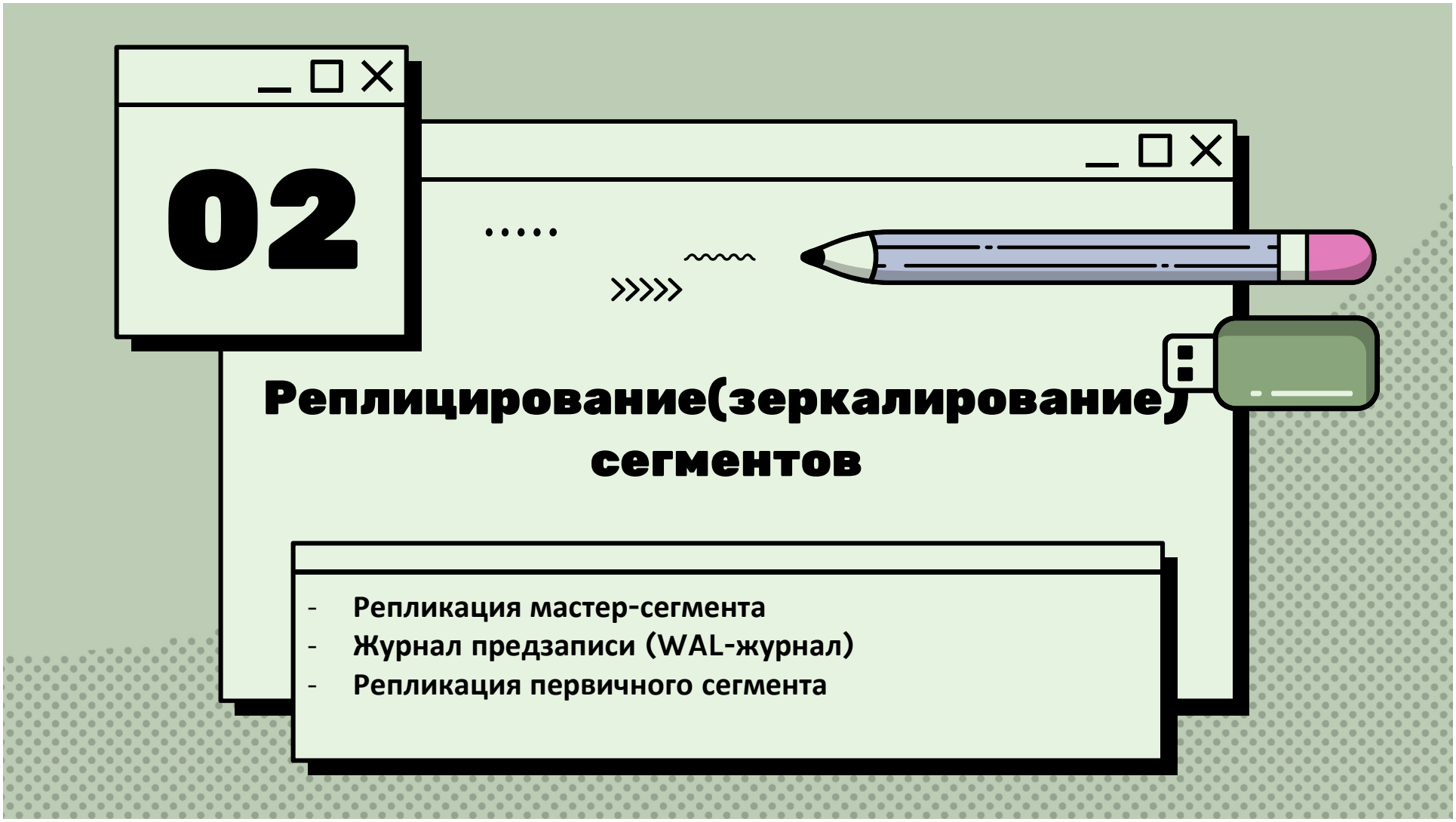
GreenPlum – это высоконагруженная, транзакционная, аналитическая система построенная на MPP архитектуре на основе PostgerSQL.

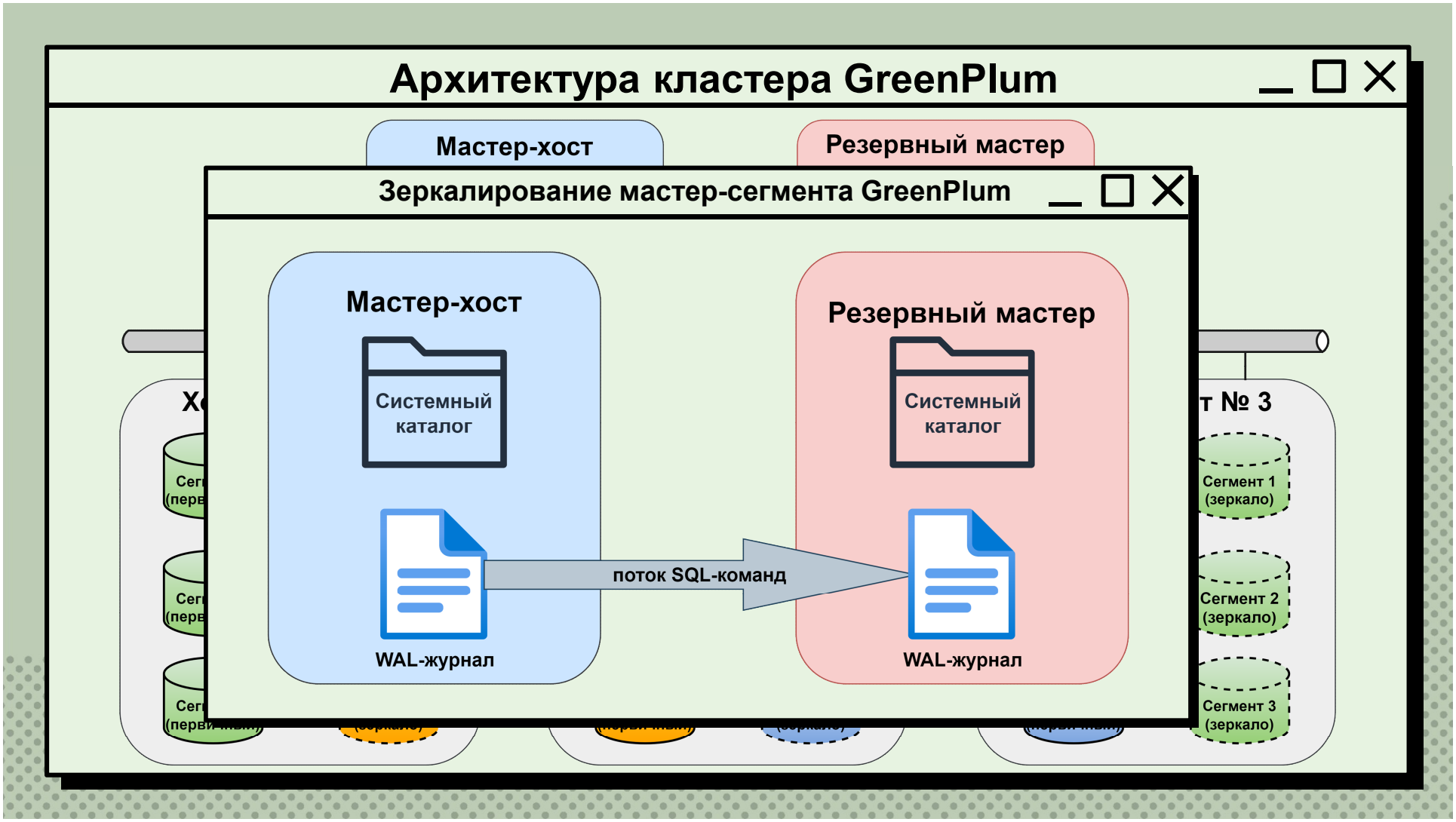

01
Архитектура GreenPlum
- SMP и MPP архитектуры
- Архитектура GreenPlum
- Операции SQL

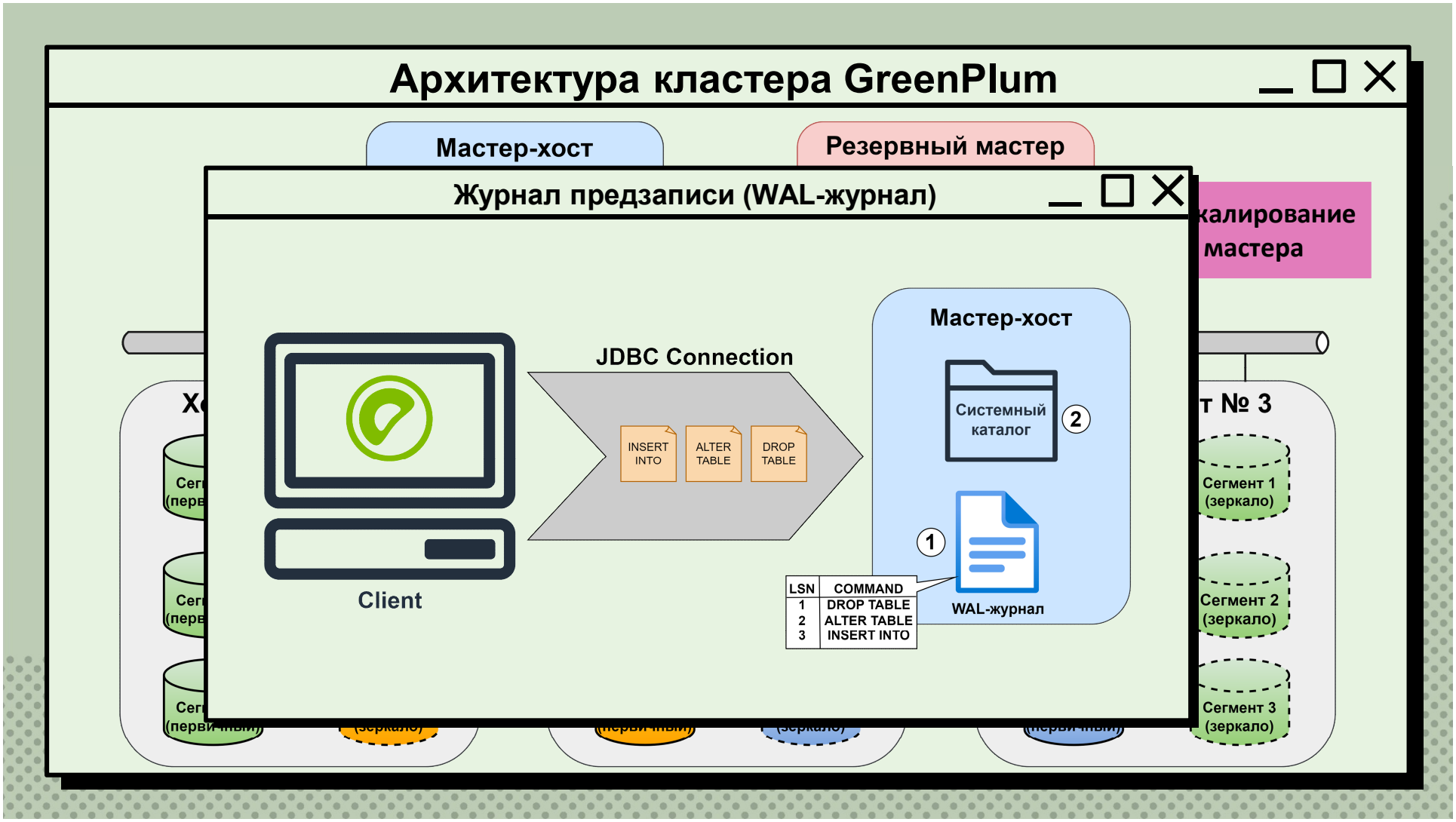

SMP vs MPP
SMP
CPU
CPU
CPU
ШИНА
MMP
МАСТЕР
CPU
СЕТЬ
УЗЕЛ
CPU
УЗЕЛ
CPU
УЗЕЛ
CPU

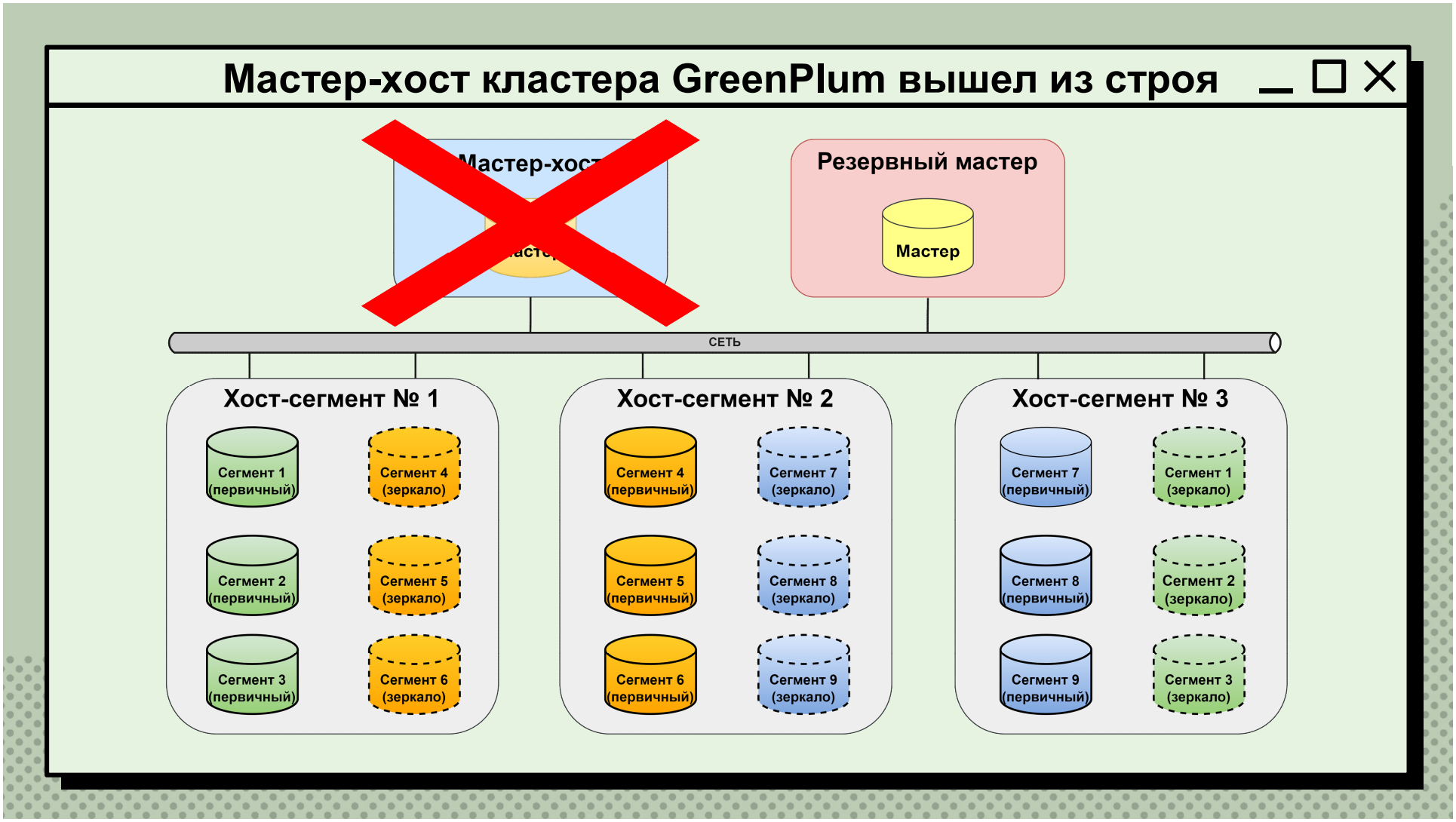

Архитектура кластера GreenPlum
Мастер-хост
Мастер
Резервный мастер
Резерв
СЕТЬ
Хост-сегмент № 1
Сегмент 1 (первичный)
Сегмент 4 (зеркало)
Сегмент 2 (первичный)
Сегмент 5 (зеркало)
Сегмент 3 (первичный)
Сегмент 6 (зеркало)
Хост-сегмент № 2
Сегмент 4 (первичный)
Сегмент 7 (зеркало)
Сегмент 5 (первичный)
Сегмент 8 (зеркало)
Сегмент 6 (первичный)
Сегмент 9 (зеркало)
Хост-сегмент № 3
Сегмент 7 (первичный)
Сегмент 1 (зеркало)
Сегмент 8 (первичный)
Сегмент 2 (зеркало)
Сегмент 9 (первичный)
Сегмент 3 (зеркало)


