

Modern Storages and Data Warehousing

Week 4 - DWH

Попов Илья, i.popov@hse.ru

Recap прошлых занятий

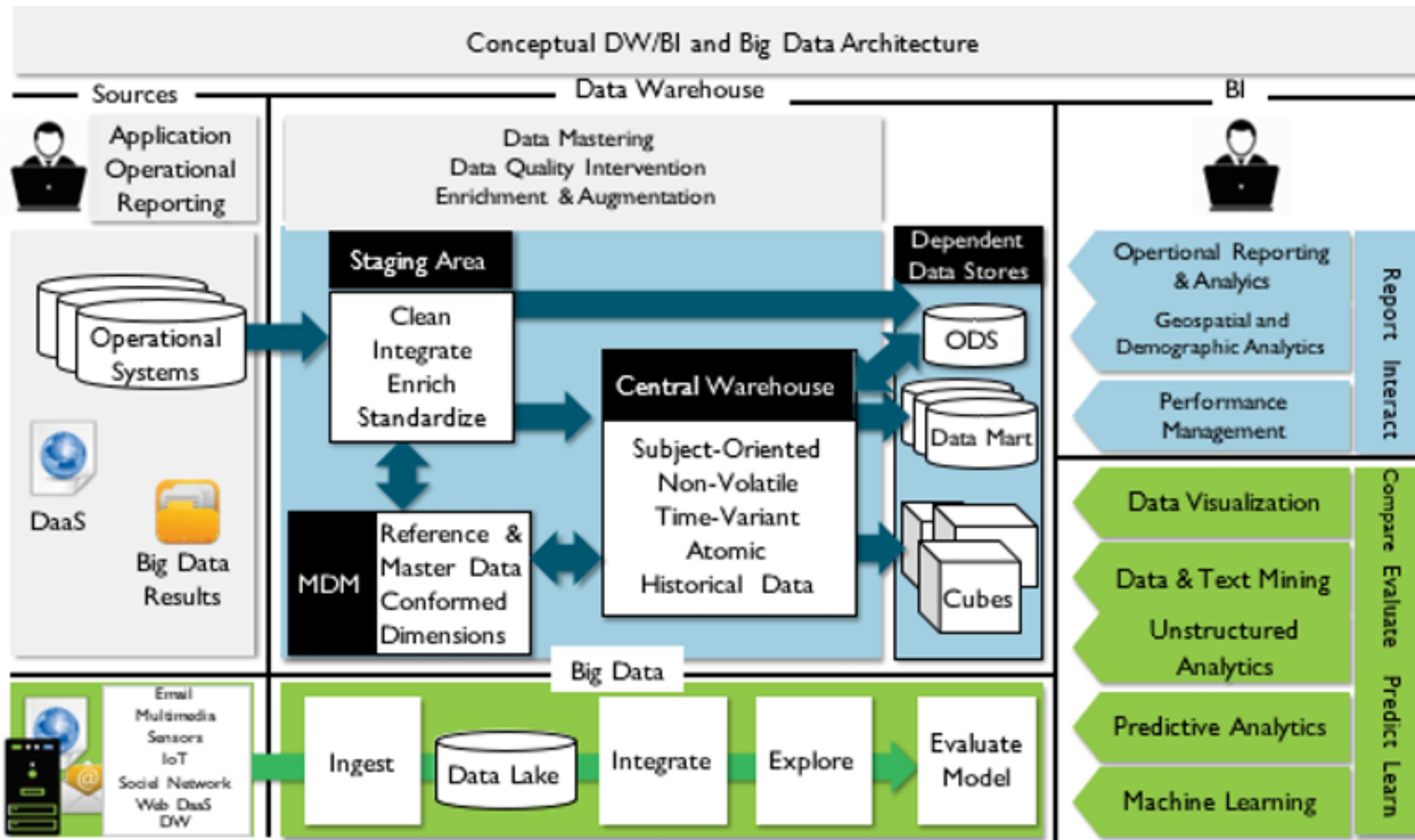


Figure 5: Date Warehouse Concept

1 - Догоняем Mass Parallel Processing

Мотивация

- › Мы все еще учимся эффективно хранить и обрабатывать большие данные
- › Мы пришли к HDFS/MapReduce, но это медленно и не соответствует ACID от реляционных СУБД
- › А что если распараллелить вычисления, но сделать так, чтобы данные обрабатывали хосты, похожие на классические СУБД

Мотивация

- › Мы все еще учимся эффективно хранить и обрабатывать большие данные
- › Мы пришли к HDFS/MapReduce, но это медленно и не соответствует ACID от реляционных СУБД
- › А что если распараллелить вычисления, но сделать так, чтобы данные обрабатывали хосты, похожие на классические СУБД
- › Так вот оказывается, что параллельно с DFS/MR разрабатывался класс систем, которые именно это и делают - MPP

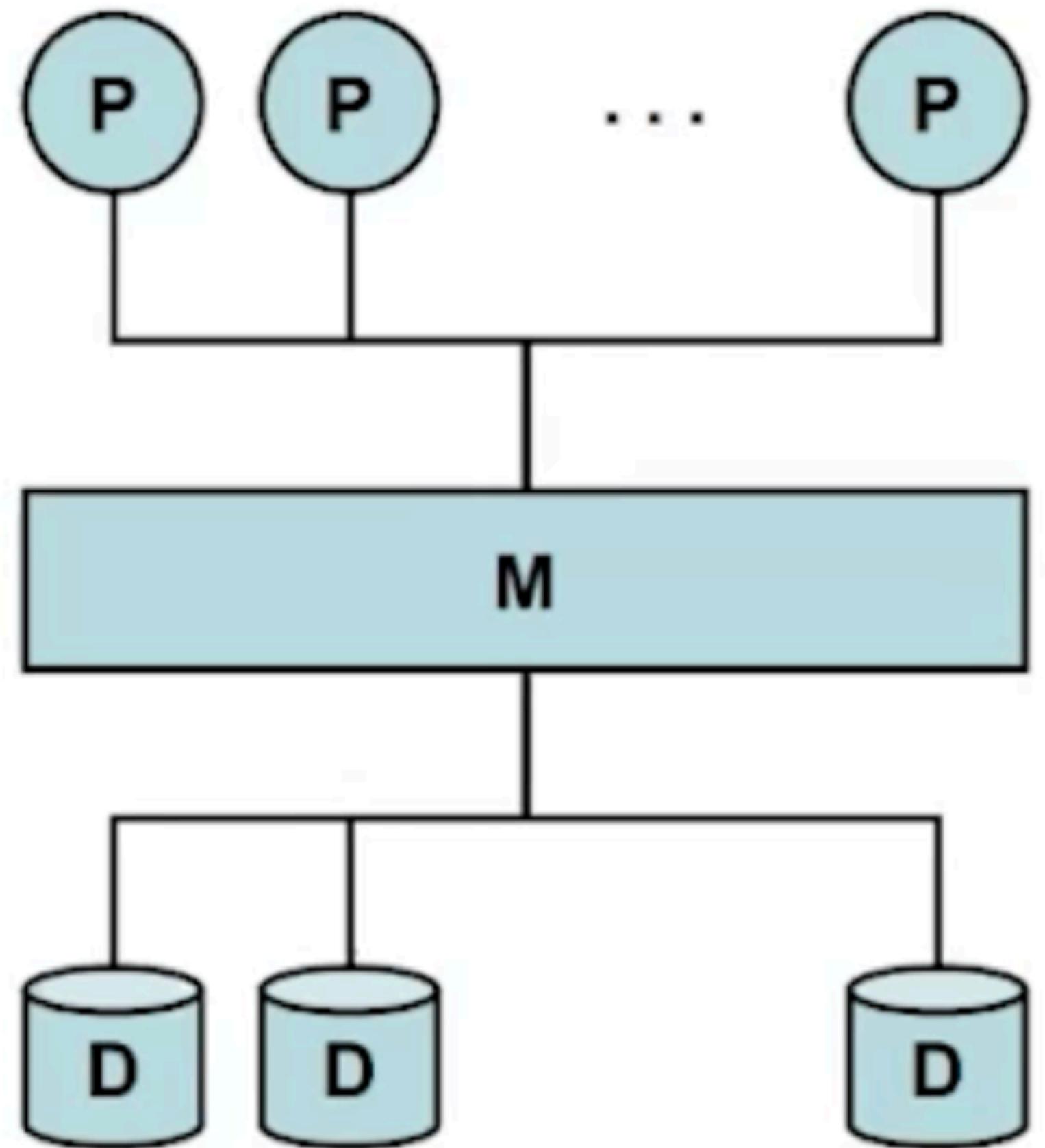
Классификация архитектур

Мы хотим создать распределенную СУБД для эффективного хранения и обработки данных.

Какие тут могут быть подходы:

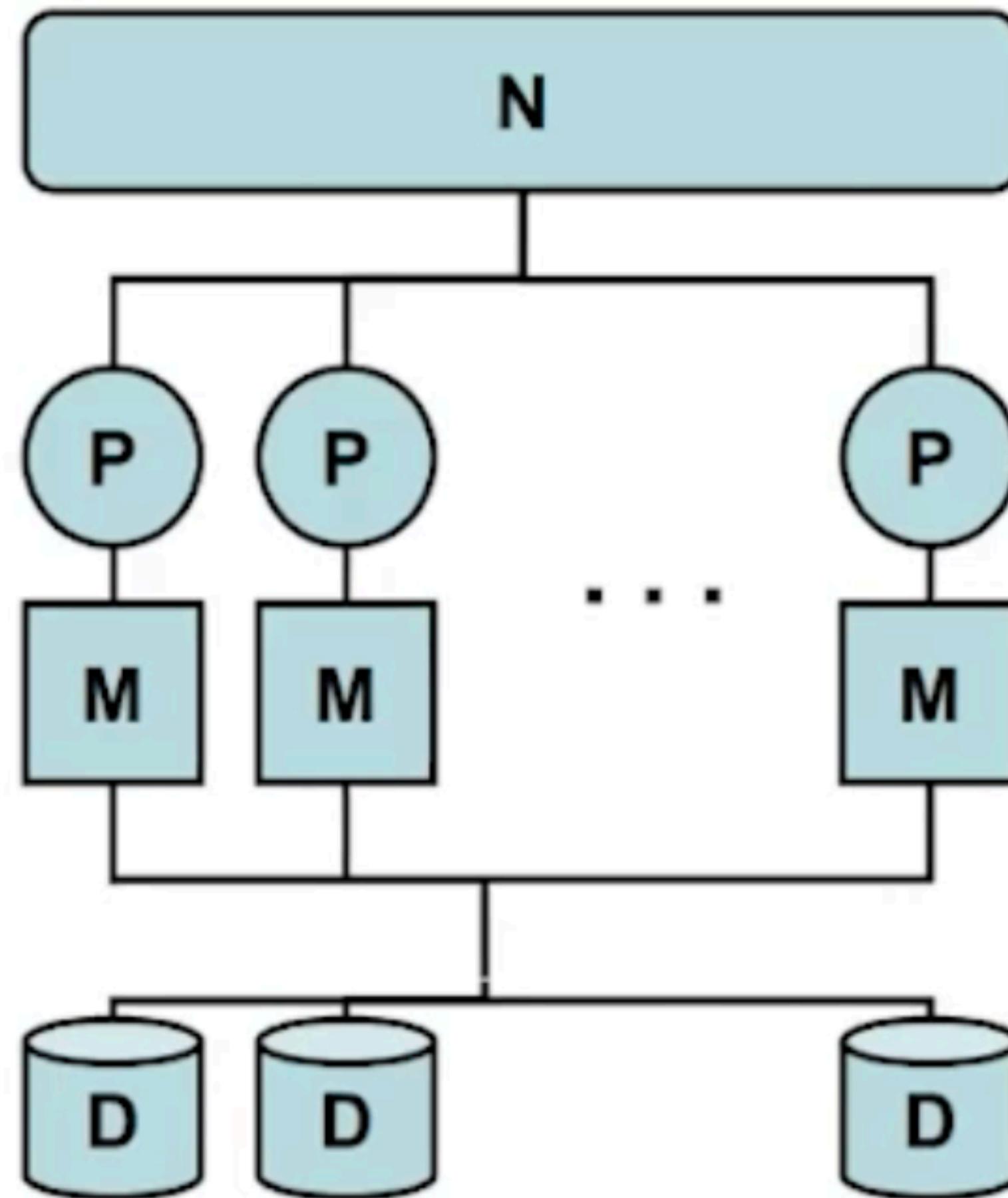
- › SE - Shared Everything - хосты делят все ресурсы (CPU, RAM, Disk)
- › SD - Shared Disks - хосты делят только диски
- › SN - Shared Nothing - нет совместного использования ресурсов
- › CD - Clustered Disk
- › CE - Clustered Everything

SE (Shared Everything)



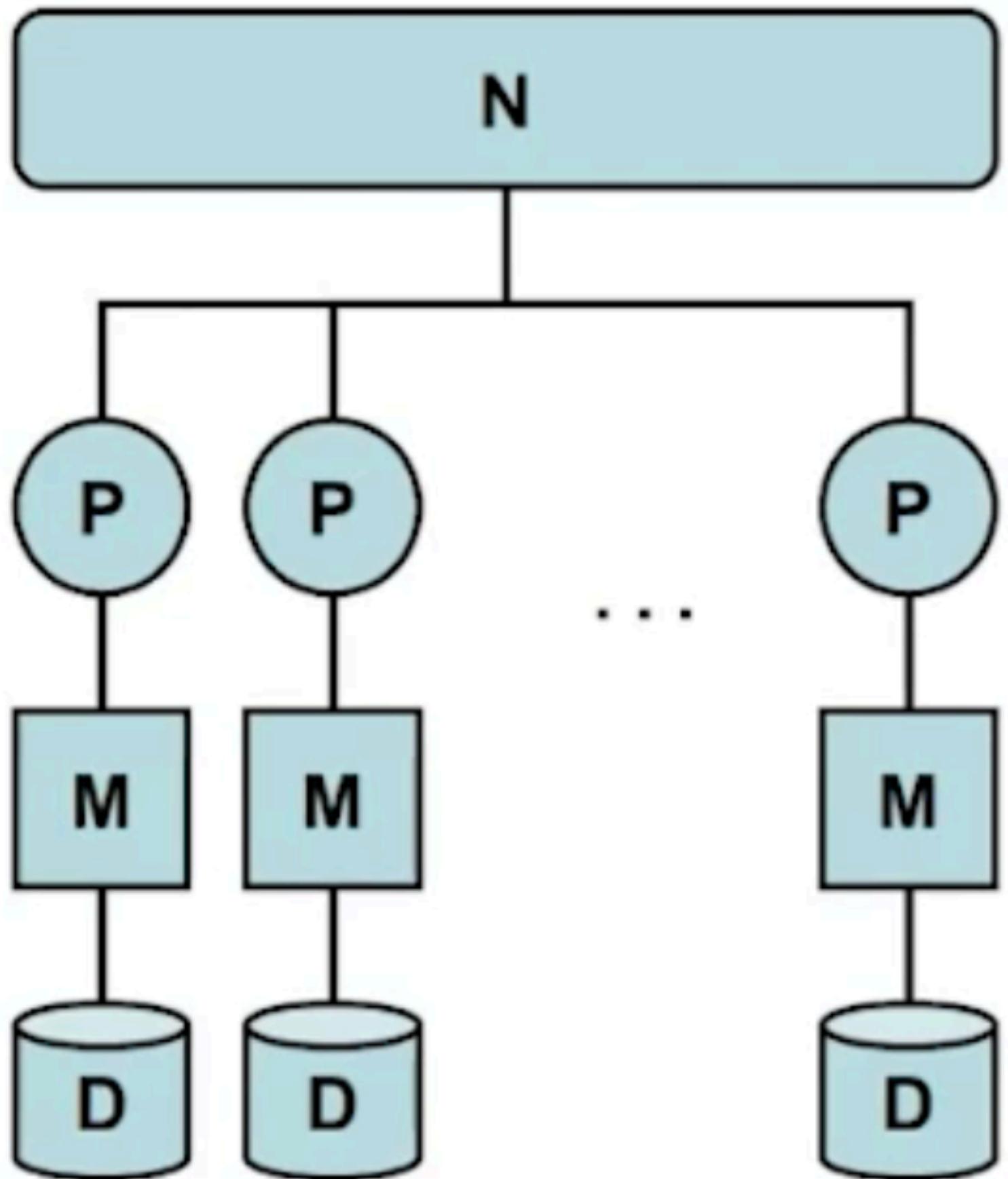
- › У всех процессоров общая RAM
- › Все диски доступны в равной степени и приоритете всем процессорам
- › Межпроцессорные коммуникации - через RAM
- › По сути - наш обычный компьютер

SD (Shared Disk)



