



PostgreSQL

Преподаватель:
Валентин Степанов

Оптимизация запросов

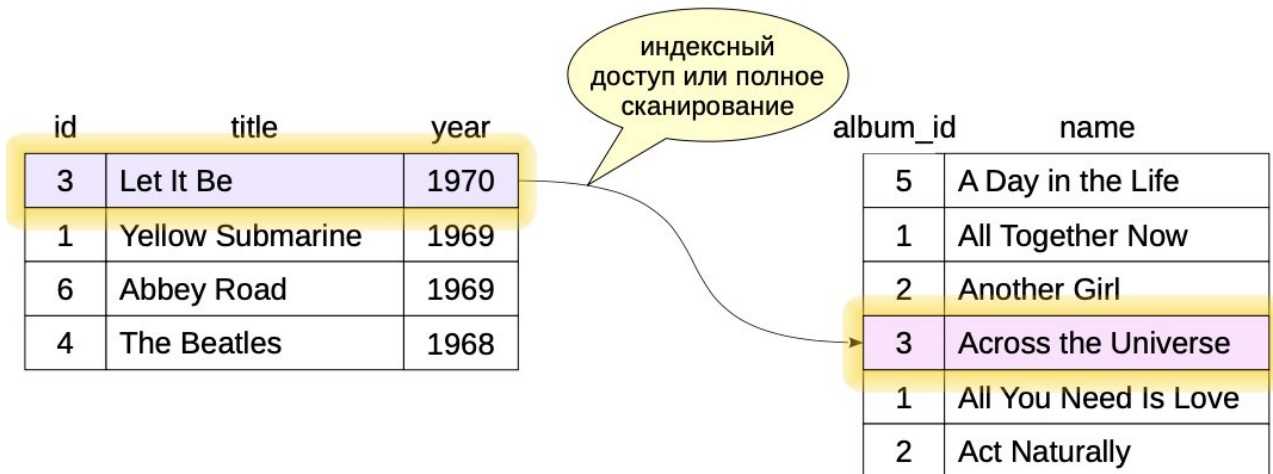
- Соединение вложенным циклом

Соединения

- Способы соединения — не соединения SQL
 - inner/left/right/full/cross join, in, exists — логические операции
 - способы соединения — механизм реализации
- Соединяются не таблицы, а наборы строк
 - могут быть получены от любого узла дерева плана
- Наборы строк соединяются попарно
 - порядок соединений важен с точки зрения производительности
 - обычно важен и порядок внутри пары

Nested Loop

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.a_id;
```



для каждой строки одного набора
перебираем подходящие строки другого набора

Nested Loop

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.a_id;
```

id	title	year
3	Let It Be	1970
1	Yellow Submarine	1969
6	Abbey Road	1969
4	The Beatles	1968

album_id	name
5	A Day in the Life
1	All Together Now
2	Another Girl
3	Across the Universe
1	All You Need Is Love
2	Act Naturally



Nested Loop

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.a_id;
```

id	title	year
3	Let It Be	1970
1	Yellow Submarine	1969
6	Abbey Road	1969
4	The Beatles	1968

album_id	name
5	A Day in the Life
1	All Together Now
2	Another Girl
3	Across the Universe
1	All You Need Is Love
2	Act Naturally



Nested Loop

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.a_id;
```

id	title	year
3	Let It Be	1970
1	Yellow Submarine	1969
6	Abbey Road	1969
4	The Beatles	1968

album_id	name
5	A Day in the Life
1	All Together Now
2	Another Girl
3	Across the Universe
1	All You Need Is Love
2	Act Naturally

Nested Loop

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.a_id;
```

id	title	year
3	Let It Be	1970
1	Yellow Submarine	1969
6	Abbey Road	1969
4	The Beatles	1968

album_id	name
5	A Day in the Life
1	All Together Now
2	Another Girl
3	Across the Universe
1	All You Need Is Love
2	Act Naturally

не все строки
внутреннего набора
рассматривались

Вычислительная сложность

$\sim N \times M$, где

- N — число строк во внешнем наборе данных,
 - M — среднее число строк внутреннего набора, приходящееся на одну итерацию
-
- Соединение эффективно только для небольшого числа строк

В параллельных планах

id	title	year
3	Let It Be	1970
1	Yellow Submarine	1969
6	Abbey Road	1969
4	The Beatles	1968

album_id	name
5	A Day in the Life
1	All Together Now
2	Another Girl
3	Across the Universe
1	All You Need Is Love
2	Act Naturally

внешний набор строк
сканируется параллельно,
внутренний — последовательно
каждым рабочим процессом

5	A	5	A
1	All Together Now	1	All Together Now
2	Another Girl	2	Another Girl
3	Across the Universe	3	Across the Universe
1	All You Need Is Love	1	All You Need Is Love
2	Act Naturally	2	Act Naturally

Оптимизация запросов

- Соединение хешированием

Однопроходное соединение

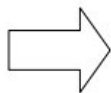
- Применяется, когда для хеш-таблицы достаточно оперативной памяти

Построение хеш-таблицы

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.album_id;
```

id	title	year
3	Let It Be	1970
1	Yellow Submarine	1969
6	Abbey Road	1969
4	The Beatles	1968

внутренний набор



	хеш-код	id	title
00	01000100	4	The Beatles
01	10100001	3	Let It Be
	10001001	6	Abbey Road
10			
11	11000111	1	Yellow Submarine

