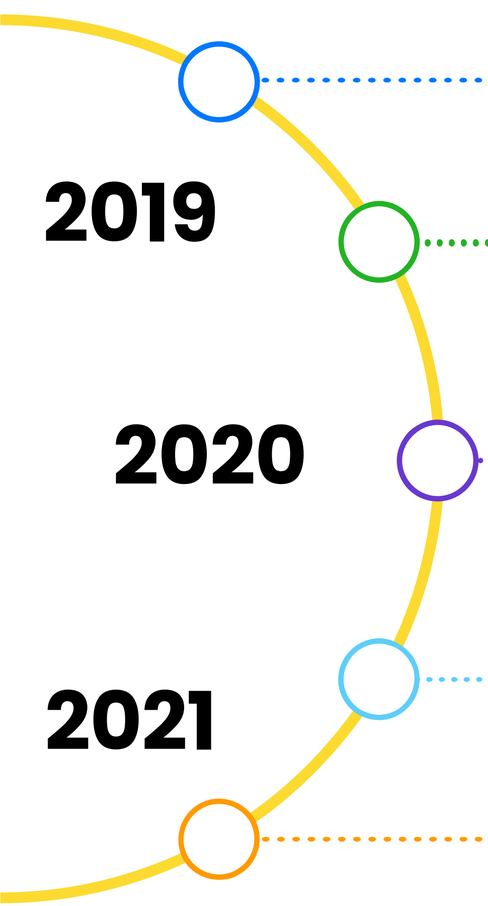

Яндекс Go

Яндекс Go

Anchor Modeling и GP: как мечты разбиваются о реальность

Евгений Ермаков, руководитель DWH
Николай Гребенщиков, руководитель команды DE RideTech

Содержание



Вступление

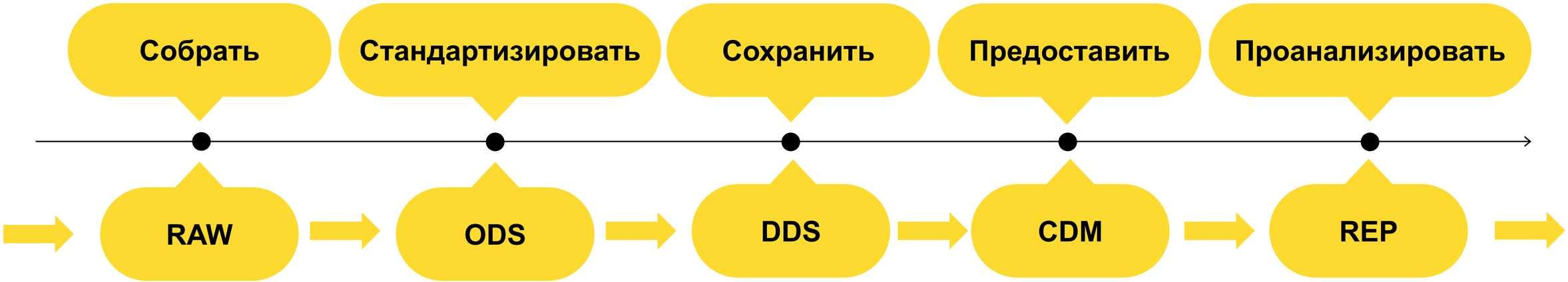
I

2019

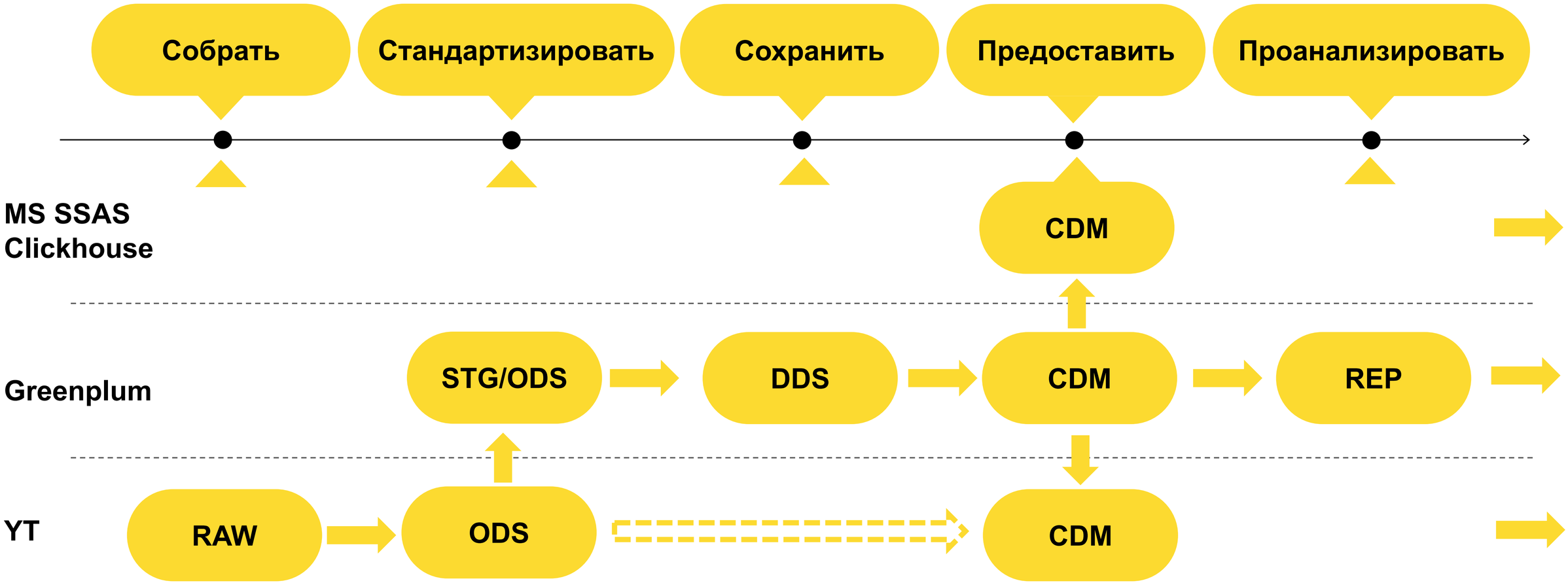
- Архитектура DWH
- Детальный слой
- Подходы к детальному слою
- Data Vault vs Anchor modeling

Архитектура слоев данных

Архитектура слоев данных



Архитектура слоев данных



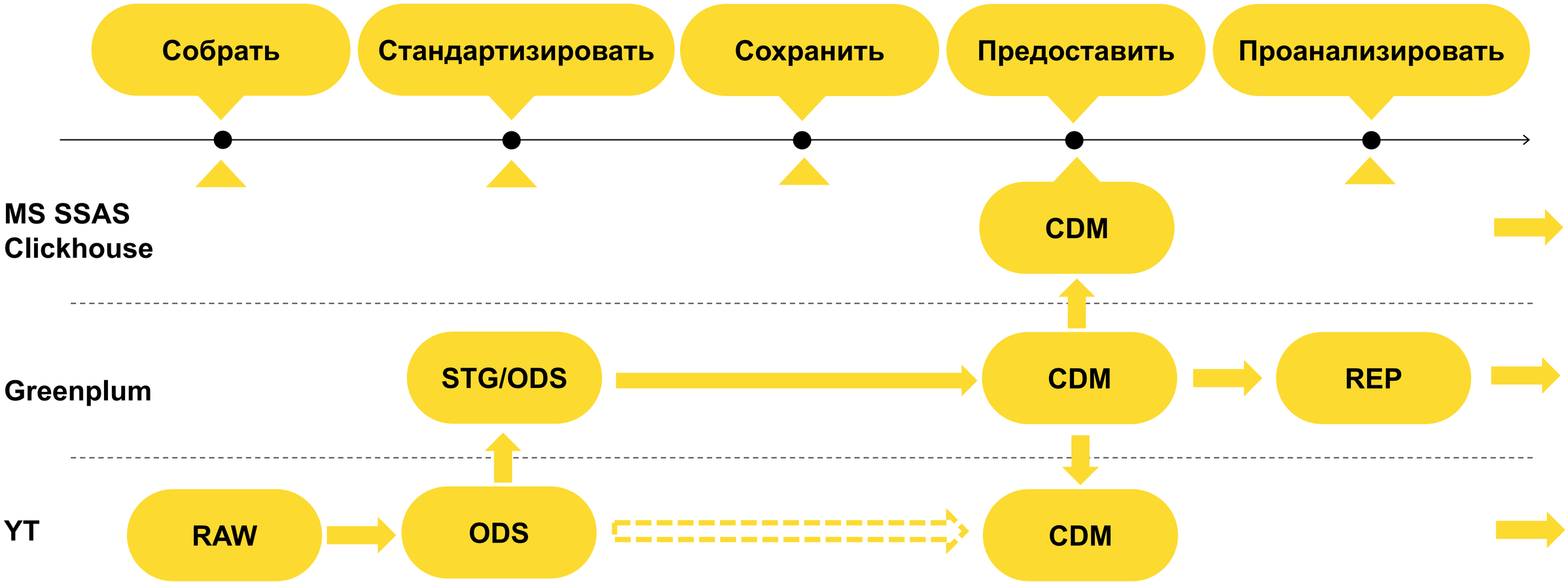
Детальный слой

Детальный слой – ключевой для построения доменной модели

- Хранит историю изменений сущностей и связей между ними
- Отвечает за консолидацию данных между источниками
- Устойчив к изменению в бизнесе (?)
- Модульный и масштабируемый (?)

Greenplum

Архитектура слоев данных



Подходы к проектированию

сложно эксплуатировать, просто вносить изменений

Никакого

- Денормализация до 1НФ
- Можно использовать без подготовки
- Неустойчиво к изменениям
- Дублирование информации
- Нет join

Звезда и снежинка

- Нормализация до 3НФ
- Можно использовать с минимальной подготовкой
- Неудобно перестраивать
- Минимальное дублирование информации
- Приемлемое количество join

легко эксплуатировать, сложно вносить изменения

Подходы к проектированию

сложно эксплуатировать, просто вносить изменений

Никакого	Звезда и снежинка	Data Vault	Anchor modeling
› Денормализация до 1НФ	› Нормализация до 3НФ	› Строгая нормализация	› Ультра строгая нормализация
› Можно использовать без подготовки	› Можно использовать с минимальной подготовкой	› Нельзя использовать без подготовки	› Нельзя использовать без подготовки
› Неустойчиво к изменениям	› Неудобно перестраивать	› Не надо перестраивать (с ограничениями)	› Не надо перестраивать
› Дублирование информации	› Минимальное дублирование информации	› Минимальное дублирование информации	› Нет дублирования информации
› Нет join	› Приемлемое количество join	› Большое количество join	› Ультра количество join

легко эксплуатировать, сложно вносить изменения

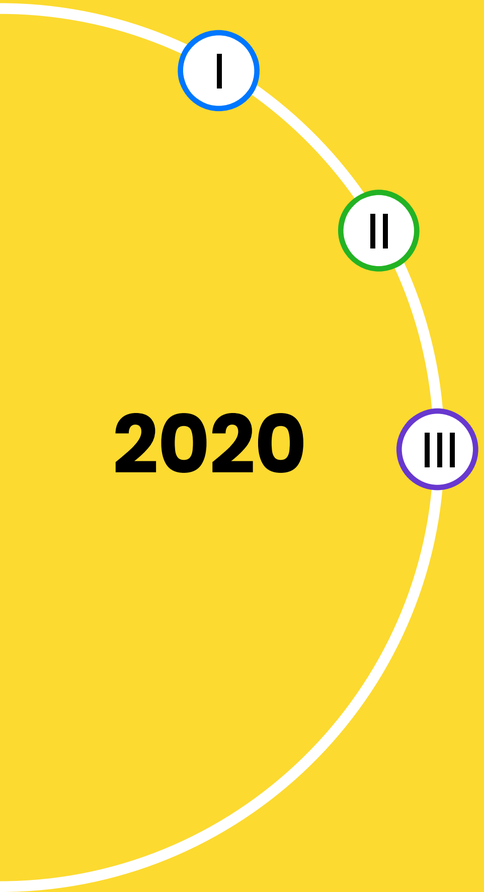
Подходы к проектированию

сложно эксплуатировать, просто вносить изменений

Никакого	Звезда и снежинка	Data Vault	**Anchor modeling**
› Денормализация до 1НФ	› Нормализация до 3НФ	› Строгая нормализация	› Ультра строгая нормализация
› Можно использовать без подготовки	› Можно использовать с минимальной подготовкой	› Нельзя использовать без подготовки	› Нельзя использовать без подготовки
› Неустойчиво к изменениям	› Неудобно перестраивать	› Не надо перестраивать (с ограничениями)	› Не надо перестраивать
› Дублирование информации	› Минимальное дублирование информации	› Минимальное дублирование информации	› Нет дублирования информации
› Нет join	› Приемлемое количество join	› Большое количество join	› Ультра количество join

легко эксплуатировать, сложно вносить изменения

Забегая вперед…



2020

~~Попытка #1~~

Data Vault и Anchor Modeling

https://learndatavault.com/

Dan Linstedt

http://www.anchormodeling.com/

Lars Rönnbäck

Обе методологии:

- Повышают градус нормализации выше 3НФ
- Вводят свои типы таблиц и накладывают жесткие ограничение на их использование
- При использовании создают **over 9000 таблиц**

Взамен обещают:

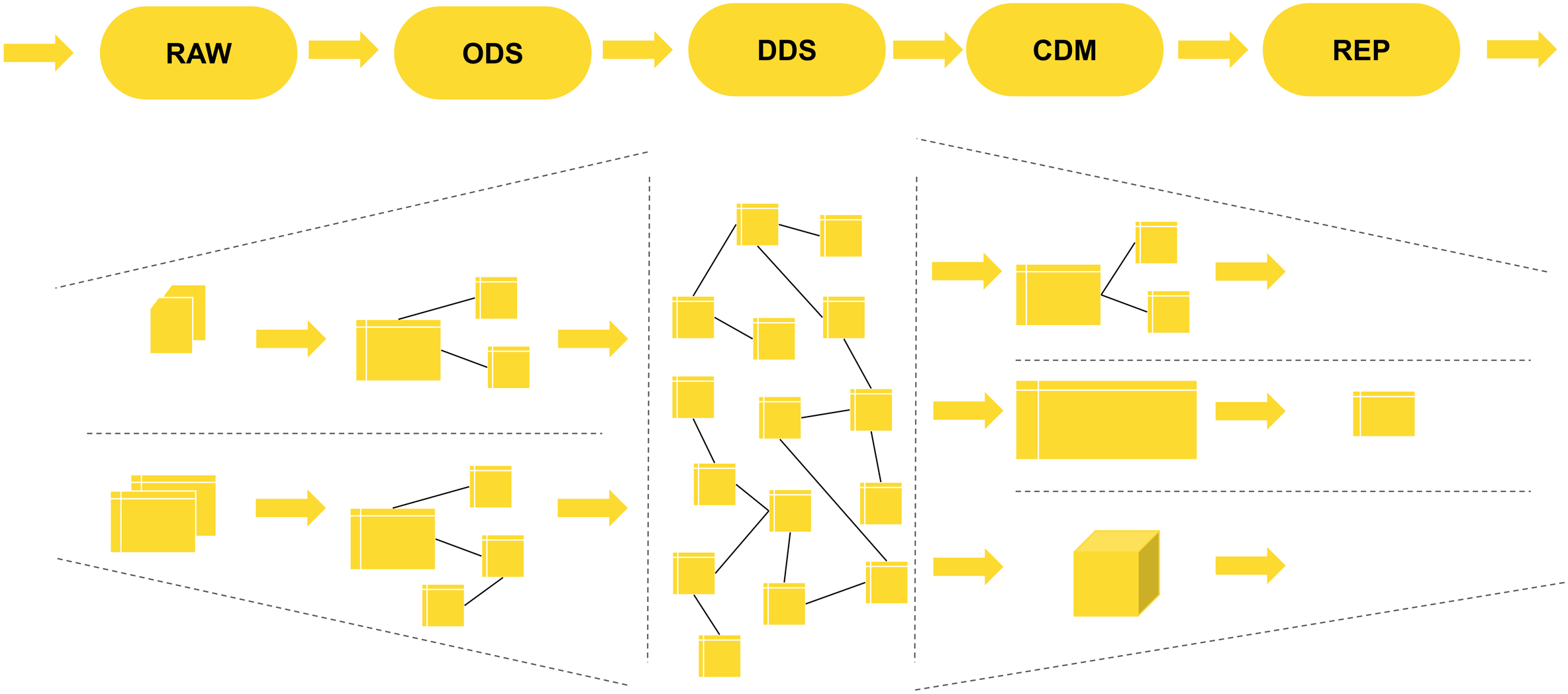
- Уменьшить постоянное дублирования данных в SCD2 от изменения всего одного атрибута
- Избавить от деструктивных изменений, только расширение модели (даже при изменении кардинальности связи, боль классического хранилища)
- Позволить дорабатывать хранилище легко и быстро (мистика)