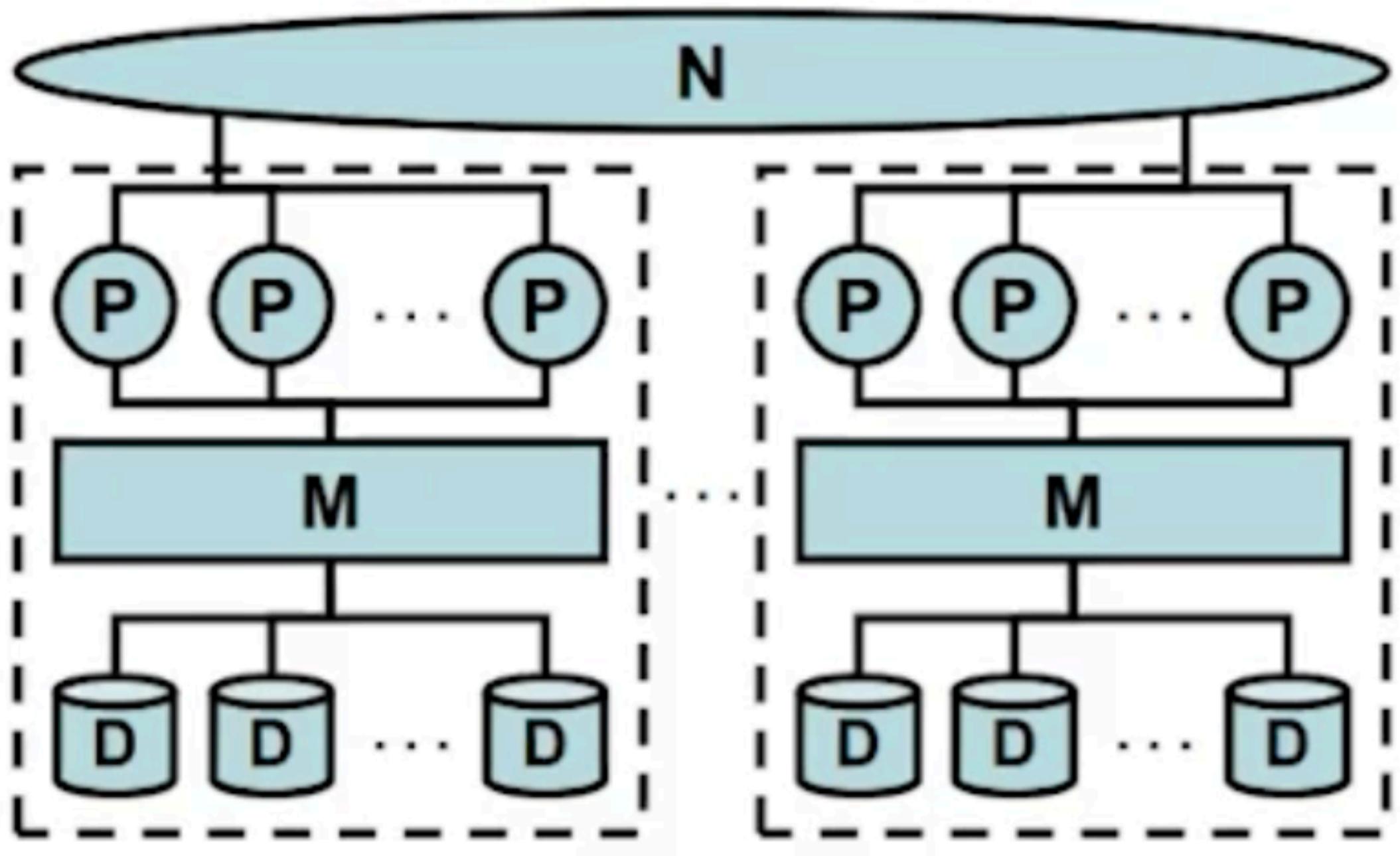
- › У каждого процессора своя RAM
- › Все диски доступны в равной степени и приоритете всем процессорам
- › Межпроцессорные коммуникации - через IO

SN (Shared Nothing)



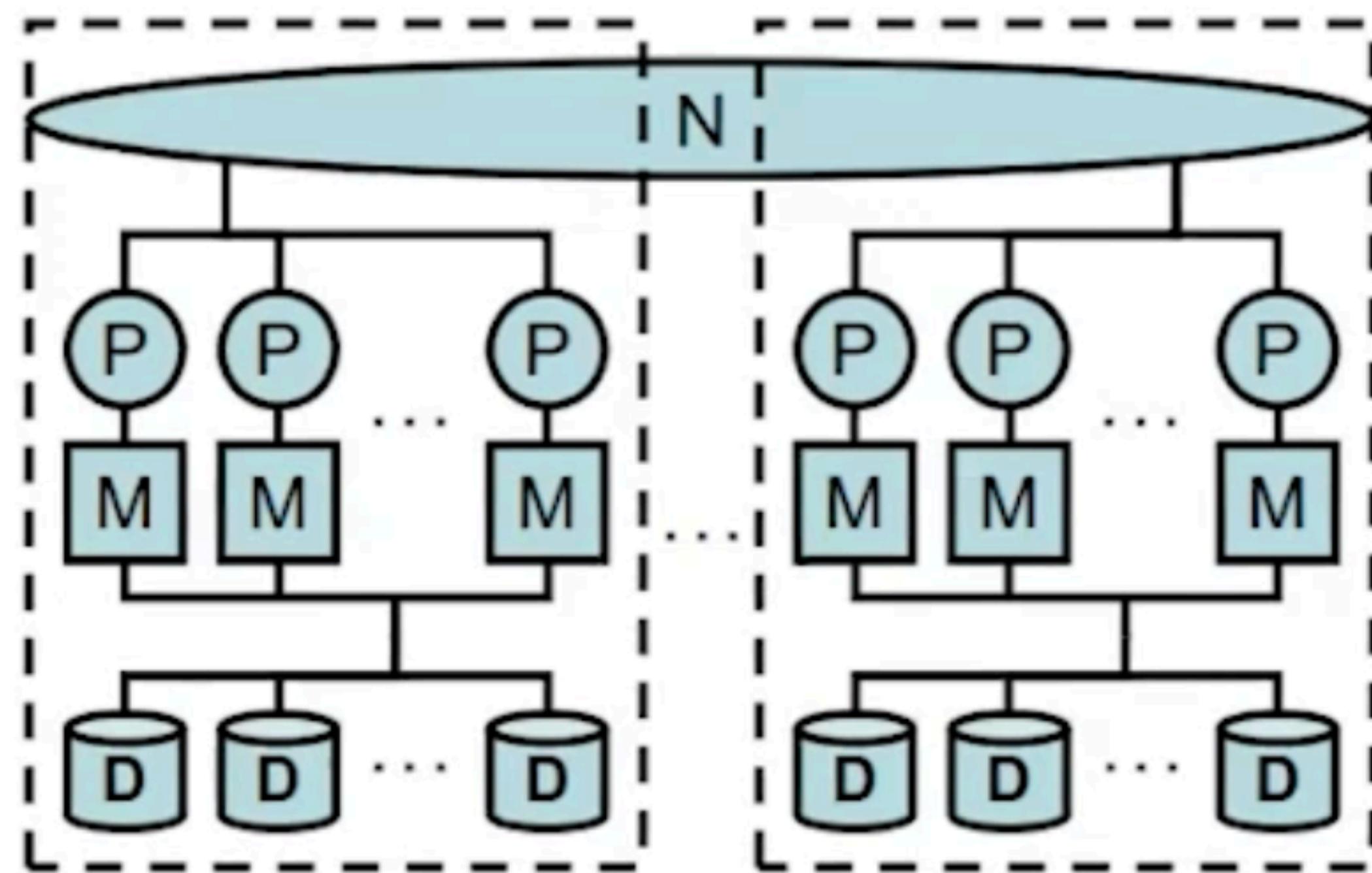
- › Каждый процессор имеет свою оперативную память и диск
- › Межпроцессорные коммуникации - через IO
- › Базовая архитектура всех MPP-расчетов: Hadoop, Vertica, GreenPlum и так далее

CE (Clustered Everything)



- › Несколько SE-кластеров, связанных сетью
- › Межпроцессорные коммуникации - через RAM
- › МежклUSTERНЫЕ коммуникации - через IO

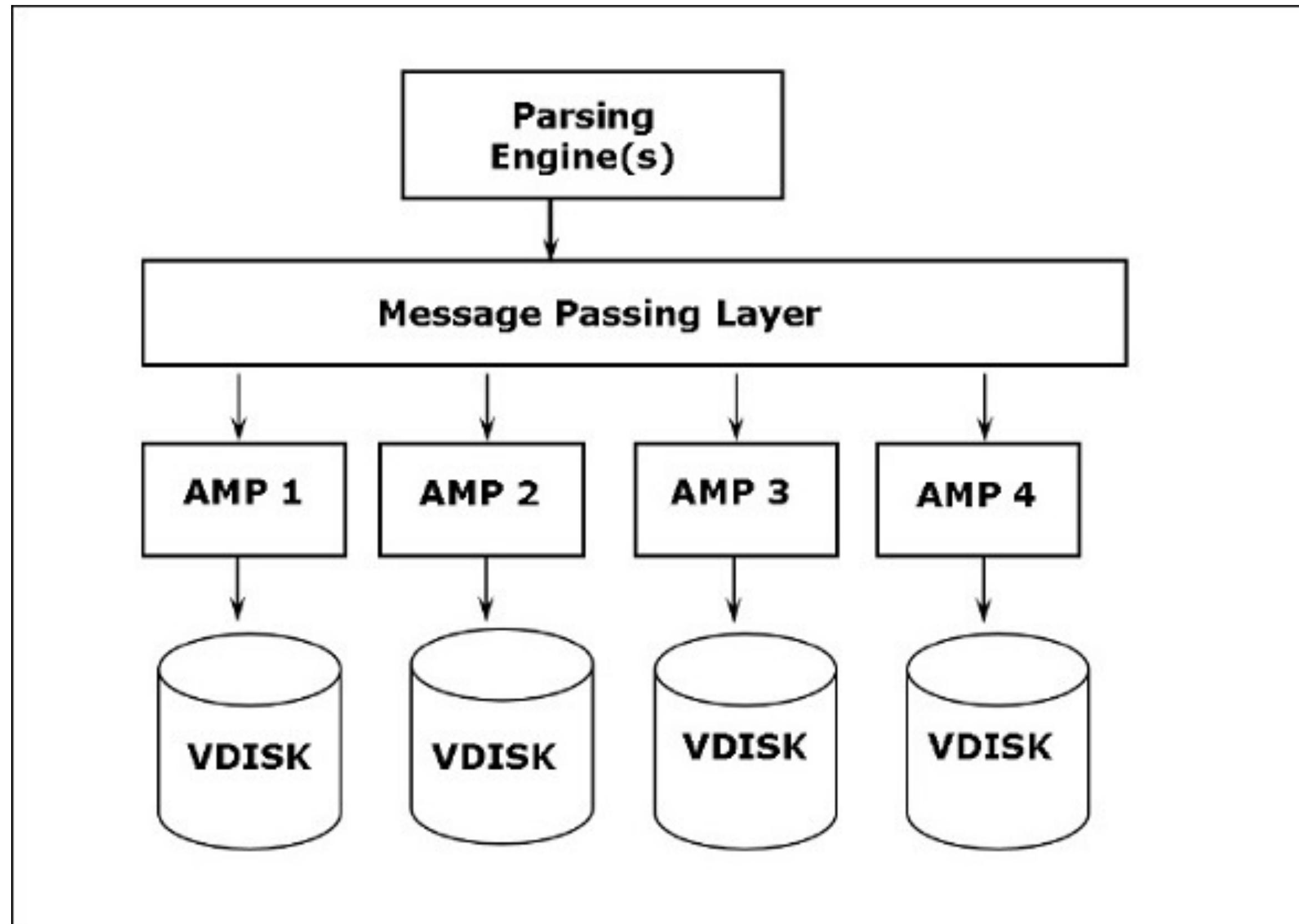
CD (Clustered Disk)



- › Несколько SD-кластеров, связанных сетью
- › Межпроцессорные коммуникации - через локальную сеть
- › МежклUSTERНЫЕ коммуникации - через глобальную сеть

2 - Архитектуры МРР

Teradata



Parsing Engines:

- › Управление сессиями пользователей
- › Чтение и оптимизация запросов
- › Выдача результатов пользователю

Message Processing Layer:

- › Посредник между РЕ и AMP

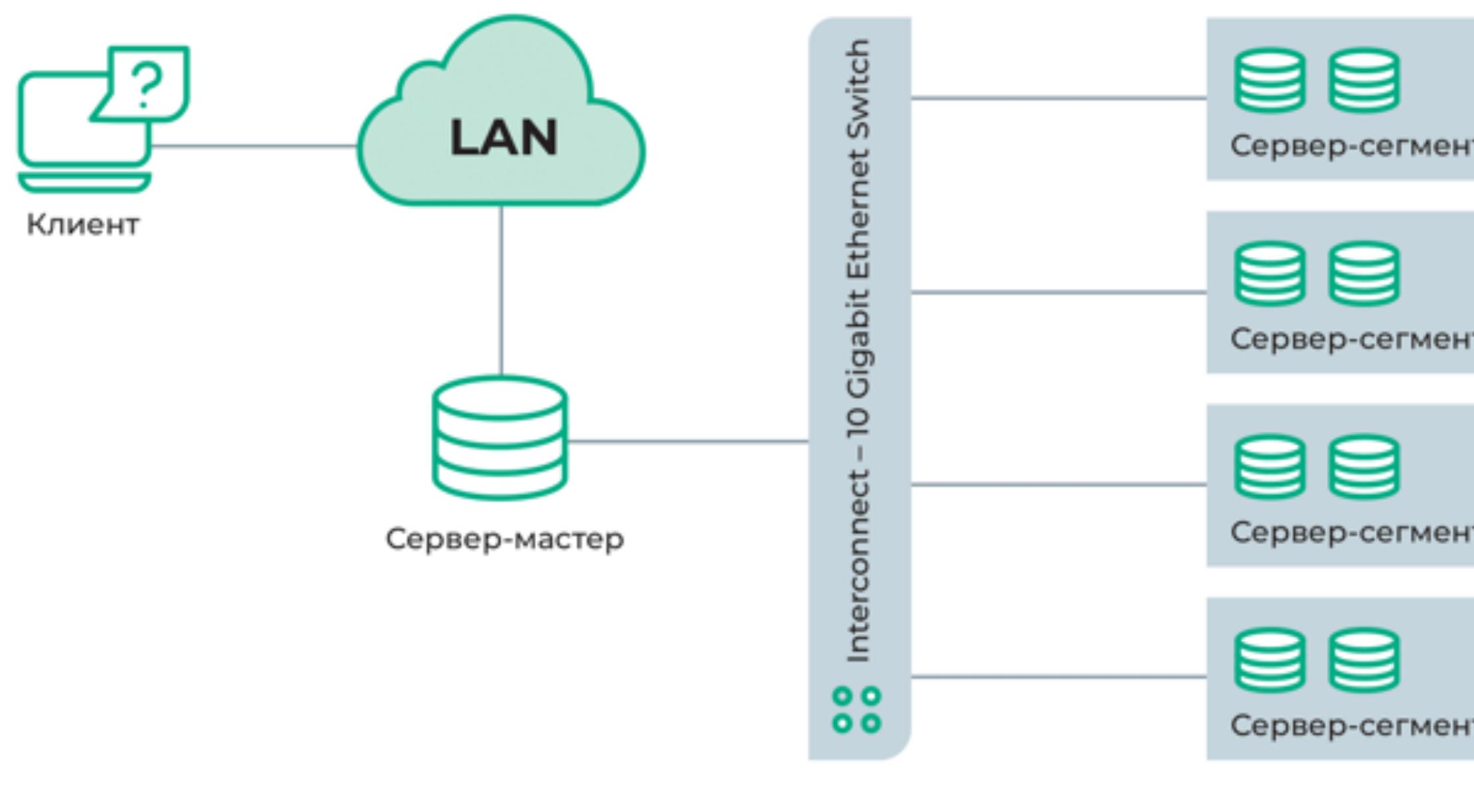
Access Module Processor:

- › Управляет выделенным пространством
- › Выполняет отправленные операции

Virtual Disks:

- › Непосредственно выделенное дисковое пространство

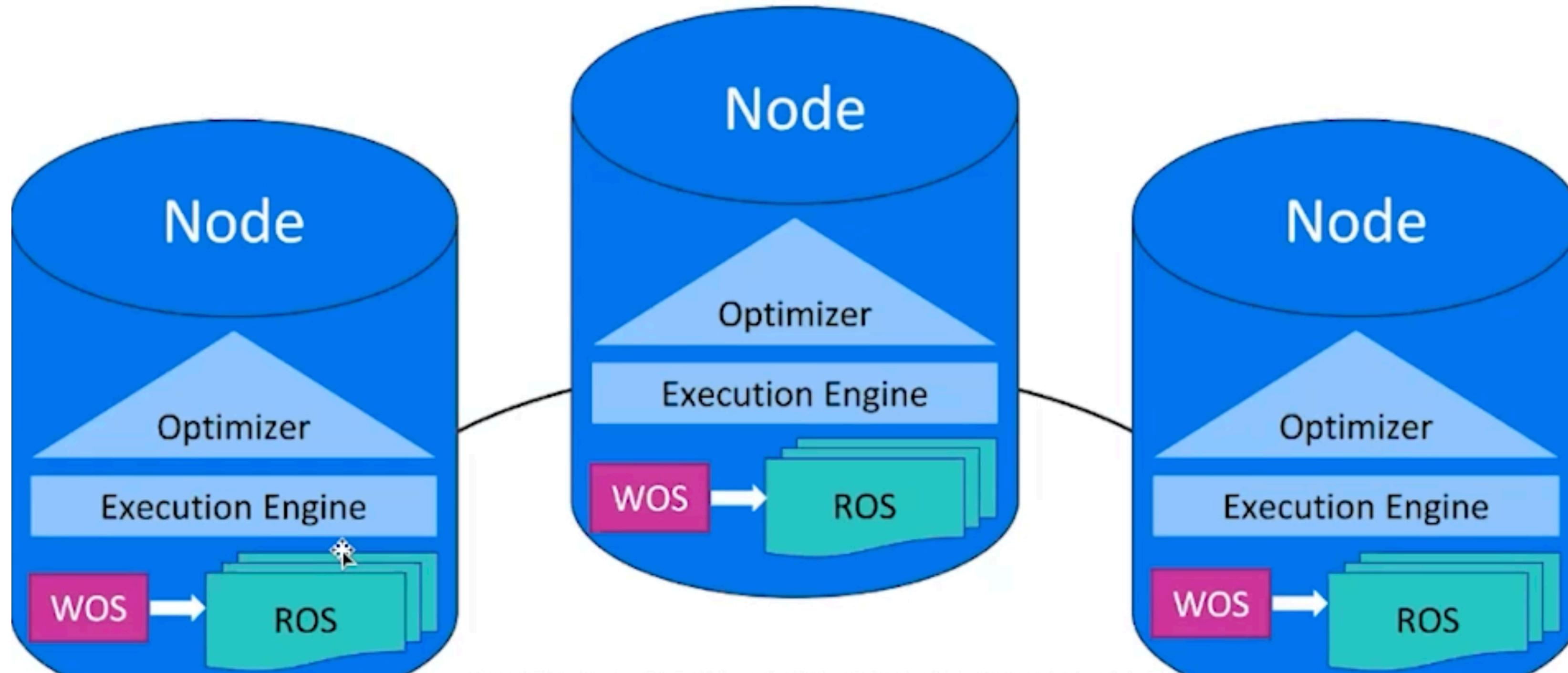
Greenplum



- › В GP все чуть хуже - есть одна (или две) мастер-ноды - это точка отказа и бутылочное горлышко в системе
- › Внутри сегментов по сути лежат маленькие постгрессы
- › Умеет как в локальные, так и в распределенные JOIN
- › Полностью соответствует ACID
- › Нативно поддерживает постгресовые приколы (PostGIS, нативная репликация по CDC)
- › Полностью соответствует ACID
- › Умеет в масштабирование “на горячую”