Архитектура кластера GreenPlum
Мастер-хост
Резервный мастер
Операции SQL
SQL
DDL
CREATE
DROP
ALTER
TRUNCATE
DML
INSERT
UPDATE
DELETE
DCL
GRANT
REVOKE
TCL
COMIT
ROLLBACK
SAVE POINT
DQL
SELECT
т № 3
Сегмент 1 (зеркало)
Сегмент 2 (зеркало)
Сегмент 3 (зеркало)

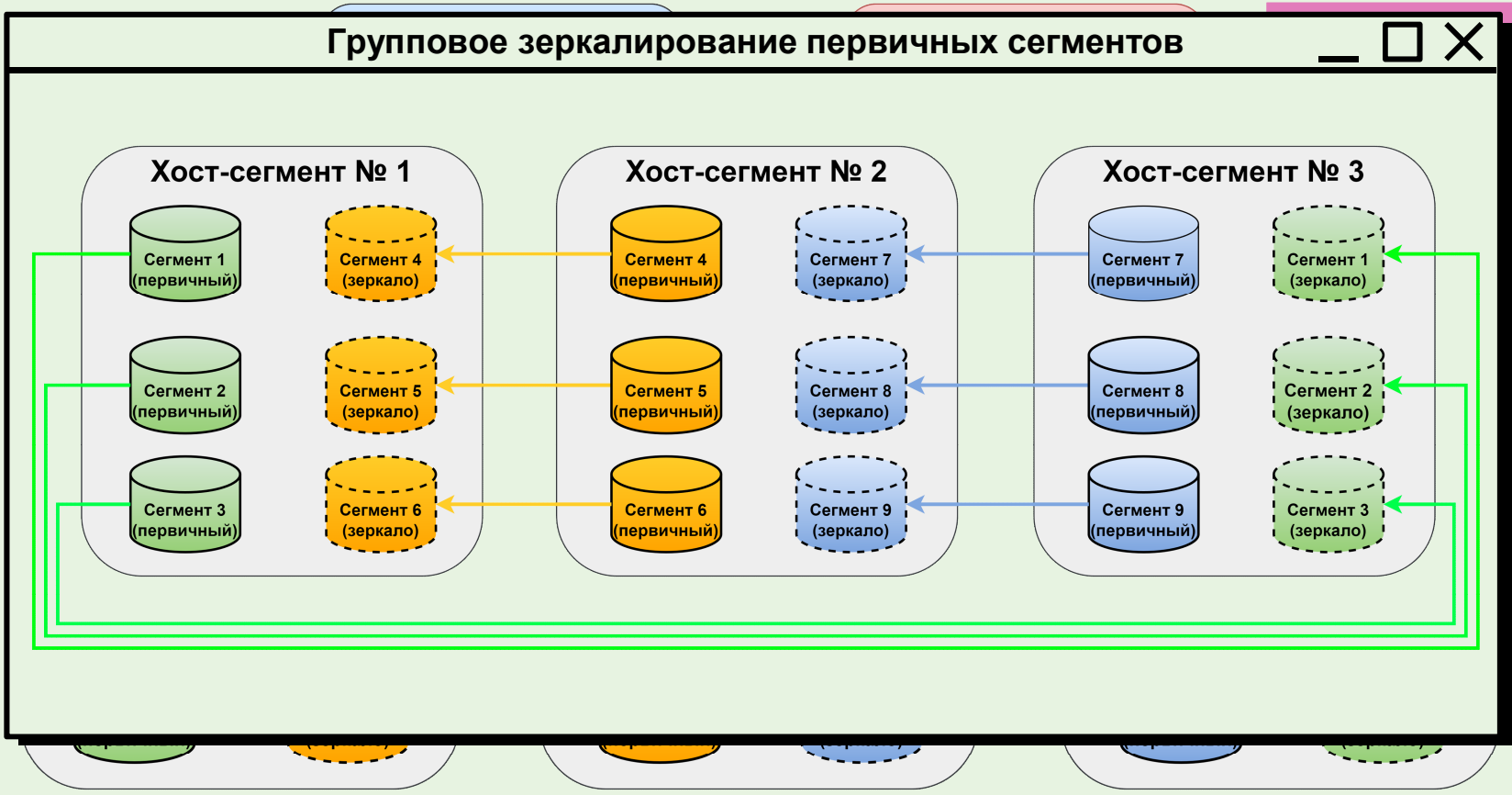

02
Репликирование(зеркалирование) сегментов
- Репликация мастер-сегмента
- Журнал предзаписи (WAL-журнал)
- Репликация первичного сегмента

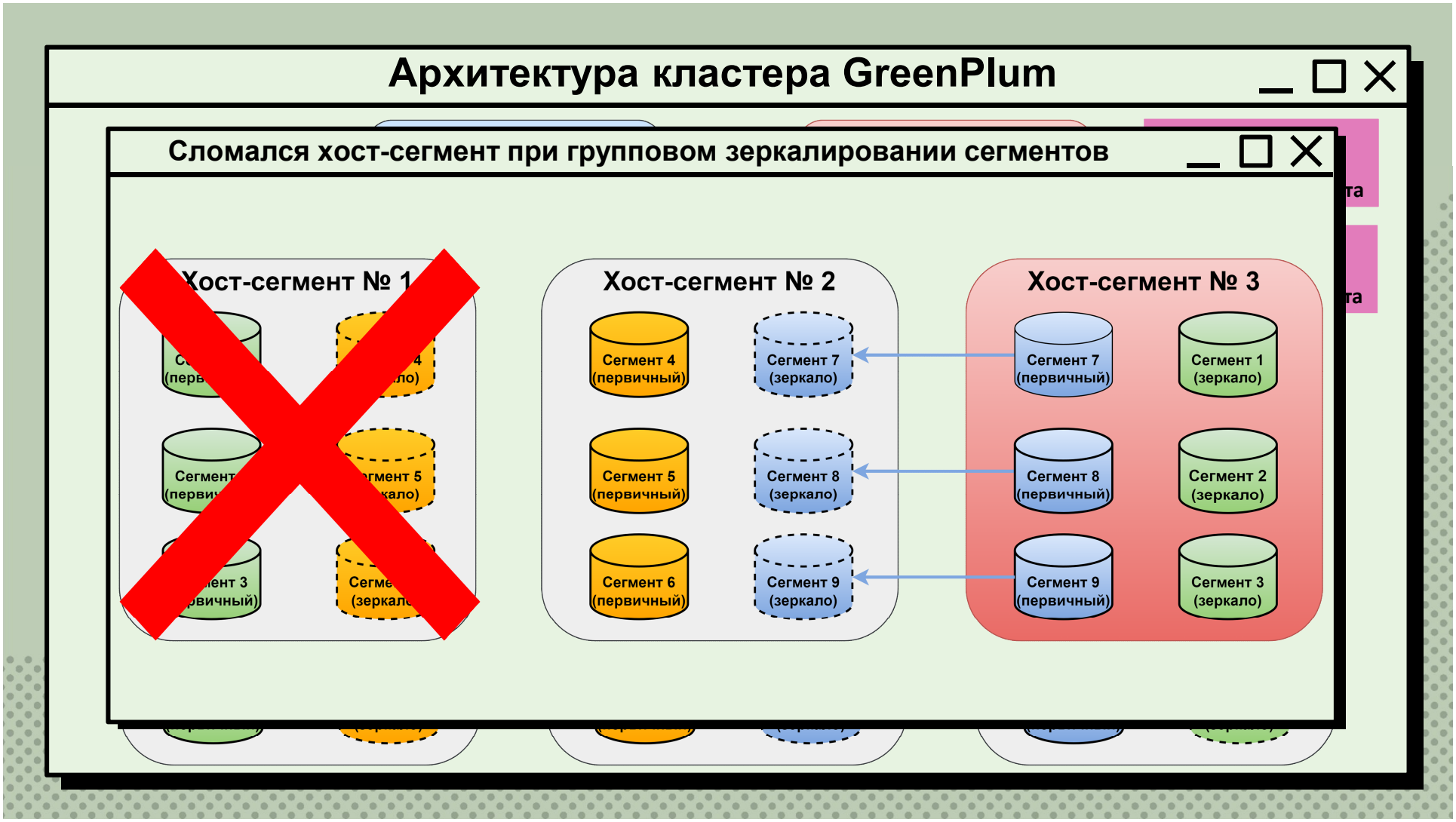

Архитектура кластера GreenPlum
Мастер-хост
Резервный мастер
Зеркалирование мастер-сегмента GreenPlum
Мастер-хост
Системный каталог
WAL-журнал
поток SQL-команд
Резервный мастер
Системный каталог
WAL-журнал
т № 3
Сегмент 1 (зеркало)
Сегмент 2 (зеркало)
Сегмент 3 (зеркало)


Архитектура кластера GreenPlum
Мастер-хост
Резервный мастер
Журнал предзаписи (WAL-журнал)
калирование
мастера
Мастер-хост
JDBC Connection
INSERT INTO
ALTER TABLE
DROP TABLE
Системный каталог
2
1
Client
WAL-журнал
LSN
COMMAND
1
DROP TABLE
2
ALTER TABLE
3
INSERT INTO
т № 3
Сегмент 1 (зеркало)
Сегмент 2 (зеркало)
Сегмент 3 (зеркало)

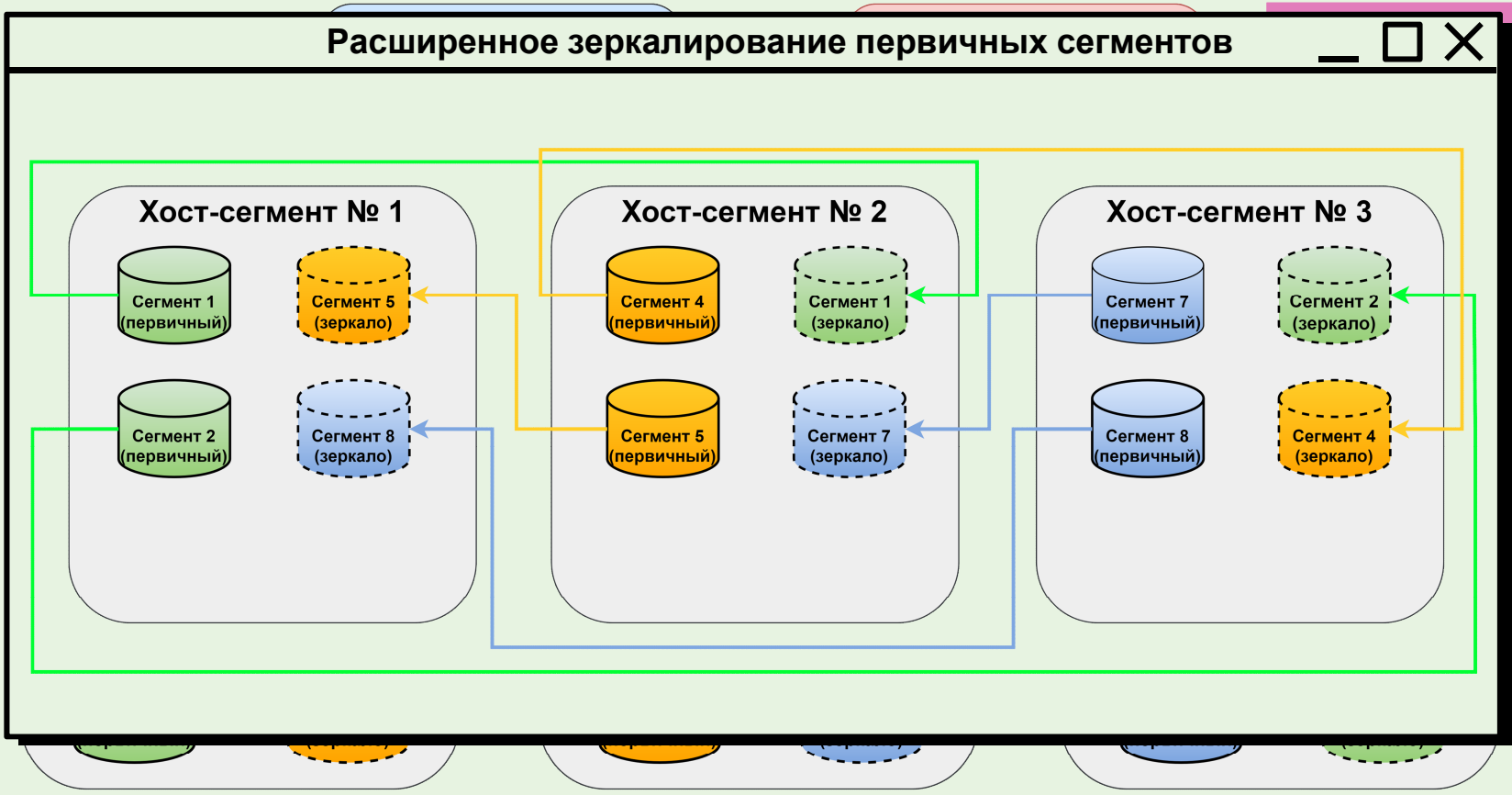

Мастер-хост кластера GreenPlum вышел из строя
Резервный мастер
Мастер
СЕТЬ
Хост-сегмент № 1
Сегмент 1 (первичный)
Сегмент 2 (первичный)
Сегмент 3 (первичный)
Сегмент 4 (зеркало)
Сегмент 5 (зеркало)
Сегмент 6 (зеркало)
Хост-сегмент № 2
Сегмент 4 (первичный)
Сегмент 5 (первичный)
Сегмент 6 (первичный)
Сегмент 7 (зеркало)
Сегмент 8 (зеркало)
Сегмент 9 (зеркало)
Хост-сегмент № 3
Сегмент 7 (первичный)
Сегмент 8 (первичный)
Сегмент 9 (первичный)
Сегмент 1 (зеркало)
Сегмент 2 (зеркало)
Сегмент 3 (зеркало)