Learn Data Vault
Business Information Intelligence
ANCHOR™


Learn Data Vault
Business Information Intelligence
ANCHOR™


Создать свою
модель

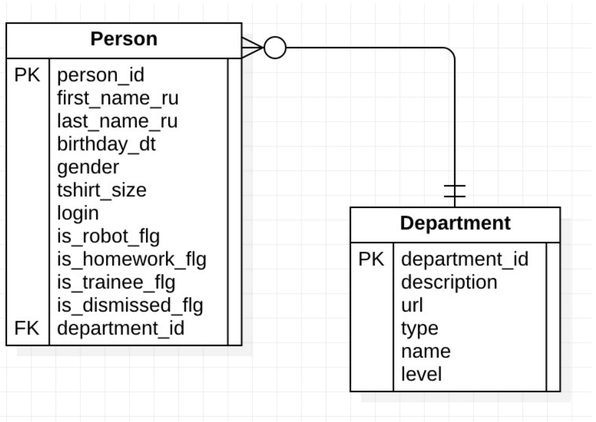
Highly Normalized Hybrid Model (hNhM)



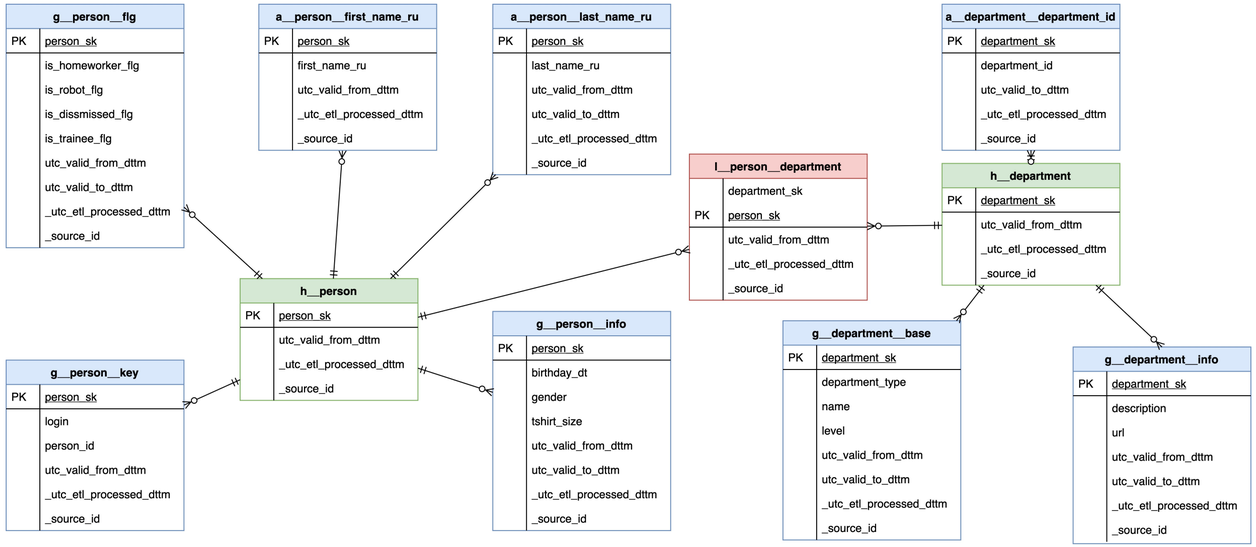
Логический и физический уровень

Закрепить разделение уровней проектирования в методологии

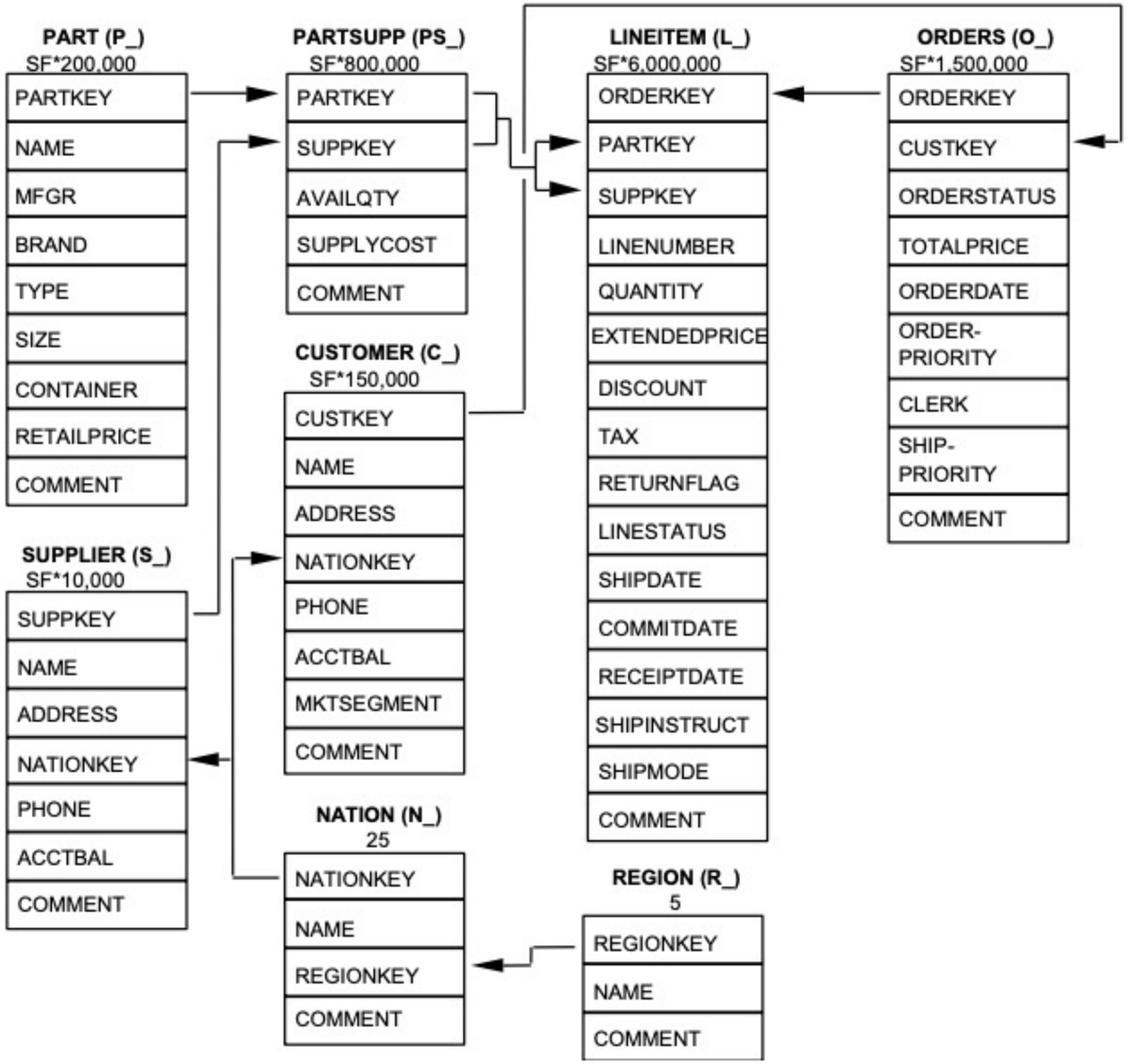
Логический уровень



Физический уровень

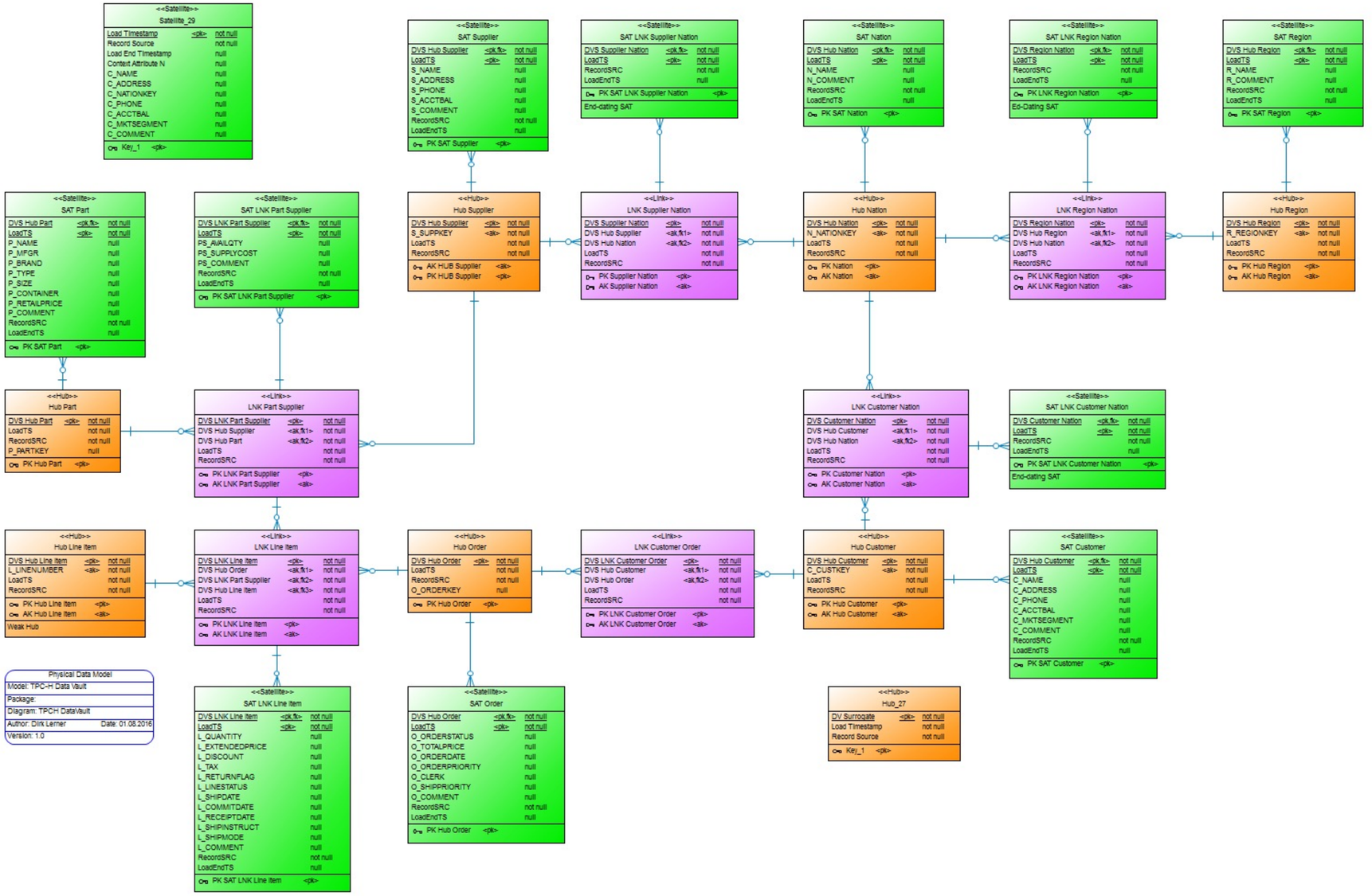


TPC-H



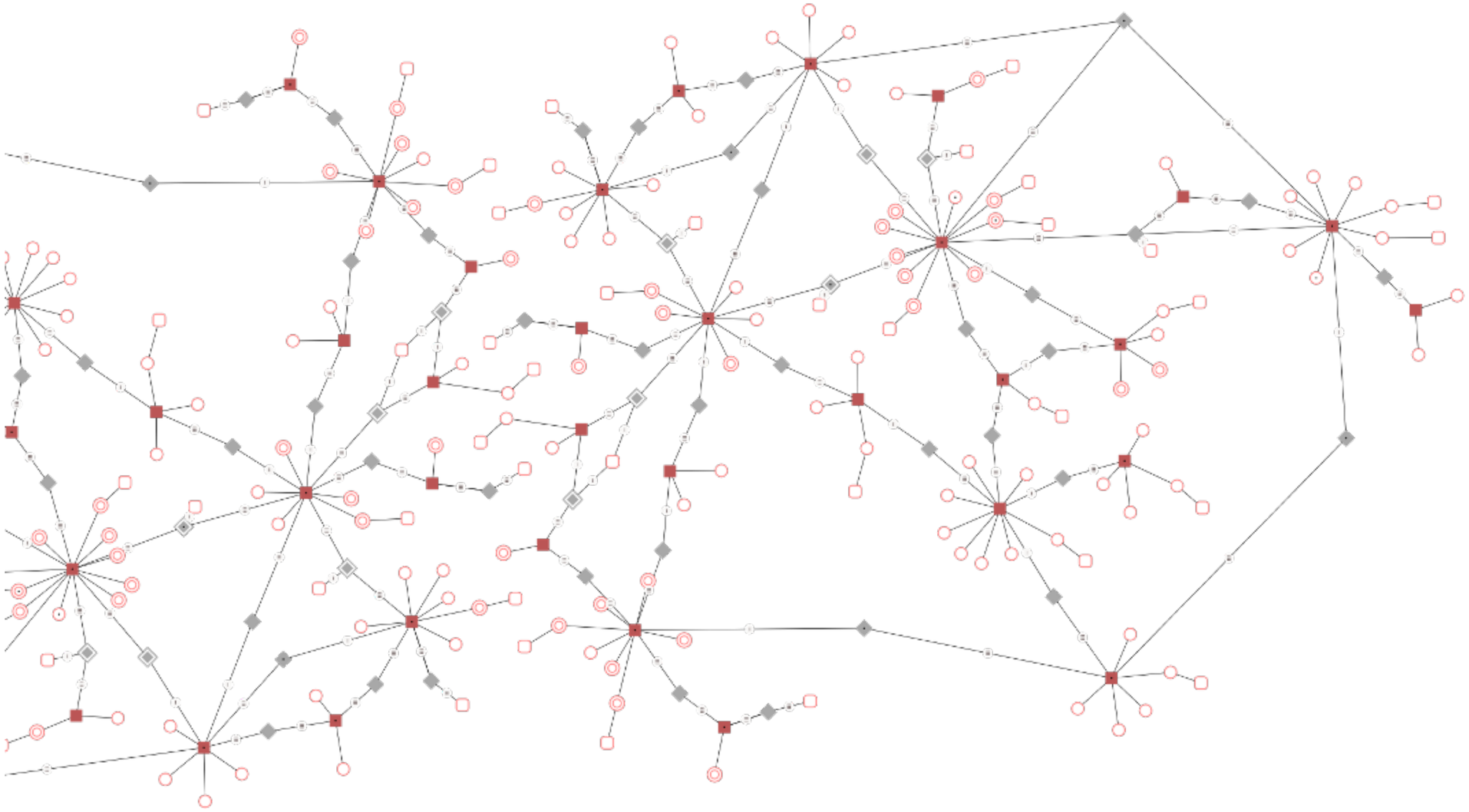
http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-h_v2.17.1.pdf