участвует
в условии
соединения

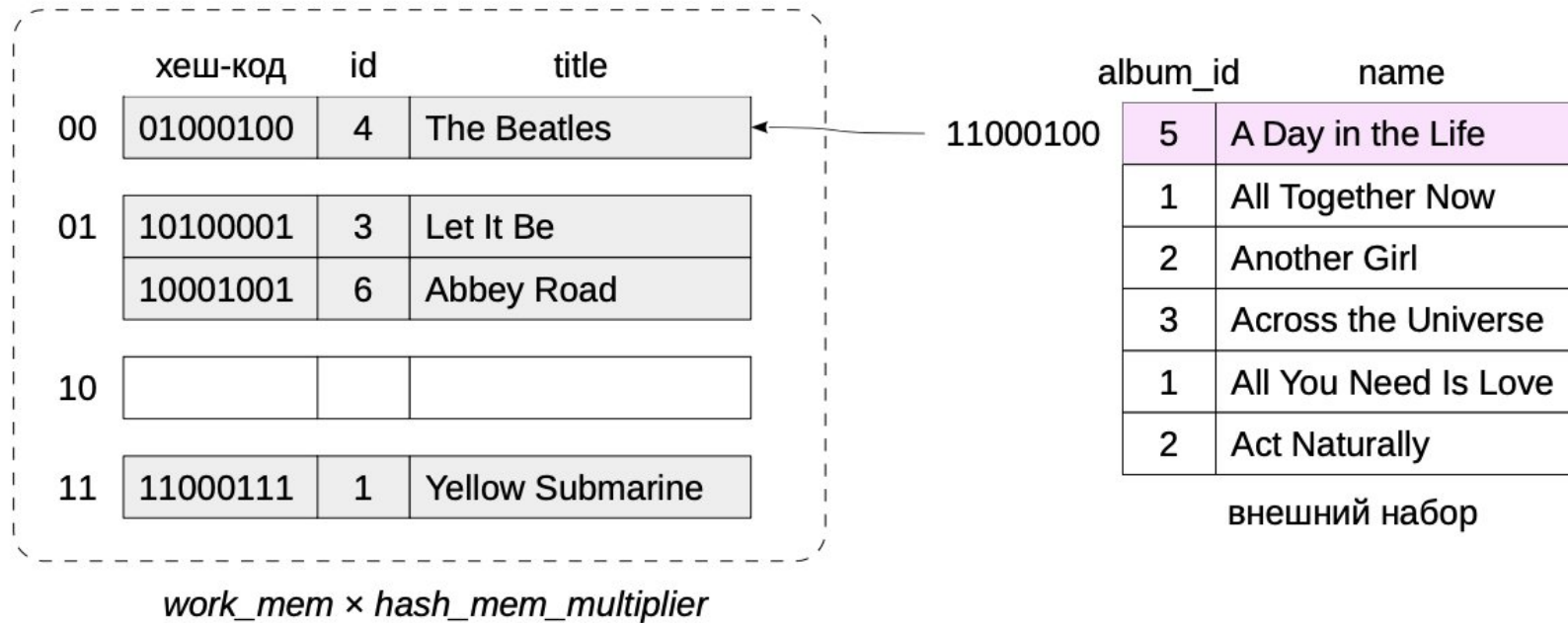
используется
в запросе

номер
корзины

$work_mem \times hash_mem_multiplier$

Сопоставление

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.album_id;
```



Сопоставление

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.album_id;
```

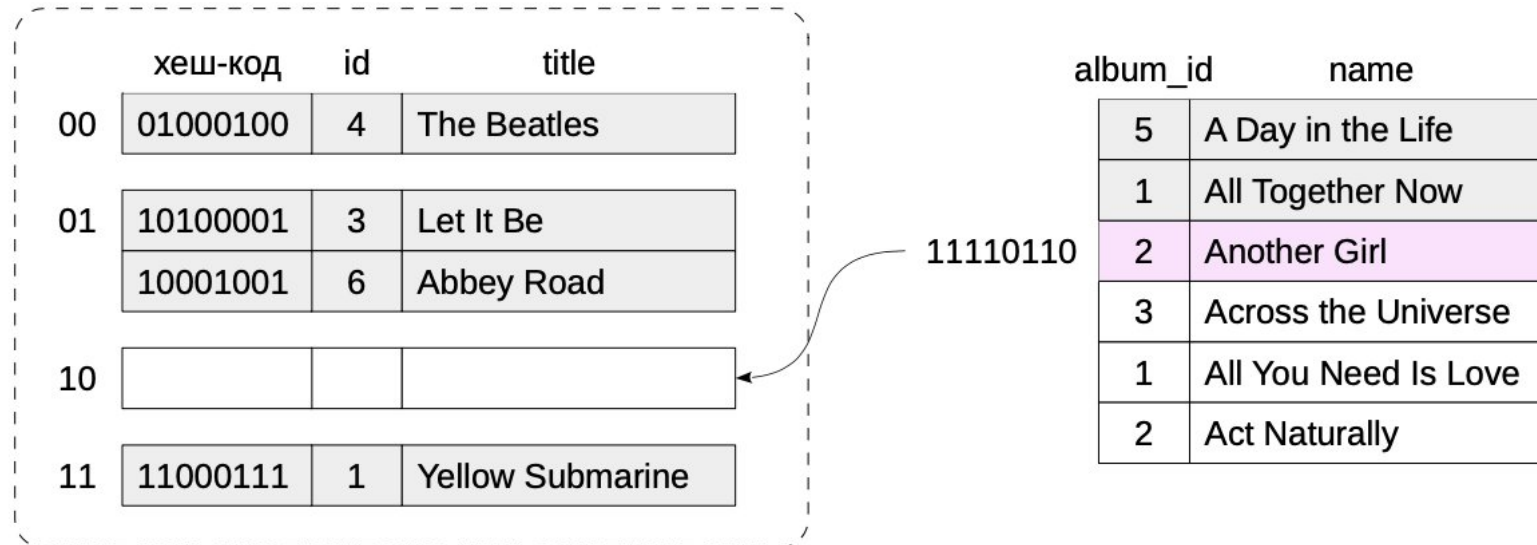
	хеш-код	id	title
00	01000100	4	The Beatles
01	10100001	3	Let It Be
	10001001	6	Abbey Road
10			
11	11000111	1	Yellow Submarine