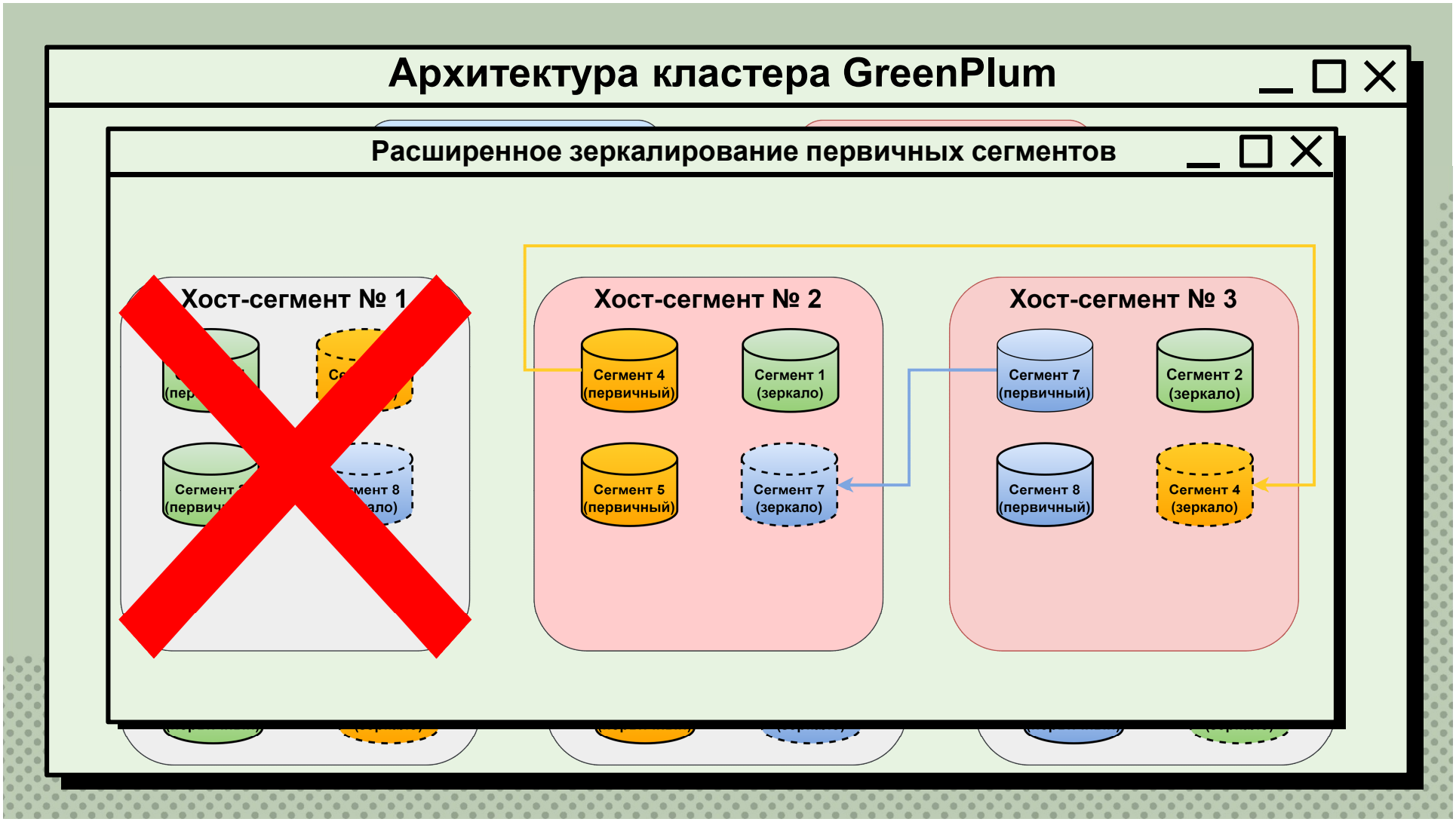
Архитектура кластера GreenPlum




Архитектура кластера GreenPlum
Сломался хост-сегмент при групповом зеркалировании сегментов
Хост-сегмент № 1
Хост-сегмент № 2
Хост-сегмент № 3
Сегмент 4 (первичный)
Сегмент 7 (зеркало)
Сегмент 5 (первичный)
Сегмент 8 (зеркало)
Сегмент 6 (первичный)
Сегмент 9 (зеркало)
Сегмент 7 (первичный)
Сегмент 1 (зеркало)
Сегмент 8 (первичный)
Сегмент 2 (зеркало)
Сегмент 9 (первичный)
Сегмент 3 (зеркало)

Архитектура кластера GreenPlum




Архитектура кластера GreenPlum
Расширенное зеркалирование первичных сегментов
Хост-сегмент № 1
Хост-сегмент № 2
Сегмент 4 (первичный)
Сегмент 1 (зеркало)
Сегмент 5 (первичный)
Сегмент 7 (зеркало)
Хост-сегмент № 3
Сегмент 7 (первичный)
Сегмент 2 (зеркало)
Сегмент 8 (первичный)
Сегмент 4 (зеркало)

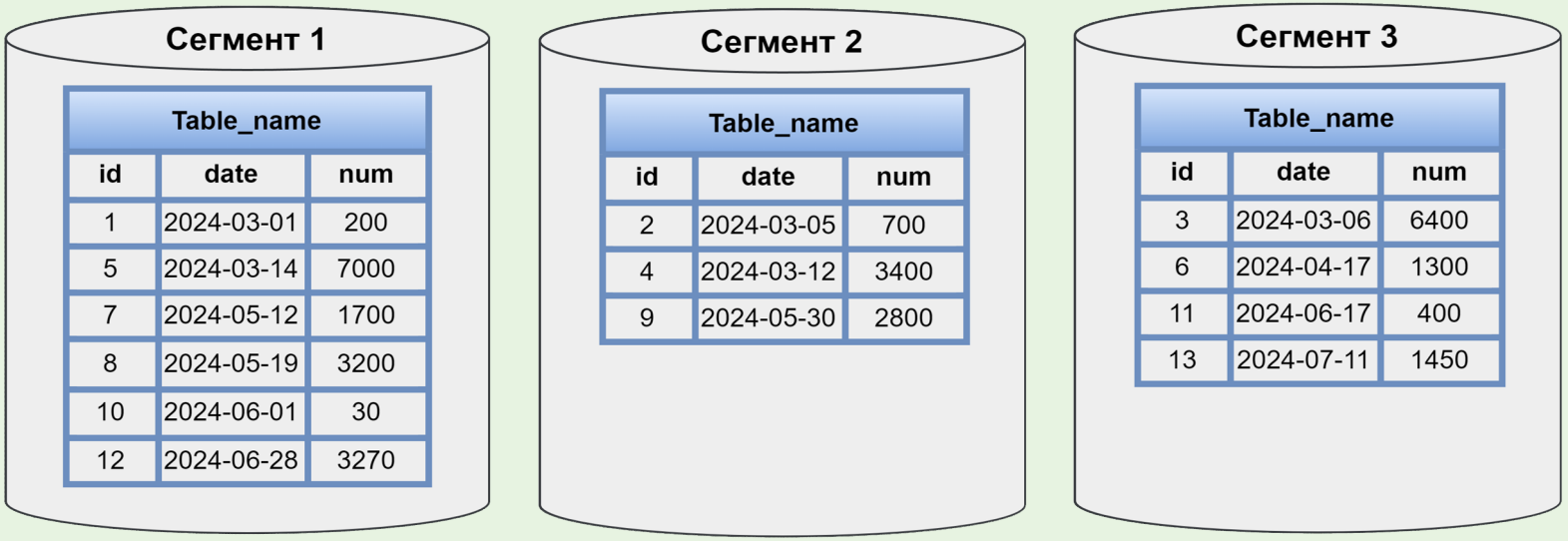
03

Дистрибуция

- Дистрибуция и их виды в GreenPlum
- Виды таблиц, применяемые в GreenPlum

Дистрибуция

Дистрибуция — это принцип проектирования базы данных, при котором данные разбиваются на части и размещаются на разных шардах\сегментах.



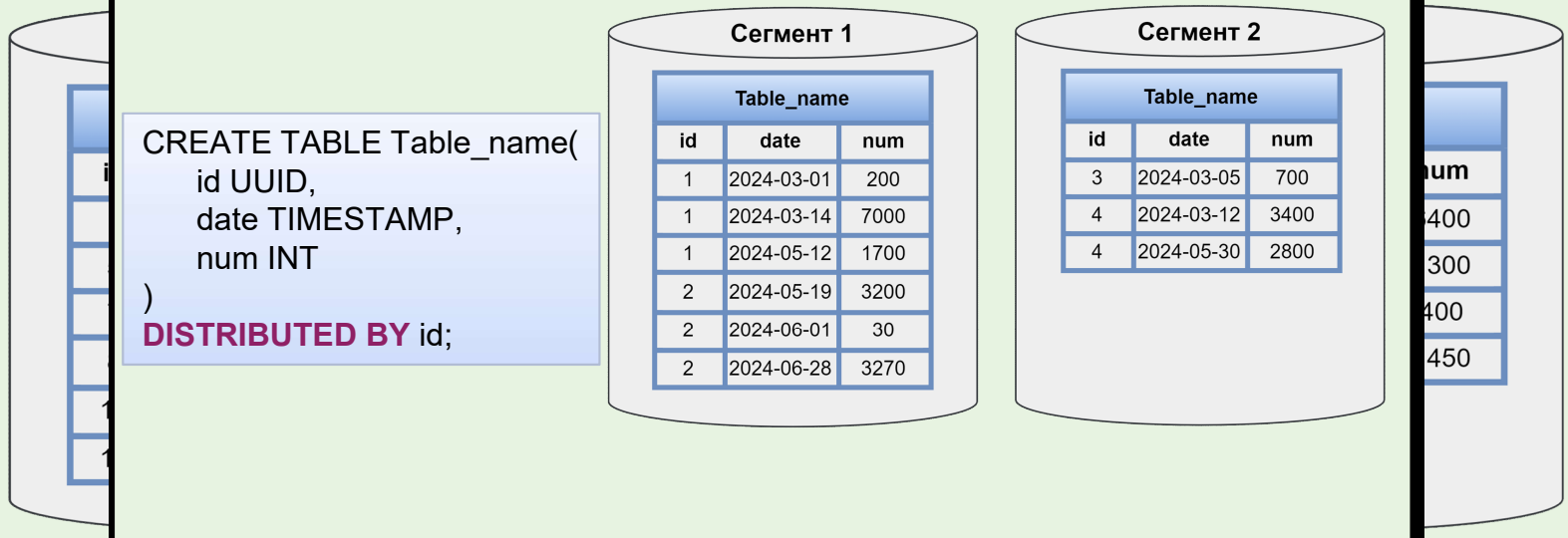
При создании таблицы для обозначения дистрибуции используется ключевое слово «**DISTRIBUTED**»

Дистрибуция \ Шардирование

Дистр
разби

Дистрибуция по ключу

```
CREATE TABLE Table_name(
    id UUID,
    date TIMESTAMP,
    num INT
)
DISTRIBUTED BY id;
```



При создании таблицы для д обозначения дистрибуции используется ключевое слово «**DISTRIBUTED**»

Дистрибуция \ Шардирование

Дистр
разби

Дистрибуция случайным образом

```
CREATE TABLE Table_name(
    id UUID,
    date TIMESTAMP,
    num INT
)
DISTRIBUTED RANDOMLY;
```

Сегмент 1

Table_name		
id	**date**	**num**
1	2024-03-01	200
2	2024-03-14	7000
2	2024-05-12	1700
3	2024-05-19	3200
4	2024-06-01	30
4	2024-06-28	3270

Сегмент 2

Table_name		
id	**date**	**num**
1	2024-03-05	700
2	2024-03-12	3400
3	2024-05-30	2800
4	2024-05-30	2800
4	2024-05-30	2800

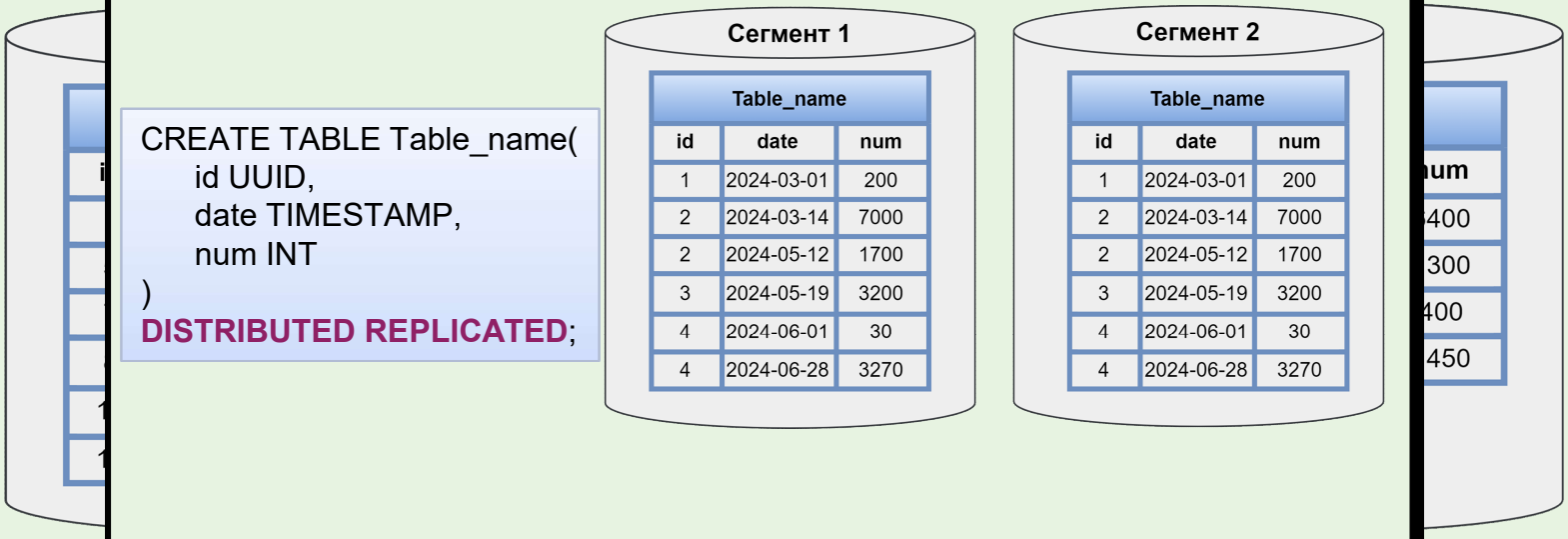
При создании таблицы для д обозначения дистрибуции используется ключевое слово **«DISTRIBUTED»**

Дистрибуция \ Шардирование

Дистр
разби

Дистрибуция таблицы по всем сегментам

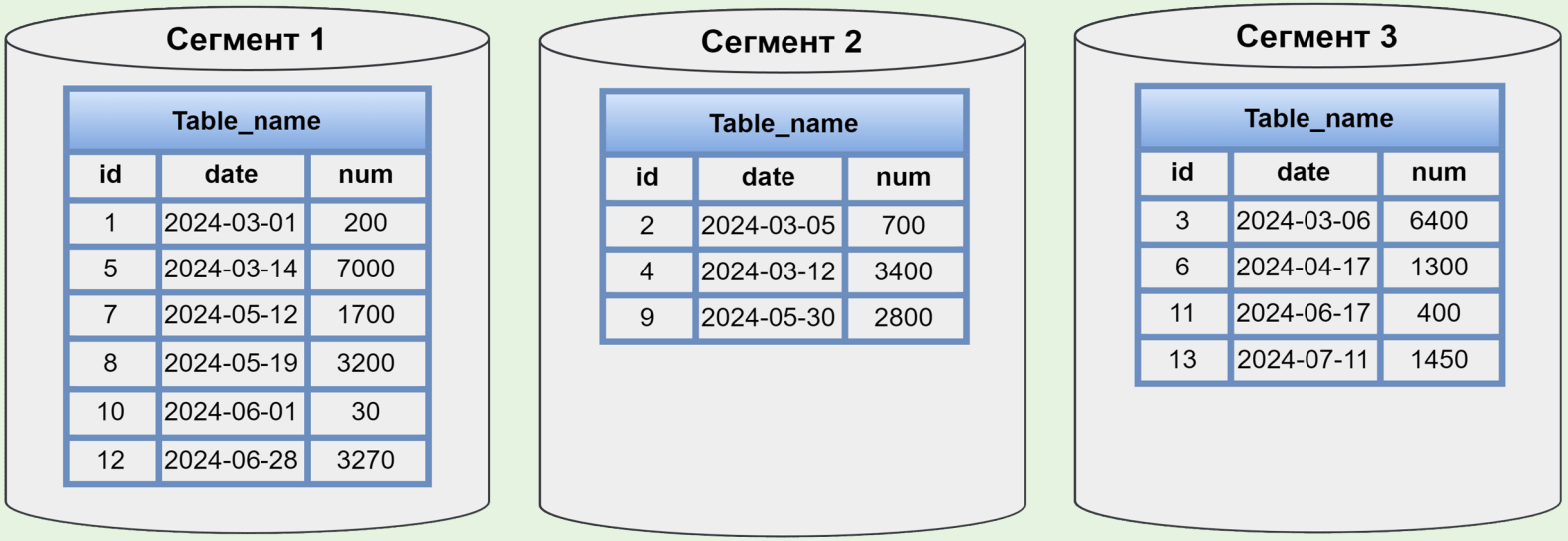
```
CREATE TABLE Table_name(
    id UUID,
    date TIMESTAMP,
    num INT
)
DISTRIBUTED REPLICATED;
```



При создании таблицы для д обозначения дистрибуции используется ключевое слово «**DISTRIBUTED**»

Дистрибуция \ Шардирование

Дистрибуция — это принцип проектирования базы данных, при котором данные разбиваются на части и размещаются на разных шардах\сегментах.