TPC-H DV



https://tedamoh.com/data-vault/79-data-vault-on-exasol-modeling-and-implementation

TPC-H AM



Возвращаясь назад…

I

2019

Постановка задачи

Цель: применить якорную модель к детальному слою в Greenplum

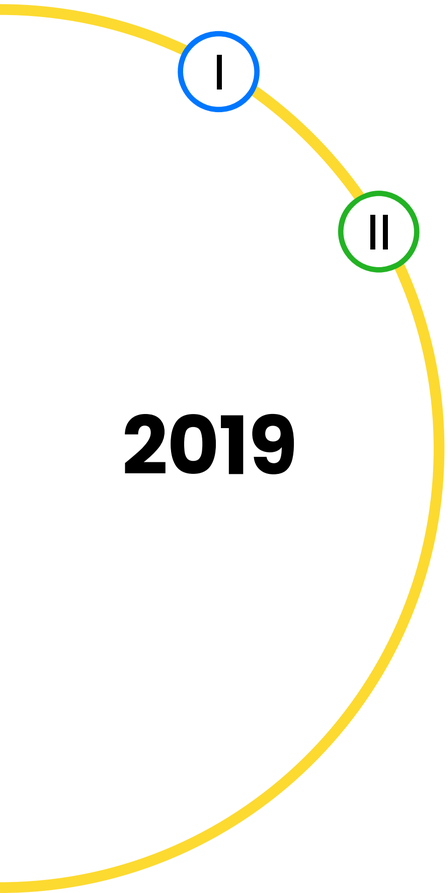
Дано:

- Рабочий кластер Greenplum5
- Понимание необходимости детального слоя
- Желание идти в ногу со временем
- Отвага и решительность

Задачи:

- Провести анализ технических оптимизаций для AM
- Проанализировать их наличие в GreenPlum5
- Собрать работающий AM
- Перевести на него построение витрин и аналитиков

Теория. Возможности Greenplum

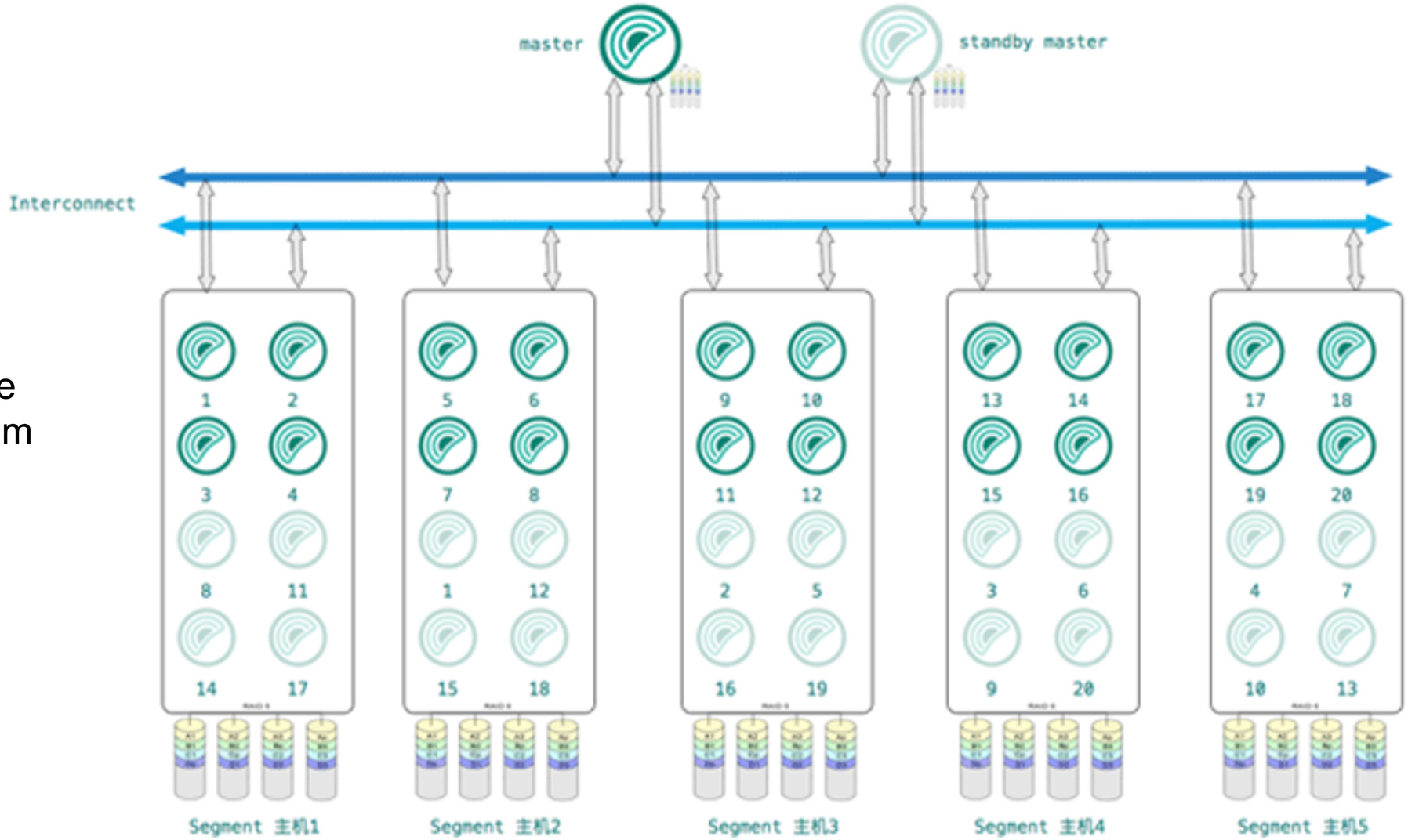


2019

- MPP RDBMS
- Heap vs AO
- Legacy vs Orca

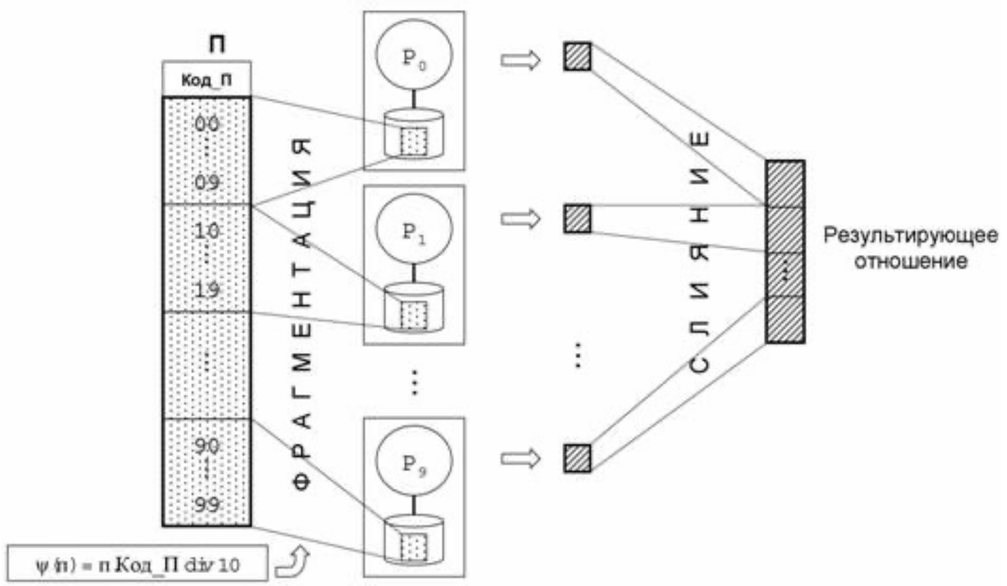
Greenplum = SN MPP RDBMS

- Share Nothing
- Massive Parallel Processing
- Relational database management system

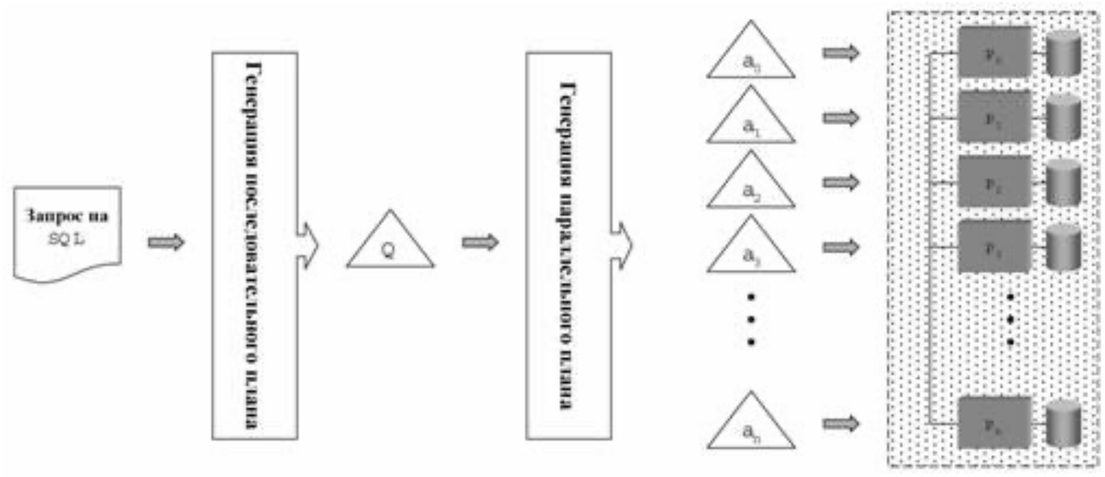


Массивно-параллельная обработка

Фрагментный параллелизм



Генерация параллельного плана запросов



Дистрибьюция

```
CREATE TABLE foo (a int, b text)
DISTRIBUTED BY (a);
```

```
CREATE TABLE bar (a int, b text)
WITH (appendoptimized=true)
DISTRIBUTED BY (a);
```

Формат хранения таблиц

```
CREATE TABLE foo (a int, b text)
DISTRIBUTED BY (a);
```

```
CREATE TABLE bar (a int, b text)
WITH (appendoptimized=true)
DISTRIBUTED BY (a);
```

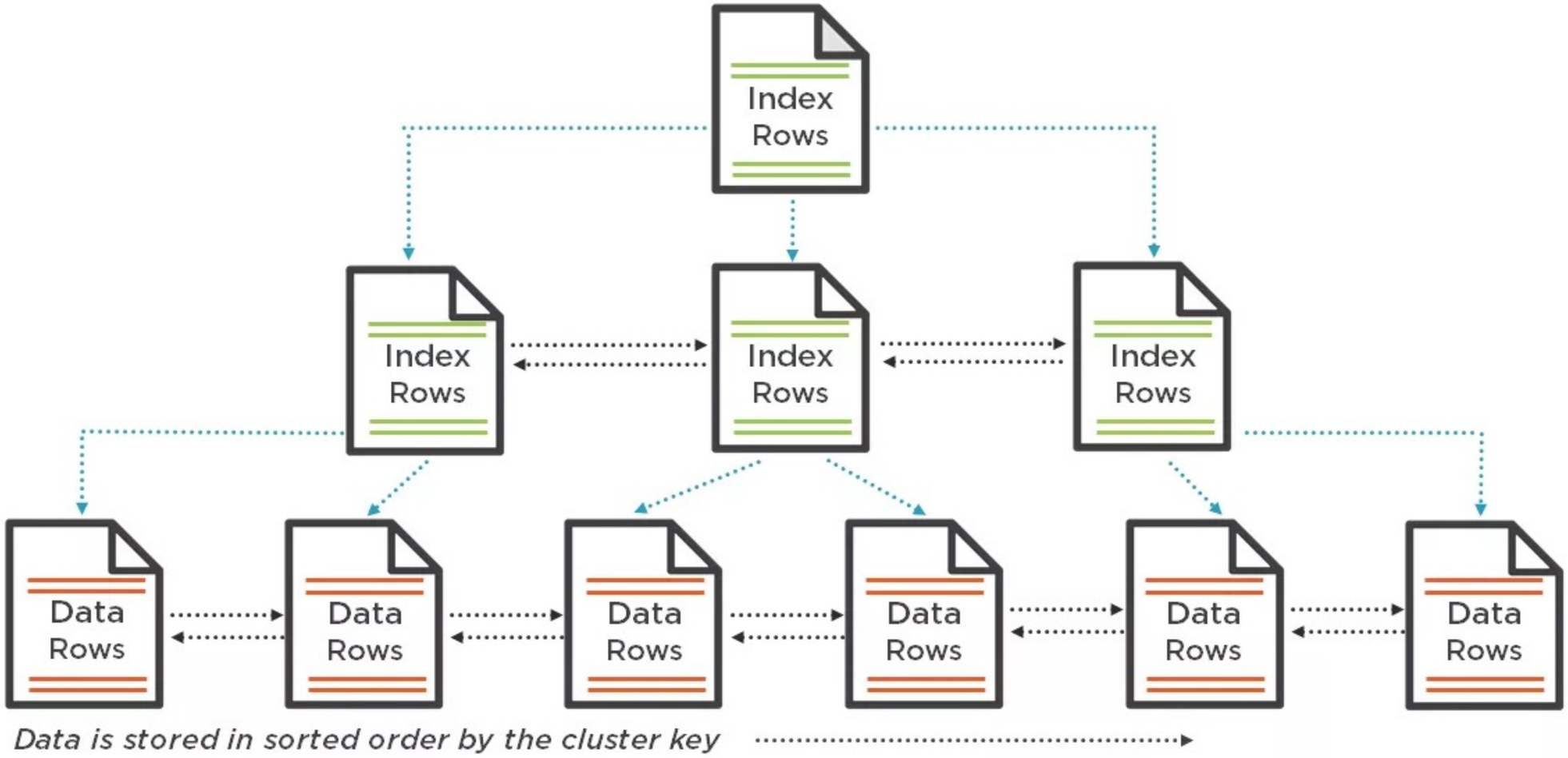
Heap-таблицы

- Стандартный вариант для Postgres
- Строчное хранение (эффективно для OLTP)
- Нет сжатия (?!)
- Есть кластерный индекс

Append-optimized таблицы

- Специальный формат хранения
- Можно хранить и строчно, и колоночно
- Есть сжатие
- Нет кластерного индекса

Cluster index



Оптимизатор запросов

Legacy (old postgres)

- Стандартный вариант для Postgres
- Поддерживает все виды join
- Предпочитает heap, но нормально работает и с AO (ранняя материализация кортежей)
- Выдерживает любое количество join

ORCA

- Специальный оптимизатор
- не поддерживает megre Join
- Предпочитает AO (но также предпочитает раннюю материализацию кортежей)
- С определенного количества join в запросе отдает управлени legacy
- В среднем шустрее legacy

Варианты работы с GP

Legacy (old postgres)

Heap

- Поддерживает все виды join
- Есть кластерный индекс
- Выдерживает любое количество join

- Нет сжатия
- Нет колоночного хранения

Append-optimized

- Поддерживает все виды join
- Есть сжатие
- Есть колоночное хранение

- Нет кластерного индекса
- Не выдерживает большое количество join

ORCA

- Есть кластерный индекс
- Выдерживает любое количество join

- Нет сжатия
- Нет колоночного хранения
- Нет merge join

- Есть сжатие
- Есть колоночное хранение
- Самый быстрый вариант

- Нет кластерного индекса
- Не выдерживает большое количество join

Теория. Оптимизации в Anchor modeling

II

2019

- Table join elimination
- Merge join
- Cluster index

Anchor Modeling

Якорное моделирование - это технология моделирования гибкой базы данных, подходящая для информации, которая со временем изменяется как по структуре, так и по содержанию.