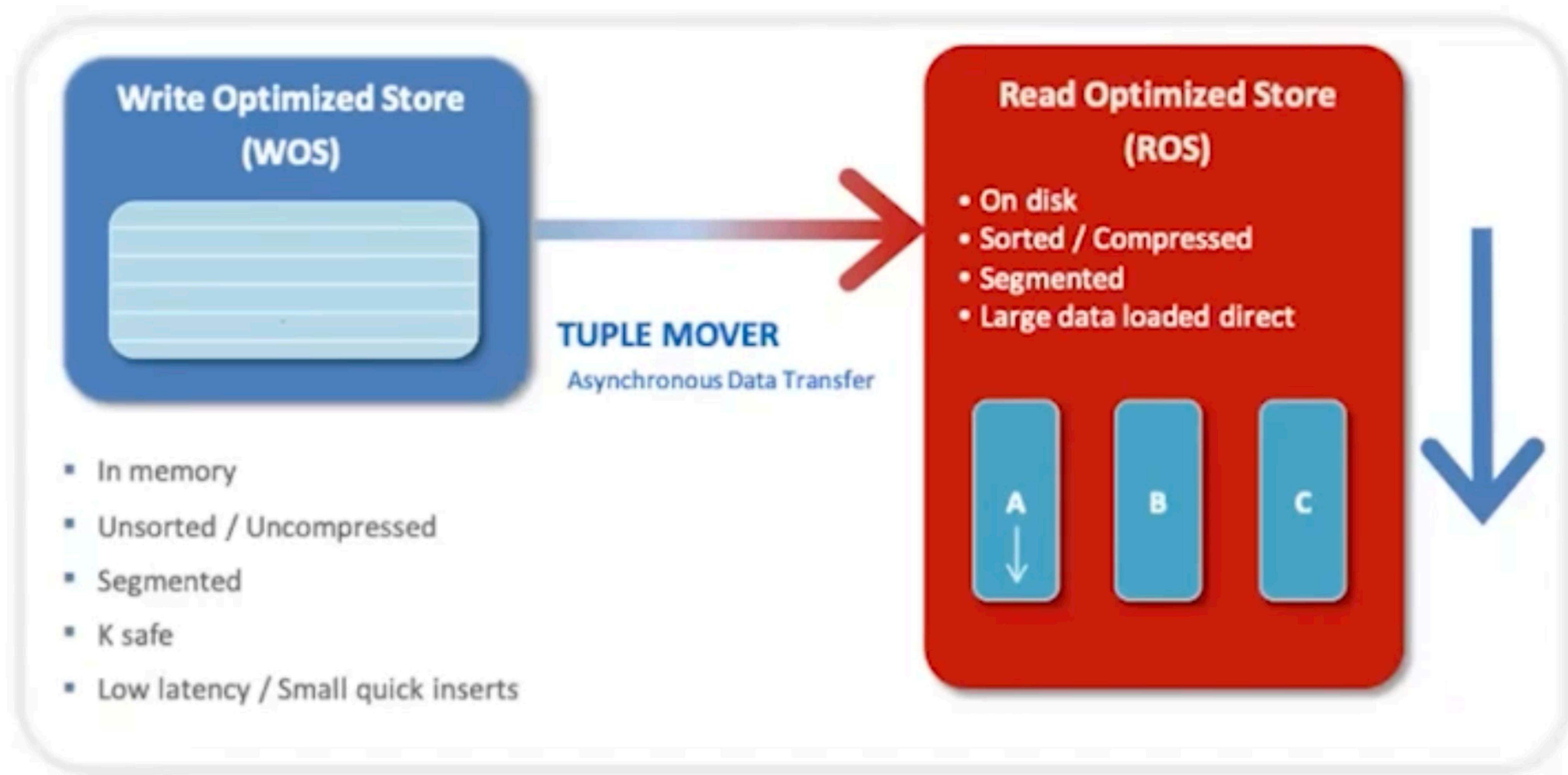
Pic. 3

Vertica



- › Похожа на Teradata
- › Исключительно колоночная СУБД (!)
- › Очень гибко администрируется
- › Шикарный инструмент проекций
- › WOS / ROS - память

Vertica



Фрагментарный параллелизм

Во всех перечисленных системах есть способ задать распределение по узлам:

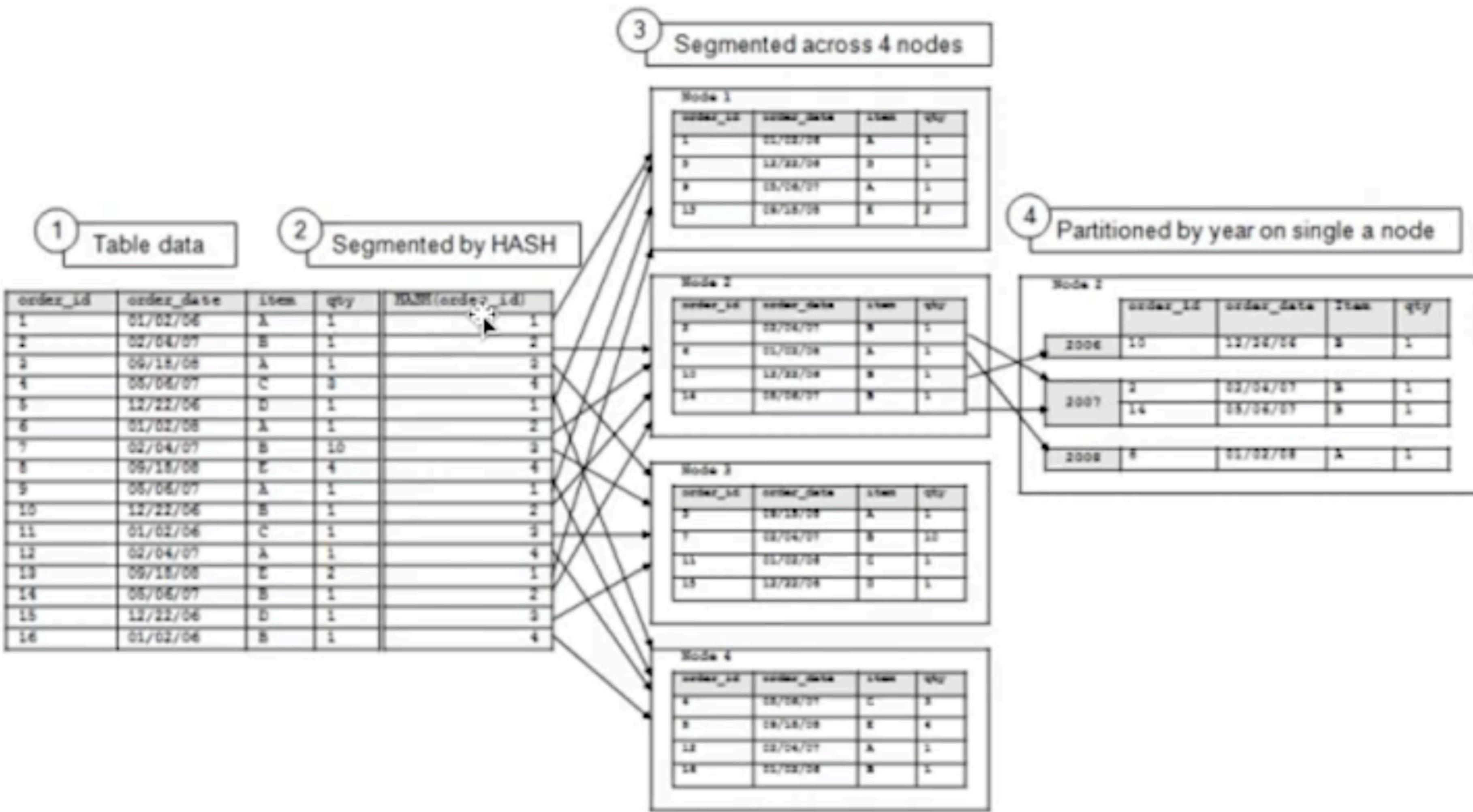
- › Teradata - primary index
- › Greenplum - distribution key
- › Vertica - segmentation key

Партицирование

Физическое разделение данных на диске по какому-то полю

- › Данные разделяются по нодам в зависимости от ключа сегментации
- › В каждой номе данные делятся на фрагменты по ключуパーティрования
- › Внутри партиции строки сортируются в зависимости от выбранного ключа сортировки
- › Ключパーティрования может не совпадать с ключом сегментации

Сегментирование иパーティционирование вместе



Проблема выбора ключа сегментации

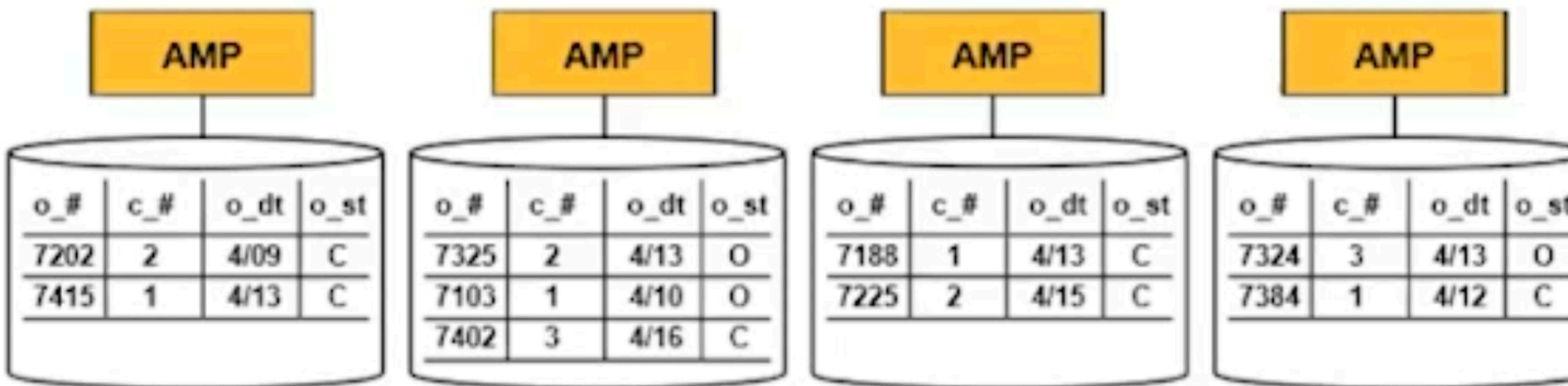
Критерии выбора ключа сегментации:

- › Доступ к данным ключа сегментации должен осуществляться чаще остальных
- › Распределение значений должно быть равномерным
- › В больших таблицах количество уникальных значений ключа должно превышать количество нод как минимум в 10 раз
- › Изменения в данных ключа сегментации не должны происходить слишком часто

Пример

Orders

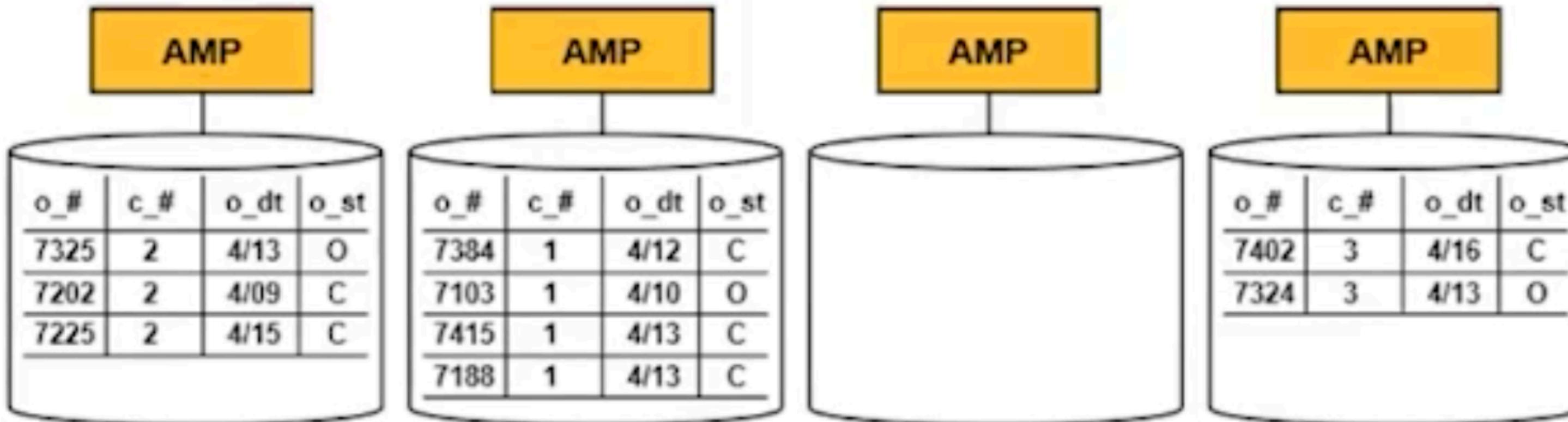
Order Number	Customer Number	Order Date	Order Status
PK			
UPI			
7325	2	4/13	O
7324	3	4/13	O
7415	1	4/13	C
7103	1	4/10	O
7225	2	4/15	C
7384	1	4/12	C
7402	3	4/16	C
7188	1	4/13	C
7202	2	4/09	C



Пример

Orders

Order Number	Customer Number	Order Date	Order Status
PK			
	NUPI		
7325	2	4/13	O
7324	3	4/13	O
7415	1	4/13	C
7103	1	4/10	O
7225	2	4/15	C
7384	1	4/12	C
7402	3	4/16	C
7188	1	4/13	C
7202	2	4/09	C



Пример

Orders

Order Number	Customer Number	Order Date	Order Status
PK			
			NUPI
7325	2	4/13	O
7324	3	4/13	O
7415	1	4/13	C
7103	1	4/10	O
7225	2	4/15	C
7384	1	4/12	C
7402	3	4/16	C
7188	1	4/13	C
7202	2	4/09	C

AMP

AMP

AMP

AMP

o_#	c_#	o_dt	o_st
7402	3	4/16	C
7202	2	4/09	C
7225	2	4/15	C
7415	1	4/13	C
7188	1	4/13	C
7384	1	4/12	C

o_#	c_#	o_dt	o_st
7103	1	4/10	O
7324	3	4/13	O
7325	2	4/13	O

Проблема использованияパーティционирования

Плюсы:

- › Если на большой таблице примененоパーティционирование, то читается только необходимый блок данных

Минусы:

- › Дополнительные накладные расходы на чтение/запись файлов
- › Запросы, которые не используют колонкиパーティционирования, работают хуже
- › Не работает MergeJoin без использования колонокパーティционирования

Фрагментарный параллелизм

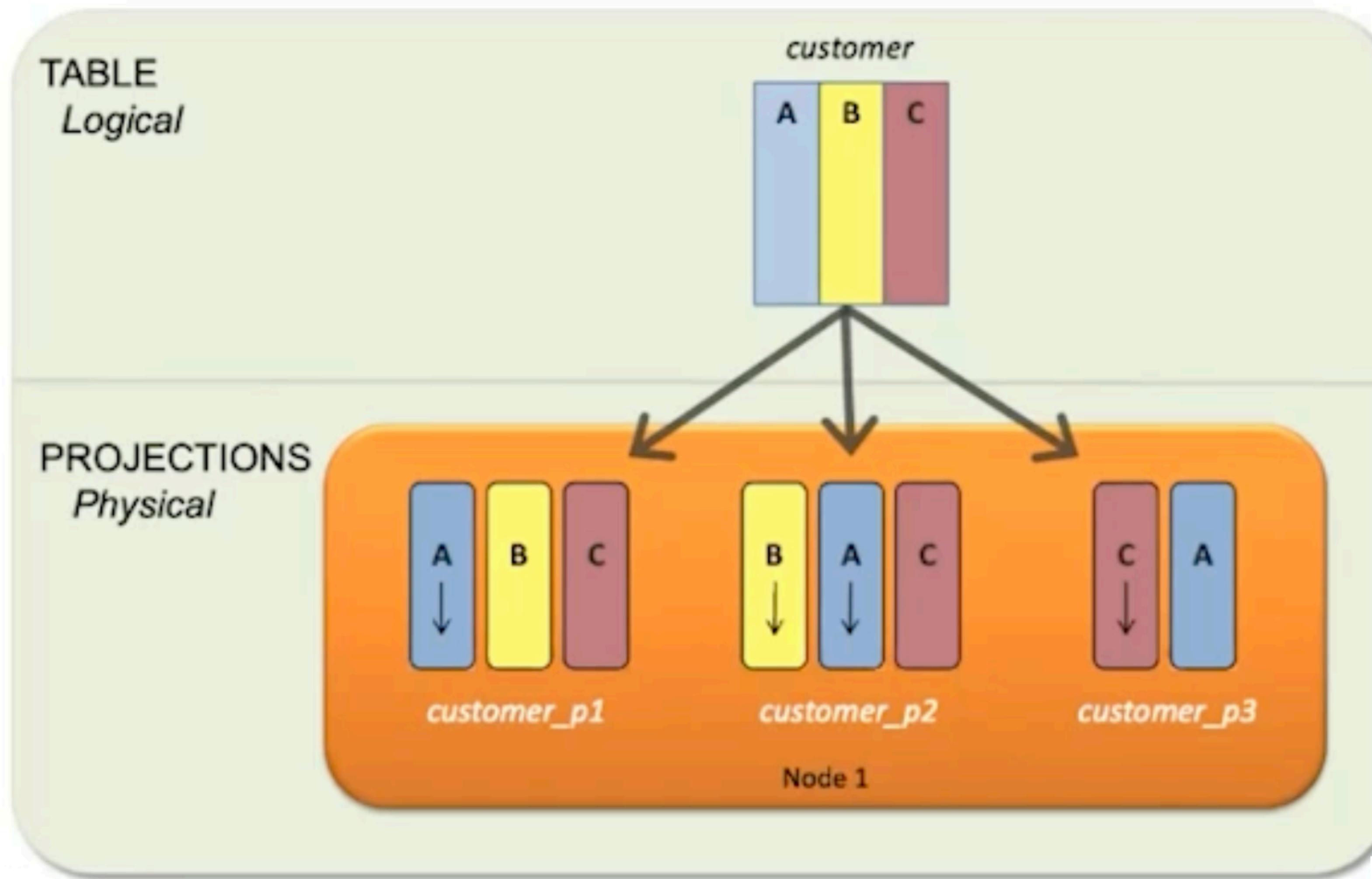
Во всех перечисленных системах есть способ задать распределение по узлам:

- › Terradata - primary index
- › Greenplum - distribution key
- › Vertica - segmentation key

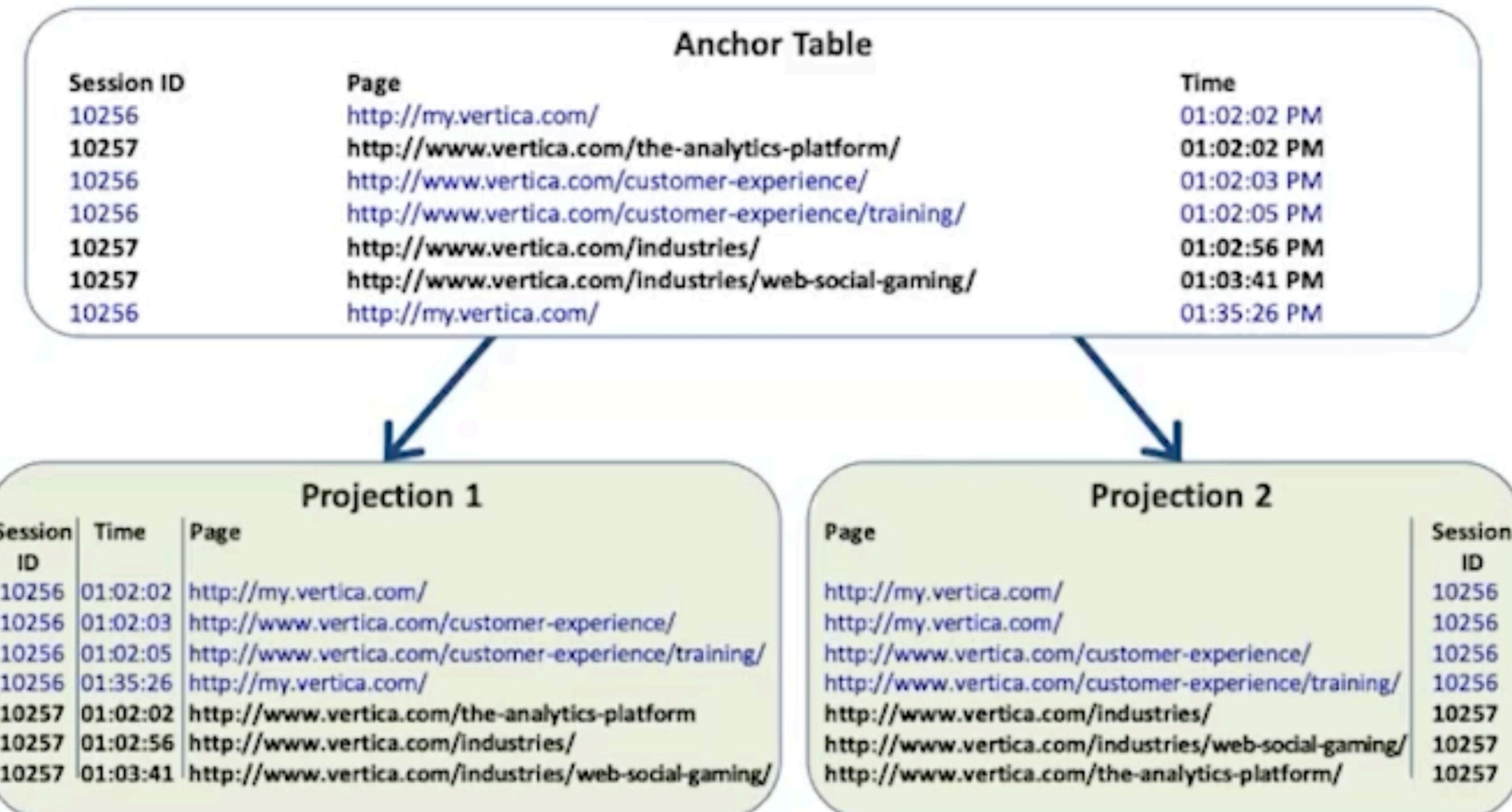
Методы доступа к данным:

- › Full scan - все МРР
- › Обращение по ключу сегментации - все МРР
- › Использование проекций - Vertica
- › Использование вторичного индекса - Greenplum

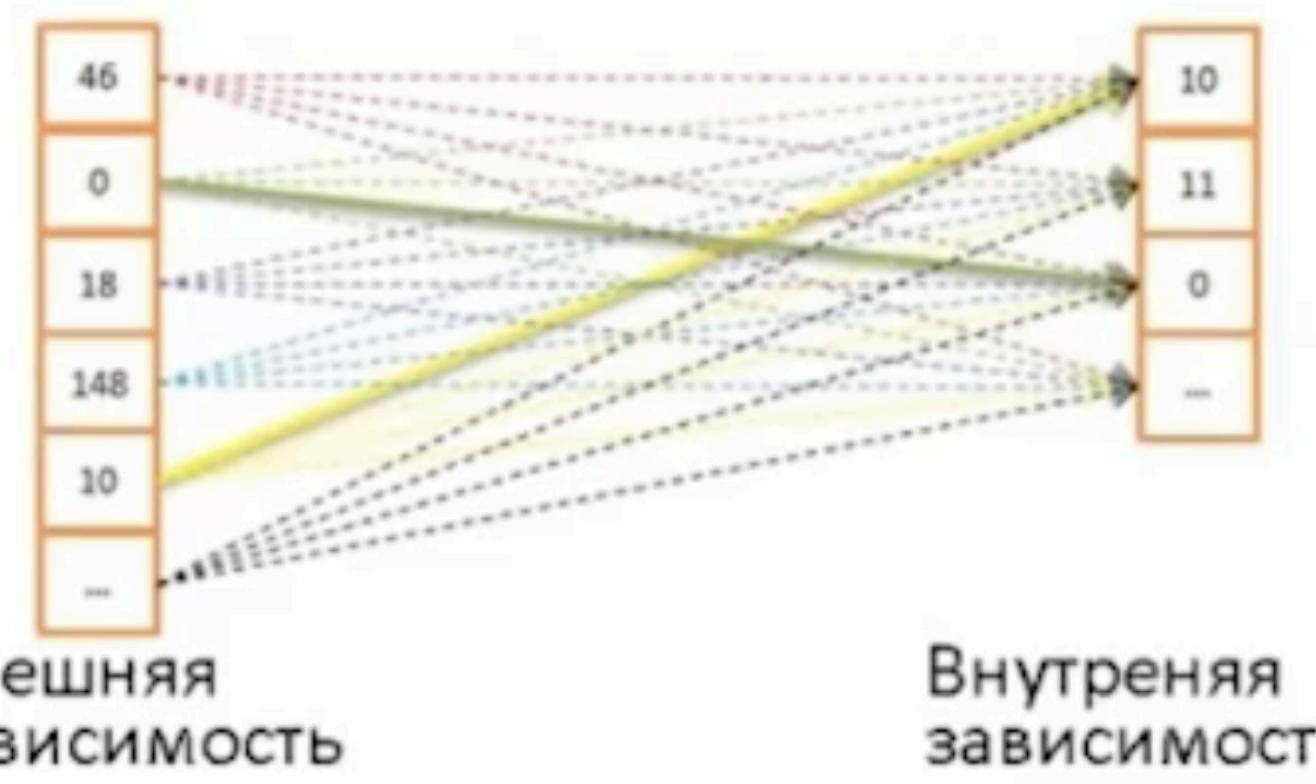
Проекции



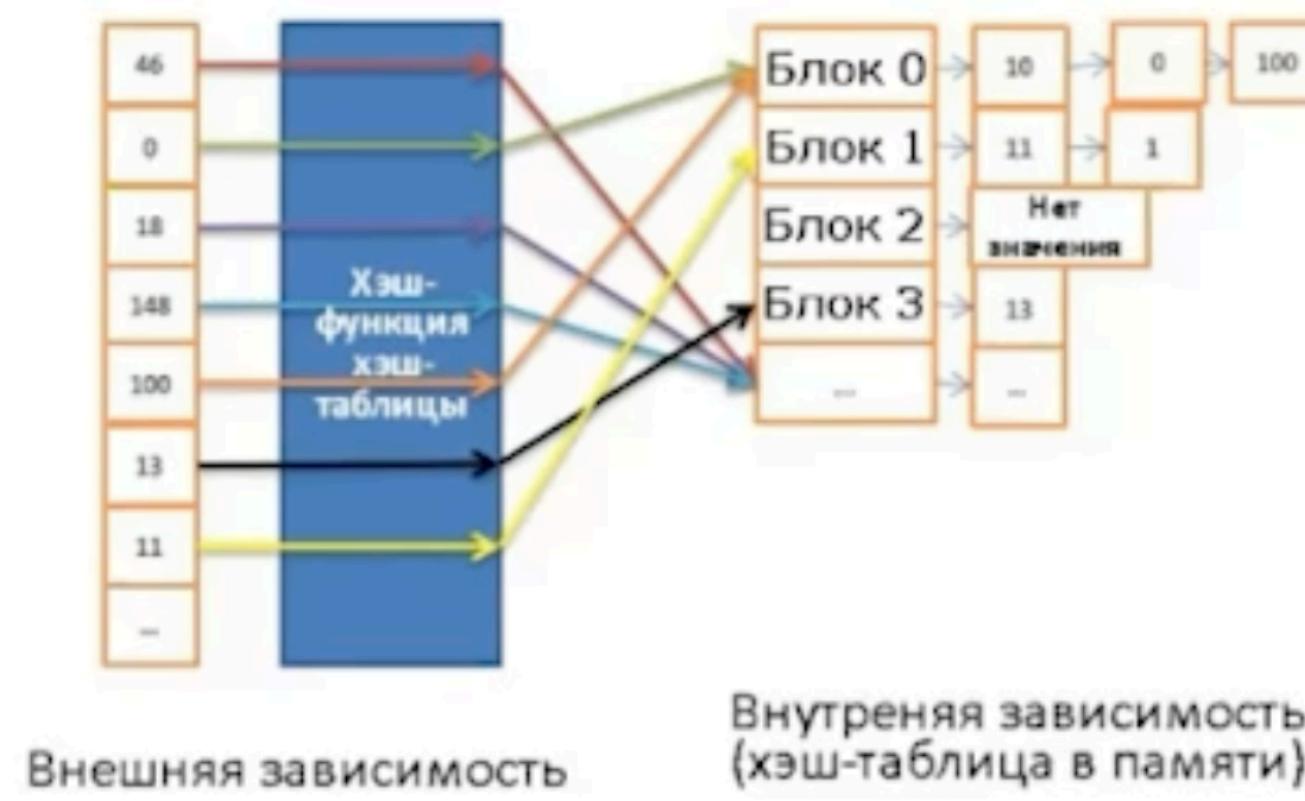
Проекции



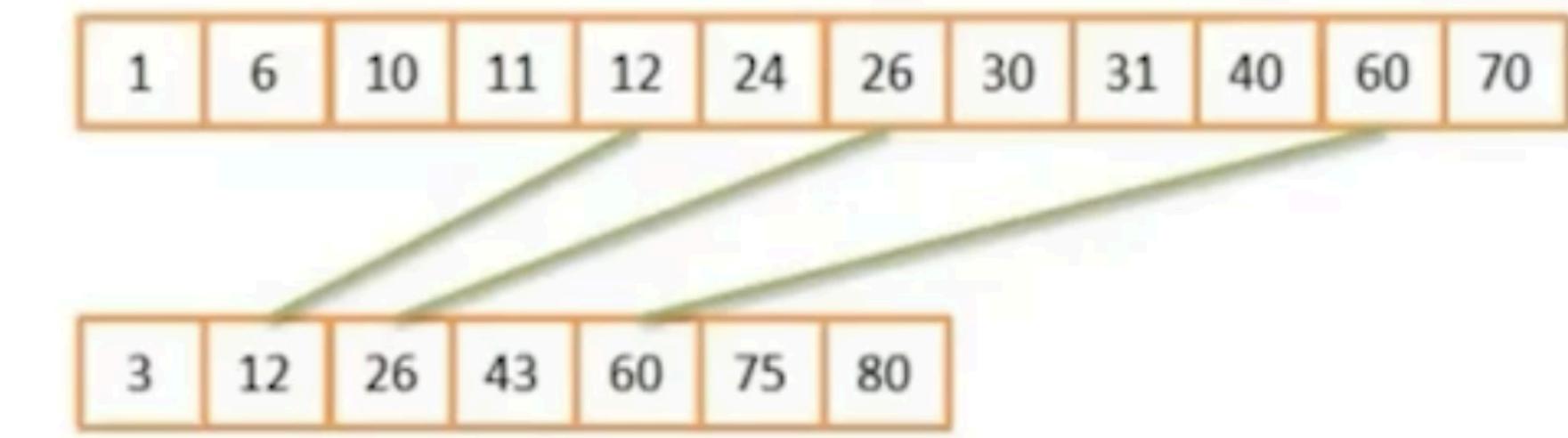
Виды JOIN



Nested loop



Hash join



Merge join

Merge join на МРР

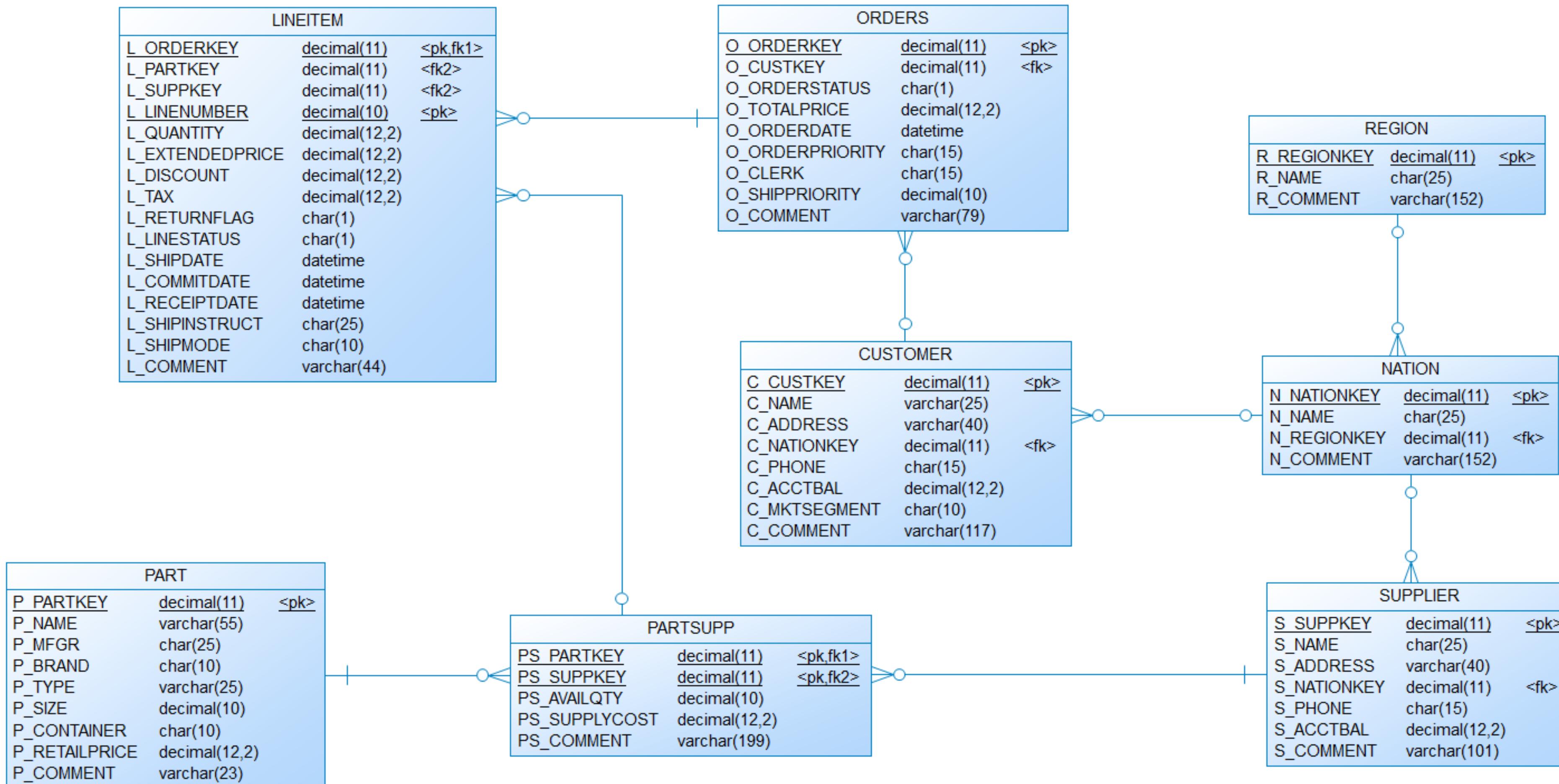
- › Необходимо, чтобы данные были одинаково распределены
- › Блоки таблицы читаются только один раз
- › В большинстве случаев быстрее других join

Как работает на стороне БД:

- › Находим меньшее множество
- › Решаем, что с ним делать - перераспределение/дублирование/сортировка
- › Делаем join

3 - Демо: GreenPlum

TCP-H



Create table

Описание таблицы

```
1 create table schema_name.table_name (
2   ...
3 )
4 distributed by (key);
5
```

Через запрос

```
1 create table schema_name_1.table_name_1 as
2 select *
3 from schema_name_2.table_name_2
4 distributed by (key);
5
```

Список полей и типов

```
select
    column_name,
case
    when data_type in ('character', 'character varying')
        then data_type || '(' ||
character_maximum_length::character varying(7) || ')'
    else data_type
end
from information_schema.columns
where 1=1
    and table_schema = 'public'
order by ordinal_position;
```

Distribution

› DISTRIBUTED BY ()

Явно задаем поля, по которым будет происходить дистрибуция данных по нодам

› DISTRIBUTED RANDOMLY

GP случайно раскидывает данные и гарантирует равномерность

› DISTRIBUTED REPLICATED

На каждой ноде создается копия таблицы

Storage Parameters

WITH (storage_parameter=value)

- › AppendOptimized - формат хранения данных, AppendOptimized или Heap
- › BlockSize - размер блока в байтах
- › Orientation - строчное или колодочное хранение (только для АО)
- › Checksum - проверка чексуммы (только для АО)
- › CompressType - алгоритм сжатия, ZLIB (default), ZSTD, RLE_TYPE или QUICKLZ1 (только для АО)
- › CompressLevel - уровень сжатия, от 1 до 12; чем больше, тем “злее” сжатие

Heap vs AO

AppendOptimized

- › Изначально не поддерживал изменения
- › Может быть строчным или поколоночным
- › Поддерживает сжатие
- › Подходит для задач обработки батчами (OLAP нагрузка)

Heap - старый формат, доставшийся в наследство от Postgres.