При создании таблицы для д обозначения дистрибуции используется ключевое слово «**DISTRIBUTED**»

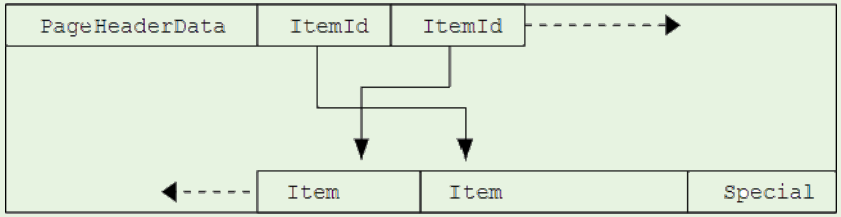
Виды таблиц применяемые в GP

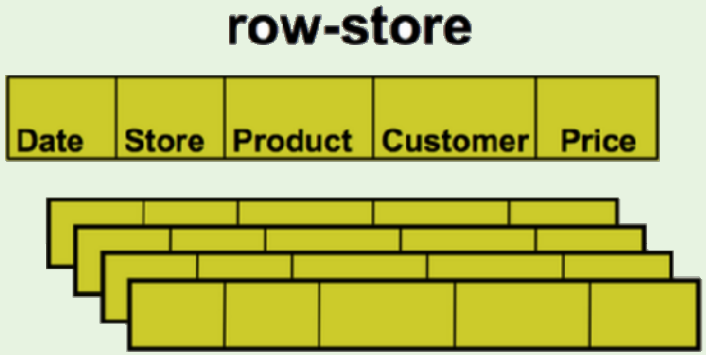
Основные виды таблиц:

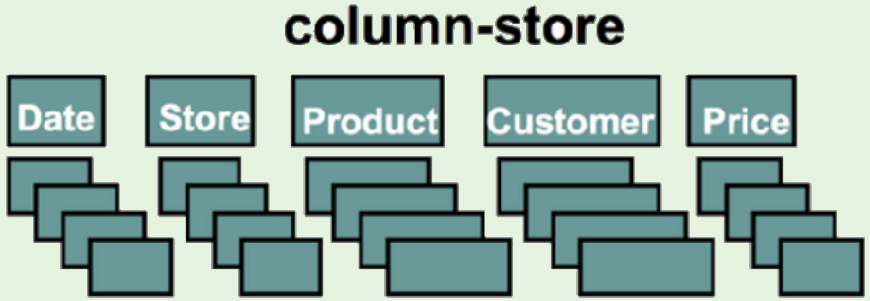
1. HEAP-таблица
2. APPEND-OPTIMIZED таблица

Дополнительные виды:

1. Временная таблица
2. Нежурналируемая таблица
3. Внешняя таблица







Виды таблиц применяемые в GP

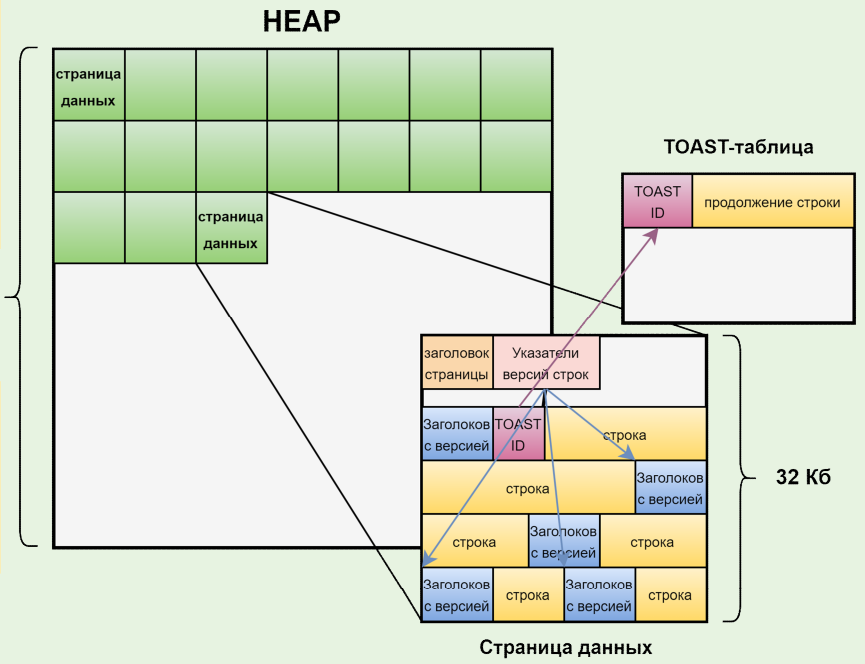
HEAP таблица

Ограничения:

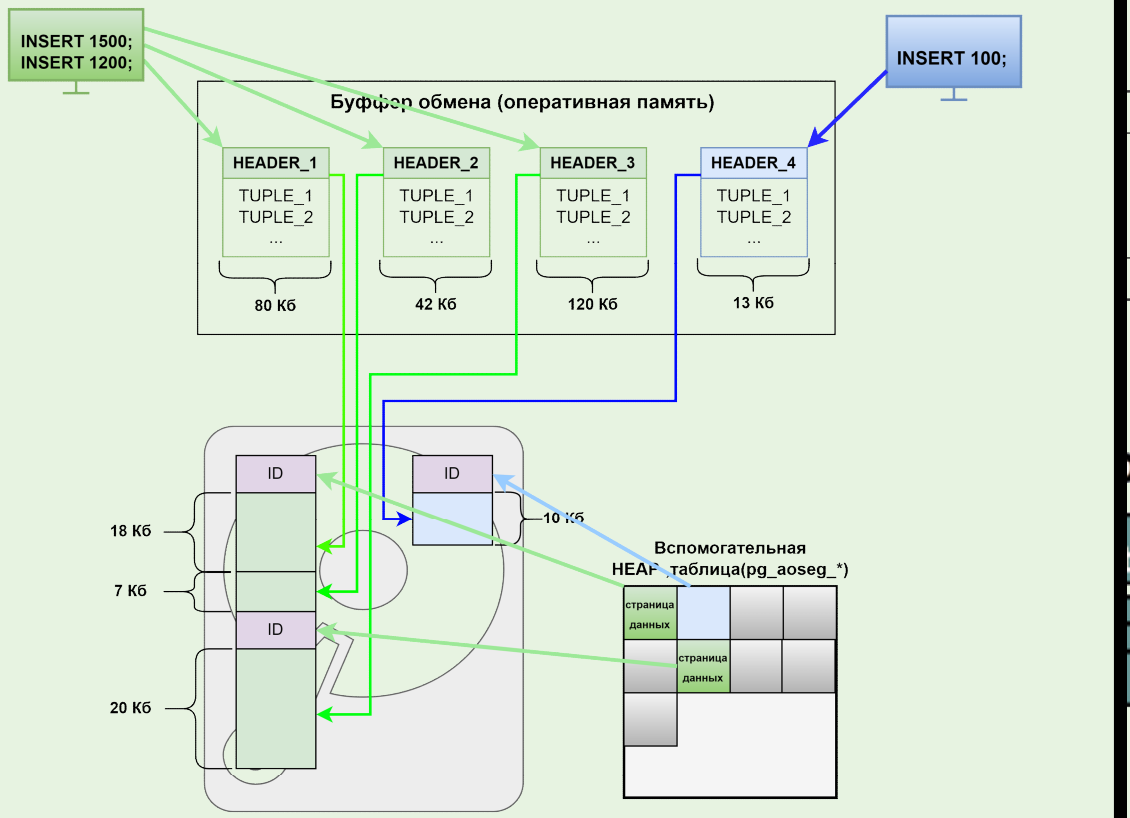
- HEAP-таблицы поддерживают только **построчно-ориентированное** хранение данных.
- Не поддерживают **сжатие**

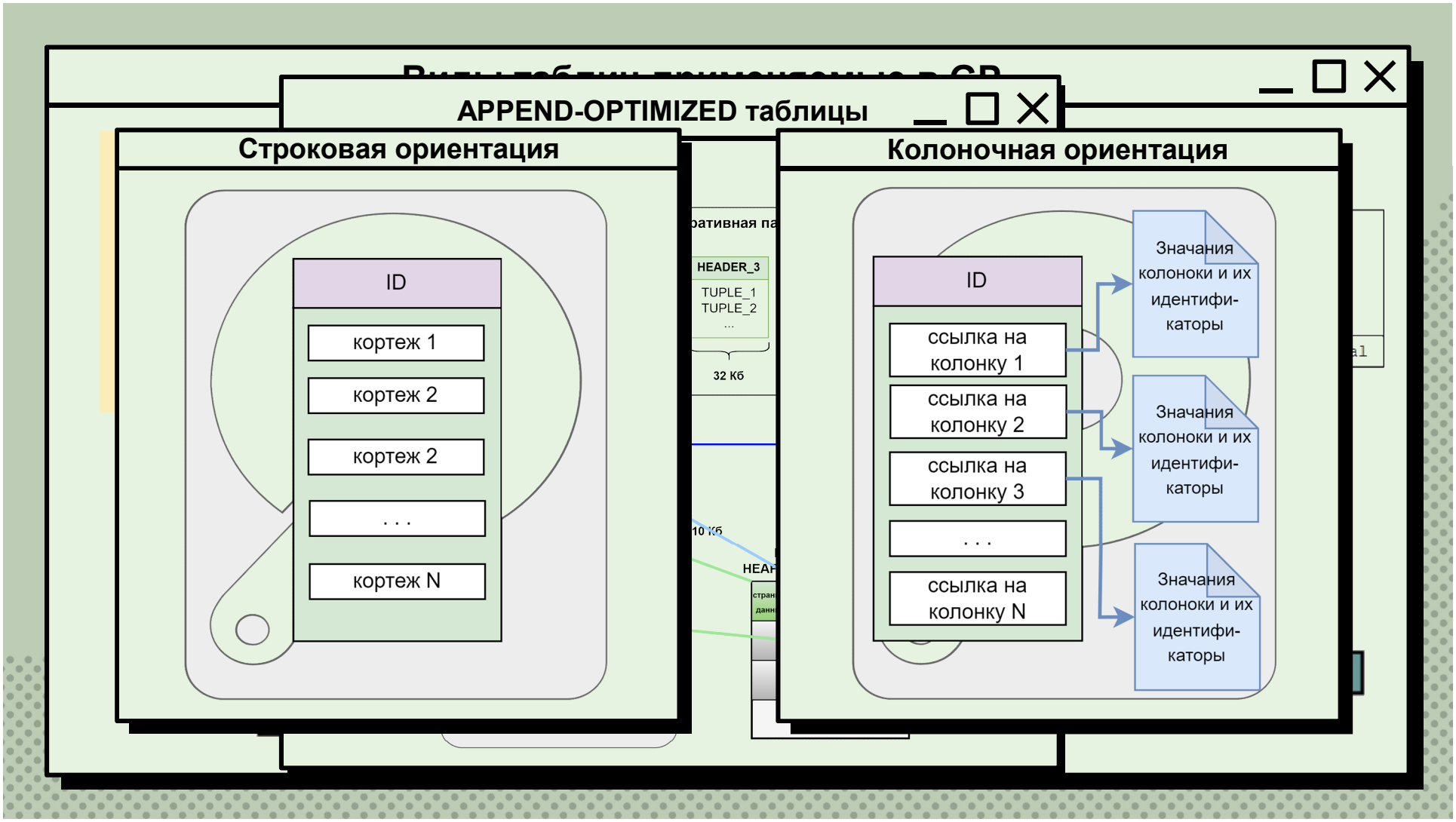
Использование:

- Если в таблице **часто меняются данные**;
- Для таблиц **малого размера**, например до Гб.



APPEND-OPTIMIZED таблицы




APPEND-OPTIMIZED таблицы
Строковая ориентация
ID
кортеж 1
кортеж 2
кортеж 2
. . .
кортеж N
HEADER_3
TUPLE_1
TUPLE_2
...
32 Кб
Колоночная ориентация
ID
ссылка на колонку 1
ссылка на колонку 2
ссылка на колонку 3
. . .
ссылка на колонку N
Значения колоноки и их идентифи-каторы
Значения колоноки и их идентифи-каторы
Значения колоноки и их идентифи-каторы

Виды таблиц применяемые в GP

Временные таблицы

Временные таблицы — живут в рамках одной сессии и автоматически уничтожаются при завершении сессии.

```
CREATE TEMPORARY TABLE temp_table(
    id_col SERIAL,
    col_2 TEXT)
WITH (
    appendoptimized=TRUE,
    orientation=column,
    compresstype=zstd,
    compresslevel=1)
DISTRIBUTED BY (id);
```

Применение:
Сохранение данных промежуточного расчёта и использования их в дальнейшем.

Виды таблиц применяемые в GP

Нежурналируемая таблица

Нежурналируемые таблицы — не записывают свои транзакции в WAL-журнал, поэтому не зеркалируются.

```
CREATE UNLOGGED TABLE unlogged_table (
   id_ col SERIAL,
   col_2  TEXT)
WITH (
   appendoptimized=true,
   orientation=row)
DISTRIBUTED BY (id);
```

Преимущества и недостатки:

✓ Скорость записи в такую таблицу увеличивается.

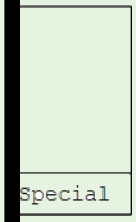
✗ Уменьшается надёжность хранения данных.

Виды таблиц применяемые в GP

Внешние таблицы

Внешние таблицы — предназначены для параллельной загрузки данных в Greenplum или выгрузки из него.