Anchor

Anchor (Якорь) — это существительное, объект реального мира.

Anchor таблица должна хранить

- Суррогатный ключ
- Временная отметка даты загрузки

Tie

Tie (Связь) — это таблица для хранения связей между объектами.

У Tie не может быть атрибутов!

Attribute

Attribute (Атрибут) — это таблица для хранения свойства, атрибута объекта, при этом на каждое свойство ровно одна таблица.

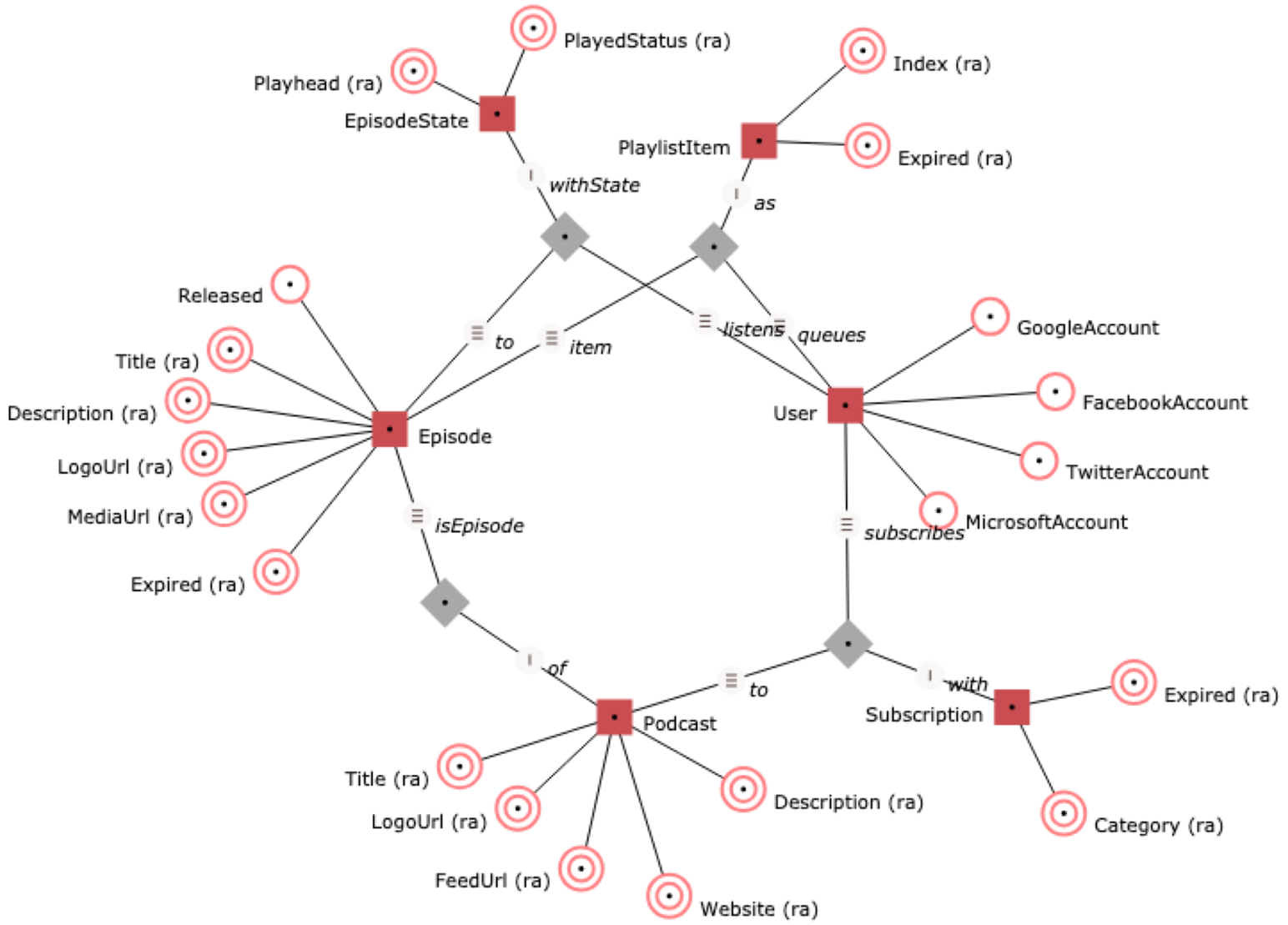
Каждая Attribute-таблица содержит

- Суррогатный ключ
- Временная отметка даты загрузки
- Непосредственно значение

Anchor Modeling

3NF

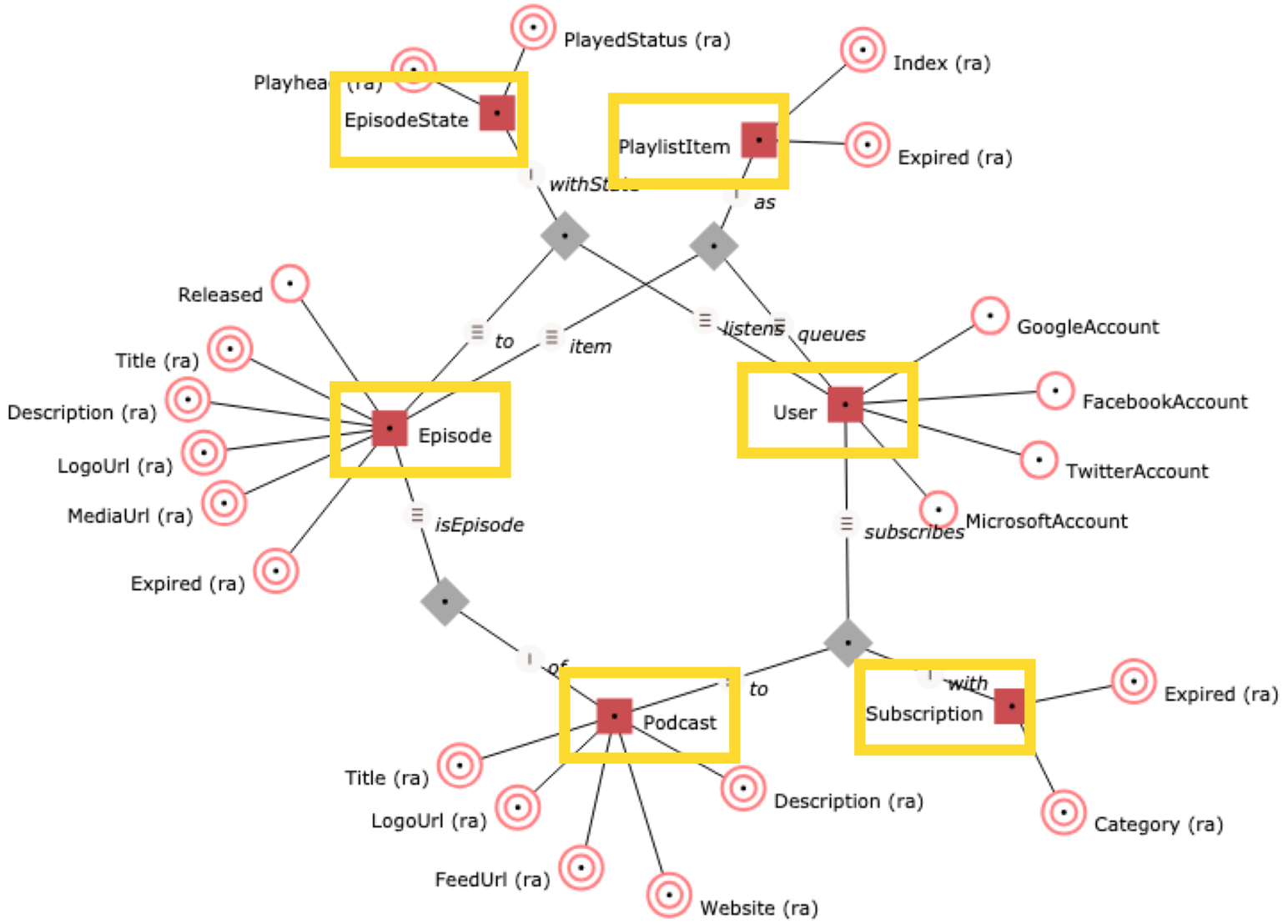
Surrogate key
Business keys
Foreign keys
Descriptors



Anchor Modeling

3NF

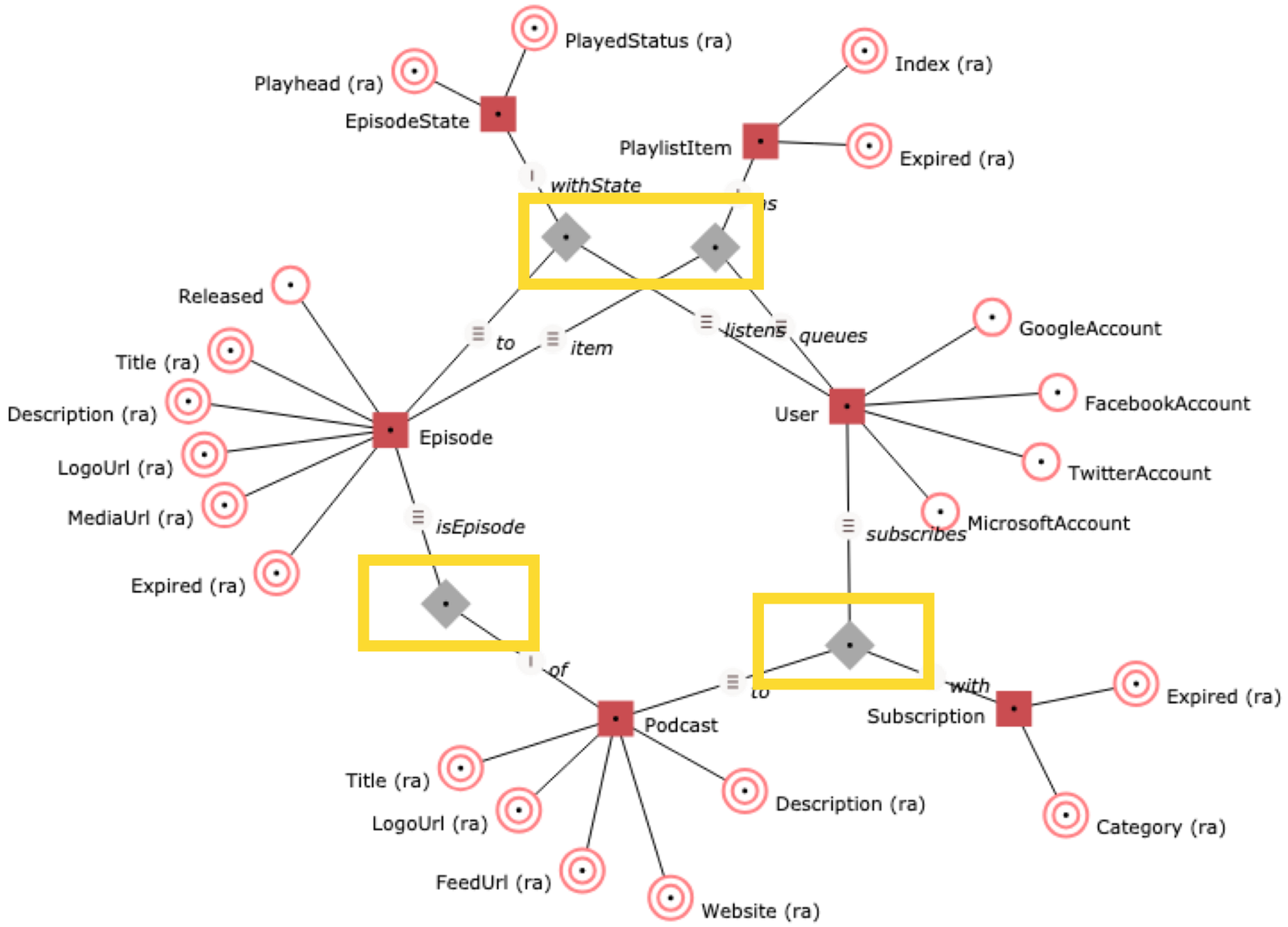
Surrogate key
Business keys
Foreign keys
Descriptors



Anchor Modeling

3NF

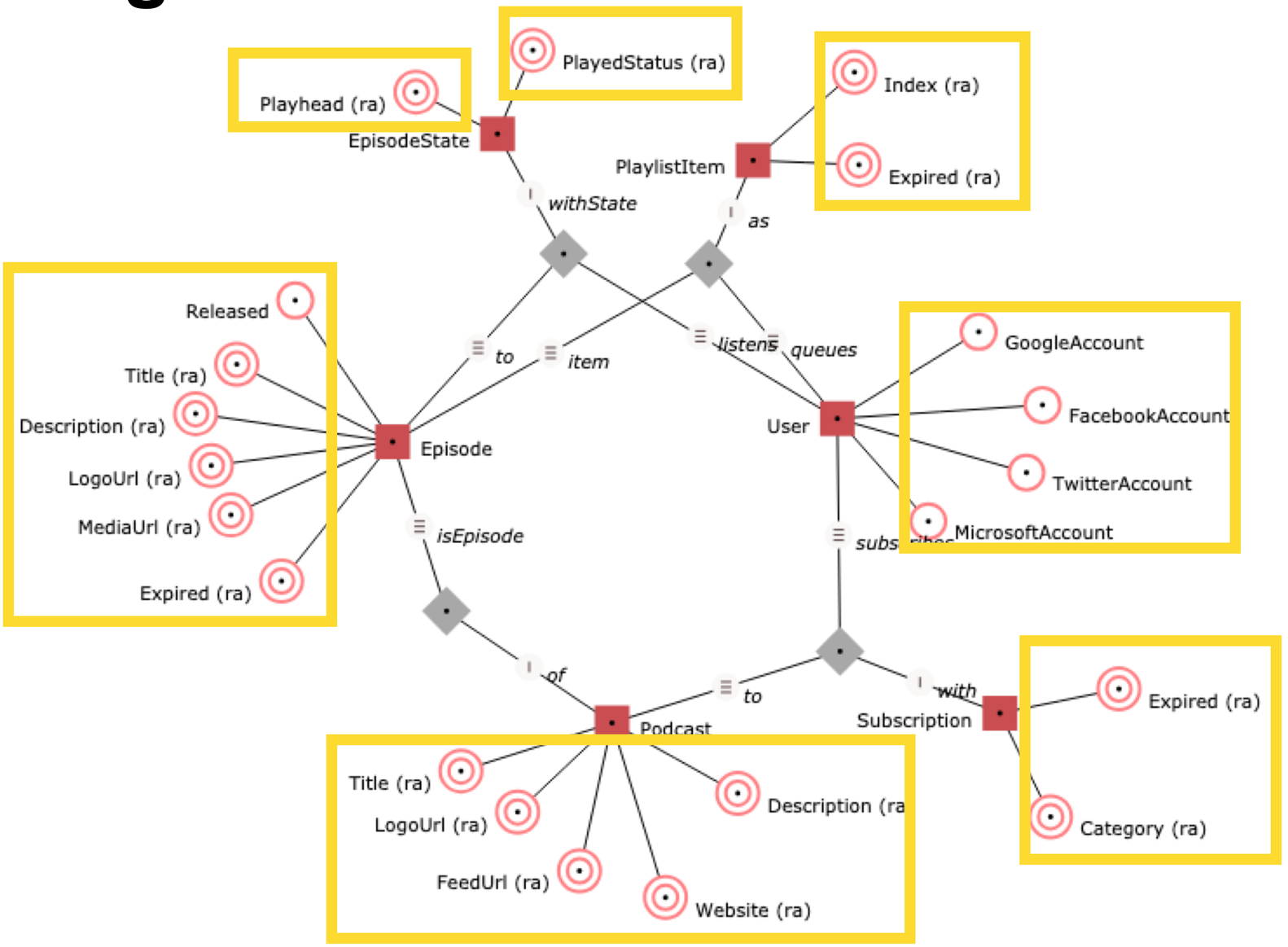
Surrogate key
Business keys
Foreign keys
Descriptors