11000111

album_id	name
5	A Day in the Life
1	All Together Now
2	Another Girl
3	Across the Universe
1	All You Need Is Love
2	Act Naturally

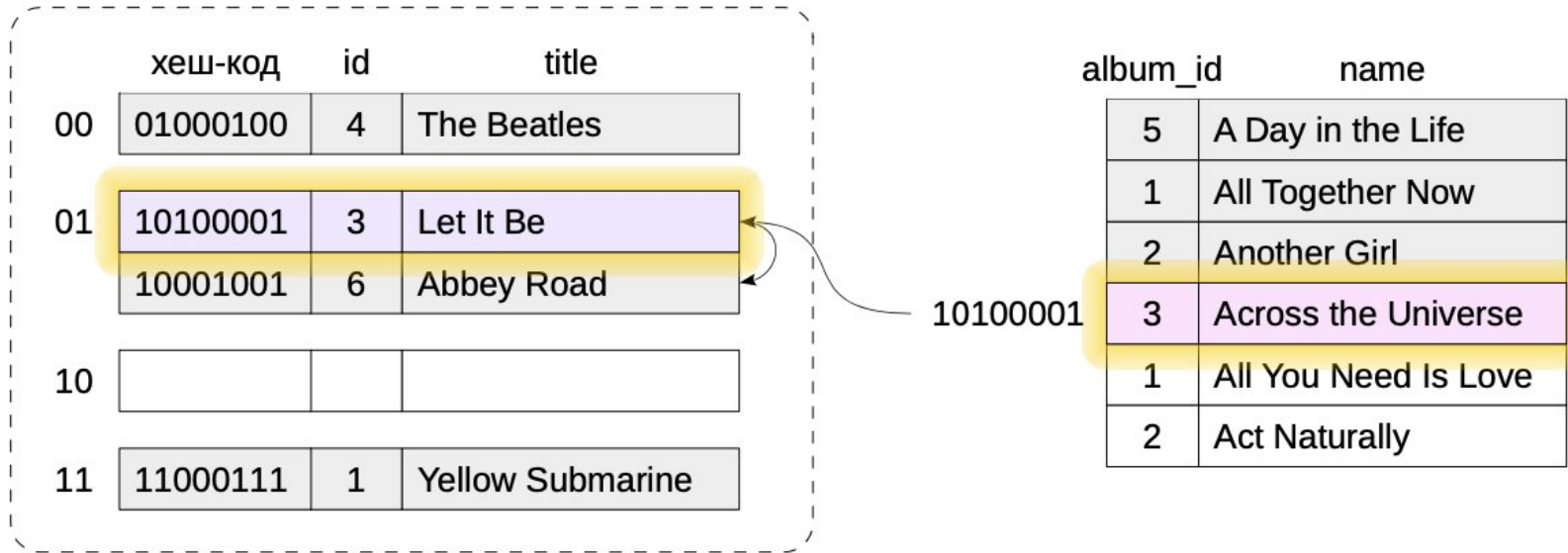
Сопоставление

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.album_id;
```



Сопоставление

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.album_id;
```



Сопоставление

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.album_id;
```

	хеш-код	id	title
00	01000100	4	The Beatles
01	10100001	3	Let It Be
	10001001	6	Abbey Road
10			
11	11000111	1	Yellow Submarine

11000111

album_id	name
5	A Day in the Life
1	All Together Now
2	Another Girl
3	Across the Universe
1	All You Need Is Love
2	Act Naturally

Сопоставление

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.album_id;
```

	хеш-код	id	title
00	01000100	4	The Beatles
01	10100001	3	Let It Be
	10001001	6	Abbey Road
10			
11	11000111	1	Yellow Submarine