- › Нет сжатия
- › Быстрее вставка и изменения
- › Подходит под задачи точечного изменения данных (OLTP нагрузка)

Heap vs AO

- › Используйте heap для таблиц и разделов, которые будут получать итеративные пакетные и одноэлементные операции UPDATE, DELETE и INSERT
- › Используйте heap, которые будут получать конкурентные операции UPDATE, DELETE и INSERT
- › Используйте AO для таблиц и разделов, которые нечасто обновляются после начальной загрузки, а последующие вставки выполняются только в больших батчевых операциях
- › Избегайте выполнения одноэлементных операций INSERT, UPDATE или DELETE в AO-таблицах
- › Избегайте одновременного выполнения пакетных операций UPDATE или DELETE для AO-таблиц. Допускаются одновременные пакетные операции INSERT

Row vs Column

- › Используйте строки для рабочих нагрузок с обновлениями или частые вставки
- › Используйте строки, когда читается много колонок
- › Используйте строки, если нагрузка смешанная
- › Используйте колоночное хранение, если выборки чаще по небольшому набору столбцов
- › Используйте колоночное хранение, если колонки меняются отдельно

Партицирование

```
CREATE TABLE sales (
    id int, year int, qtr int,
    c_rank int, code char(1), region text
)
DISTRIBUTED BY (id)
PARTITION BY LIST (code)
(
    PARTITION sales VALUES ('S'),
    PARTITION returns VALUES ('R')
);
```

Партицирование

```
CREATE TABLE sales (
    id int, year Int, qtr int,
    c_rank int, code char(1), region text
)
DISTRIBUTED BY (id)
PARTITION BY LIST (code)
    SUBPARTITION BY RANGE (c_rank)
        SUBPARTITION by LIST (region)

(PARTITION sales VALUES ("S")
( SUBPARTITION cr1 START (1) END (2) )
( SUBPARTITION ca VALUES ("CA") ),
SUBPARTITION cr2 START (3) END (4)
( SUBPARTITION ca VALUES ("CA") )
),

(PARTITION returns VALUES ("R")
( SUBPARTITION cr1 START (1) END (2)
( SUBPARTITION ca VALUES ("CA") ),
SUBPARTITION c2 START (3) END (4)
( SUBPARTITION ca VALUES ("CA") )
)
);
```

Партицирование

```
CREATE TABLE sales (
    id int, year Int, qtr int,
    c_rank int, code char(1), region text
)
DISTRIBUTED BY (id)
PARTITION BY LIST (code)
    SUBPARTITION BY RANGE (c_rank)
        SUBPARTITION by LIST (region)

(PARTITION sales VALUES ("S")
    ( SUBPARTITION cr1 START (1) END (2) )
    ( SUBPARTITION ca VALUES ("CA") ),
    SUBPARTITION cr2 START (3) END (4)
    ( SUBPARTITION ca VALUES ("CA") )
),
(PARTITION returns VALUES ("R")
    ( SUBPARTITION cr1 START (1) END (2)
    ( SUBPARTITION ca VALUES ("CA") ),
    SUBPARTITION c2 START (3) END (4)
    ( SUBPARTITION ca VALUES ("CA")) )
)
);
```

```
CREATE TABLE sales (
    id int, year int, qtr int,
    c_rank int, code char(1), region text
)
DISTRIBUTED BY (id)
PARTITION BY RANGE (year)
    SUBPARTITION BY RANGE (qtr)
        SUBPARTITION TEMPLATE (
            START (1) END (5) EVERY (1),
            DEFAULT SUBPARTITION bad_qtr )
    SUBPARTITION BY LIST (region)
        SUBPARTITION TEMPLATE (
            SUBPARTITION usa VALUES ('usa'),
            SUBPARTITION europe VALUES ('europe'),
            SUBPARTITION asia VALUES ('asia'),
            DEFAULT SUBPARTITION other_regions )
    ( START (2009) END (2011) EVERY (1), DEFAULT
    PARTITION outlying_years );
```

Добавление колонки

› В случае **AppendOnly** таблиц (**только поколоночных!**) сжатие и его уровень не наследуются. Во время добавления новой колонки необходимо явно указывать тип и уровень сжатия (**значения в compressstype и compresslevel вводить такие же как у таблицы**):

```
ALTER TABLE schema_name,table name  
ADD COLUMN column _name COLUMN_TYPE ENCODING  
(compressstype=zlib,compresslevel=2);
```

Оптимизатор

- › В GreenPlum есть 2 оптимизатора: Legacy и ORCA
- › ORCA написана командой GP с упором на работу с колодочными данными
- › Legacy - остался в наследство от PostgreSQL

- › Для использования ORCA - `set optimizer = on;` (default)
- › Для использования Legacy - `set optimizer = off;`

CTE

```
WITH regional_sales AS (
    SELECT region, SUM(amount) AS total sales
    FROM orders
    GROUP BY region
), top regions AS (
    SELECT region
    FROM regional_sales
    WHERE 1=1
        and total sales > (SELECT SUM(total sales)/10 FROM regional sales)
)
SELECT
    region, product,
    SUM(quantity) AS product_ units, SUM(amount) AS product sales
FROM orders
WHERE 1=1
    and region IN (SELECT region FROM top regions)
GROUP BY region, product;
```

Итого

Для **Greenplum** операции работы с дисками и сетью обходятся дорого, поэтому стоит придерживаться некоторым правилам:

- › стараться делать **фильтрацию данных** как можно раньше;
- › к partitionированным таблицам **применять фильтр по полю партиции**;
- › избегать большого количества подзапросов в **СТЕ конструкции** (WITH ... AS (...));
- › для долгих и "тяжелых" расчетов **использовать временные таблицы** (CREATE TEMPORARY TABLE ...), которые удаляются автоматически при завершении транзакции;
- › изучать **план запроса** (ANALYZE ...);
- › не делать **JOIN'ы больших таблиц** по не уникальному ключу, чтобы не генерировать дубли (оценочное количество записей во время выполнения запроса можно узнать в плане запроса);
- › по возможности избегать **Redistribute Motion** (перераспределение данных по сегментам) в планах запроса;
- › избегать **Broadcast Motion** (копирование данных целиком на каждый сегмент), когда это будет применено к таблице с большим количеством данных.

4 - DWH. Теормин

Первичный ключ

Будем называть первичным ключом подмножество атрибутов отношения, если оно соответствует условиям:

- › **Уникальность** - в отношении не может быть более одного кортежа с одинаковым значением
- › **Неизбыточность** - никакое его подмножество не обладает свойством уникальности

Потенциальный ключ из одного атрибута называется **простым**, из нескольких атрибутов - **составным**.

Отношение может иметь несколько потенциальных ключей. Тогда среди них выделяются первичный и альтернативные.

Нормальные формы

Здесь и далее - примеры из моей любимой [статьи на Хабре](#)

Отношение находится в **1НФ**, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице.

Нормальные формы

Здесь и далее - примеры из моей любимой статьи на Хабре

Отношение находится в **1НФ**, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице.

Вопрос: ниже 1НФ?

Фирма	Модели
BMW	M5, X5M, M1
Nissan	GT-R

1НФ

Нарушение нормализации 1НФ происходит в моделях BMW, т.к. в одной ячейке содержится список из 3 элементов: M5, X5M, M1, т.е. он не является атомарным.

Настоящая 1НФ:

Фирма	Модели
BMW	M5
BMW	X5M
BMW	M1
Nissan	GT-R

2НФ

Отношение находится во 2НФ, если оно находится в 1НФ и каждый не ключевой атрибут неприводимо зависит от первичного ключа.

2НФ

Отношение находится во 2НФ, если оно находится в 1НФ и каждый не ключевой атрибут неприводимо зависит от первичного ключа.

Вопрос: ниже 2НФ?

<u>Модель</u>	<u>Фирма</u>	Цена	Скидка
M5	BMW	5500000	5%
X5M	BMW	6000000	5%
M1	BMW	2500000	5%
GT-R	Nissan	5000000	10%

2НФ

Цена машины зависит от модели и фирмы.

Скидка зависят от фирмы, то есть зависимость от первичного ключа неполная.

Исправляется это путем декомпозиции на два отношения, в которых не ключевые атрибуты зависят от ПК.

Модель	Фирма	Цена
M5	BMW	5500000
X5M	BMW	6000000
M1	BMW	2500000
GT-R	Nissan	5000000

Фирма	Скидка
BMW	5%
Nissan	10%

ЗНФ

Отношение находится в ЗНФ, когда находится во 2НФ и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы.

Вопрос: ниже ЗНФ?

<u>Модель</u>	Магазин	Телефон
BMW	Риал-авто	87-33-98
Audi	Риал-авто	87-33-98
Nissan	Некст-Авто	94-54-12

ЗНФ

Таблица находится во 2НФ, но не в ЗНФ.

В отношении атрибут «Модель» является первичным ключом. Личных телефонов у автомобилей нет, и телефон зависит исключительно от магазина.

В результате разделения исходного отношения получаются два отношения, находящиеся в ЗНФ:

<u>Магазин</u>	Телефон
Риал-авто	87-33-98
Некст-Авто	94-54-12

<u>Модель</u>	Магазин
BMW	Риал-авто
Audi	Риал-авто
Nissan	Некст-Авто

4НФ, 5НФ и 6НФ

- › **4НФ:** 3НФ + все нетривиальные многозначные зависимости фактически являются функциональными зависимостями от ее потенциальных ключей
- › **5НФ:** 4НФ + отсутствуют сложные зависимые соединения между атрибутами
- › **6НФ:** 5НФ + не может быть далее декомпозирована без потерь

Слабая vs сильная нормализация

Критерий	Слабая нормализация (1НФ, 2НФ)	Сильная нормализация (3НФ+)
Адекватность базы данных предметной области	Хуже (-)	Лучше (+)
Легкость разработки и поддержки базы данных	Сложнее (-)	Легче (+)
Скорость вставки, обновления, удаления	Медленнее (-)	Быстрее (+)
Скорость выборки данных	Быстрее (+)	Медленнее (-)

5 - Dimensional Modeling

Философия

- › **Сущность** - это предмет, который существует независимо от других предметов
- › **Атрибут** - это средство, с помощью которого определяются свойства сущности или её связи
- › **Связь** - это отношение между экземплярами двух (или более) разных сущностей

Философия

- › **Сущность** - это предмет, который существует независимо от других предметов
- › **Атрибут** - это средство, с помощью которого определяются свойства сущности или её связи
- › **Связь** - это отношение между экземплярами двух (или более) разных сущностей
- › Вопрос: какие виды связей вы знаете?

Философия

- › **Сущность** - это предмет, который существует независимо от других предметов
- › **Атрибут** - это средство, с помощью которого определяются свойства сущности или её связи
- › **Связь** - это отношение между экземплярами двух (или более) разных сущностей
- › Вопрос: какие виды связей вы знаете?

Ноль или один

Только один

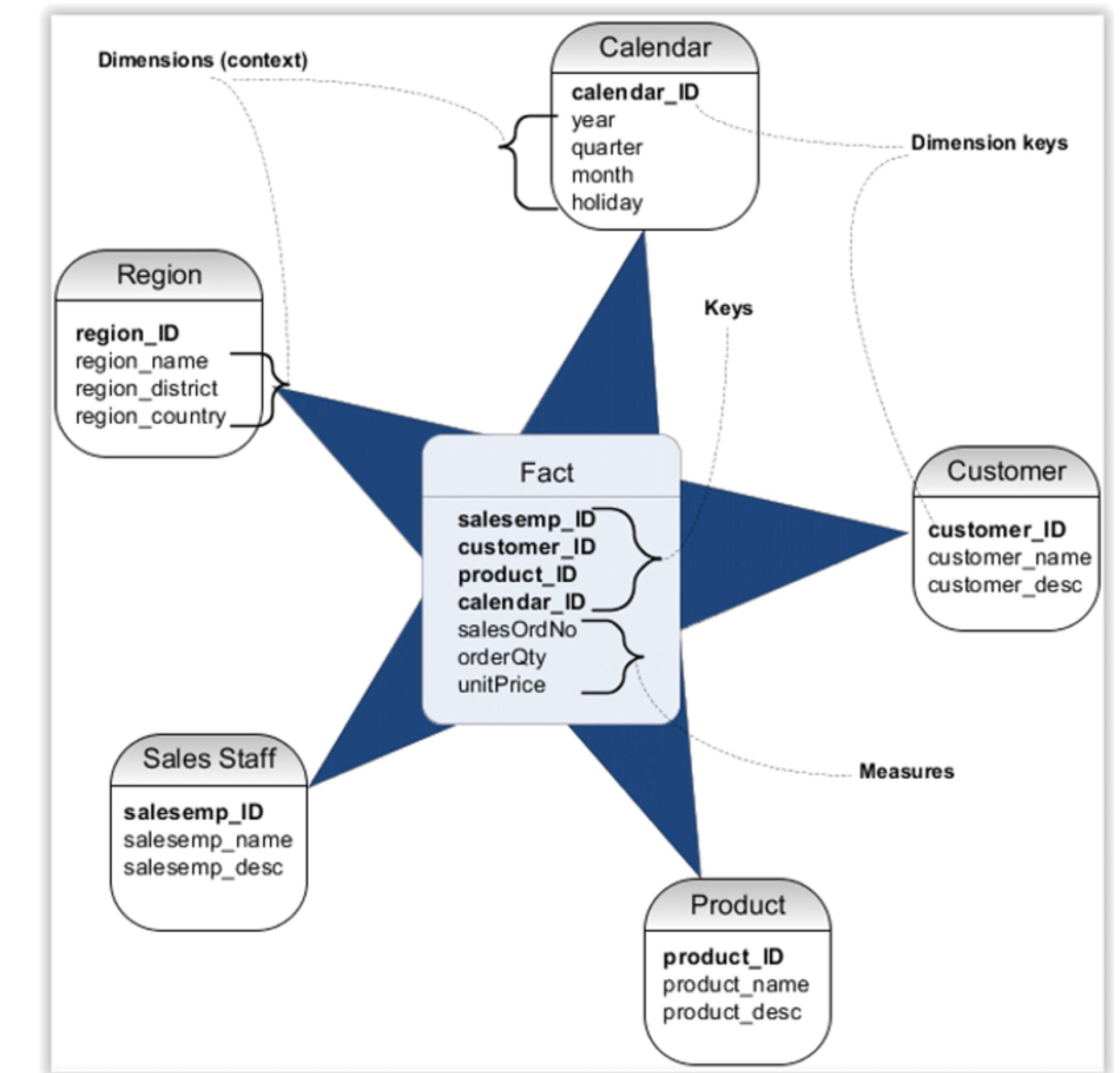
Ноль или много

Много

Один или много

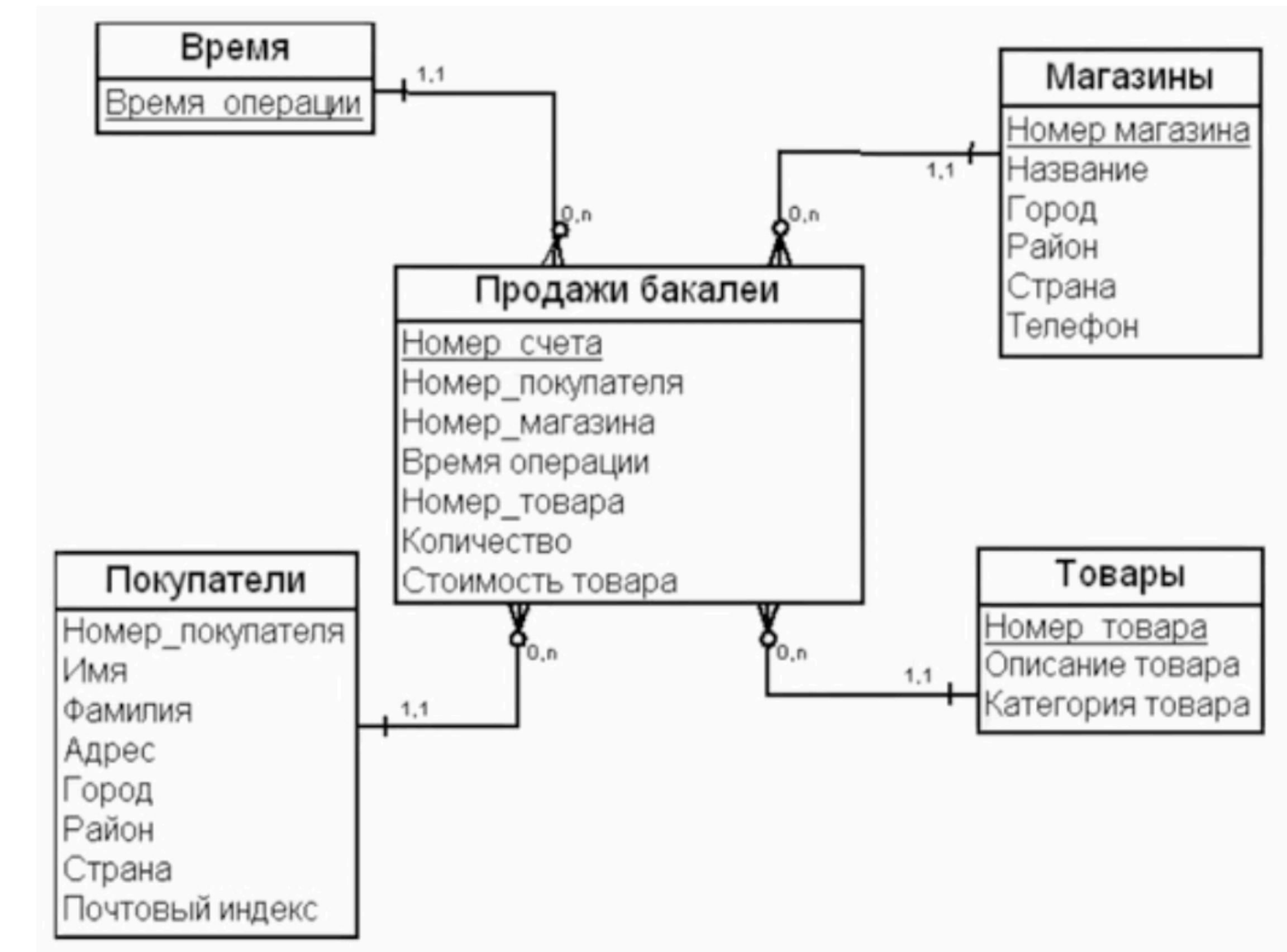
Модель “Звезда”