Anchor Modeling

3NF

- Surrogate key
- Business keys
- Foreign keys
- Descriptors



Три кита AM

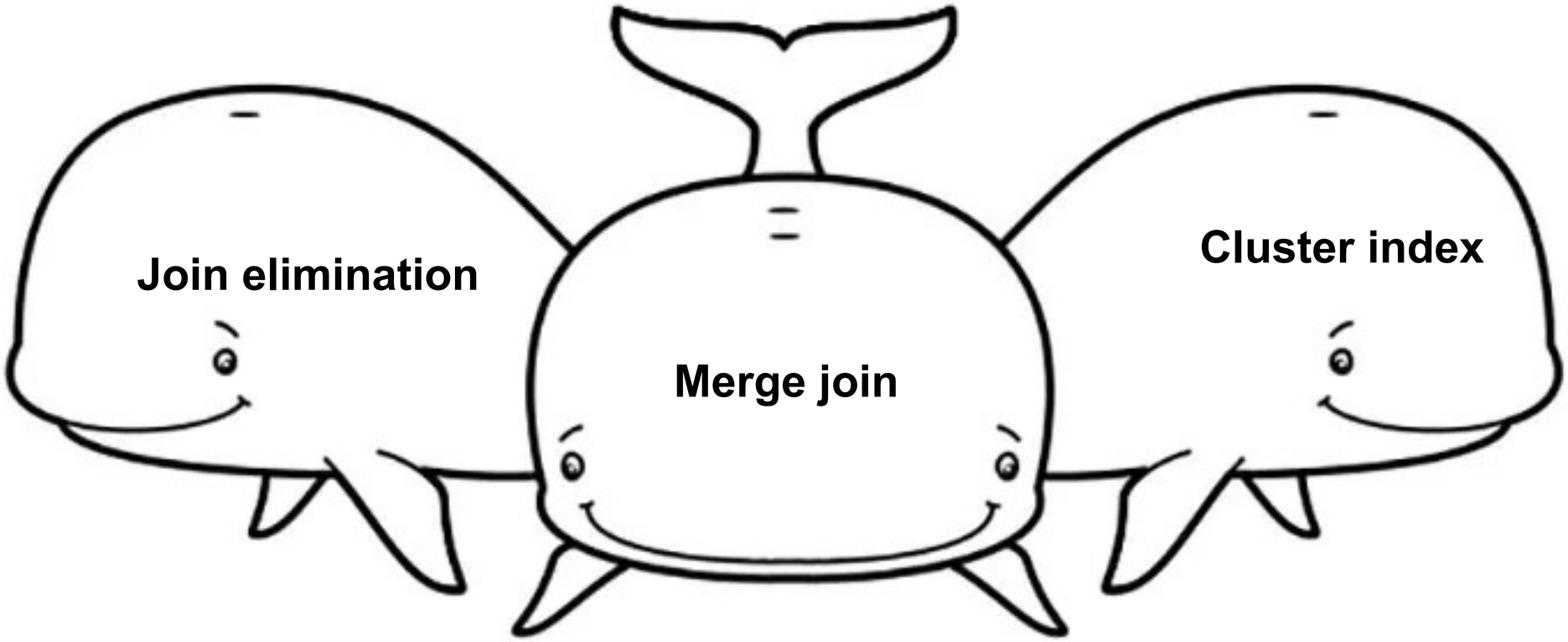


Table join elimination

Оптимизация, позволяющая исключить таблицу из плана запроса, если она не нужна

```
select A.colA
  from tableA  A
  left join tableB  B
    on B.id = A.id;
```

```
select A.colA
  from tableA  A
  left join tableC  C
    on C.id = A.id
   and C.fromDate = (
       select max(sub.fromDate)
         from tableC sub
        where sub.id = A.id );
```

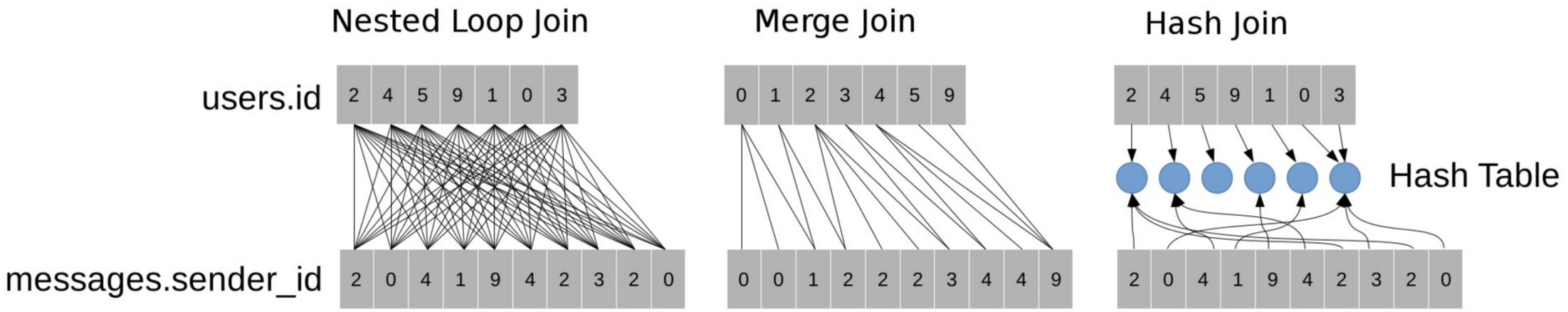
Table join elimination

Оптимизация, позволяющая исключить таблицу из плана запроса, если она не нужна

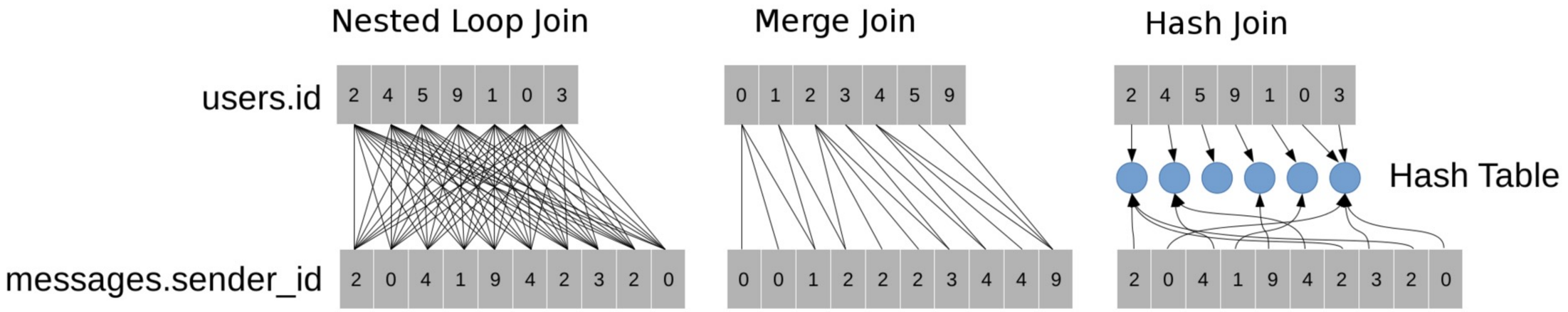
Database Engine	Support	Example
Microsoft SQL Server 2005	full	SQL Server Example
Microsoft SQL Server 2008	full	SQL Server Example
Oracle 10gR2 Express Edition*	partial	Oracle Example
Oracle 11gR1 Enterprise/Express Edition	full	Oracle Example
IBM DB2 v9.5	full	IBM DB2 Example
PostgreSQL v8.4 beta	full	PostgreSQL Example
Teradata v12**	partial	Teradata Example
MySQL v5.0.70	none	MySQL Example
MySQL v6.0.10 alpha	none	MySQL Example
MariaDB v5.1	full	MariaDB Example
Sybase	not tested	

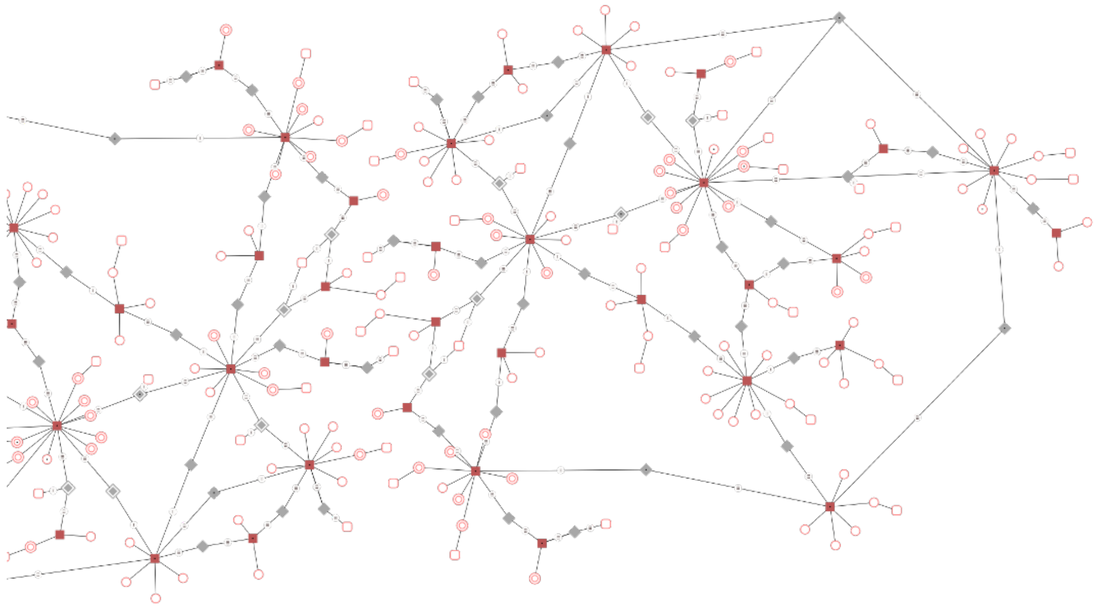
```
select A.colA
  from tableA  A
  left join tableC  C
    on C.id = A.id
   and C.fromDate = (
       select max(sub.fromDate)
         from tableC sub
        where sub.id = A.id );
```

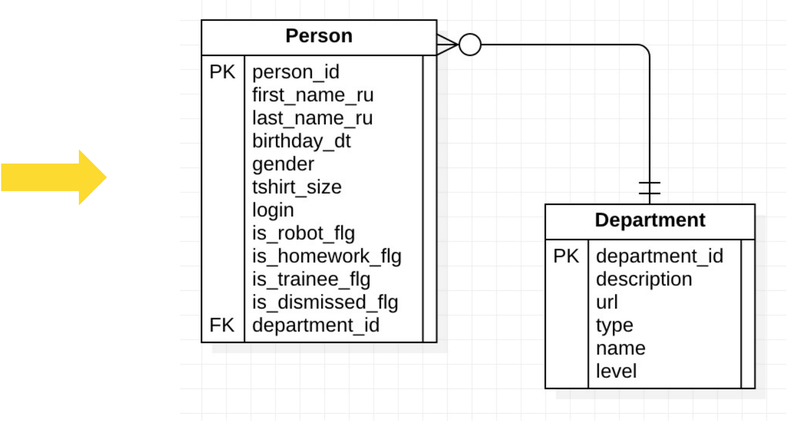
Merge join


Nested Loop Join
Merge Join
Hash Join
users.id
messages.sender_id
2 4 5 9 1 0 3
2 0 4 1 9 4 2 3 2 0
0 1 2 3 4 5 9
0 0 1 2 2 2 3 4 4 9
2 4 5 9 1 0 3
Hash Table
2 0 4 1 9 4 2 3 2 0

Merge join

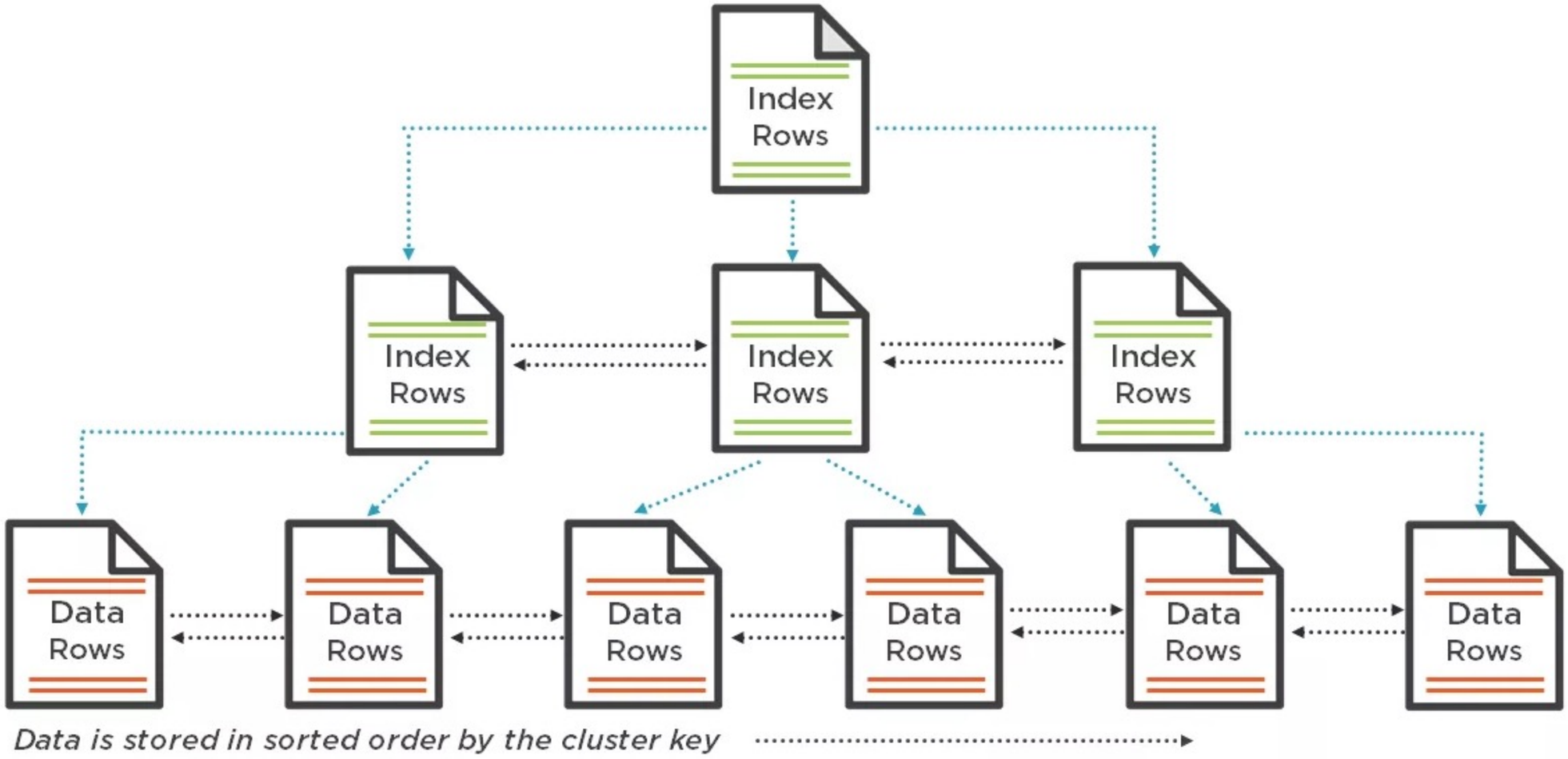






SQL QUERY OPTIMIZATION FOR HIGHLY NORMALIZED BIG DATA https://bijournal.hse.ru/data/2015/10/12/1076327532/1.pdf