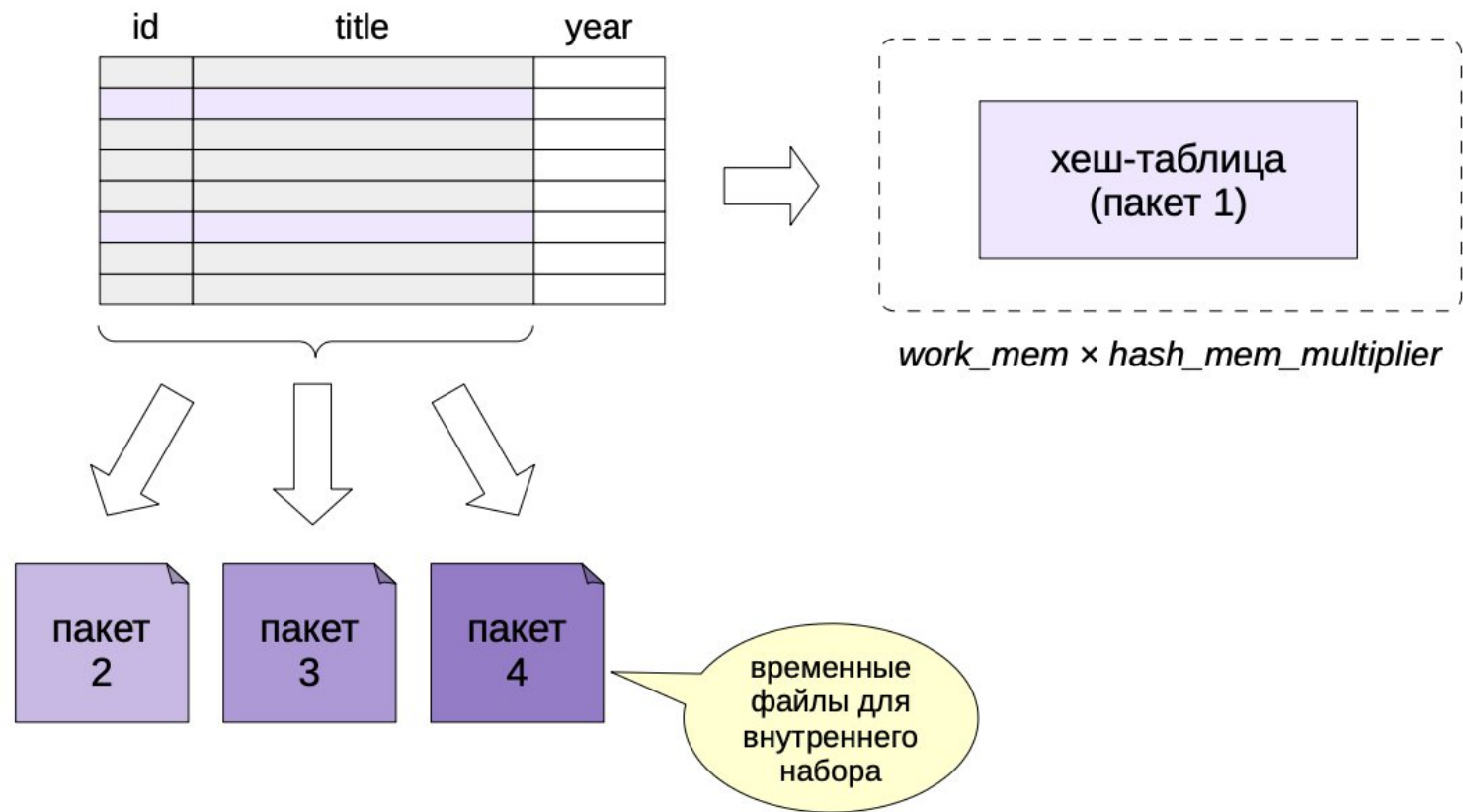
album_id	name
5	A Day in the Life
1	All Together Now
2	Another Girl
3	Across the Universe
1	All You Need Is Love
2	Act Naturally