- › Она же пространственная модель, модель фактов и измерений, модель “сущность-связь”
- › Придумал Ральф Кимбалл
- › Состоит из 2 видов таблиц - **таблиц измерений и таблиц фактов**



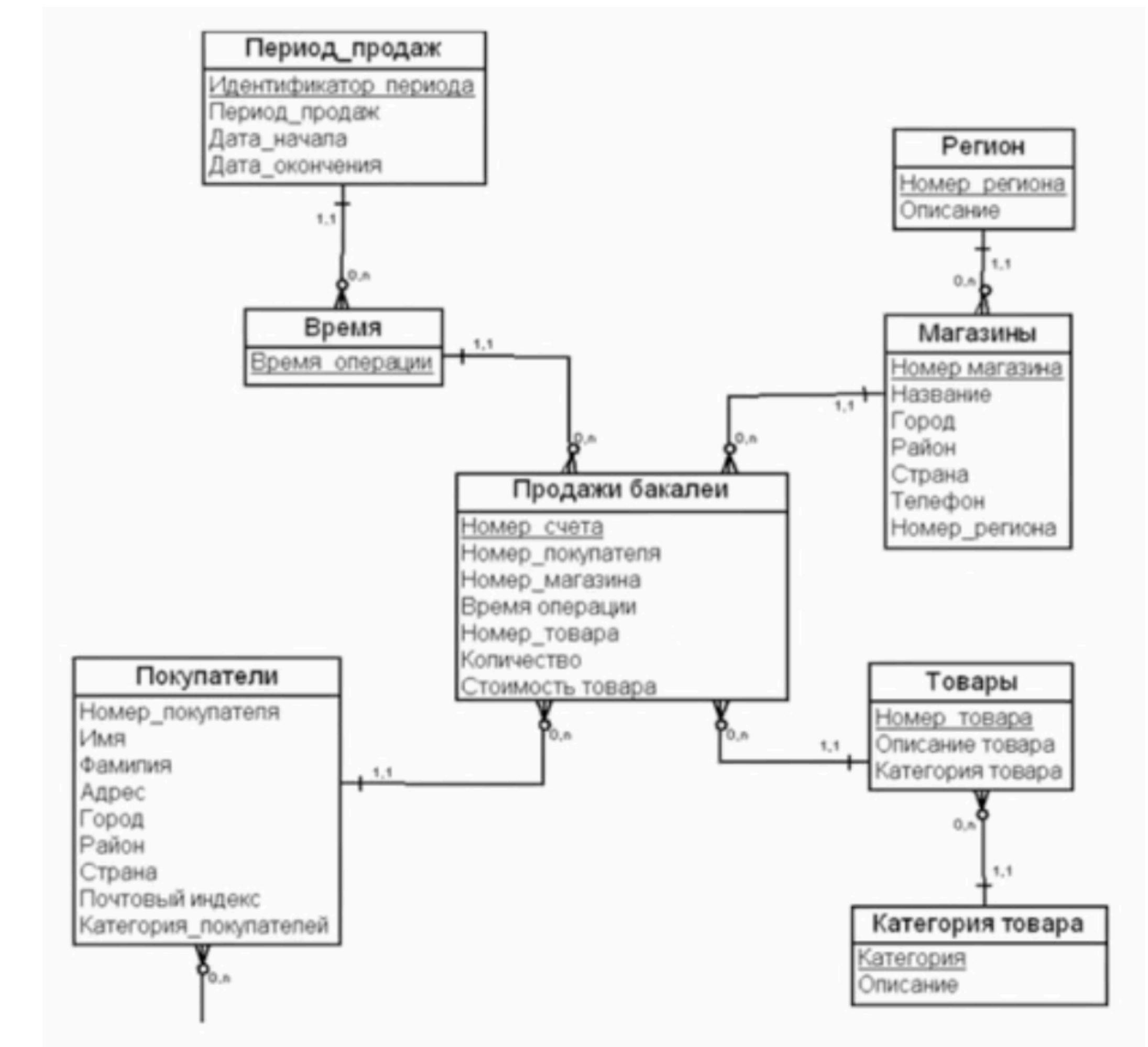
Модель “Звезда”

- › Она же пространственная модель, модель фактов и измерений, модель “сущность-связь”
- › Придумал Ральф Кимбалл
- › Состоит из 2 видов таблиц - **таблиц измерений и таблиц фактов**



Модель “Снежинка”

- › Добавляет в модель “Звезда” дополнительные уровни иерархии
- › Измерение “Регион” группирует магазины по регионам
- › Измерение “Категория товара” группирует товары по категориям
- › Измерение “Категория покупателей” группирует покупателям по категориям
- › Измерение “Период продаж” группирует продажи по временными периодам



Модель “Созвездие”

› Несколько таблиц фактов связи между собой какой-то логикой через атрибуты

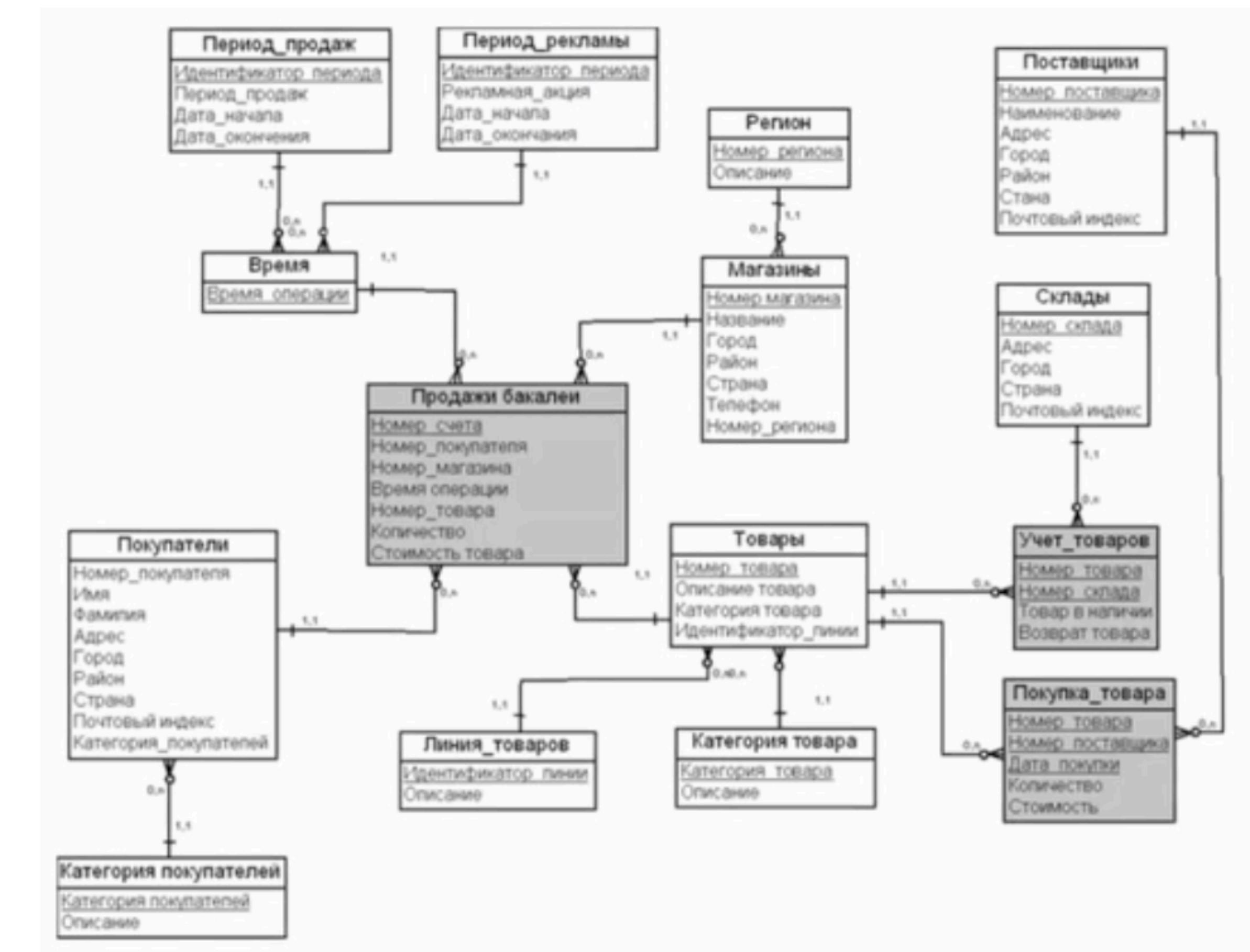


Таблица фактов

Таблица фактов — является основной таблицей хранилища данных. Как правило, она содержит сведения об объектах, событиях или процессах, совокупность которых будет в дальнейшем анализироваться.

Характеристики таблиц фактов:

- › Таблица фактов содержит числовые параметры (метрики),
- › Каждая таблица фактов имеет составной ключ, состоящий из первичных ключей таблиц измерений.
- › Первичный ключ таблицы измерений является внешним ключом в таблице фактов.

Виды фактов

- | **Аддитивные факты (Additive facts).** Факт называется аддитивным, если его имеет смысл использовать с любыми измерениями для выполнения операций суммирования с целью получения какого-либо значимого результата. Например, количество продаж, объем продаж и т.д.
- | **Полуаддитивные факты (Semiadditive facts).** Факт называется полуаддитивным, если его имеет смысл использовать совместно с некоторыми измерениями для выполнения операций суммирования с целью получения какого-либо значимого результата. Например, остаток на счете, уровень запасов на складе и т.д.;

Виды фактов

- | **Неаддитивные факты (Non-additive facts).** Факт называется неаддитивным, если его не имеет смысла использовать совместно с каким-либо измерением для выполнения операций суммирования с целью получения какого-либо значимого результата. Например, измерение комнатной температуры.
- | **Числовые меры интенсивности (Numerical Measures of Intensity).** Факт называется числовой мерой интенсивности, если он, являясь неаддитивным по времени, допускает агрегацию и суммирование по некоторому числу временных периодов. Например, остаток на счете.

Вопросы

Какие виды фактов есть в таблице ниже:

По измерению "Время":					
Дата	Товар	Магазин	Количество продаж	Количество покупателей	Суммарная прибыль
23.01.2009	CD диск	Компьютер	10	10	1500
24.01.2009	CD диск	Компьютер	35	30	5250
25.01.2009	CD диск	Компьютер	20	15	3000
			65	55	9750

По измерению "Товар":					
Дата	Товар	Магазин	Количество продаж	Количество покупателей	Суммарная прибыль
23.01.2009	CD диск	Компьютер	10	6	1500
23.01.2009	Принтер	Компьютер	1	1	5000
23.01.2009	Сканер	Компьютер	2	2	3000
			13		9500

По измерению "Магазин":					
Дата	Товар	Магазин	Количество продаж	Количество покупателей	Суммарная прибыль
23.01.2009	CD диск	Компьютер	10	10	1500
23.01.2009	CD диск	Принтеры	10	5	1500
23.01.2009	CD диск	Оргтехника	20	7	3000
			40	22	6000

Вопросы

Какие виды фактов есть в таблице ниже:

Суммирование метрики "Количество покупателей" по измерению "Товары":					
Дата	Товар	Магазин	Количество продаж	Количество покупателей	Суммарная прибыль
23.01.2009	Бумага для факсов	Компьютер	10	6	1000
23.01.2009	Бумага для принтера	Компьютер	12	7	1320
				13	

Свойства таблиц фактов

Свойства данных в таблицах фактов:

- › Числовые параметры используются для агрегации и суммирования;
- › Значения данных должны обладать свойствами аддитивности или полуаддитивности и по отношению к измерениям, для того чтобы их можно было суммировать;
- › Все данные таблицы фактов должны быть однозначно идентифицированы через ключи таблиц измерений, чтобы обеспечить доступ к ним через таблицы измерений;

Таким образом, таблицу фактов можно разделить на две части. Первая часть состоит из первичных ключей измерений, вторая — из числовых параметров функционально зависящих от ключей таблиц измерений.

Виды таблиц фактов

› Транзакционная таблица фактов (Transaction facts)

В такой таблице фактов сохраняют факты, которые фиксируют определенные события (транзакции). Это факты, описывающие каждое событие бизнеса. Например, продажи товара

› Таблица фактов периодических моментальных снимков (Snapshot)

В такой таблице собирают факты, фиксирующие текущее состояние определенного направления бизнеса. Это факты, которые описывают текущее состояние определенного направления бизнеса для любой комбинации значений измерений за данный период времени. Например, продажи организации на определенную дату (ежедневно)

› Таблица фактов кумулятивных моментальных снимков (Accumulated Snapshot)

В такой таблице собирают факты, фиксирующие некоторое итоговое состояние определенного направления бизнеса на текущий момент времени. Это факты, которые описывают промежуточные итоги деятельности организации по определенному направлению бизнеса для любой комбинации значений измерений за данный период времени. Например, продажи этого года на определенную дату

Виды таблиц фактов

	Транзакционная таблица фактов	Таблица фактов периодических моментальных снимков	Таблица фактов кумулятивных моментальных снимков
Определение гранулярности таблицы фактов	Одна строка на одну операцию	Одна строка на период	Одна строка для периода завершенного события
Измерения	Факты на самом низком уровне детализации “дата/ время”	Факты на конец периода измерения “дата/время”	Факты по нескольким измерениям “дата/время” для фиксации результатов в контрольных точках
Факты	Факты связаны с операционной деятельностью	Факты связаны с периодической деятельностью	Факты связаны с деятельностью, которая имеет определенное время существования
Обновления	Не допускаются	Не допускаются	Допускаются
Кардинальность таблицы фактов	Растет быстро	Растет медленнее, чем в транзакционных таблицах	Растет быстрее, чем в таблицах фактов периодических снимков

Таблицы измерений

Таблица измерений (англ. dimension table) — таблица в структуре многомерной базы данных, которая содержит атрибуты событий, сохраненных в таблице фактов. Атрибуты представляют собой текстовые или иные описания, логически объединенные в одно целое. Таблица измерения имеет первичный ключ и атрибуты, описывающие факты с точки зрения некоторого направления деятельности организации.

Характеристики измерений:

- › Таблицы измерений содержат данные о детализации фактов;
- › Таблицы измерений содержат описательную информацию о числовых значениях в таблице фактов, т.е. они содержат атрибуты фактов;
- › Как правило, таблицы измерений содержат большое количество полей;
- › Таблицы измерений содержат обычно значительно меньше строк, чем таблицы фактов;
- › Атрибуты таблиц измерений обычно используются при визуализации данных в отчетах и запросах;

Виды измерений

› **Медленно меняющимися измерениями (Slowly Changing Dimensions)** называются таблицы измерений, в которых некоторые атрибуты могут изменить свои значения по истечении некоторого периода времени, причем частота таких изменений является небольшой.

Всего существует 8 основных типов SCD, которые определяют, как история изменений может быть отражена в модели.

› **Быстро меняющимися измерениями (Rapidly Changing Dimensions)** называются таблицы измерений, в которых некоторые атрибуты могут часто менять свои значения в короткий период времени.

Виды SCD

SCD Type	Dimension Table Action	Impact on Fact Analysis
Type 0	No change to attribute value	Facts associated with attribute's original value
Type 1	Overwrite attribute value	Facts associated with attribute's current value
Type 2	Add new dimension row for profile with new attribute value	Facts associated with attribute value in effect when fact occurred
Type 3	Add new column to preserve attribute's current and prior values	Facts associated with both current and prior attribute alternative values
Type 4	Add mini-dimension table containing rapidly changing attributes	Facts associated with rapidly changing attributes in effect when fact occurred
Type 5	Add type 4 mini-dimension, along with overwritten type 1 mini-dimension key in base dimension	Facts associated with rapidly changing attributes in effect when fact occurred, plus current rapidly changing attribute values
Type 6	Add type 1 overwritten attributes to type 2 dimension row, and overwrite all prior dimension rows	Facts associated with attribute value in effect when fact occurred, plus current values
Type 7	Add type 2 dimension row with new attribute value, plus view limited to current rows and/or attribute values	Facts associated with attribute value in effect when fact occurred, plus current values

5 - Data Vault

Data Vault

Data Vault — набор уникально связанных нормализованных таблиц, содержащих детальные данные, отслеживающих историю изменений и предназначенных для поддержки одной или нескольких функциональных областей бизнеса.

Автор: **Дэн Линстедт** (Dan E. Linstedt)

Дизайн Data Vault — гибок, масштабируем, последователен и приспособливаем к потребностям предприятия. Это — модель данных, спроектированная специально для удовлетворения потребностей хранилищ данных предприятия.

В модели Data Vault содержаться структуры, которые знакомые и соответствуют традиционным определениям схемы звезда и ЗНФ, включая: измерения, связи многие ко многим и стандартные табличные структуры. Различия заключаются в представлении отношений, структурировании полей и хранении детальных данных, основанных на времени.

Data Vault

Для того чтобы сохранить дизайн простым и изящным, используется минимальное количество типов сущностей: **Хаб** (Hub), **Связь** (Link) и **Сателлит** (Satellite).