Cluster index



Итого по теории

Вывод: технические сложности возможны, но принципиальный блокеров нет

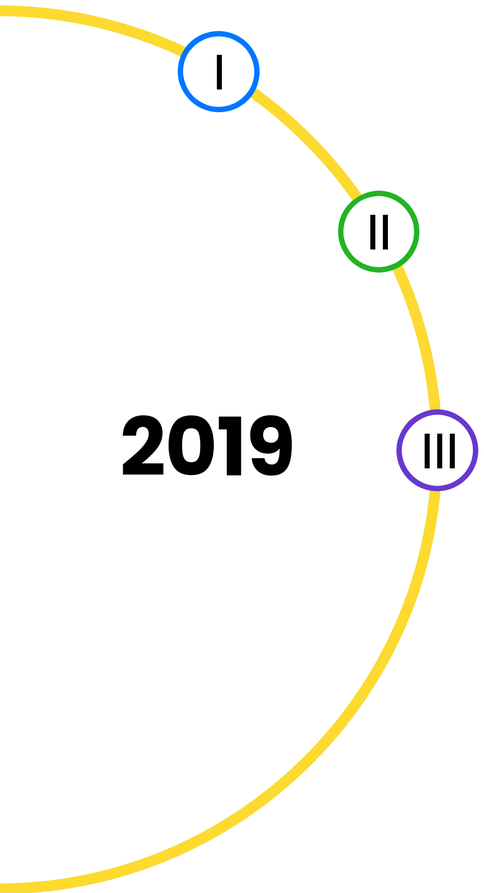
Greenplum:

- SN MPP RDBMS
- Heap vs AO
- ORCA vs Legacy

Anchor Modeling:

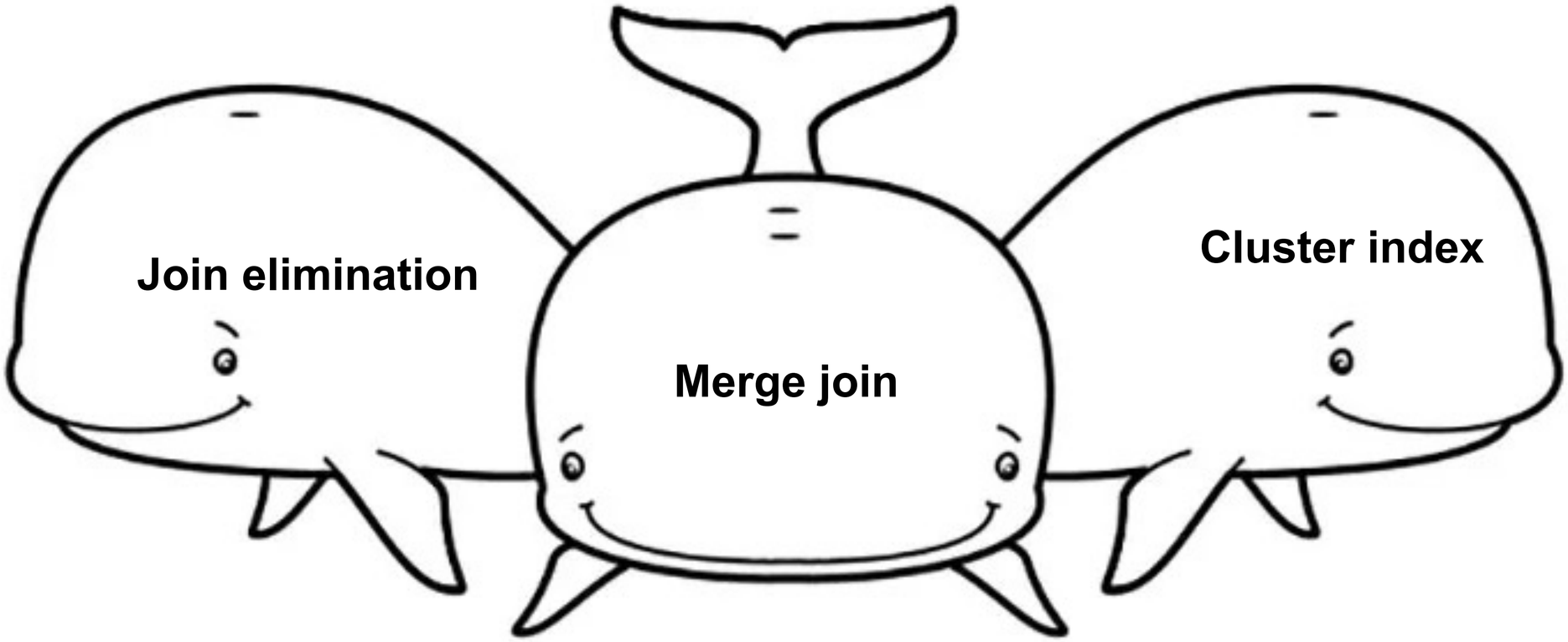
- Merge join (legacy over heap)
- Cluster index (heap)
- Join elimination (postgres 8.4+)

Попытка #1. Эксперименты с Greenplum 5

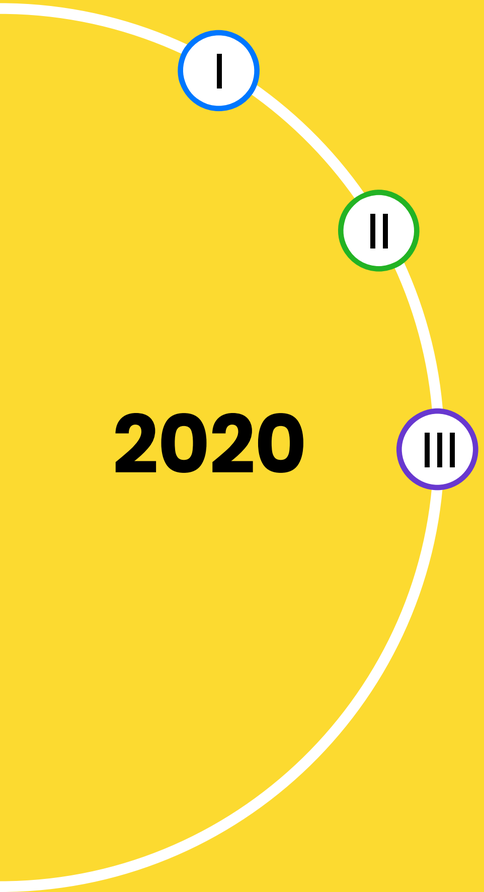


- Что могло пойти не так?
- Как мы выкрутились?
- Предпосылки hNhM

Три кита АМ

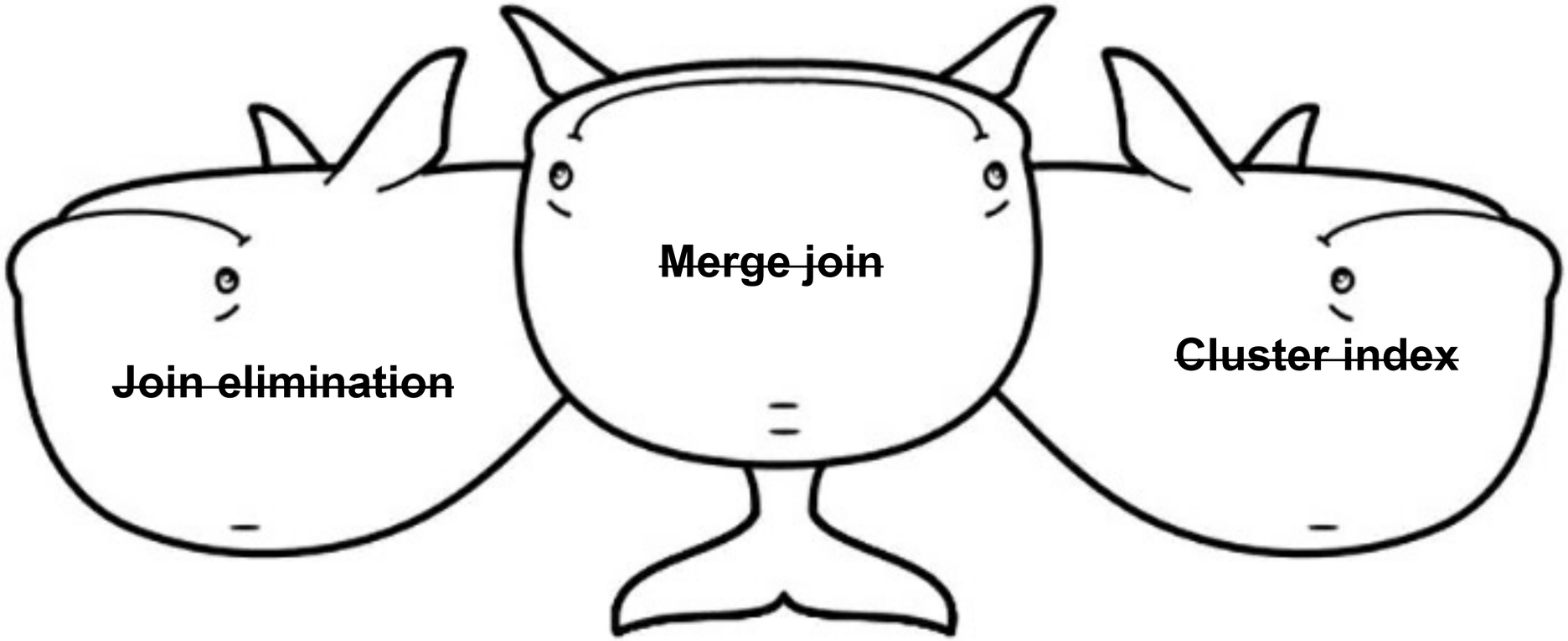


Сколько оптимизаций «из коробки» заработало в Greenplum 5?



Попытка #1

Три кита



Существует в более свежей версии PG чем та, что в GP5

Отсутствует в ORCA

Работает только для Heap, а не AO

MVP на GP5

Собираем MVP из того, что есть

- GreenPlum 5
- AO таблицы
- Column oriented
- Hash join
- Оптимизации формирования запросов

MVP работает, жить можно – но на душе скребет, потому что не завелось идеально

Highly Normalized Hybrid Model (hNhM)

Ключевая идея: выбирать оптимальный формат хранения для каждого конкретного случая

Требования:

- Разделение логического и физического моделирования
- Возможность эмулировать как Data Vault, так и Anchor Modeling, высокая нормализация (вплоть до предельной 6НФ)
- Параллельная загрузка из разных источников, идемпотентность при повторной загрузке
- Устойчивость к изменению в бизнесе, модульность и масштабируемость
- Удобство использования при построении витрин

Доступ к сущностям из Python

> **Был разработано несколько классов, которые позволяют сформировать SQL запрос к определенной сущности и затем использовать его в построении витрины**

```python
def load():
    department = HistoryEntityCte(Department).project(
        Department.department_id,
        Department.department_type,
        Department.description
    )
    person = ActualEntityCte(Person).project(
        Person.person_id,
        Person.gender,
        Person.tshirt_size,
        Person.is_robot_flg,
        staff_login=Person.login
    )
    link_person_department = ActualLinkCte(LinkPersonDepartment)
```



```sql
WITH department AS (
    {department}
), person AS (
    {person}
), link_person_department AS (
    {link_person_department}
)
SELECT *
FROM department d
  LEFT JOIN link_person_department lpdr
        ON d.id = lpdr.department
  LEFT JOIN person p
        ON p.id = lpdr.person
```

Первый опыт. Еда

В качестве первого проекта, выбрали Яндекс.Еда

Почему:

- Достаточно большая, чтобы было интересно
- Новый бизнес, источник может сильно меняться

Что получили:

- ~ 25 сущностей (заказ, курьер, ресторан, промокоды, смена…)
- 6 систем источников
- 2 аналитические витрин
- ETA – 40 минут от источника до финальных витрин

Загрузка данных Яндекс.Такси

Перевод основных расчетов Такси на новый детальный слой

Yandex Taxi

Что пошло не так:

- Объем данных на порядок больше
- Большая глубина обновления заказов

Что получили:

- ~ 40 сущностей
- Более 30 систем источников
- Более 15 аналитических витрин
- ETA – 2 часа от источника до финальных витрин

Итого по Попытке#1

«Из коробки» не завелось, но в ходе борьбы родился hNhM

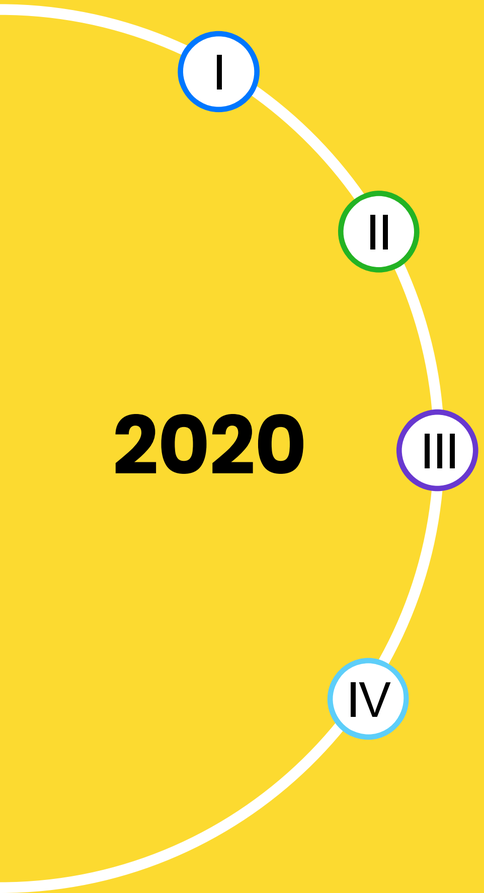
Яндекс Еда:

- ~ 25 сущностей (заказ, курьер, ресторан, промокоды, смена…)
- 6 систем источников
- 2 аналитические витрин
- ETA – 40 минут от источника до финальных витрин

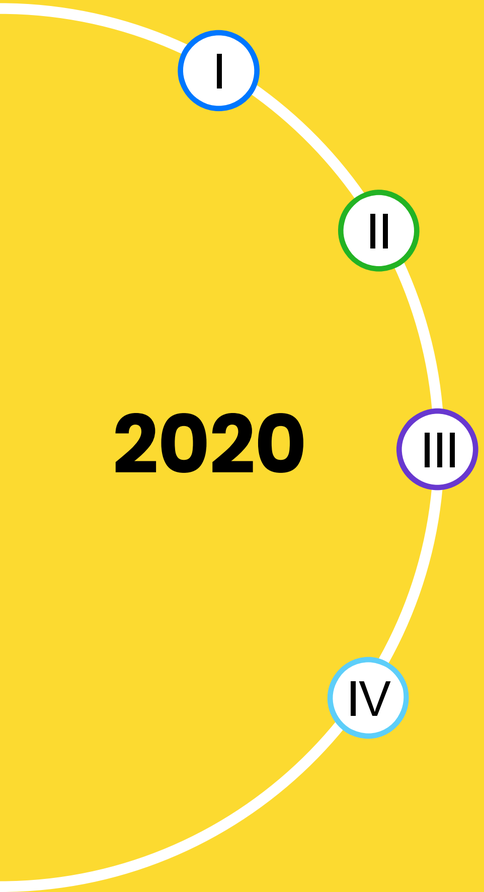
Яндекс Такси:

- ~ 40 сущностей
- Более 30 систем источников
- Более 15 аналитических витрин
- ETA – 2 часа от источника до финальных витрин

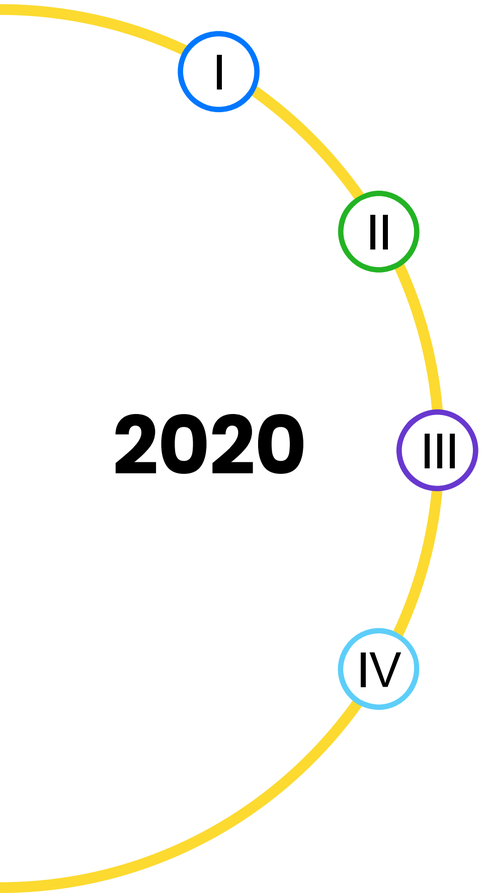
Прошло полгода…



… приехало новое железо и вышел GP6

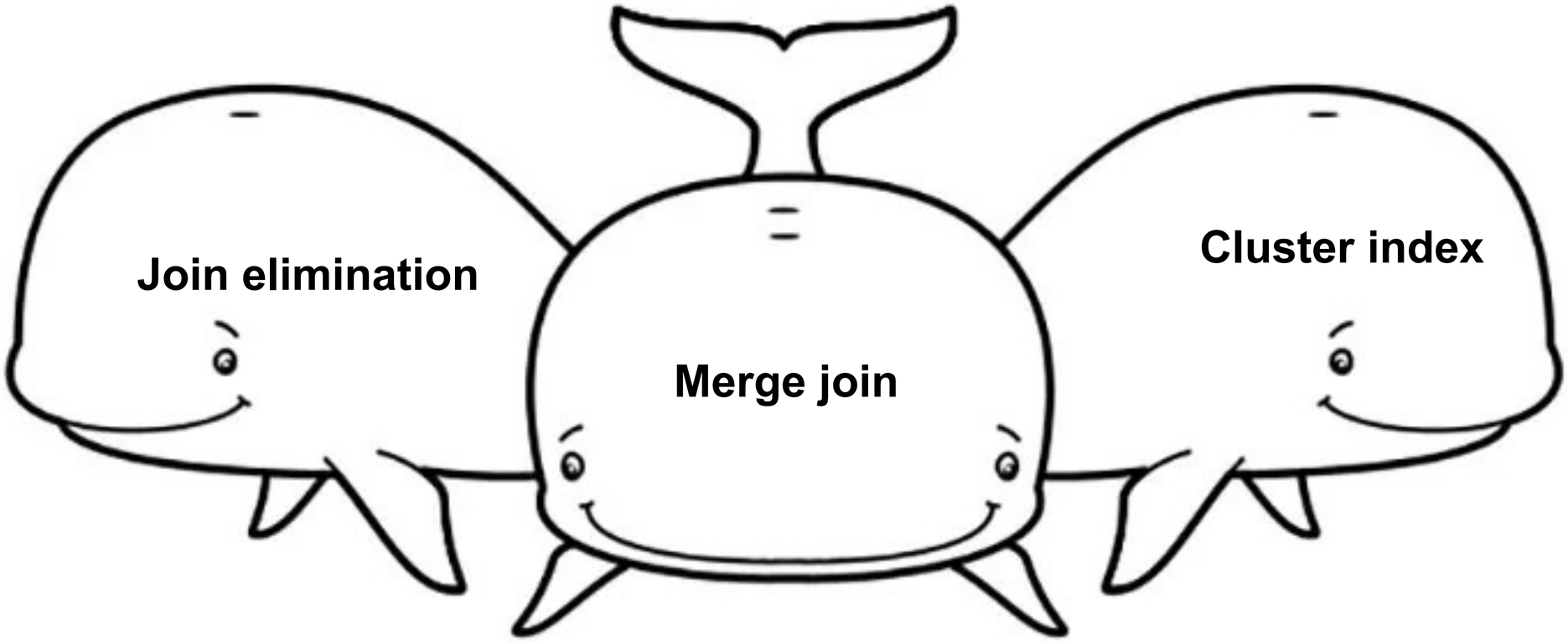


Попытка #2. Эксперименты с Greenplum 6

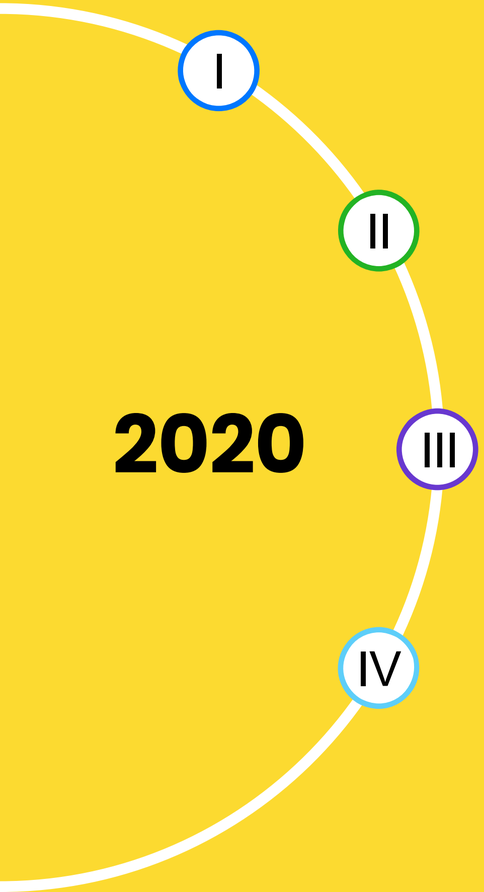


- А теперь что могло пойти не так?
- На какие жертвы мы готовы были идти?
- Стоит ли повторять наш путь?

Три кита АМ

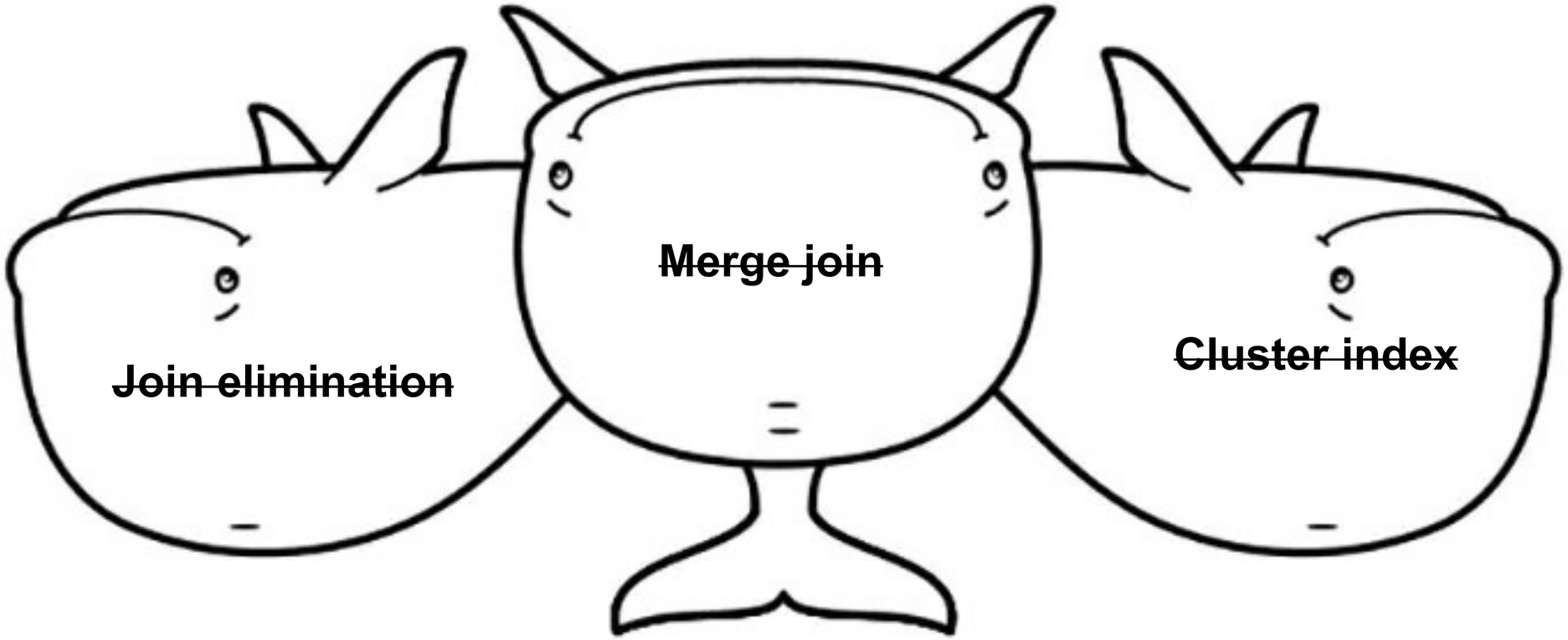


Сколько оптимизаций «из коробки» заработало в Greenplum 6?



Попытка #2

Три кита



Версия PG более свежая, но оптимизации GP убили JE

Отсутствует в ORCA (и не предвидится)

Работает только для Heap, а не AO (предвидится в GP7)

Варианты работы с GP

	Legacy (old postgres)	ORCA
Heap	› Поддерживает все виды join › Есть кластерный индекс › Выдерживает любое количество join › Нет сжатия › Нет колоночного хранения	› Есть кластерный индекс › Выдерживает любое количество join › Нет сжатия › Нет колоночного хранения › Нет merge join
Append-optimized	› Поддерживает все виды join › Есть сжатие › Есть колоночное хранение › Нет кластерного индекса › Не выдерживает большое количество join	› Есть сжатие › Есть колоночное хранение › Самый быстрый вариант › Нет кластерного индекса › Не выдерживает большое количество join

У нас есть чистый кластер, почему бы не поэкспериментировать с heap?

Тесты на GP6

У нас есть чистый кластер, почему бы не поэкспериментировать с heap

Плюсы:

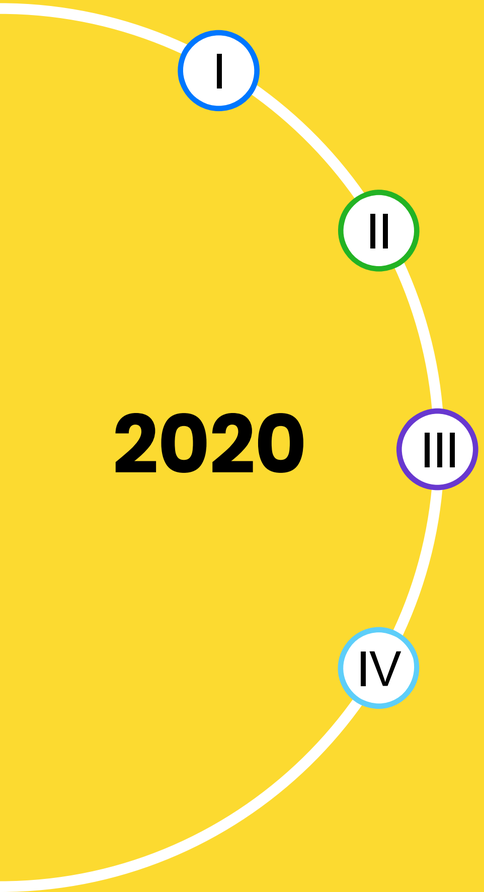
- У heap есть кластерный индекс
- Легаси оптимизатор выдает MergeJoin для heap (с нюансами)
- Таблицы в АМ небольшие, поэтому колоночное хранение не нужно

Минусы:

- У heap ***невозможно*** задать сжатие

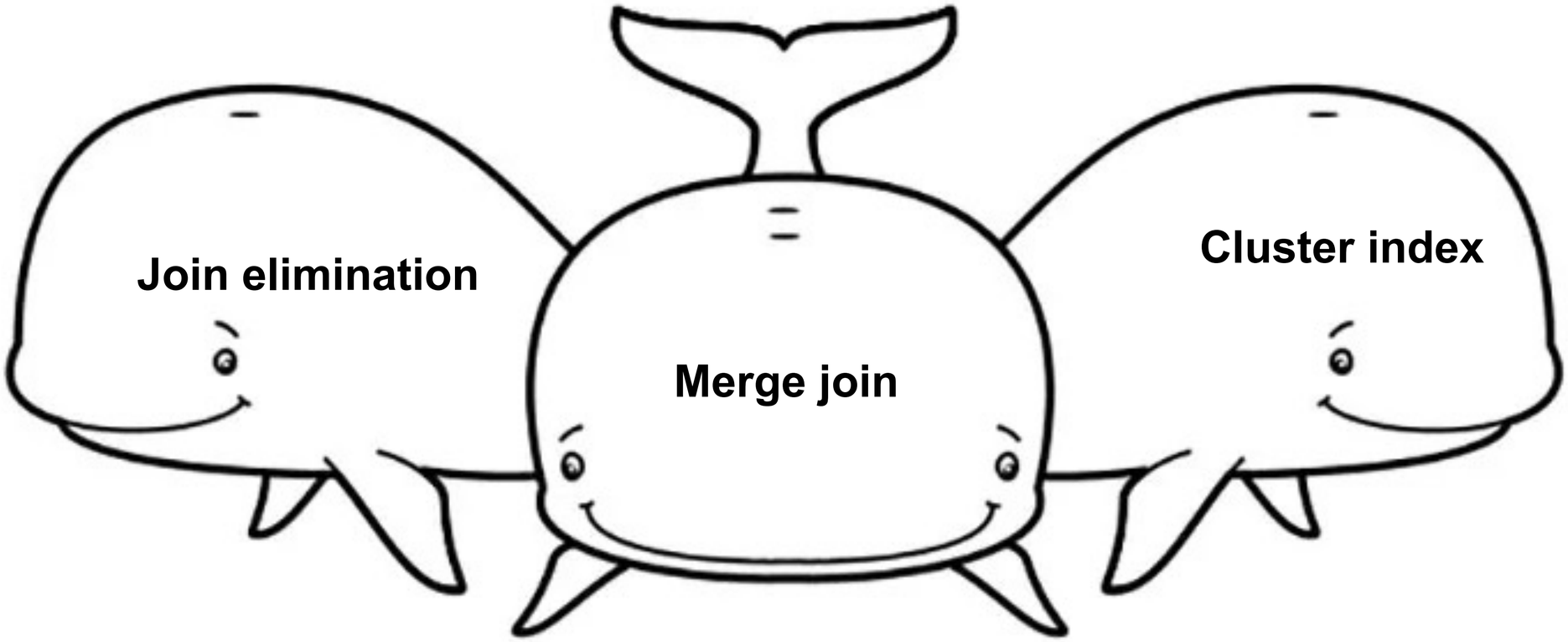
Отсутствие сжатия покрываем файловой системой со сжатием BTFRS

Сколько оптимизаций «из коробки» заработало в сборке heap over BTFRS?