11110110

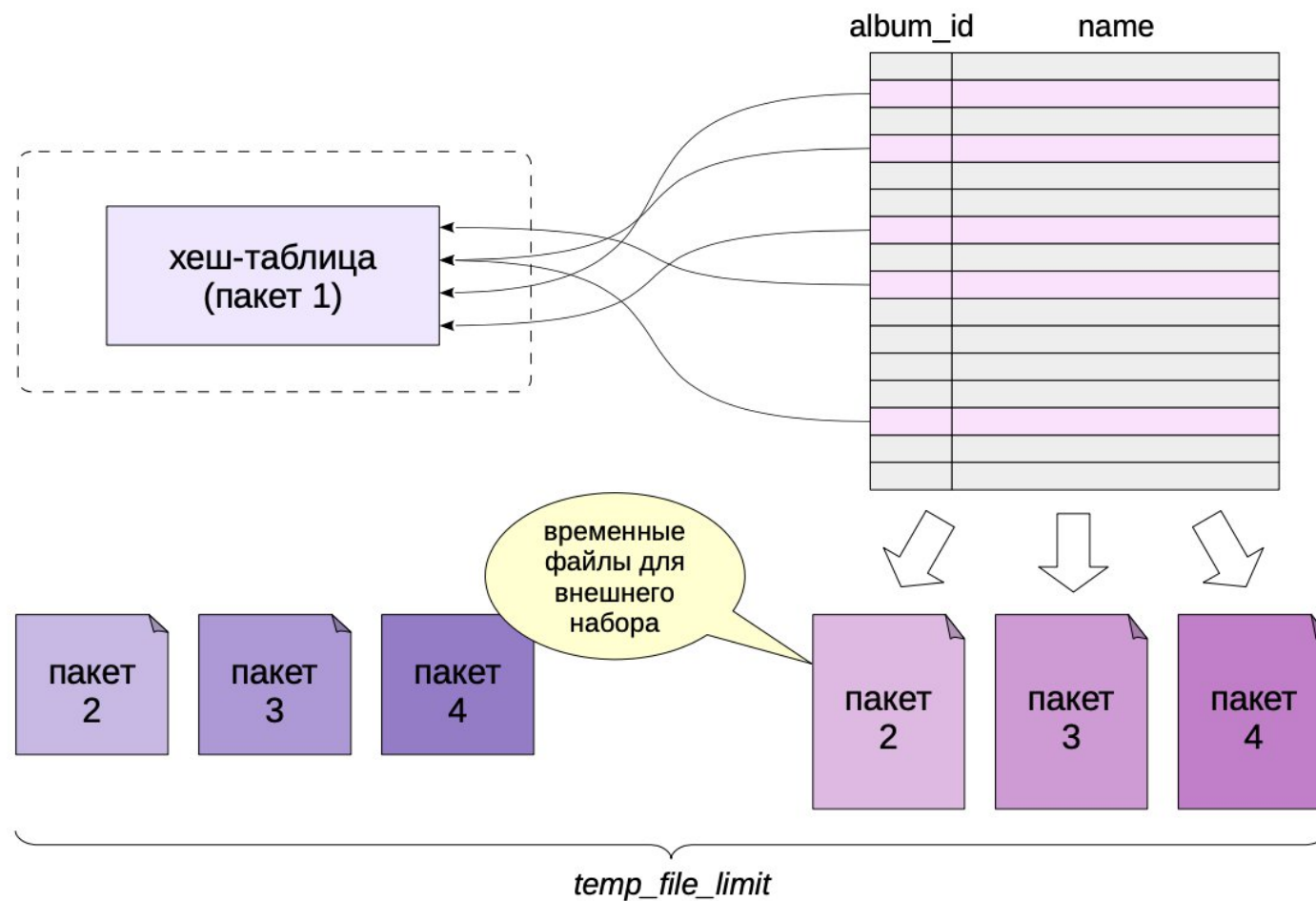
Двухпроходное соединение

- Применяется, когда хеш-таблица не помещается в оперативную память: наборы данных разбиваются на пакеты и последовательно соединяются