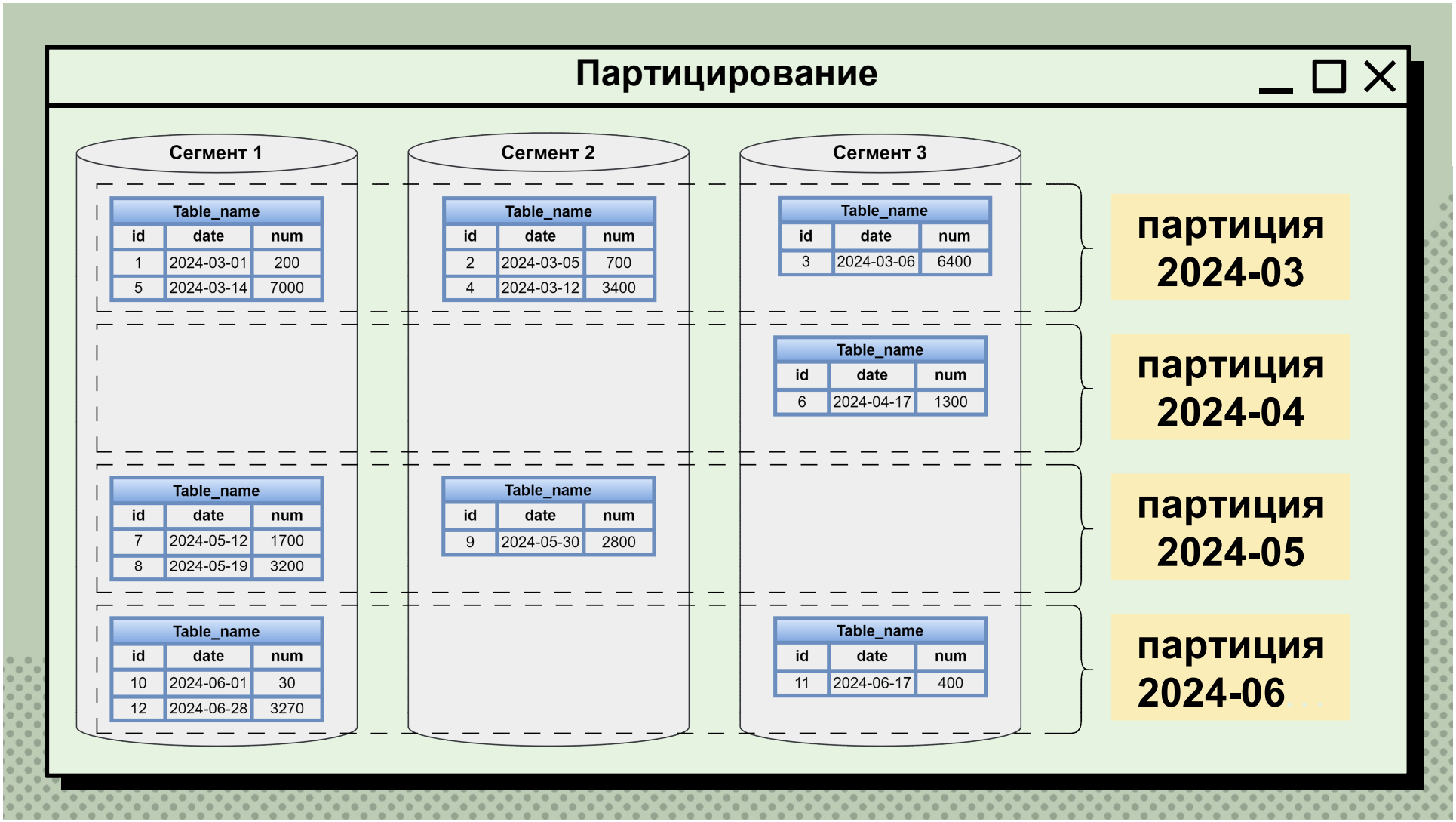
```
CREATE EXTERNAL TABLE ext_expenses (
id_ col int,
col_2  TEXT)
LOCATION (
    'gpfdist://localhost:2107/*.csv',
    'gpfdist://localhost:2108/*.csv ')
FORMAT 'CSV' ( DELIMITER ';' );
```



04

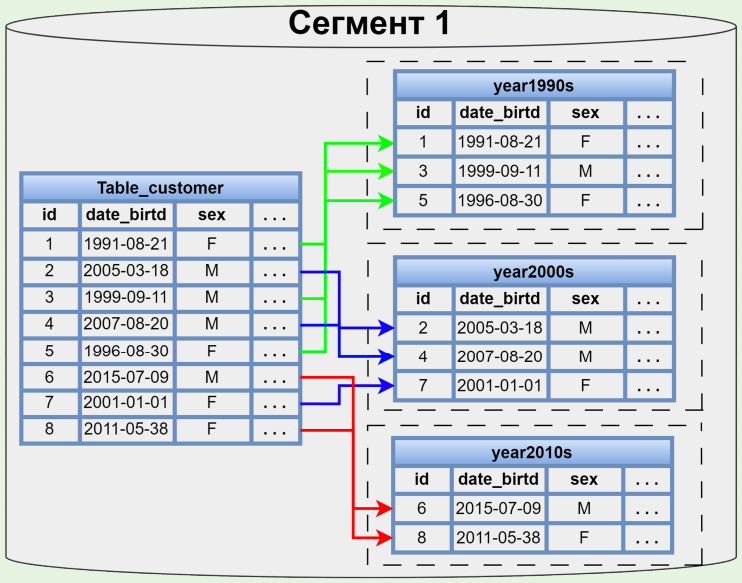
Партицирование

- Партицирование и их виды в GreenPlum
- Групповое партицирование в GreenPlum


Партицирование
Сегмент 1
Сегмент 2
Сегмент 3
Table_name
id date num
1 2024-03-01 200
5 2024-03-14 7000
Table_name
id date num
2 2024-03-05 700
4 2024-03-12 3400
Table_name
id date num
3 2024-03-06 6400
партиция 2024-03
Table_name
id date num
6 2024-04-17 1300
партиция 2024-04
Table_name
id date num
7 2024-05-12 1700
8 2024-05-19 3200
Table_name
id date num
9 2024-05-30 2800
партиция 2024-05
Table_name
id date num
10 2024-06-01 30
12 2024-06-28 3270
Table_name
id date num
11 2024-06-17 400
партиция 2024-06

Партицирование

Партицирование по диапазону

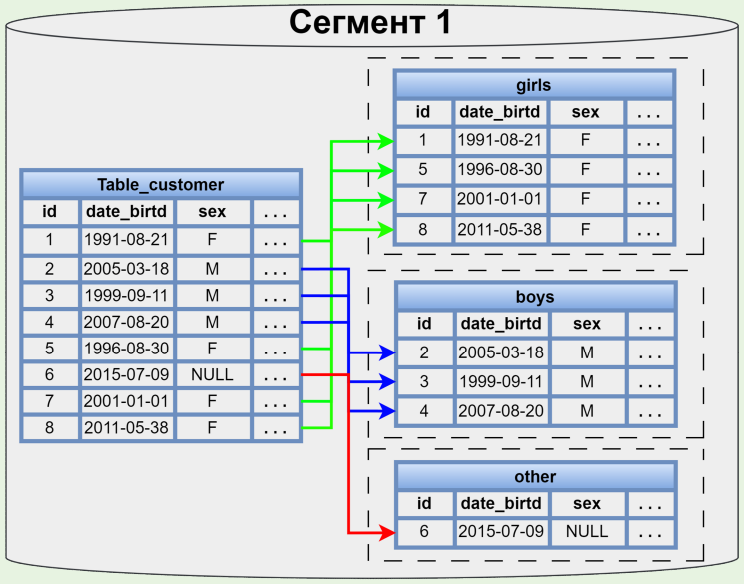



```
CREATE TABLE …
WITH …
DISTRIBUTED BY …
PARTITION BY RANGE (date_birth)
(
PARTITION year1990s START ('1990-01-01')
                END ('2000-01-01')
                WITH (appendoptimized=TRUE,
                      compresstype=zstd),
PARTITION year2000s START ('2000-01-01')
                END ('2010-01-01'),
PARTITION year2010s START ('2010-01-01')
                END ('2020-00-01'));
```

```
(START ('1990-01-01 00:00:00')
   END ('2020-01-01 00:00:00')
   EVERY (INTERVAL '10 YEAR'));
```

Партицирование

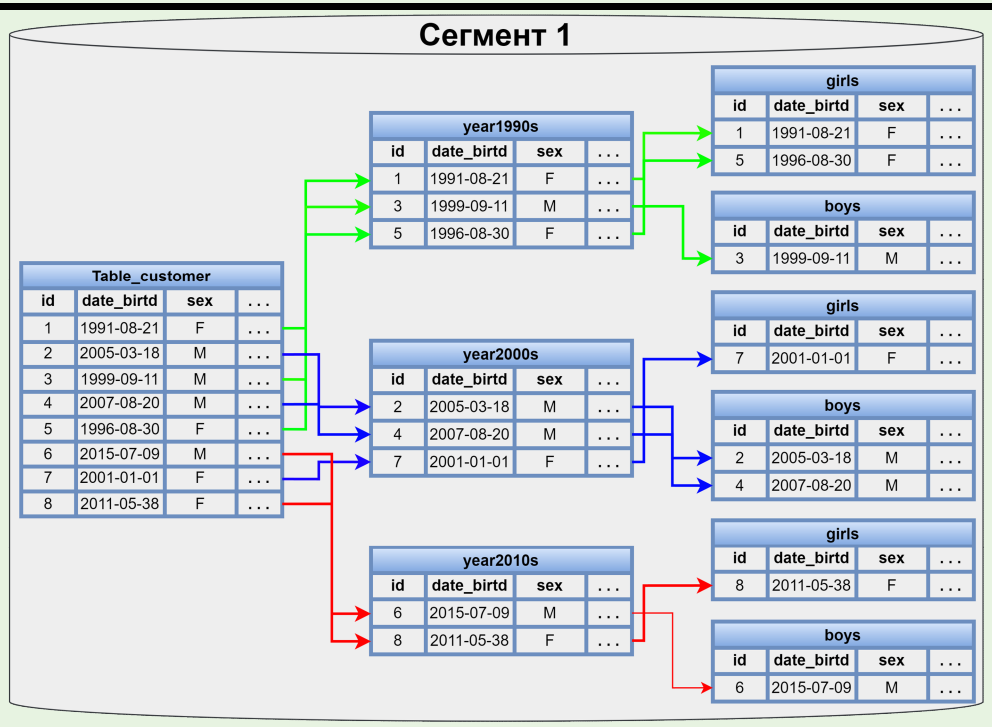
Партицирование по списку




```
CREATE TABLE ...
WITH…
DISTRIBUTED BY ...
PARTITION BY LIST (sex)
( PARTITION girls VALUES ('F'),
PARTITION boys VALUES ('M'),
DEFAULT PARTITION other );
```

Партицирование

Многоуровневое партицирование




```
CREATE TABLE ...
WITH ...
DISTRIBUTED BY ...
PARTITION BY RANGE (date_birth)
SUBPARTITION BY LIST (sex)
  SUBPARTITION TEMPLATE (
    SUBPARTITION girls VALUES ('F'),
    SUBPARTITION boys VALUES ('M')
      )
(START ('1990-01-01')
 END ('2025-01-01')
 EVERY (INTERVAL '10 YEAR'));
```

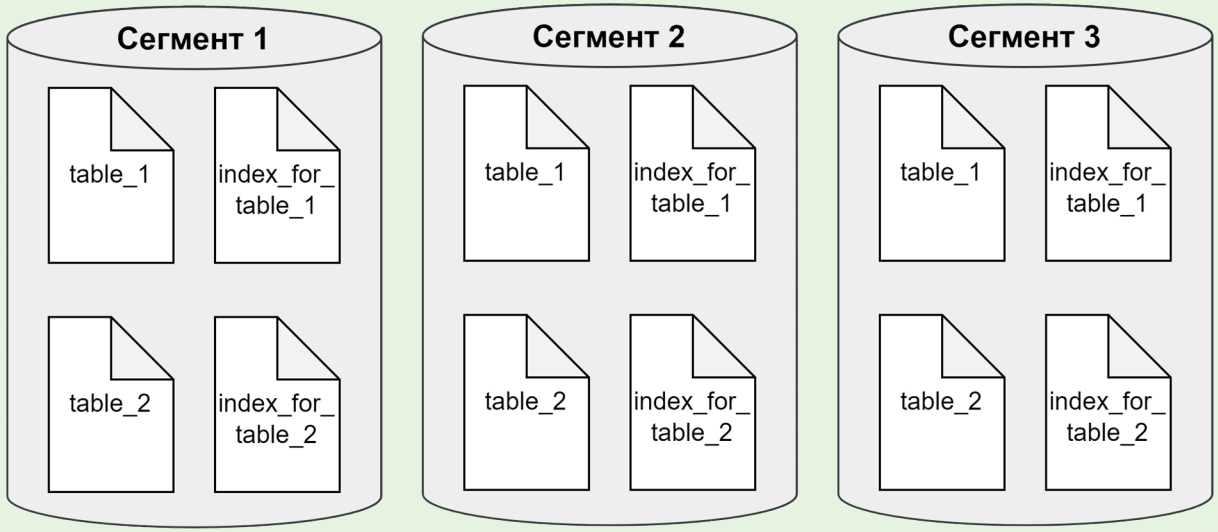
05

Индексы

- Индексы и их виды в GreenPlum
- B-tree индекс
- Bitmap индекс

Партицирование

Индекс базы данных – это отдельная структура данных, которая создается на основе с данных в таблице, которая ускоряют процесс извлечения этих данных, и тем самым повышая производительность запросов.



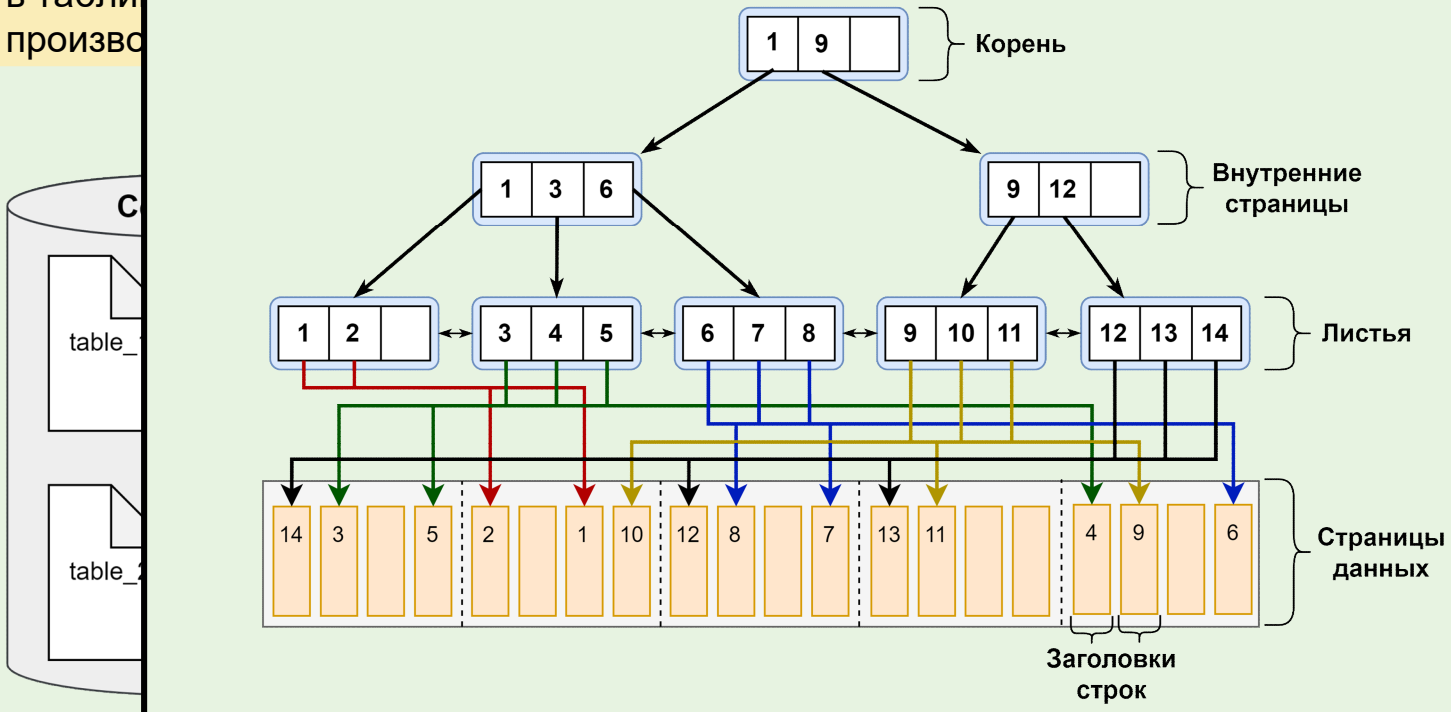
Виды индексов:

- B-tree
- Bitmap
- GIST
- SP-GIST
- GIN

Партицирование

Индекс
в табли
произво
ных

B-tree индекс

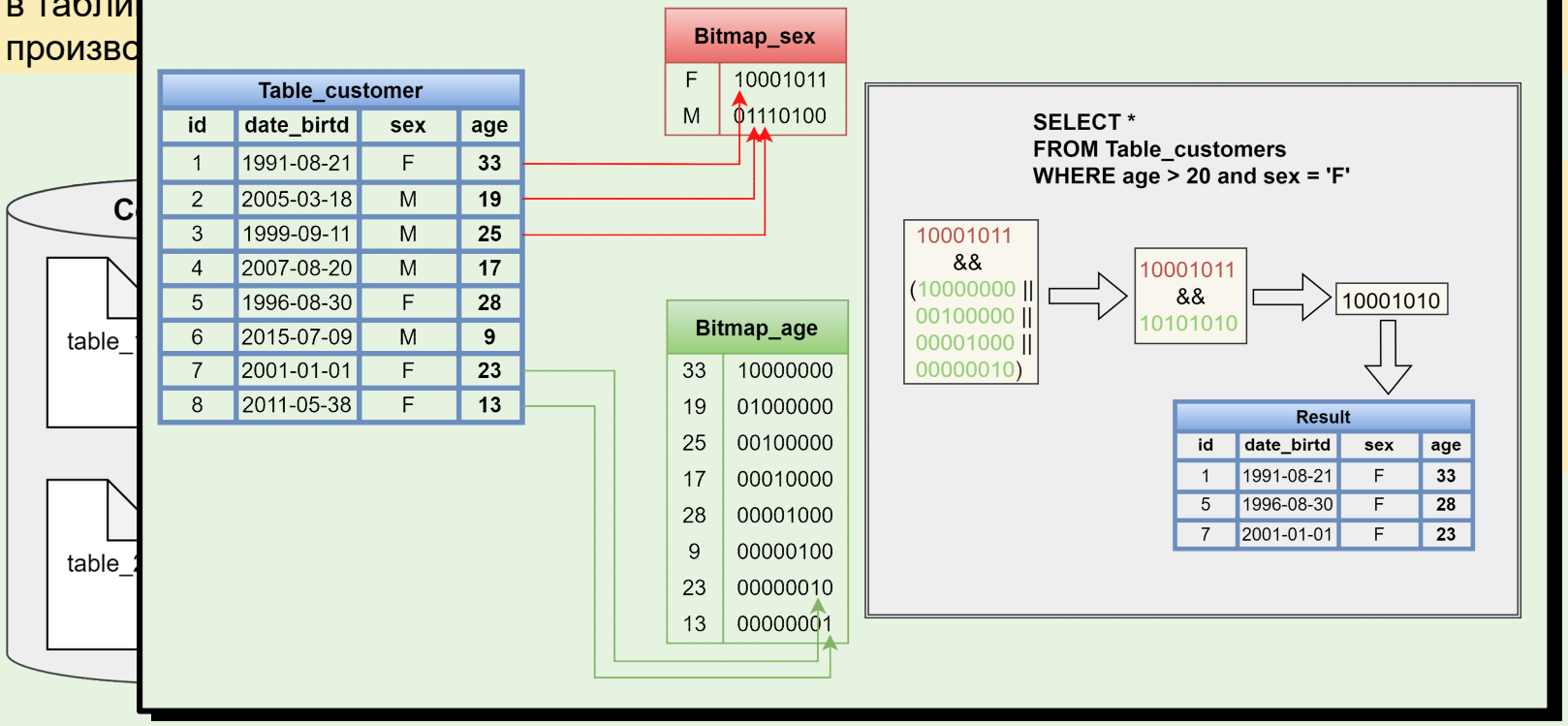


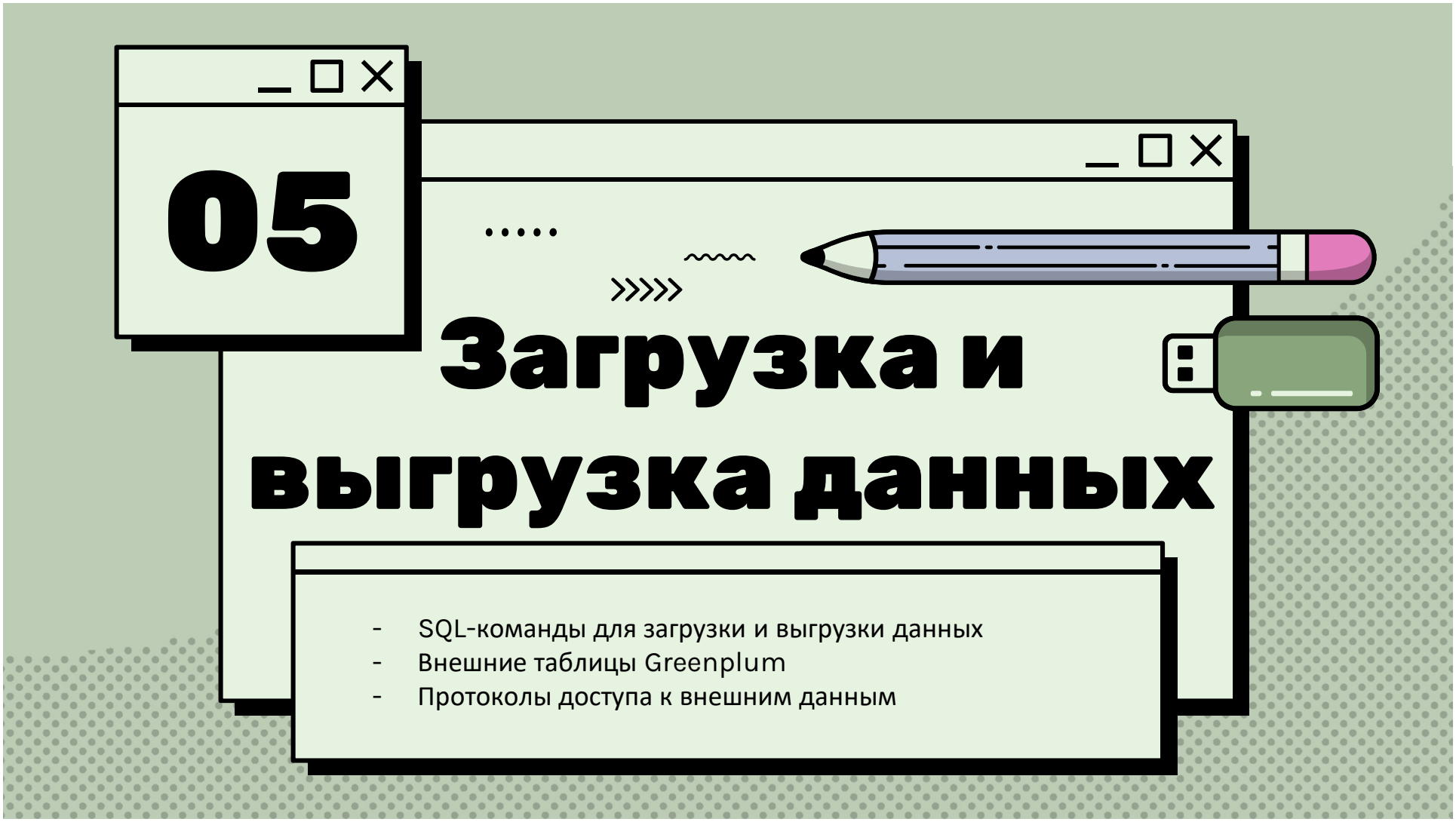
Партицирование

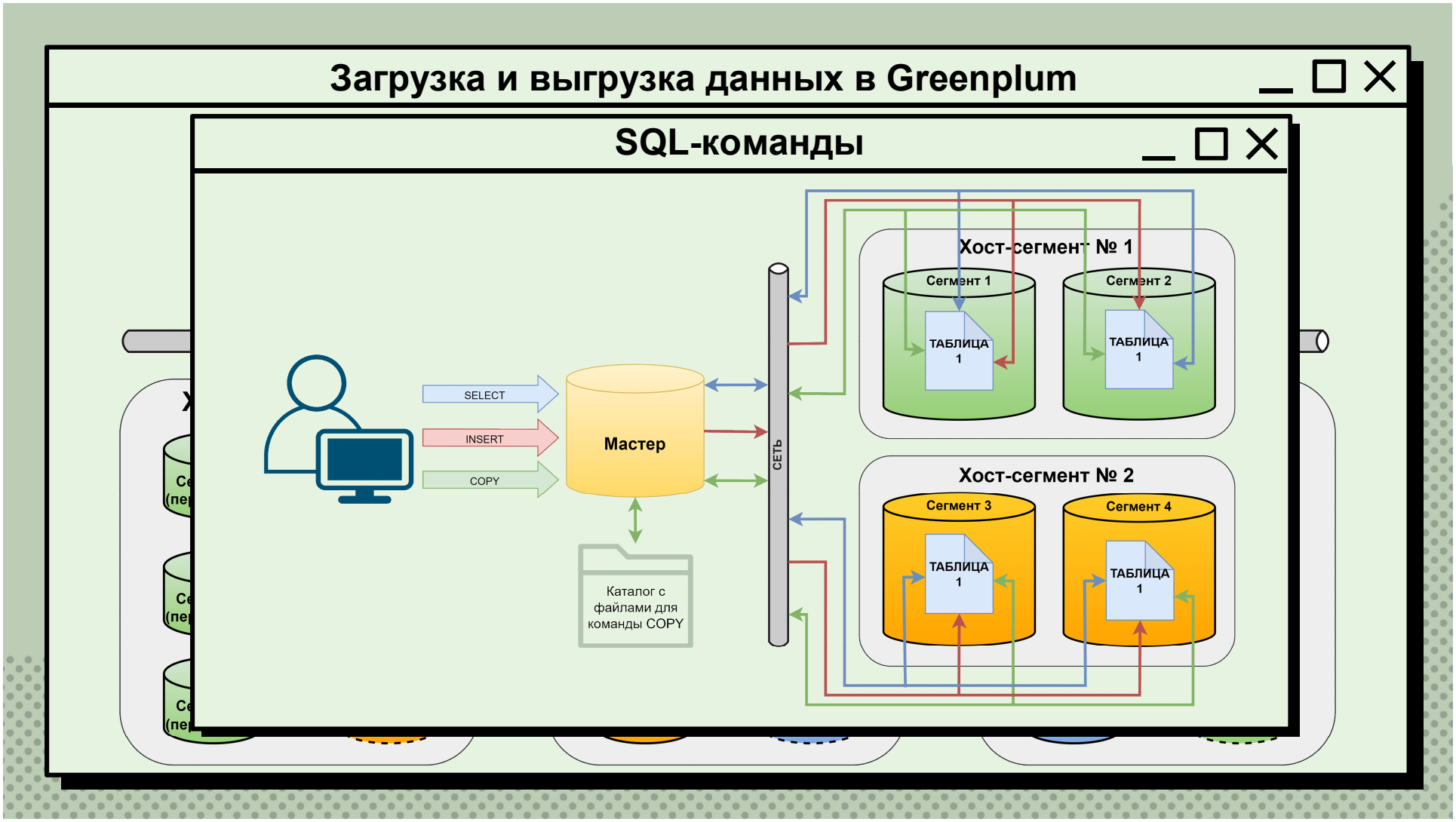
Индекс
в табли
произво

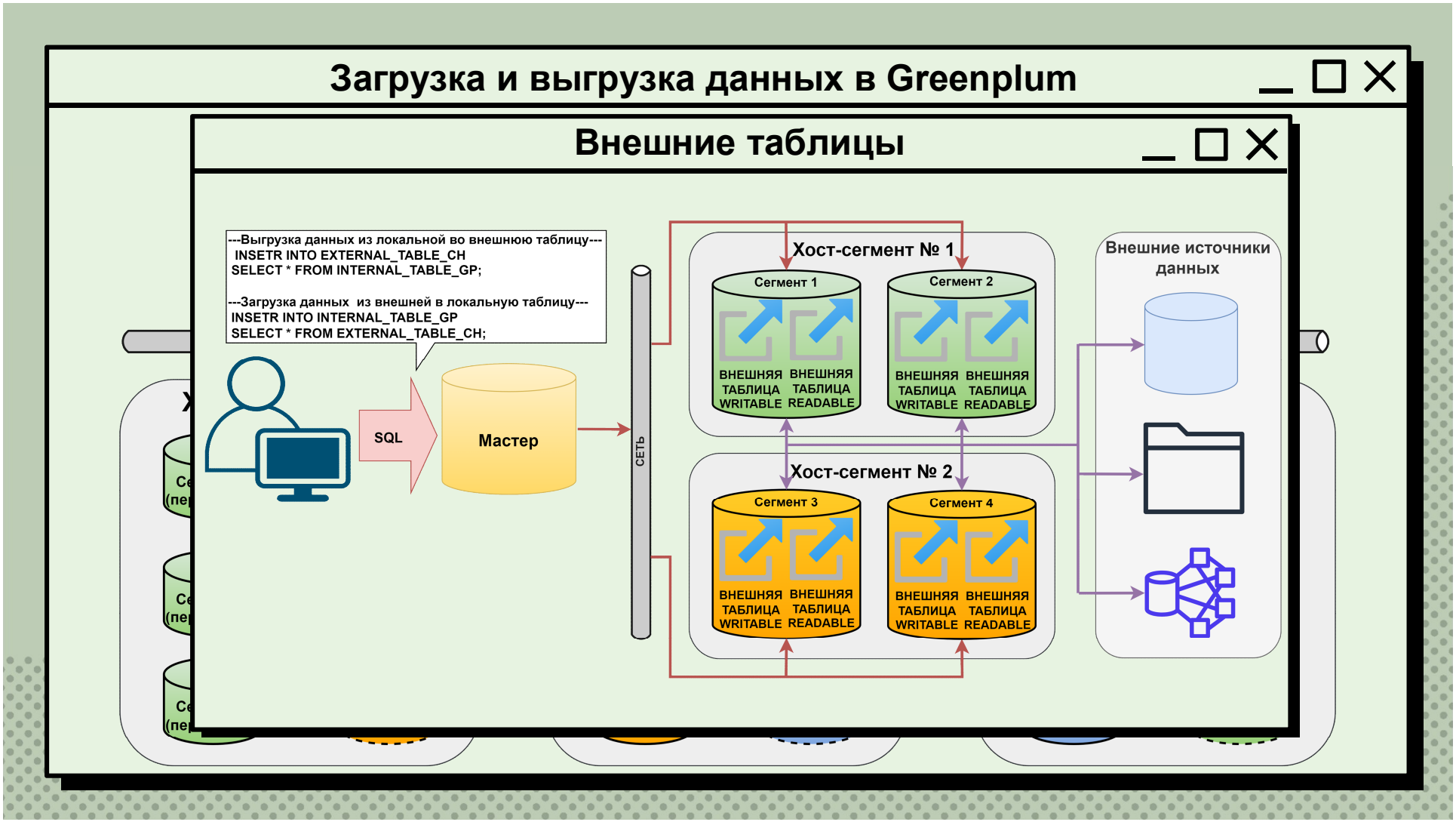
ных

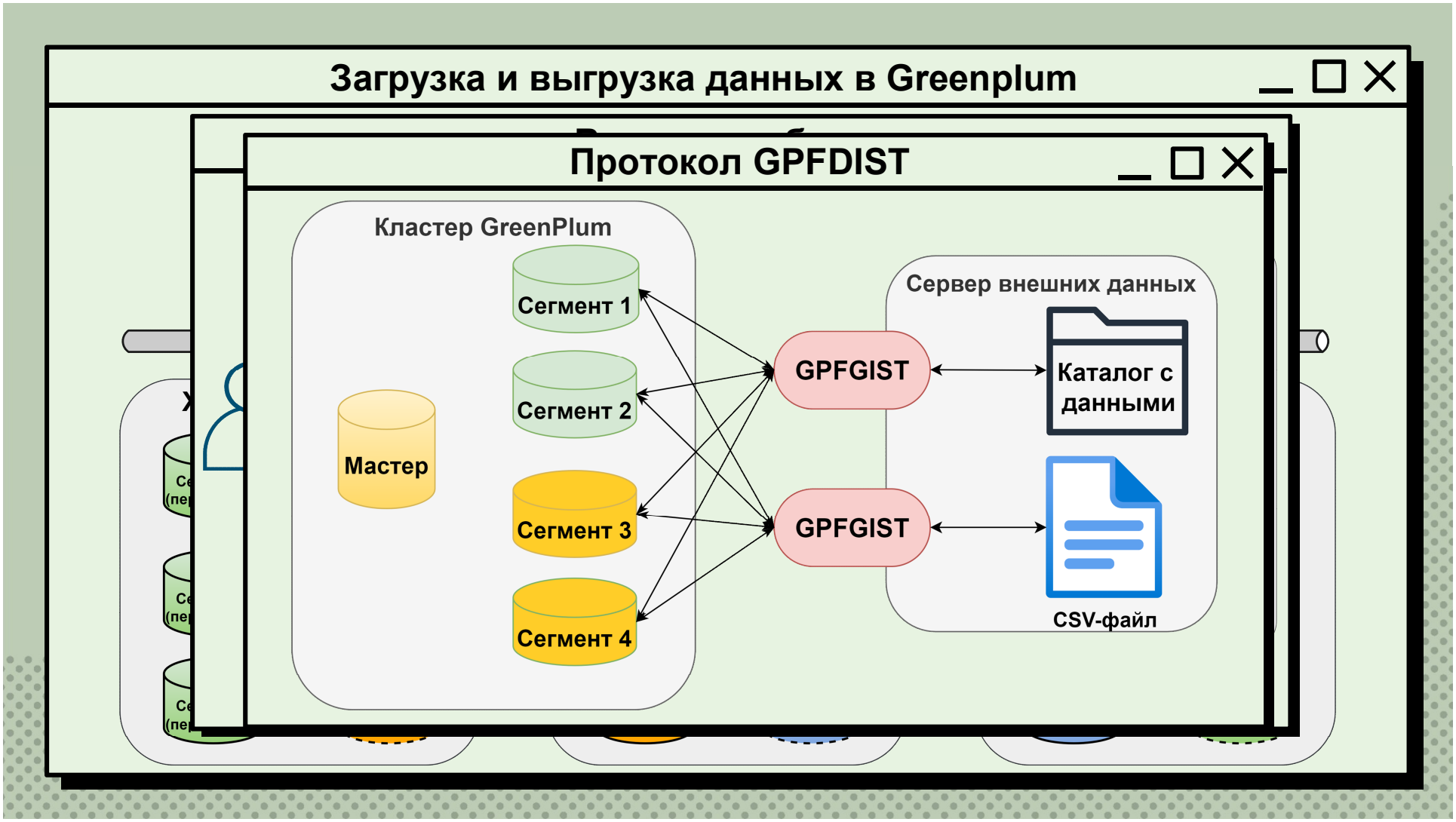
B-tree индекс

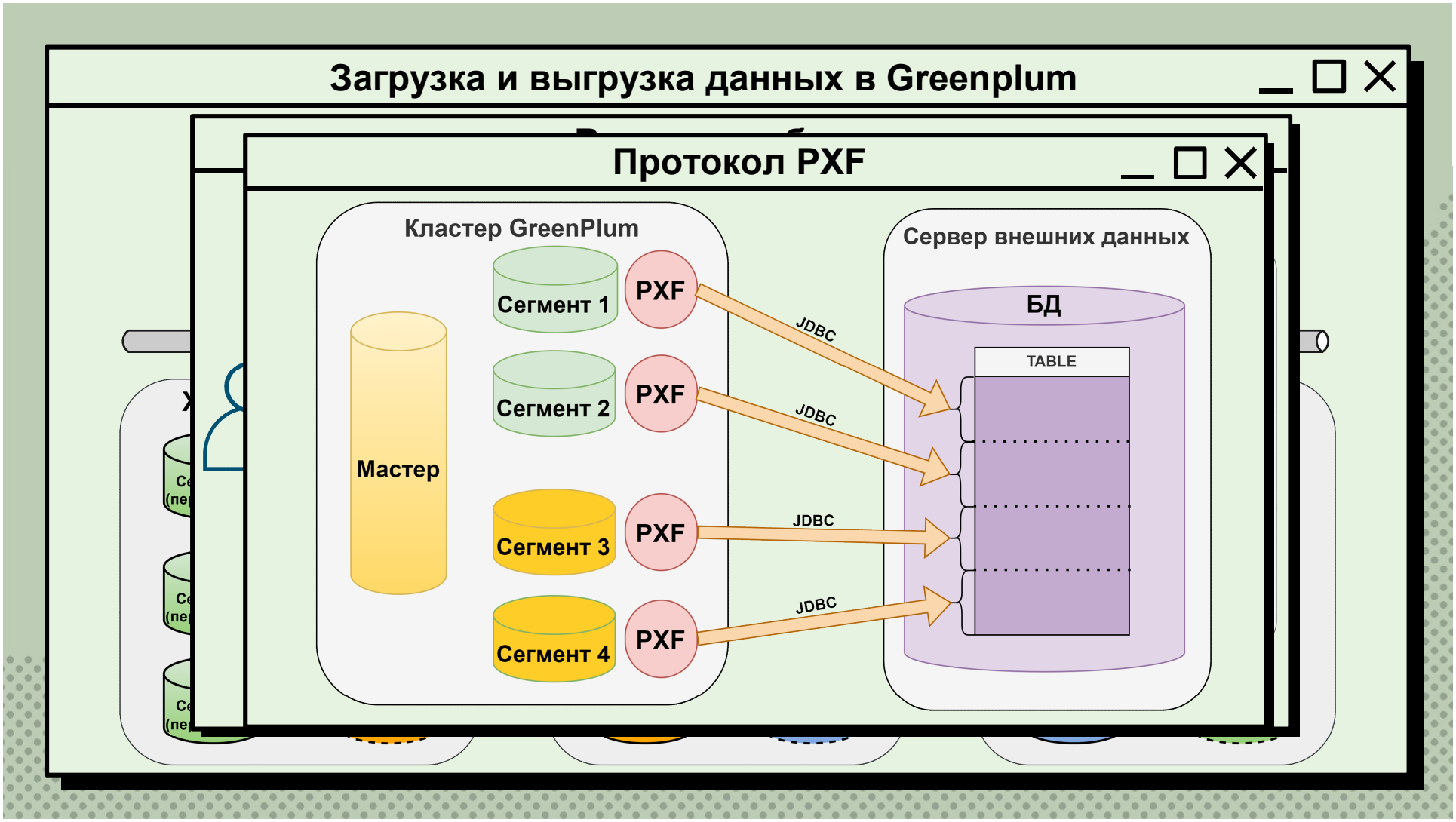


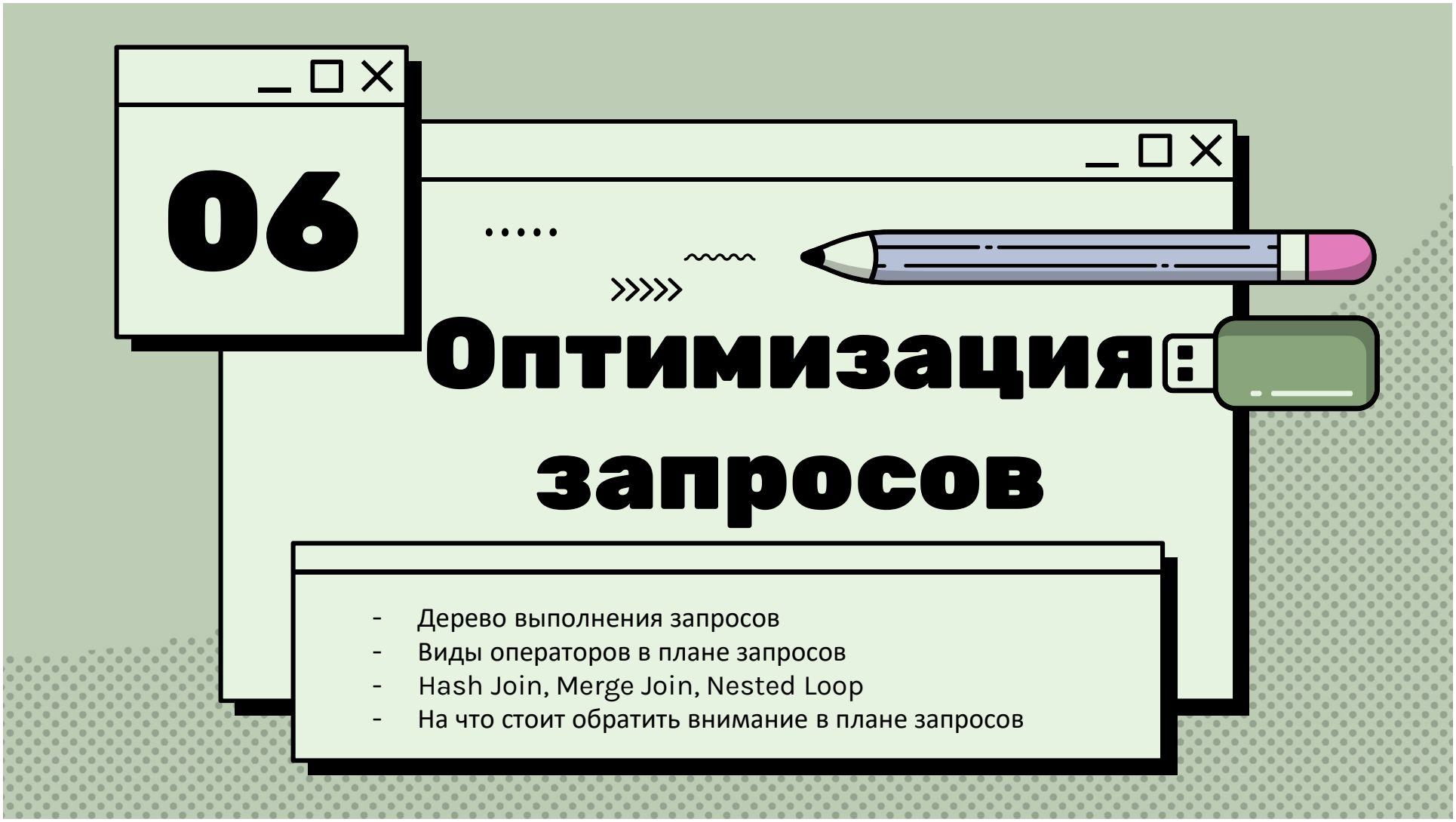

05
Загрузка и выгрузка данных
- SQL-команды для загрузки и выгрузки данных
- Внешние таблицы Greenplum
- Протоколы доступа к внешним данным


Загрузка и выгрузка данных в Greenplum
SQL-команды
SELECT
INSERT
COPY
Мастер
Каталог с файлами для команды COPY
СЕТЬ
Хост-сегмент № 1
Сегмент 1
Сегмент 2
ТАБЛИЦА 1
ТАБЛИЦА 1
Хост-сегмент № 2
Сегмент 3
Сегмент 4
ТАБЛИЦА 1
ТАБЛИЦА 1


Загрузка и выгрузка данных в Greenplum
Внешние таблицы
---Выгрузка данных из локальной во внешнюю таблицу---
INSETR INTO EXTERNAL_TABLE_CH
SELECT * FROM INTERNAL_TABLE_GP;
---Загрузка данных из внешней в локальную таблицу---
INSETR INTO INTERNAL_TABLE_GP
SELECT * FROM EXTERNAL_TABLE_CH;
SQL
Мастер
СЕТЬ
Хост-сегмент № 1
Сегмент 1
Сегмент 2
ВНЕШНЯЯ ТАБЛИЦА WRITABLE
ВНЕШНЯЯ ТАБЛИЦА READABLE
Хост-сегмент № 2
Сегмент 3
Сегмент 4
Внешние источники данных


Загрузка и выгрузка данных в Greenplum
Протокол GPFDIST
Кластер GreenPlum
Сегмент 1
Сегмент 2
Мастер
Сегмент 3
Сегмент 4
Сервер внешних данных
GPFGIST
GPFGIST
Каталог с данными
CSV-файл


Загрузка и выгрузка данных в Greenplum
Протокол PXF
Кластер GreenPlum
Мастер
Сегмент 1
Сегмент 2
Сегмент 3
Сегмент 4
PXF
PXF
PXF
PXF
JDBC
JDBC
JDBC
JDBC
Сервер внешних данных
БД
TABLE


06
Оптимизация запросов
- Дерево выполнения запросов
- Виды операторов в плане запросов
- Hash Join, Merge Join, Nested Loop
- На что стоит обратить внимание в плане запросов

Дерево выполнения запроса

EXPLAIN

```
QUERY PLAN