Дизайн Data Vault сосредоточен вокруг функциональных областей бизнеса:

- › **Хаб** (Hub) хранит сущности
- › **Связь** (Link) обеспечивает транзакционную интеграцию между Хабами (связи между сущностями)
- › **Сателлит** (Satellite) предоставляет контекст первичного ключа Хаба (атрибуты, описания)

Каждая сущность предназначена для обеспечения максимальной гибкости и масштабируемости, сохраняя при этом большинство традиционных навыков моделирования данных.

Hub

Хабы (Hub) являются отдельными таблицами, содержащими как минимум уникальный список бизнес ключей.

Атрибуты Хаба включают:

- › Ключ бизнес-сущности из внешней системы
- › Суррогатный ключ (Surrogate Key)
- › Временная отметка даты загрузки (Load Date Time Stamp), регистрирующая, когда ключ впервые был загружен в хранилище
- › Источник данных (Record Source) - регистрация исходной системы, используется для обратной трассировки (отслеживания) данных

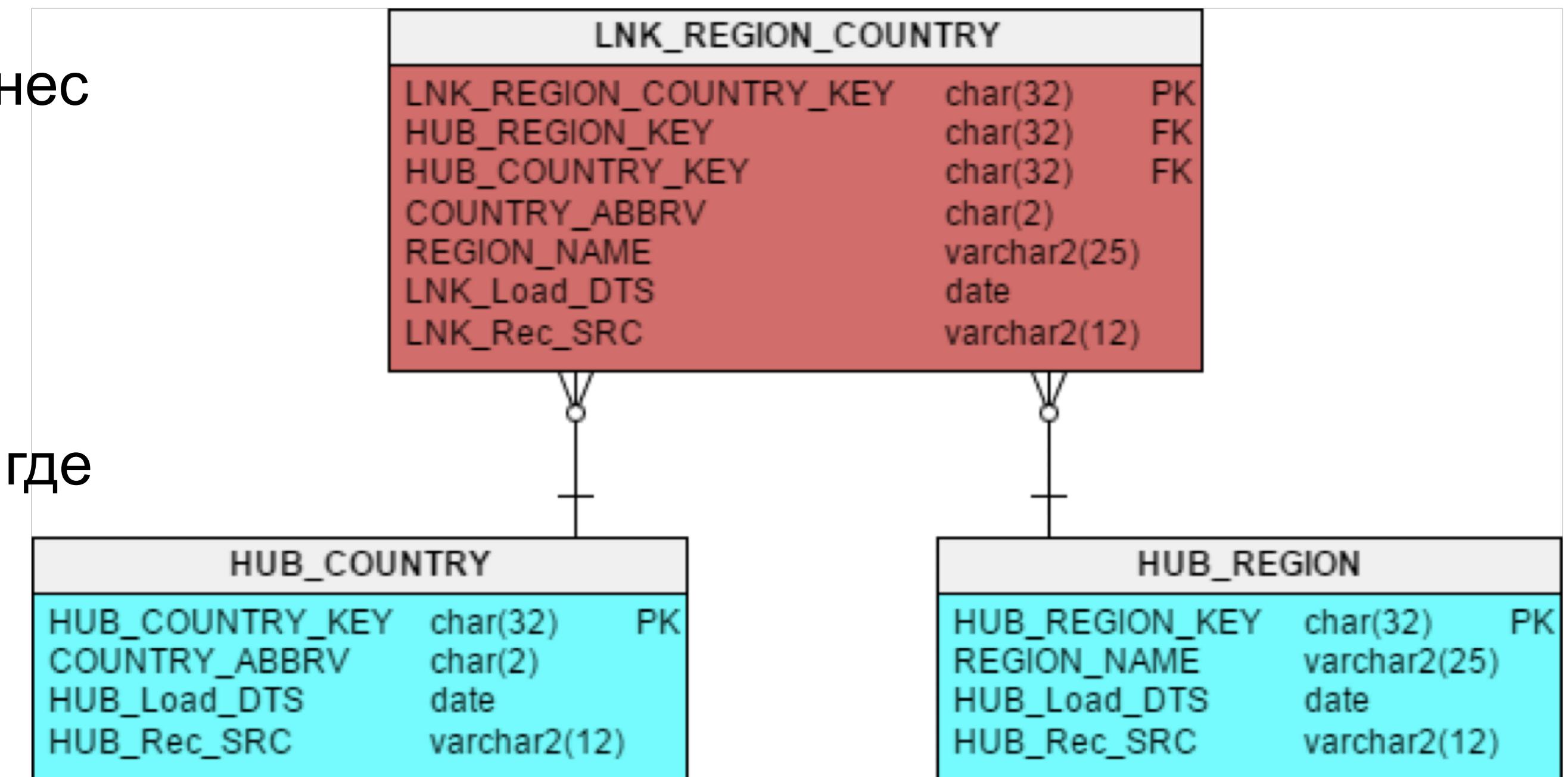
HUB_LOCATION		
HUB_LOCATION_KEY	char(32)	PK
LOCATION_NAME	varchar2(50)	
HUB_Load_DTS	date	
HUB_Rec_SRC	varchar2(12)	

Link

Связи (Link) представляют собой отношения или транзакцию между двумя или более компонентами бизнеса (два или более бизнес ключа).

Атрибуты линка включают:

- › Суррогатный ключ (Surrogate Key)
- › Ключи Хабов: от 2-ух Хабов до N Хабов, где N – количество входящих в Link Хабов
- › Временная отметка даты загрузки
- › Код источника данных

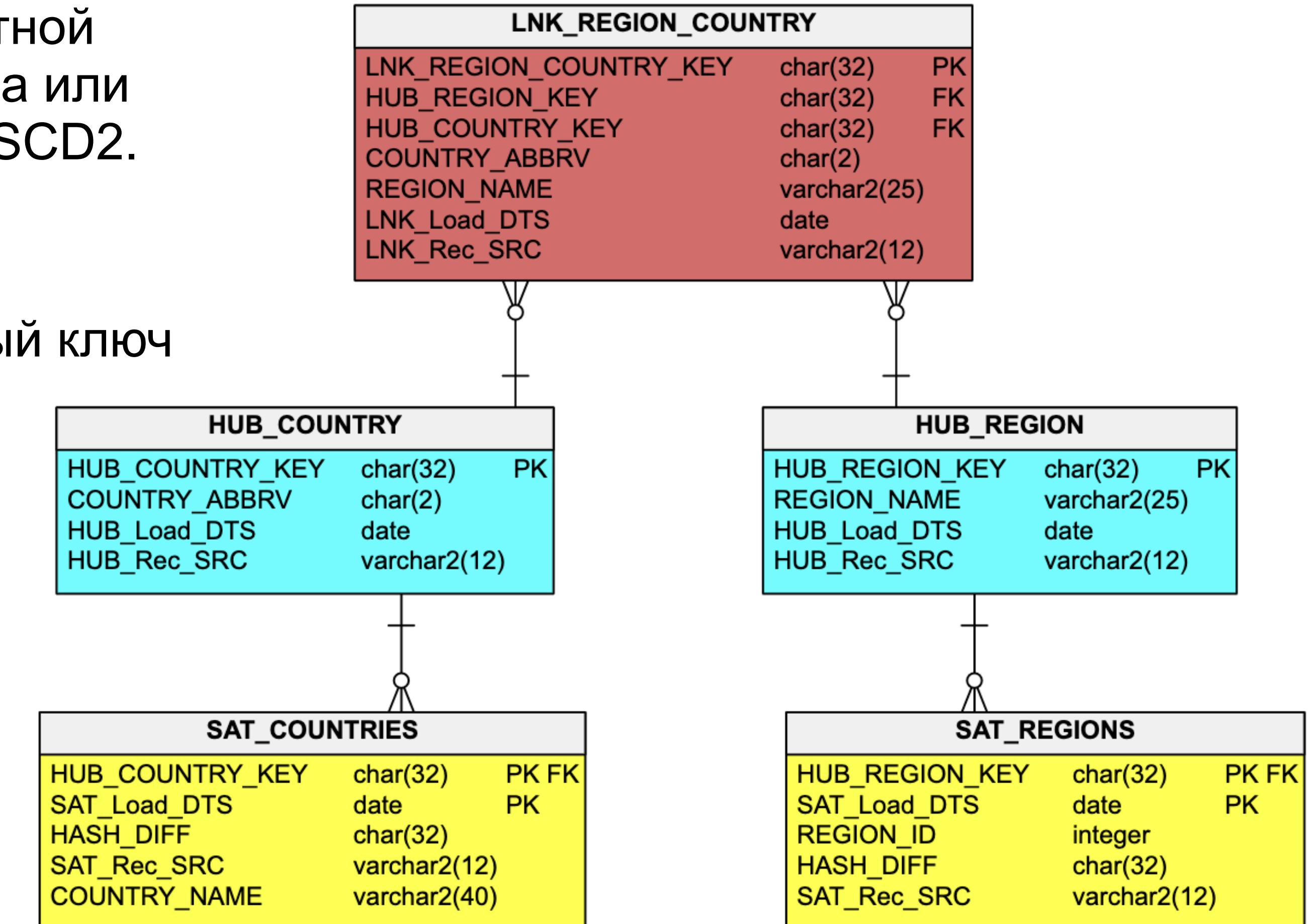


Satellite

Сателлиты (Satellite) являются контекстной (описательной) информацией ключа Хаба или Линка (Связи), обычно с историзмом по SCD2.

Атрибуты сателлита включают:

- › Первичный ключ Сателлита: Первичный ключ Хаба или первичный ключ Связи
- › Даты действия записи (SCD2)
- › Временная отметка даты загрузки
- › Код источника данных



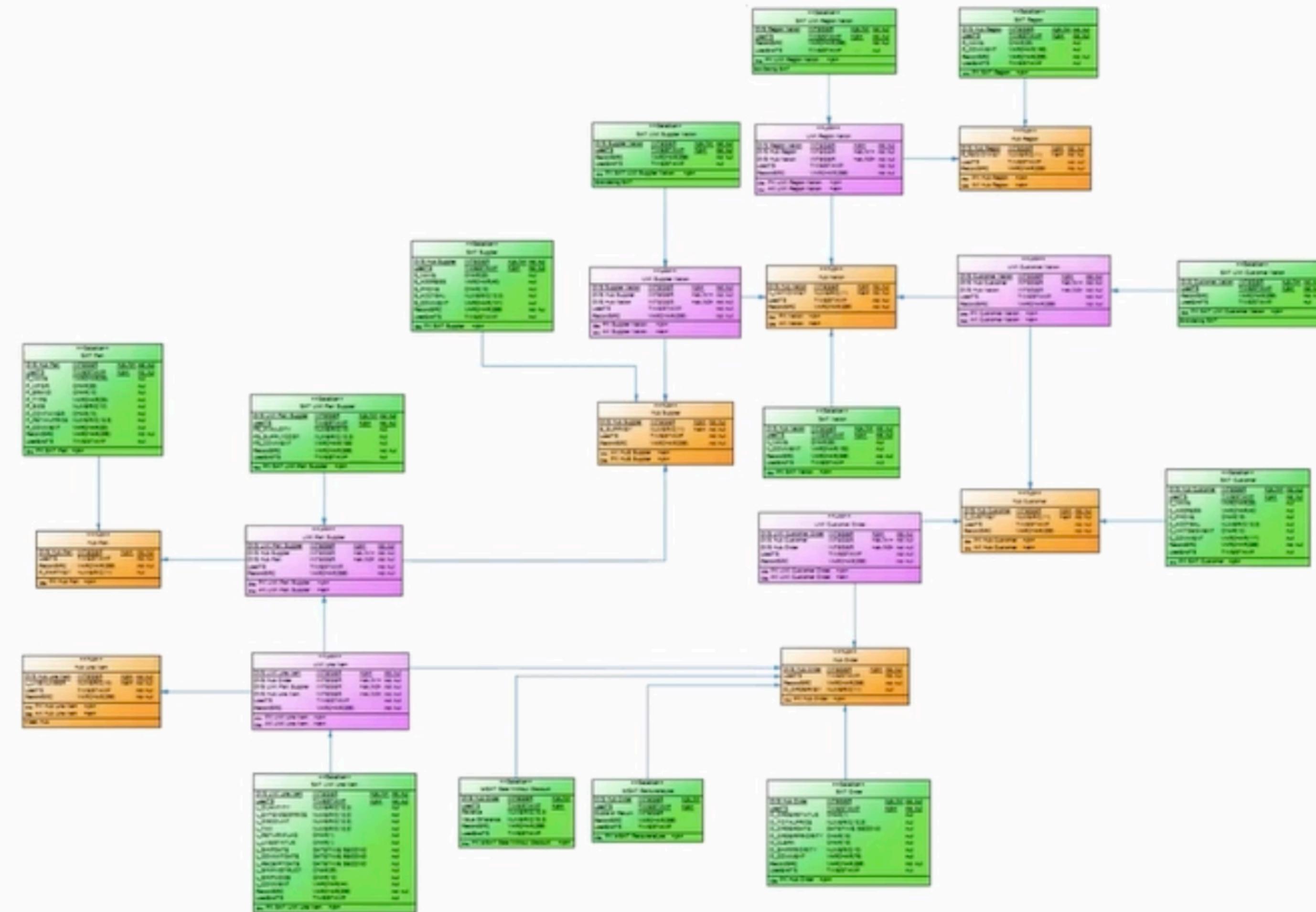
Data Vault

Data vault 1.0:

- › Хаб (Hub),
- › Связь (Link)
- › Сателлит (Satellite)

Data Vault 2.0:

- › Peg-legged link
- › Pit – Point in time
- › Bridge



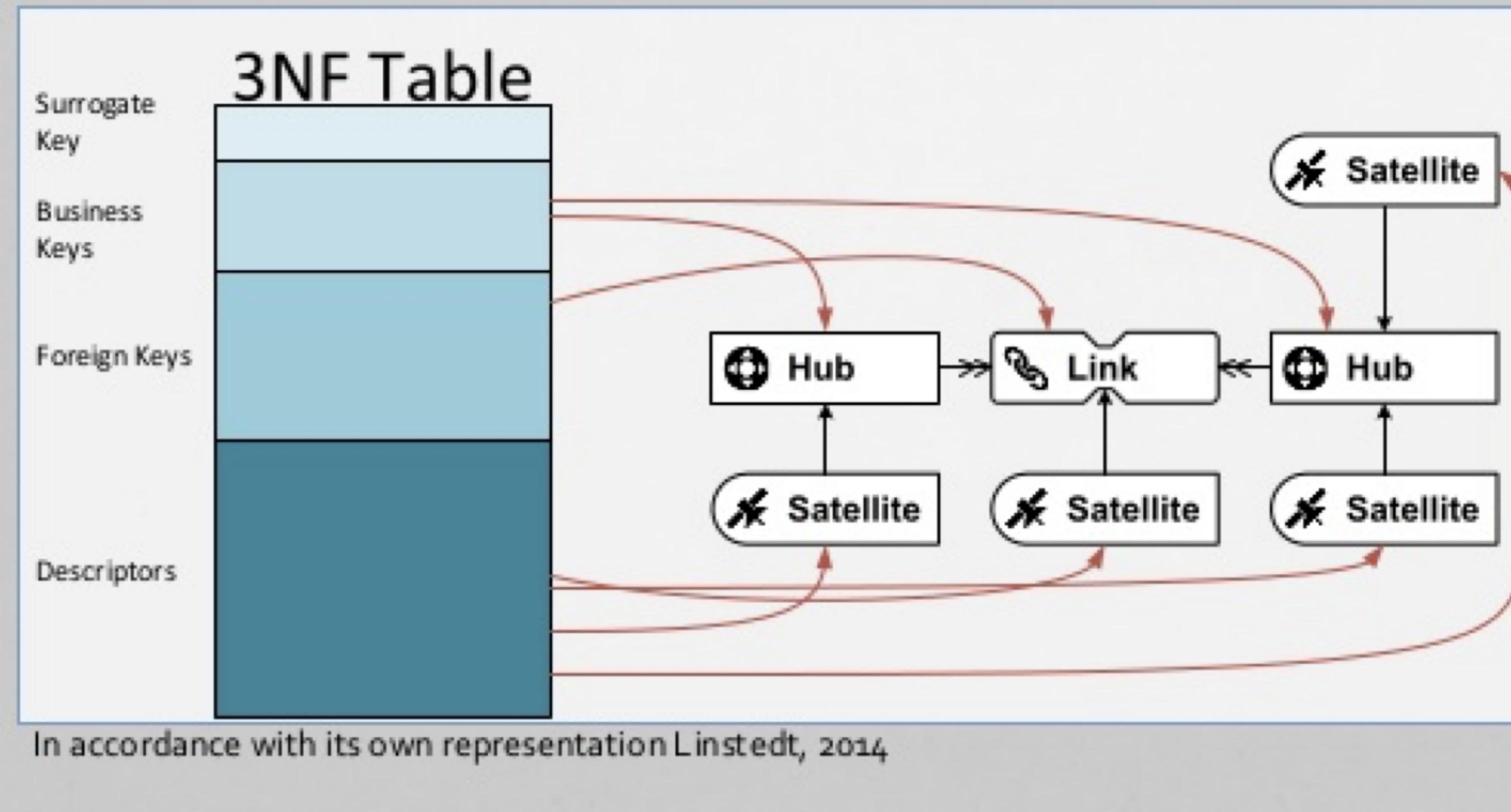
Проектирование Data Vault

Загрузка происходит следующим образом:

- › Из источников данные собираются в **стейджинг**
- › Стейджинг прогружается в **хабы**
- › Из хабов генерируются **суррогатные ключи**
- › Имея суррогатные ключи, можно генерировать **сателлиты и линки**