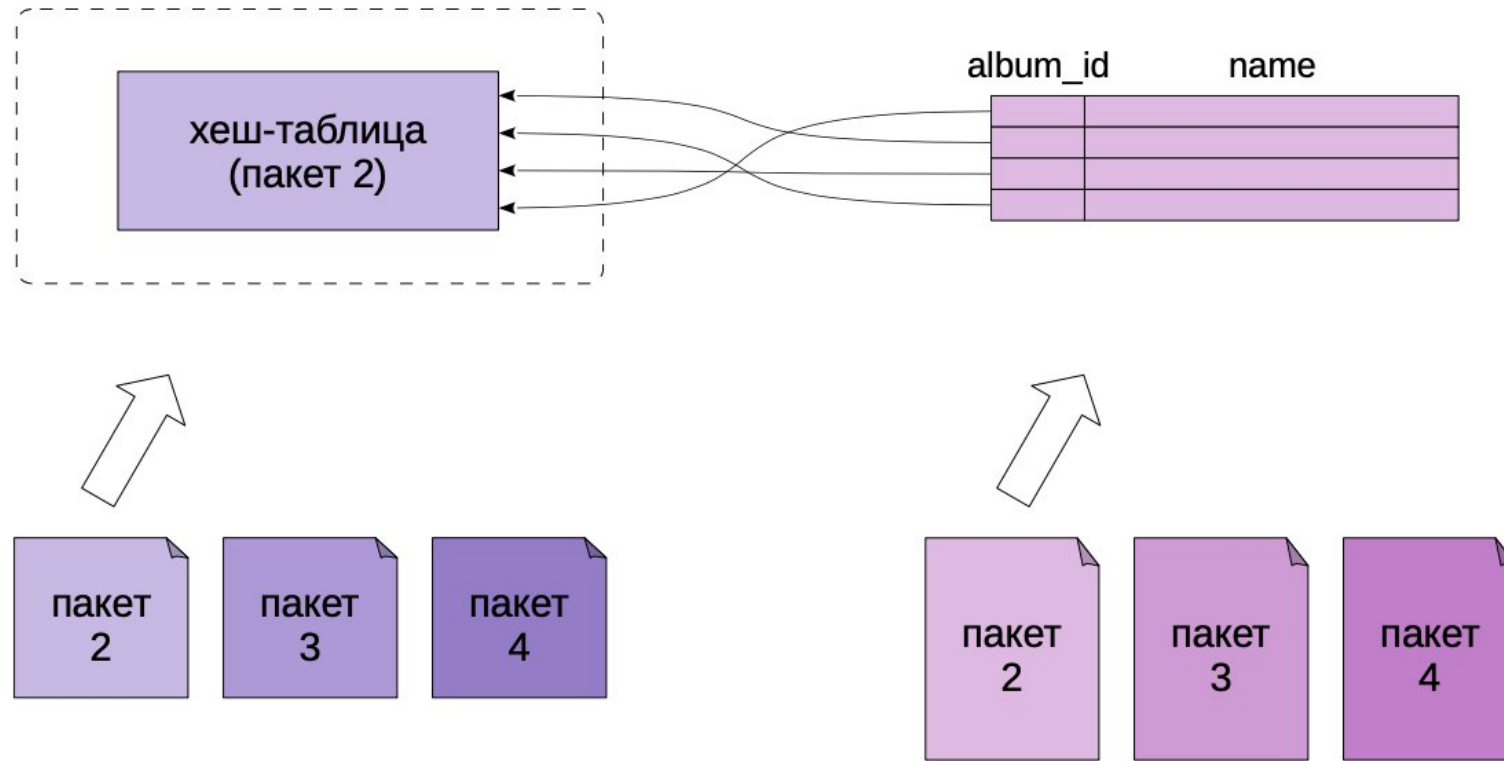
Построение хеш-таблицы



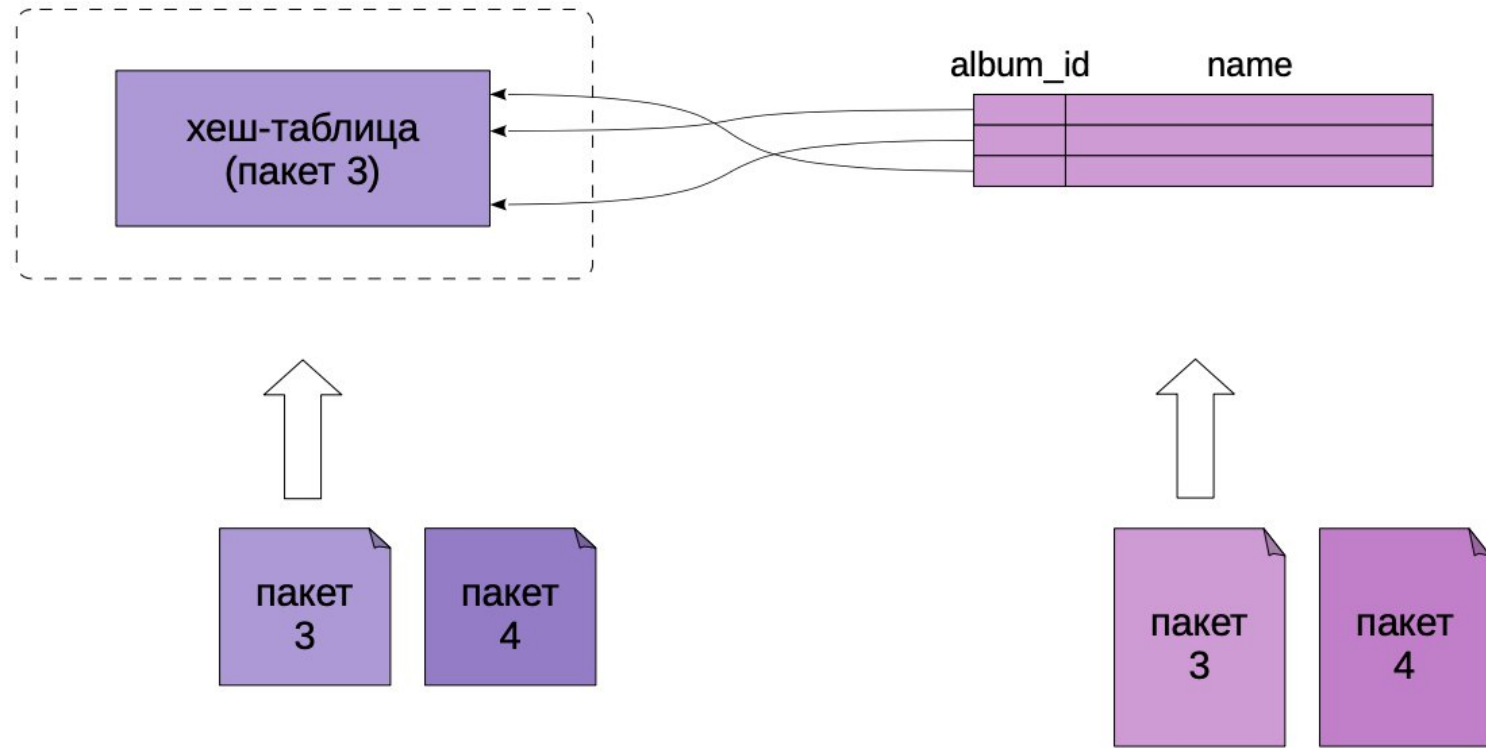
Сопоставление – пакет 1



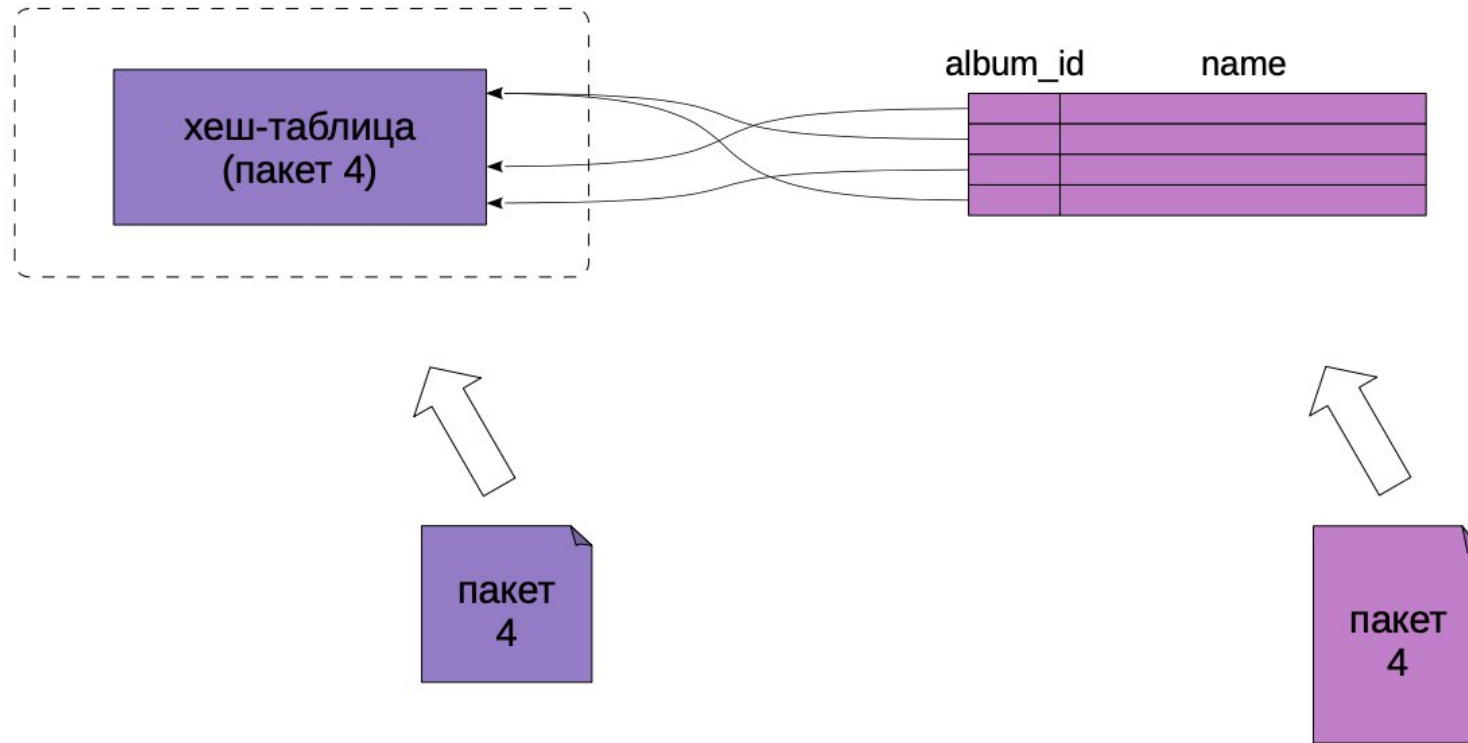
Сопоставление – пакет 2



Сопоставление – пакет 3