Gather Motion 24:1  (slice1; segments: 24)  (cost=0.00..1293.11 rows=190 width=98)
  ->  Hash Left Join  (cost=0.00..1293.06 rows=8 width=98)
        Hash Cond: (info_ref_vitrin_ratan.user_sk = info_ref_vitrin_ratan_2.user_sk)
        Join Filter: (info_ref_vitrin_ratan_2.date >= (min(info_ref_vitrin_ratan.date)))
        ->  Hash Left Join  (cost=0.00..862.05 rows=8 width=102)
              Hash Cond: (info_ref_vitrin_ratan.user_sk = info_ref_vitrin_ratan_1.user_sk)
              Join Filter: (info_ref_vitrin_ratan_1.date >= (min(info_ref_vitrin_ratan.date)))
              ->  HashAggregate  (cost=0.00..431.01 rows=8 width=90)
                    Group Key: info_ref_vitrin_ratan.user_sk, info_ref_vitrin_ratan.client_region, info_ref_vitrin_ratan.client_a
                    ->  Seq Scan on info_ref_vitrin_ratan  (cost=0.00..431.01 rows=8 width=90)
                          Filter: issued_conversions
              ->  Hash  (cost=431.01..431.01 rows=34 width=12)
                    ->  HashAggregate  (cost=0.00..431.01 rows=34 width=12)
                          Group Key: info_ref_vitrin_ratan_1.user_sk, info_ref_vitrin_ratan_1.date
                          ->  Seq Scan on info_ref_vitrin_ratan info_ref_vitrin_ratan_1  (cost=0.00..431.01 rows=34 width=12)
                                Filter: conversions
        ->  Hash  (cost=431.01..431.01 rows=8 width=12)
              ->  GroupAggregate  (cost=0.00..431.01 rows=8 width=12)
                    Group Key: info_ref_vitrin_ratan_2.user_sk, info_ref_vitrin_ratan_2.date
                    ->  Sort  (cost=0.00..431.01 rows=8 width=12)
                          Sort Key: info_ref_vitrin_ratan_2.user_sk, info_ref_vitrin_ratan_2.date
                          ->  Seq Scan on info_ref_vitrin_ratan info_ref_vitrin_ratan_2  (cost=0.00..431.00 rows=8 width=12)
                                Filter: issued_conversions
Optimizer: Pivotal Optimizer (GPORCA)
```