Проектирование Data Vault

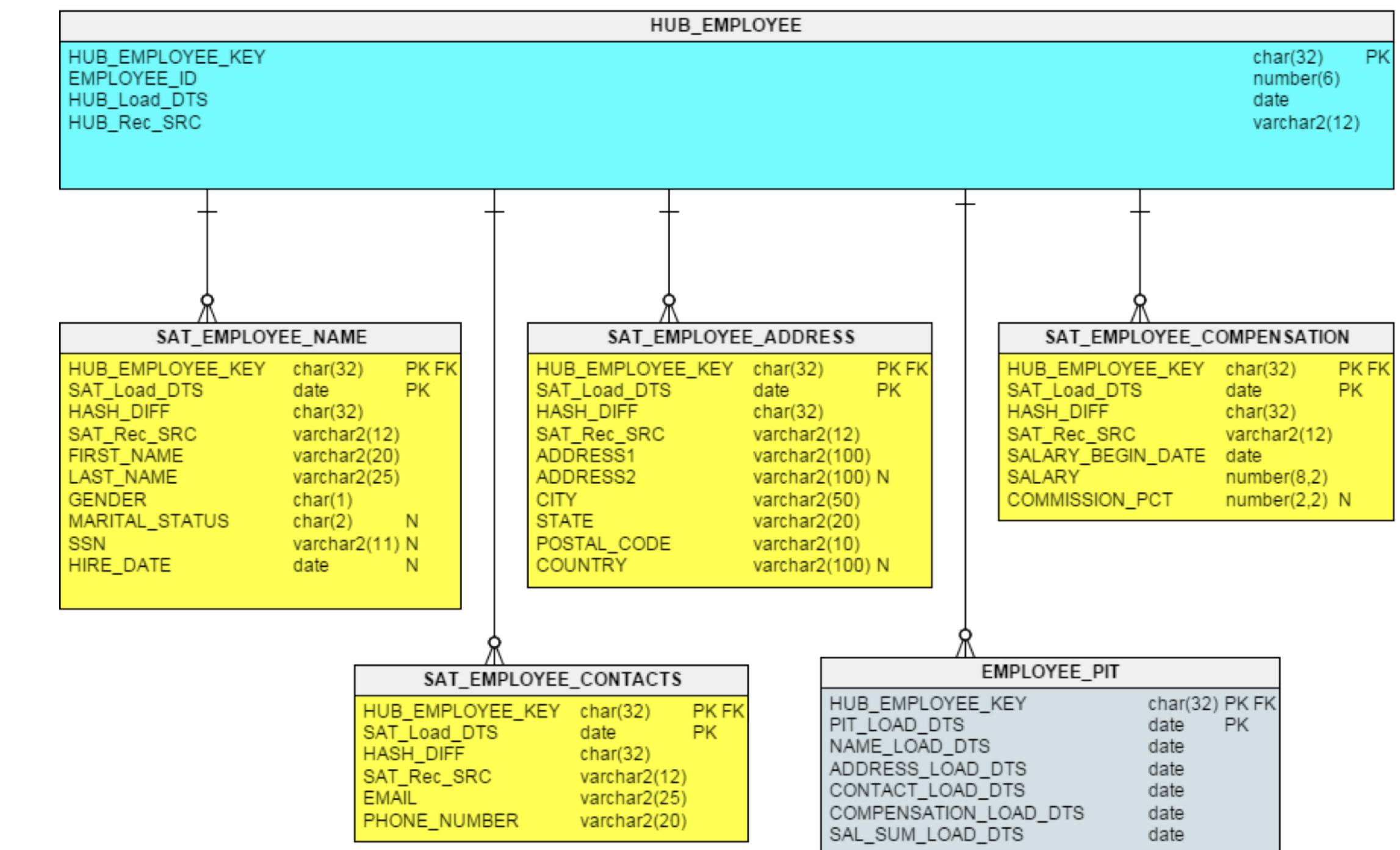
DATA VAULT 2.0 MODELING



DÖRFFLER
PARTNER

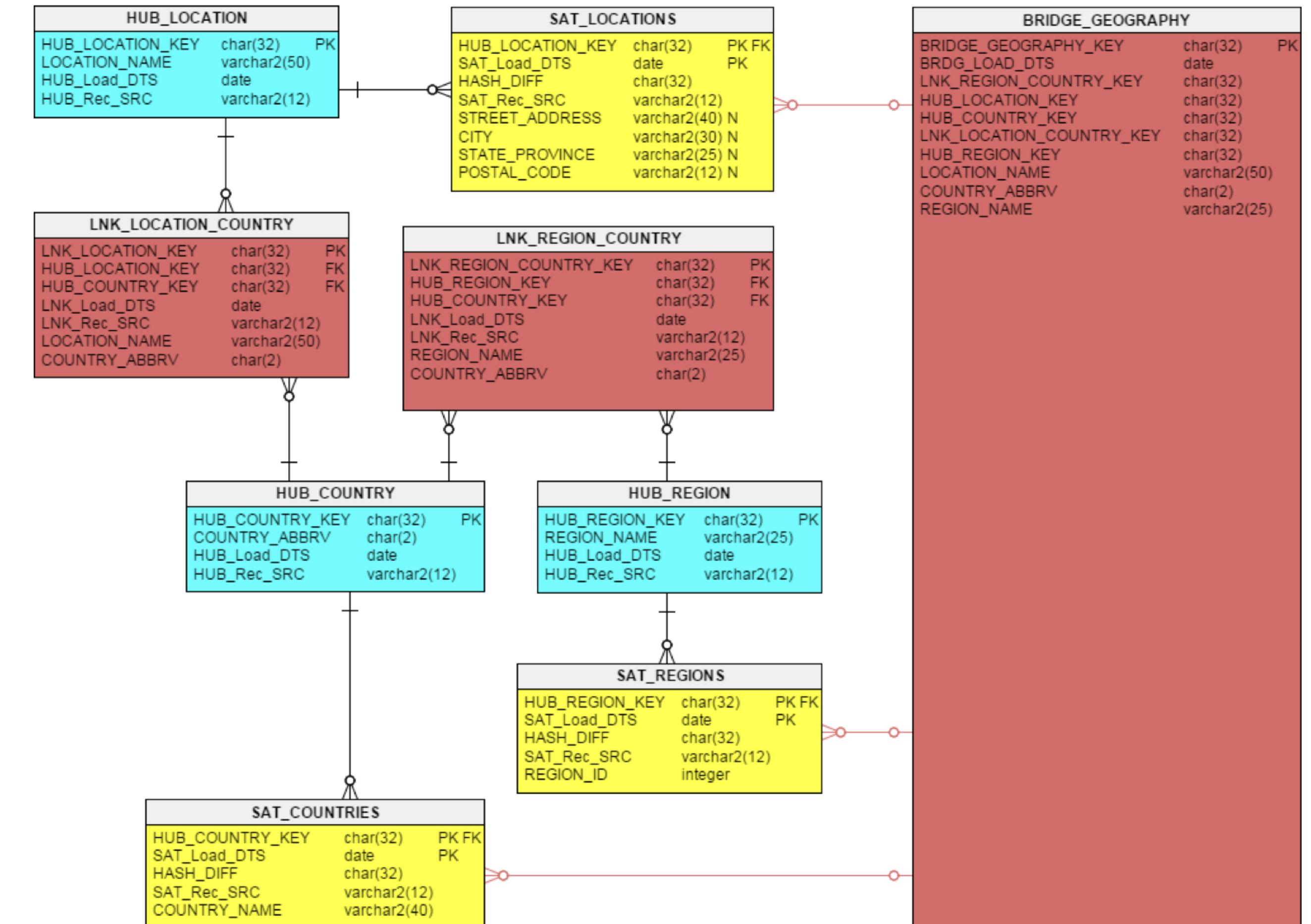
DataVault 2.0 - РiТ

- › РiТ (point in time) упрощает получение информации из Сателлитов одной сущности с разной частотой обновления.
- › Так как частота изменений каждого отдельного Сателлита может быть разной, не все диапазоны SCD2 будут пересекаться. В связи с этим могут возникнуть трудности с восстановлением состояния записи на некоторый момент времени.
- › В РiТ-таблице добавляются записи на каждое изменение каждого Сателлита. Фактически, это позволяет с помощью join'ов Хаба на РiТ и Сателлиты воссоздать всю сущность на необходимый момент времени.



DataVault 2.0 - Bridge

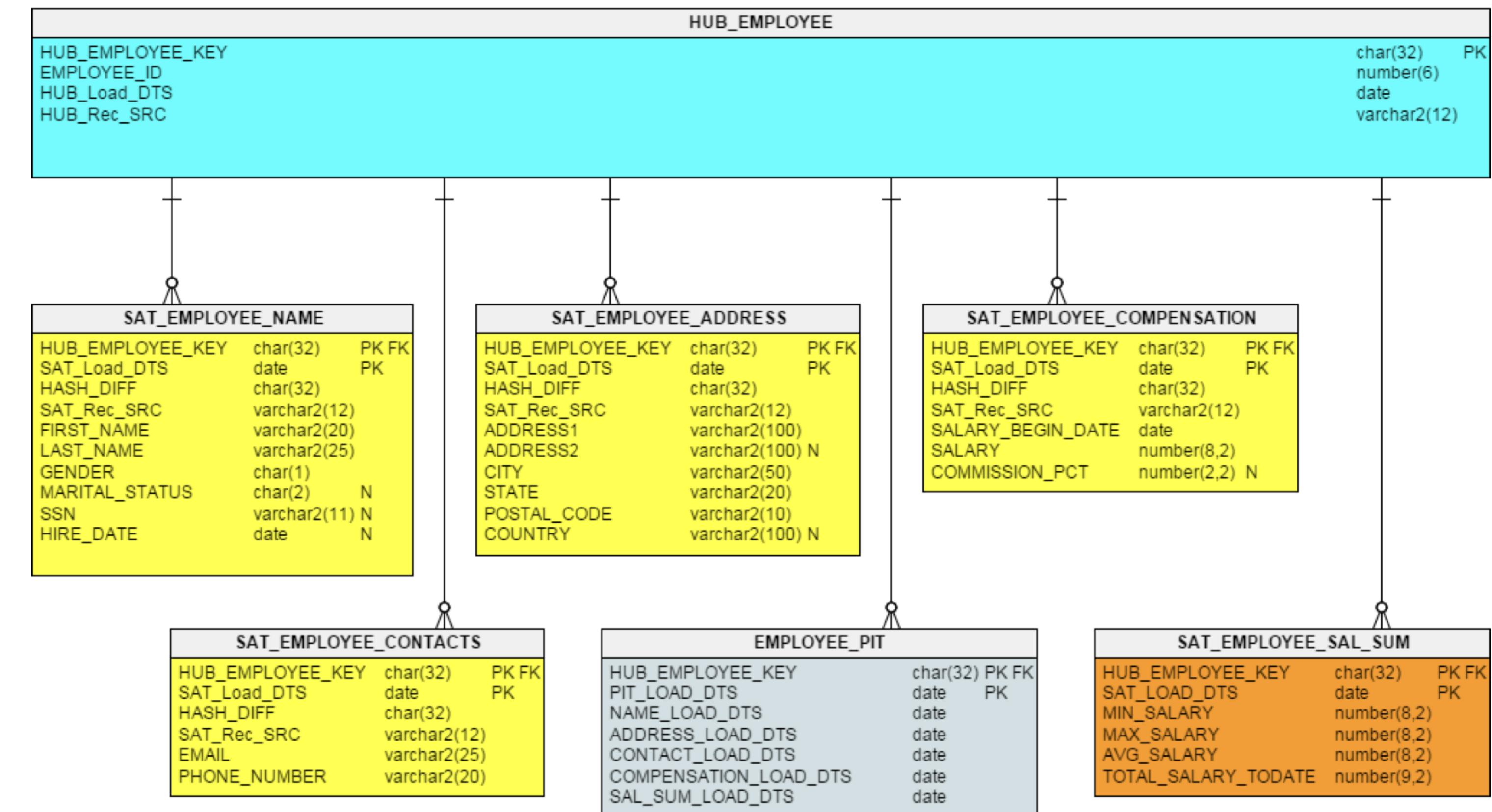
- › Мост (Bridge) упрощает соединение данных через несколько связей.
- › При наличии нескольких Линков, соединение всех таблиц в некоторую сущность может быть затруднено. Bridge упрощает это, так как фактически содержит в себе все Линки.



DataVault 2.0 - Predefined Derivations

› **Predefined Derivations** (предагрегаты) представляет собой готовые, предагрегированные (или предкалькулированные) данные, такие как дата первой поездки или минимальное время нахождения на сервисе.

› Эти данные появляются на следующем этапе после загрузки Сателлитов.



Почему это модно?



6 - Anchor Modeling

Anchor Modeling

Автор: Ларс Рёнбэк (Lars Rönnbäck)

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

В методике моделирования используются четыре модели моделирования: **якорь атрибут, связь и узел**, каждый из которых отражает различные аспекты моделируемого домена. Полученные модели могут быть переведены в физические проекты баз данных с использованием формализованных правил. Когда такой перевод сделан, таблицы в реляционной базе данных будут в основном в шестой нормальной форме.

Anchor Modeling

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

Anchor таблица должна хранить только суррогатный ключ и несколько технических полей (система-источник, дата-время загрузки).

Пример - товар, пользователь, платеж

Attribute (Атрибут) — это таблица для хранения свойства, атрибута объекта

Пример - название товара, логин и дата рождения пользователя, суммы платежа.

Одно свойство у объекта — одна Attribute-таблица.

Каждая Attribute-таблица содержит суррогатный ключ объекта, которым является ссылка на соответствующий Anchor, поле для значения атрибута, и, опционально, дату для историчности и технические поля.

Anchor Modeling

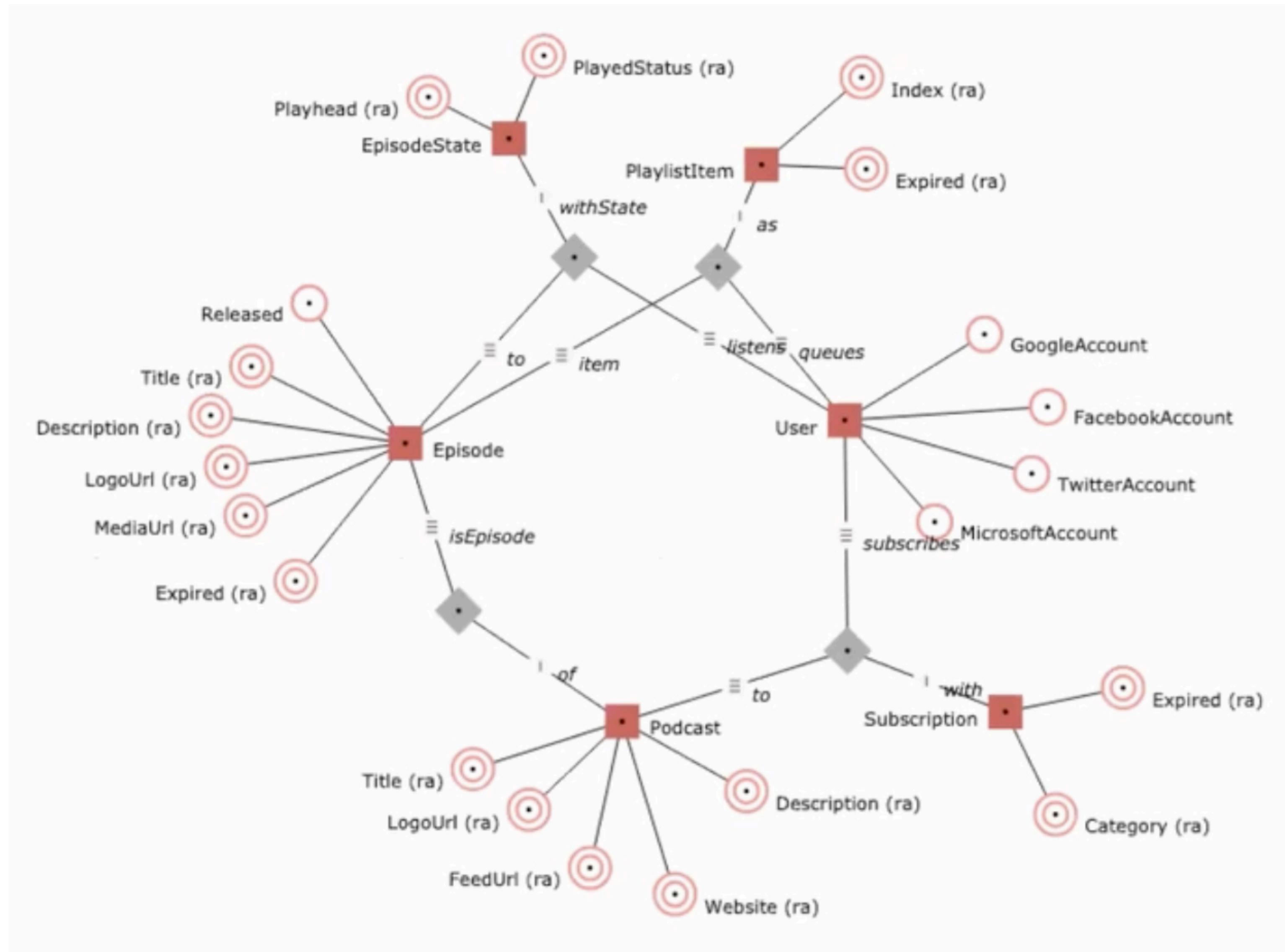
Tie (Связь) — это таблица для хранения связей между объектами.

Таблица должна содержать суррогатный ключ левого объекта (customer_id), правого объекта (country_id) и, по необходимости, даты историчности и технических полей.

Knot (Узел) - таблица-состояние, справочник. Узлы можно рассматривать как сочетание якоря и одного атрибута.

Пример – справочник, в котором содержатся поля.

Как это выглядит?



Проектирование

- › Найдите основные бизнес-сущности (это якоря)
- › Исследуйте их (найдите другие якоря, связи)
- › Опишите их (атрибуты, узлы)
- › Историзируйте их, если нужно (атрибуты)
- › Определите взаимоотношения сущностей (связи)
- › Историзируйте взаимоотношения (связи)

Технические нюансы реализации Anchor modeling

- › Якорная модель крайне чувствительна к техническим возможностям СУБД
- › ОГРОМНОЕ количество join
- › Table (join) elimination
- › Cluster index
- › Insert Only
- › Необходима автоматизация

Что выбрать?

сложность эксплуатации, простота внесения изменений



Никакого	Звезда и снежинка	Data Vault	Anchor modeling
› Денормализация	› Нормализация	› Строгая нормализация	› Ультра 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	› Не надо перестраивать
› Дублирование информации	› Минимальное дублирование информации	› Нет дублирования информации	› Нет дублирования информации
› Нет join	› Приемлемое количество join	› Большое количество join	› Ультра количество join

легкость эксплуатации, сложность внесения изменений

Дilemma проектирования DWH