Сопоставление – пакет 4



Вычислительная сложность

$\sim N+M$, где

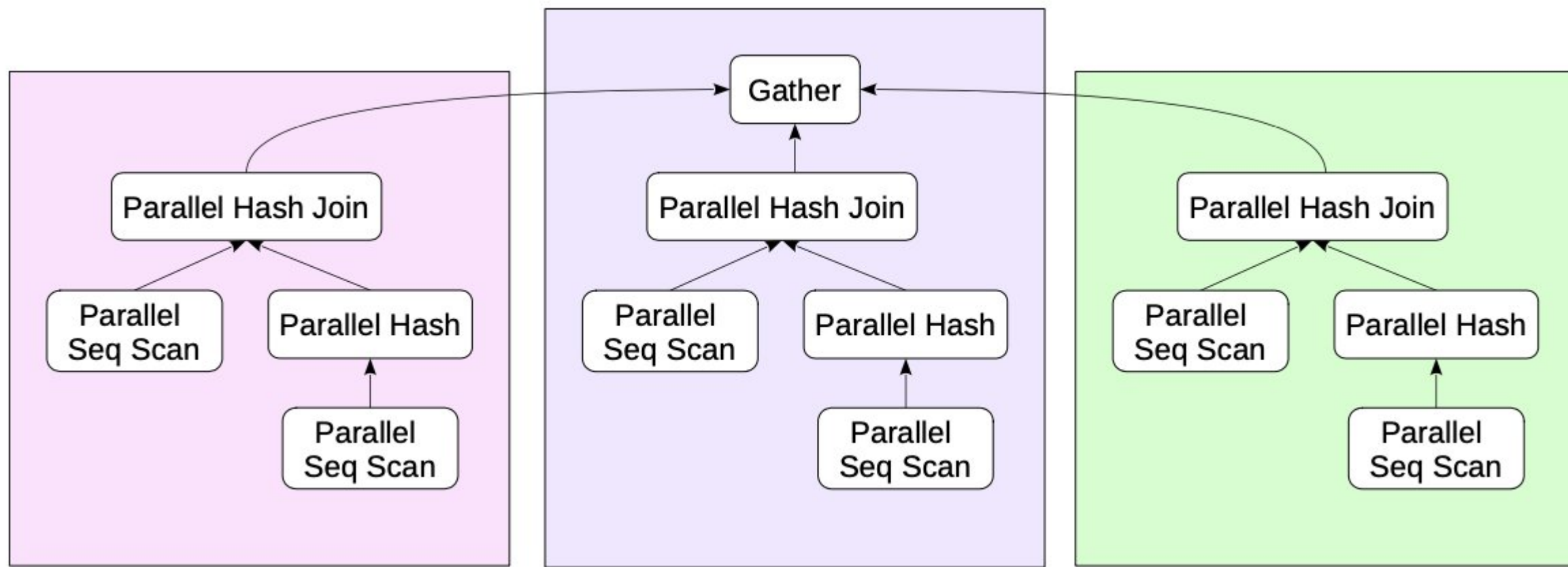
N и M — число строк в первом и втором наборах данных

- Начальные затраты на построение хеш-таблицы
- Эффективно для большого числа строк