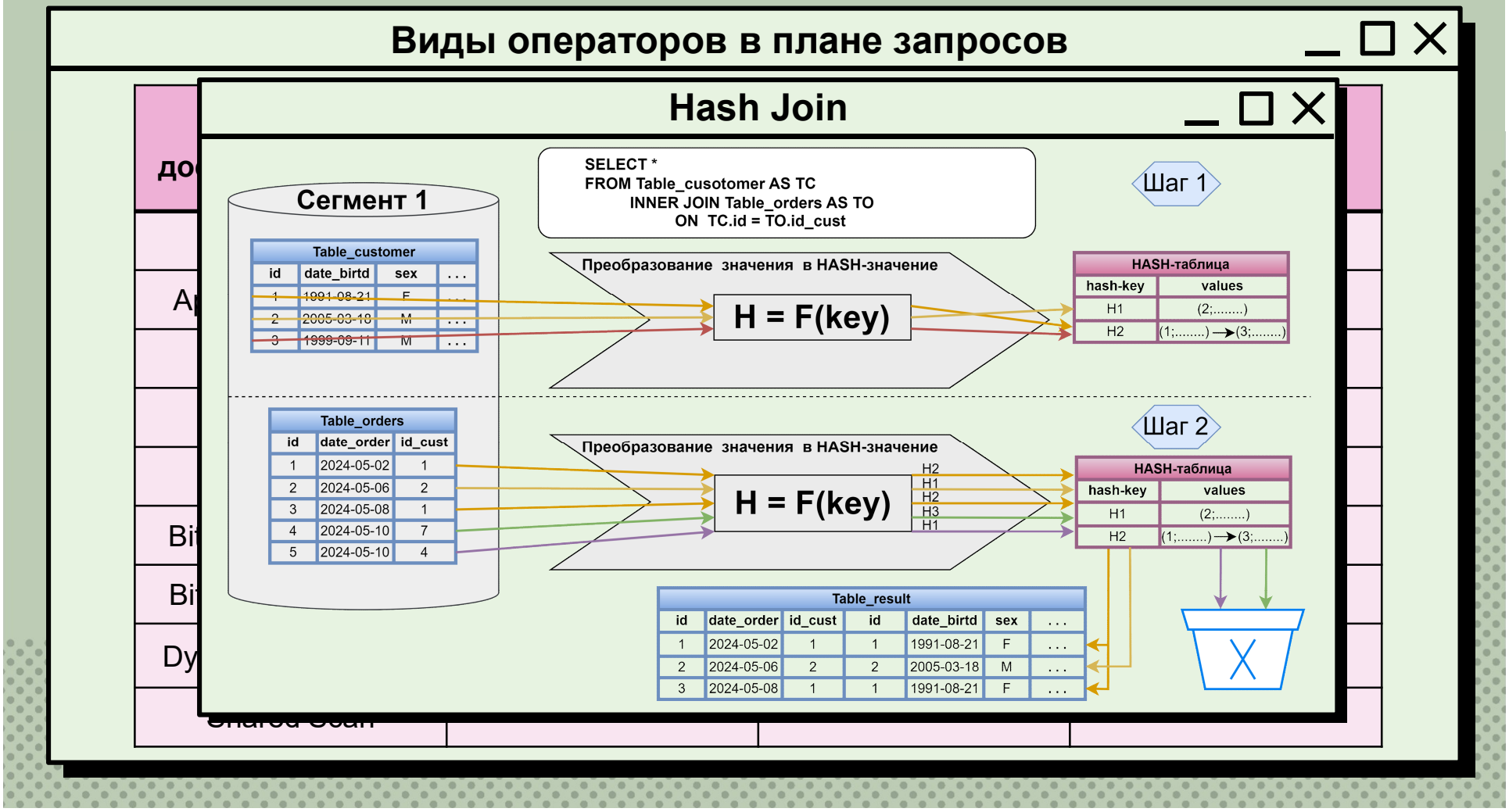
EXPLAIN ALALYZE

```
QUERY PLAN
Gather Motion 24:1  (slice1; segments: 24)  (cost=0.00..1293.11 rows=190 width=98) (actual time=1.819..2.266 rows=561 loops=1)
  ->  Hash Left Join  (cost=0.00..1293.06 rows=8 width=98) (actual time=0.938..1.162 rows=70 loops=1)
        Hash Cond: (info_ref_vitrin_ratan.user_sk = info_ref_vitrin_ratan_2.user_sk)
        Join Filter: (info_ref_vitrin_ratan_2.date >= (min(info_ref_vitrin_ratan.date)))
        Extra Text: (seg12)  Hash chain length 2.4 avg, 7 max, using 5 of 65536 buckets.Hash chain length 2.4 avg, 8 max, using 17 of 65536 buckets.Hash chain leng

        Extra Text: (seg19)  Hash chain length 1.9 avg, 6 max, using 7 of 65536 buckets.Hash chain length 2.4 avg, 7 max, using 5 of 65536 buckets.Hash chain lengt

        ->  Hash Left Join  (cost=0.00..862.05 rows=8 width=102) (actual time=0.582..0.693 rows=18 loops=1)
              Hash Cond: (info_ref_vitrin_ratan.user_sk = info_ref_vitrin_ratan_1.user_sk)
              Join Filter: (info_ref_vitrin_ratan_1.date >= (min(info_ref_vitrin_ratan.date)))
              Extra Text: (seg21)  Hash chain length 2.0 avg, 7 max, using 18 of 65536 buckets.Hash chain length 1.0 avg, 1 max, using 8 of 32 buckets; total 0 expansic

              Extra Text: (seg23)  Hash chain length 2.4 avg, 8 max, using 17 of 65536 buckets.Hash chain length 2.0 avg, 7 max, using 18 of 65536 buckets.Hash chain

              ->  HashAggregate  (cost=0.00..431.01 rows=8 width=90) (actual time=0.145..0.153 rows=8 loops=1)
                    Group Key: info_ref_vitrin_ratan.user_sk, info_ref_vitrin_ratan.client_region, info_ref_vitrin_ratan.client_employment_type, info_ref_vitrin_ratan.client_
                    Extra Text: (seg8)  Hash chain length 1.0 avg, 1 max, using 8 of 32 buckets; total 0 expansions.

                    ->  Seq Scan on info_ref_vitrin_ratan  (cost=0.00..431.01 rows=8 width=90) (actual time=0.094..0.123 rows=16 loops=1)
                          Filter: issued_conversions
              ->  Hash  (cost=431.01..431.01 rows=34 width=12) (actual time=0.128..0.128 rows=40 loops=1)
                    ->  HashAggregate  (cost=0.00..431.01 rows=34 width=12) (actual time=0.097..0.113 rows=40 loops=1)
                          Group Key: info_ref_vitrin_ratan_1.user_sk, info_ref_vitrin_ratan_1.date
                          Extra Text: (seg23)  Hash chain length 2.1 avg, 5 max, using 19 of 32 buckets; total 0 expansions.

                          ->  Seq Scan on info_ref_vitrin_ratan info_ref_vitrin_ratan_1  (cost=0.00..431.01 rows=34 width=12) (actual time=0.029..0.052 rows=58 loops=1)
                                Filter: conversions
        ->  Hash  (cost=431.01..431.01 rows=8 width=12) (actual time=0.125..0.125 rows=13 loops=1)
              ->  GroupAggregate  (cost=0.00..431.01 rows=8 width=12) (actual time=0.108..0.121 rows=13 loops=1)
                    Group Key: info_ref_vitrin_ratan_2.user_sk, info_ref_vitrin_ratan_2.date
                    ->  Sort  (cost=0.00..431.01 rows=8 width=12) (actual time=0.102..0.106 rows=16 loops=1)
                          Sort Key: info_ref_vitrin_ratan_2.user_sk, info_ref_vitrin_ratan_2.date
                          Sort Method:  quicksort  Memory: 792kB
                          ->  Seq Scan on info_ref_vitrin_ratan info_ref_vitrin_ratan_2  (cost=0.00..431.00 rows=8 width=12) (actual time=0.051..0.069 rows=16 loops=1)
                                Filter: issued_conversions
Planning time: 44.778 ms
  (slice0)    Executor memory: 1930K bytes.
  (slice1)    Executor memory: 2823K bytes avg x 24 workers, 2838K bytes max (seg12).  Work_mem: 33K bytes max.
Memory used:  126976kB
Optimizer: Pivotal Optimizer (GPORCA)
Execution time: 4.555 ms
```

Виды операторов в плане запросов

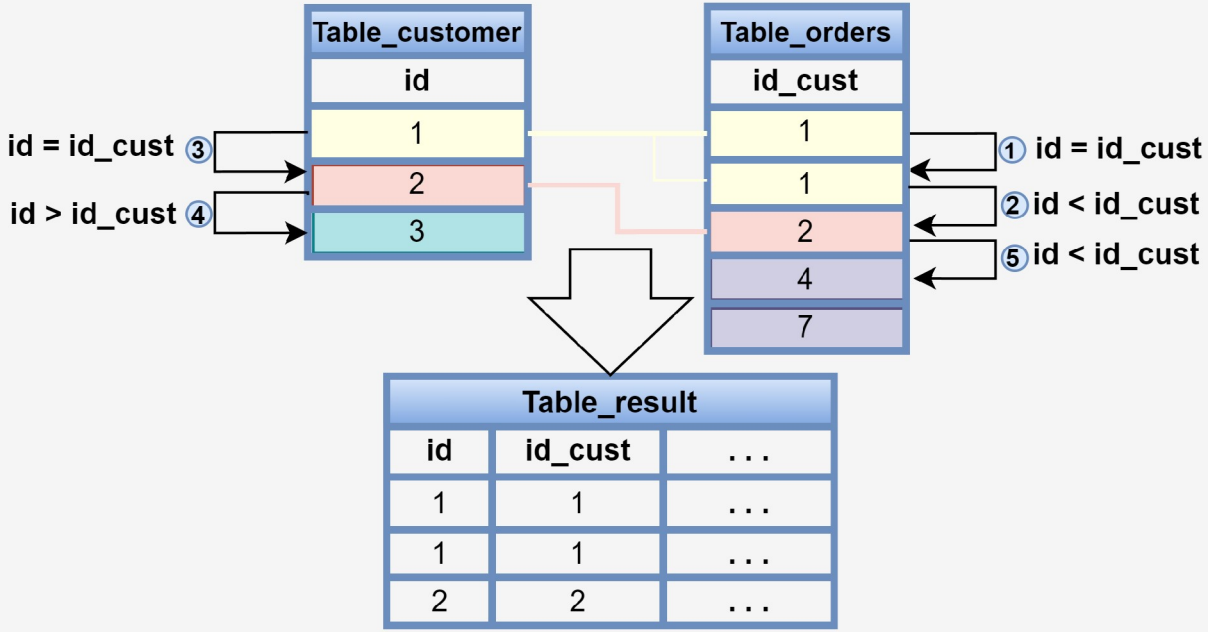
Операторы доступа к данным	Операторы преобразований данных	Операторы перемещения данных	Служебные операторы
Seq Scan	Limit	Redistridute Motion	Result
AppendOnly Scan	Sort	Broadcast Motion	Append
Column Scan	Hash Join	Gather Motion	Sequence
External Scan	Merge Join		Subplan
Index Scan	Nested Loop Join		Materialize
Bitmap Index Scan	HashAggregate		Partition Selector
Bitmap Heap Scan	GroupAggregate		Hash
Dynamyc Seq Scan	WindowAgg		
Shared Scan			

Виды операторов в плане запросов



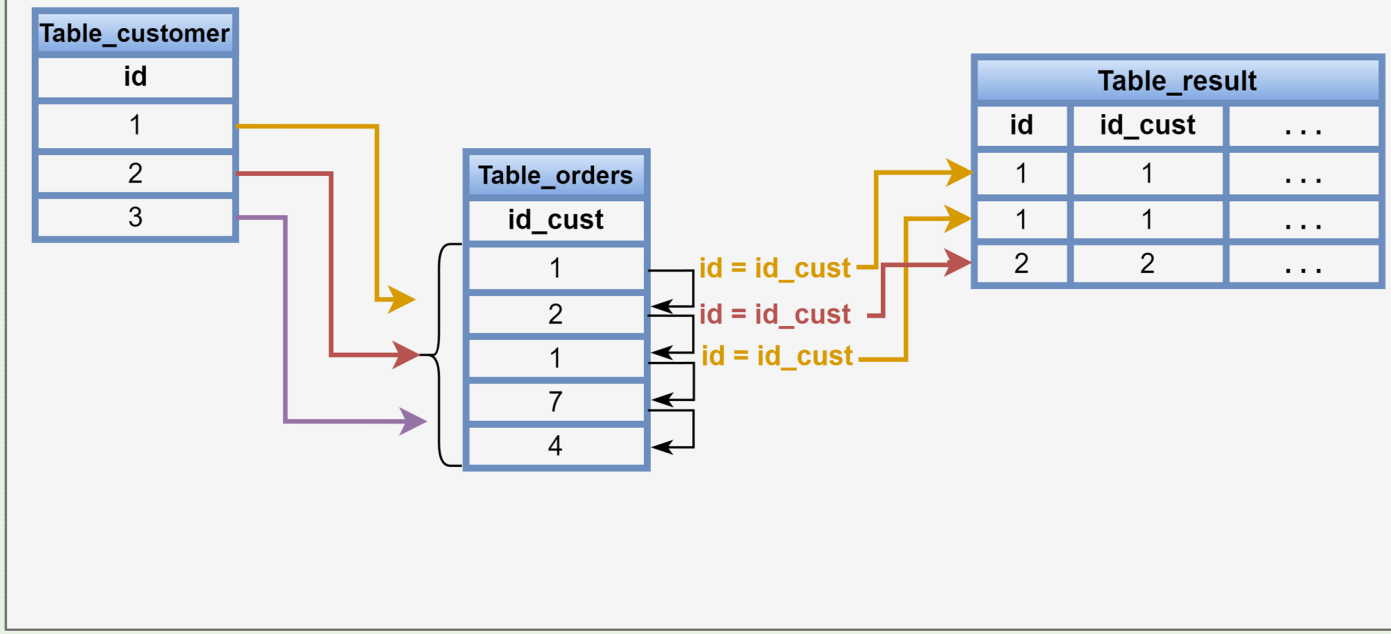
Виды операторов в плане запросов

Merge Join



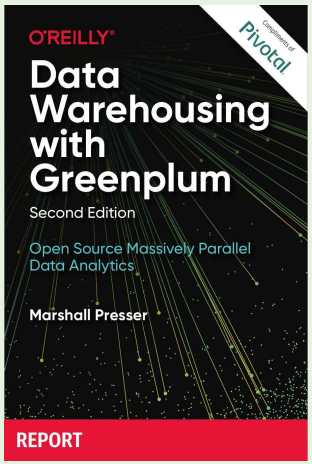
Виды операторов в плане запросов

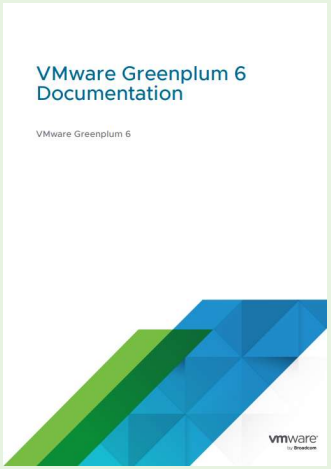
Nested Loop



Литература и ROADMAP-проект

Литература





Roadmap