Попытка #2

Все завелось!

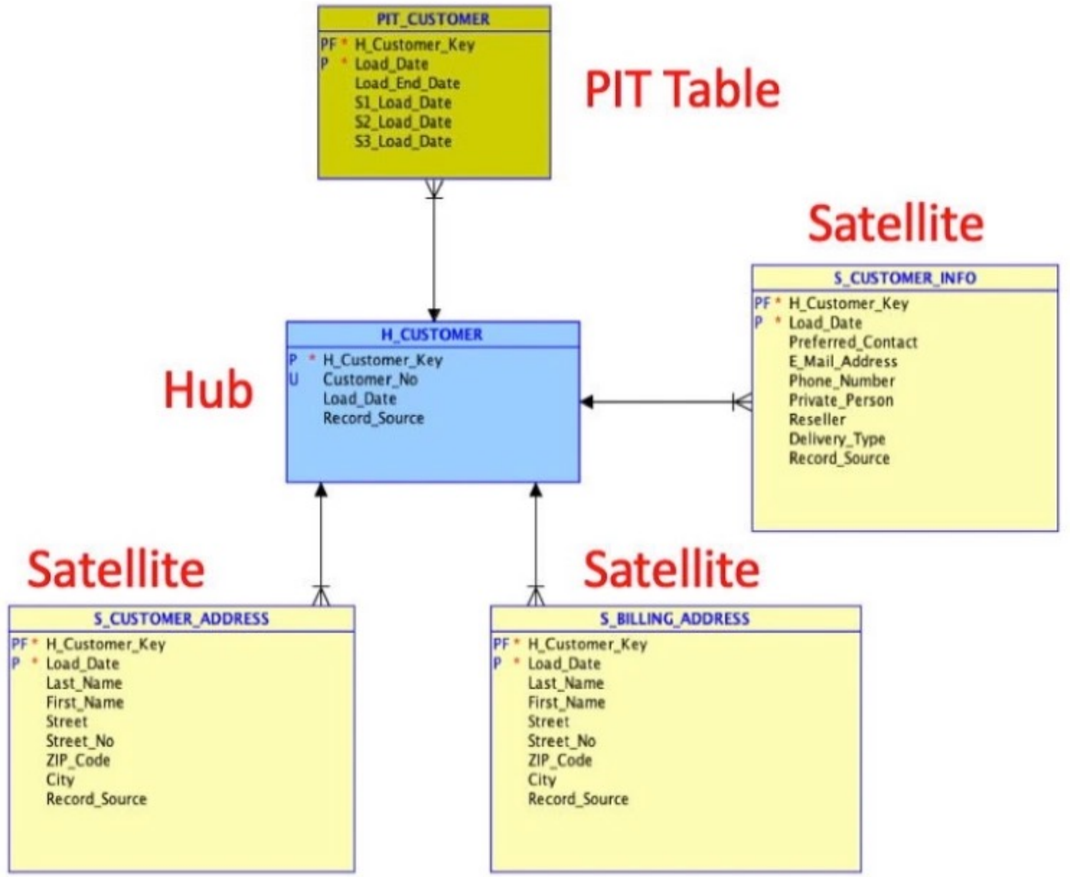


Но есть нюансы…

Нюансы table elimination

Он как бы работает, но нет

- Работает только с выключенной ORCA
- Не работает на партицированных таблицах
- Нужна PIT таблица



Нюансы merge join

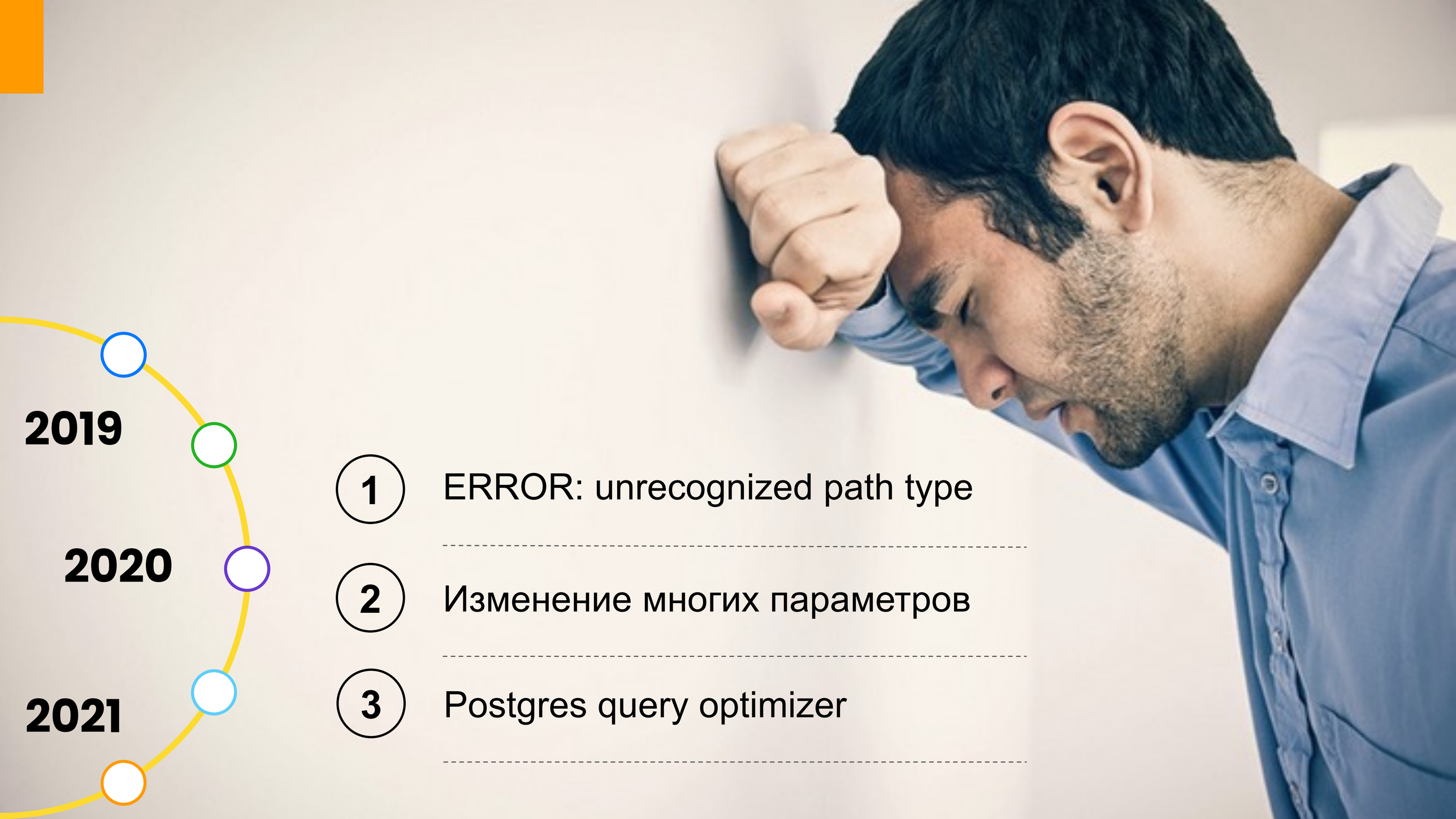
Он как бы есть и даже шустрее, но его как бы нет

- Работает только с выключенной ORCA
- Нужно уменьшить random_page_cost
- При больше 20 таблицах может рандомно покрашиться
- Надо явно кластеризовать таблицу и обновлять статистику (GP любит сортировать, если что-то не понял)

Тип соединения	Время выполнения (с)
MERGE	17
HASH	13
MERGE CLUSTER INDEX	5

Рандомно покрашится ☹ (слайд боли)


```
194             case T_NestLoop:
195                 {
196                     NestPath   *nestpath = (NestPath *)path;
197 
198                     v = pathnode_walk_node(nestpath->outerjoinpath, walker, context);
199                     if (v != CdbVisit_Walk)     /* stop */
200                         break;
201                     v = pathnode_walk_node(nestpath->innerjoinpath, walker, context);
202                     if (v != CdbVisit_Walk)     /* stop */
203                         break;
204                 }
205                 break;
206             case T_Append:
207                 v = pathnode_walk_list(((AppendPath *)path)->subpaths, walker, context);
208                 break;
209             case T_MergeAppend:
210                 v = pathnode_walk_list(((MergeAppendPath *)path)->subpaths, walker, context);
211                 break;
212             case T_Material:
213                 v = pathnode_walk_node(((MaterialPath *)path)->subpath, walker, context);
214                 break;
215             case T_Unique:
216                 v = pathnode_walk_node(((UniquePath *)path)->subpath, walker, context);
217                 break;
218             case T_Motion:
219                 v = pathnode_walk_node(((CdbMotionPath *)path)->subpath, walker, context);
220                 break;
221             default:
222                 v = CdbVisit_Walk;  /* keep compiler quiet */
223                 elog(ERROR, "unrecognized path type: %d", (int)path->pathtype);
224         }
225         return v;
226     }                               /* pathnode_walk_kids */
```



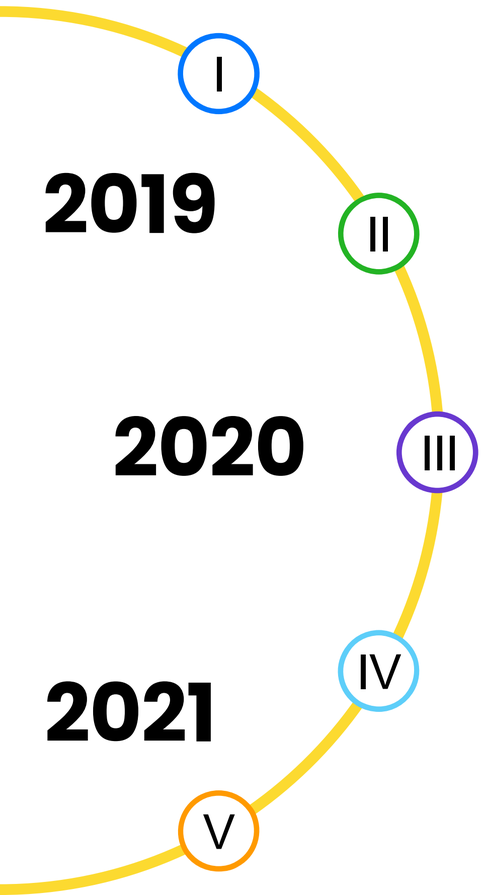
2019

2020

2021

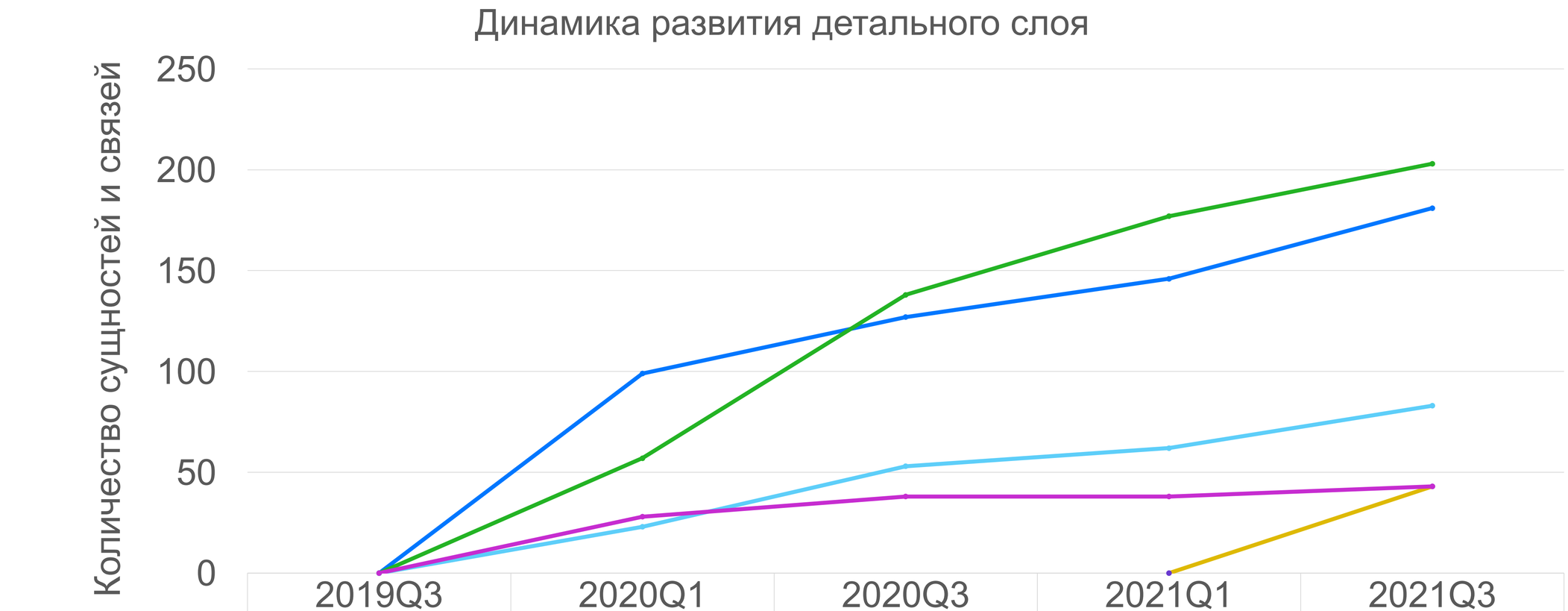
1. ERROR: unrecognized path type
2. Изменение многих параметров
3. Postgres query optimizer

Резюме



- К чему мы пришли?
- И что в итоге?

Все работает на MVP DDS



	2019Q3	2020Q1	2020Q3	2021Q1	2021Q3
FoodTech	0	99	127	146	181
RideTech	0	57	138	177	203
Market				0	43
CrossService	0	23	53	62	83
Meta	0	28	38	38	43

Итог

Разочарование

Мы не завели на GP запросы для AM в чистом виде

- Нет Join Elimination
- Нет Merge join для ORCA
- Нет Cluster Index для AO
- Не работает на партицированных таблицах
- Нужна PIT таблица
- Нужно сжатие

Итог

Разочарование

Мы не завели на GP запросы для AM в чистом виде

- Нет Join Elimination
- Нет Merge join для ORCA
- Нет Cluster Index для AO
- Не работает на партицированных таблицах
- Нужна PIT таблица
- Нужно сжатие

Принятие

hNhM (наша смесь DV и AM) завелась и работает

- 162 сущности
- 1678 атрибутов
- 306 линков

Итог

Разочарование

Мы не завели на GP запросы для AM в чистом виде

- Нет Join Elimination
- Нет Merge join для ORCA
- Нет Cluster Index для AO
- Не работает на партицированных таблицах
- Нужна PIT таблица
- Нужно сжатие

Принятие

hNhM (наша смесь DV и AM) завелась и работает

- 162 сущности
- 1678 атрибутов
- 306 линков

Надежда

С выходом GP7 мы будем продолжать эксперименты

- Cluster Index для AO
- Отказ от ORCA в пользу legacy optimizer (которые перестанет быть легаси)
- Или доработка merge join в ORCA (см последний GP meetup)
- Свежие версии PG

Яндекс Такси

Спасибо

Евгений Ермаков

Руководитель DWH

jkermakov@yandex-team.ru

@iJKos

iJKos.com

Николай Гребенщиков

Руководитель Data Engineer RideTech

ngrbnshchkv@yandex-team.ru

@ngrebenshchikov