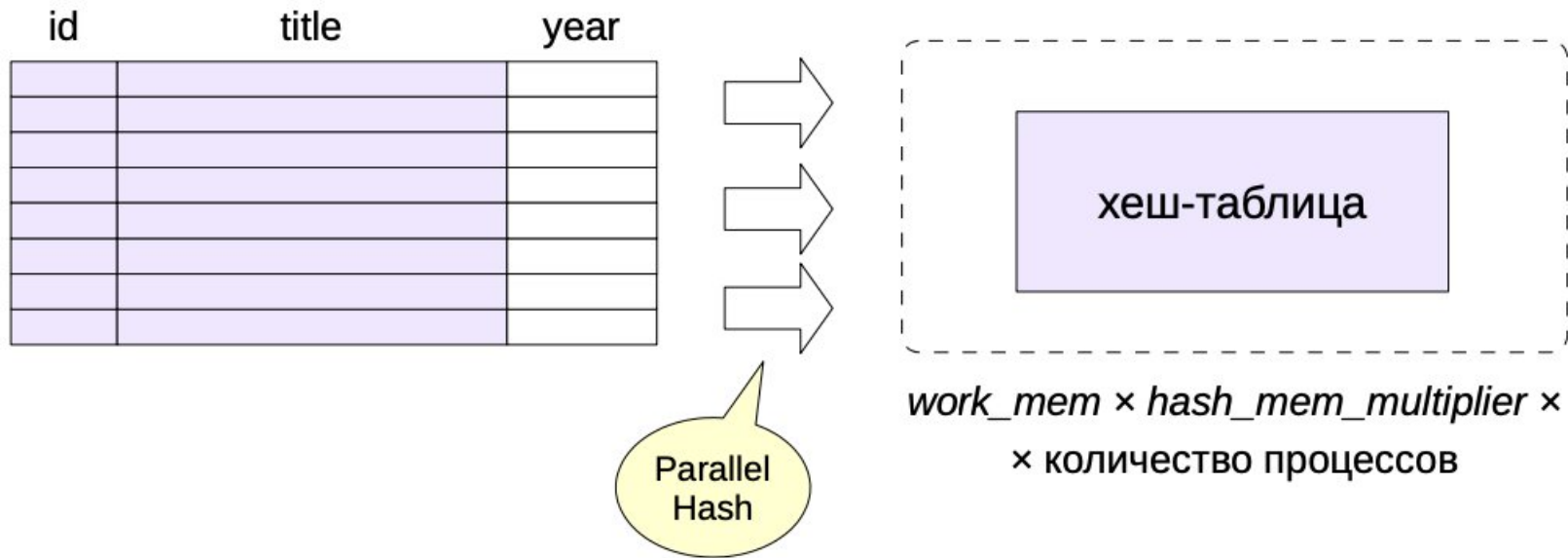
Параллельно, один проход

- Процессы используют общую хеш-таблицу

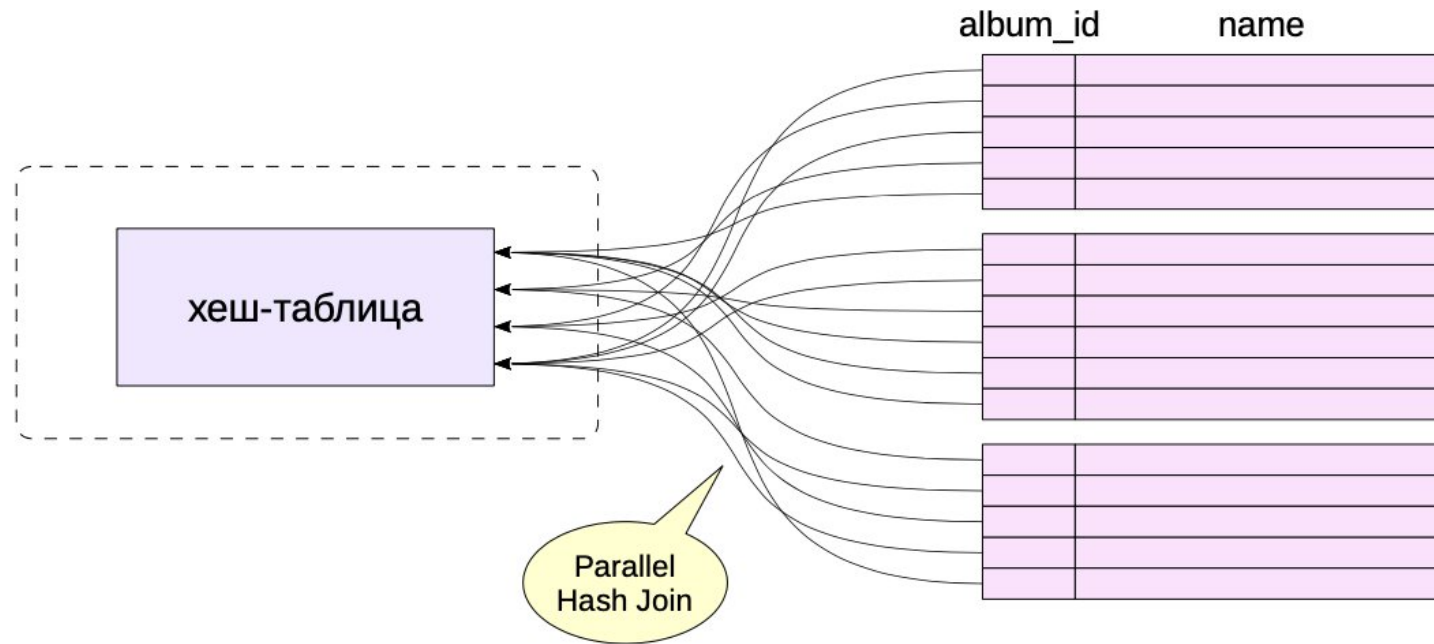
Параллельный алгоритм



Построение хеш-таблицы



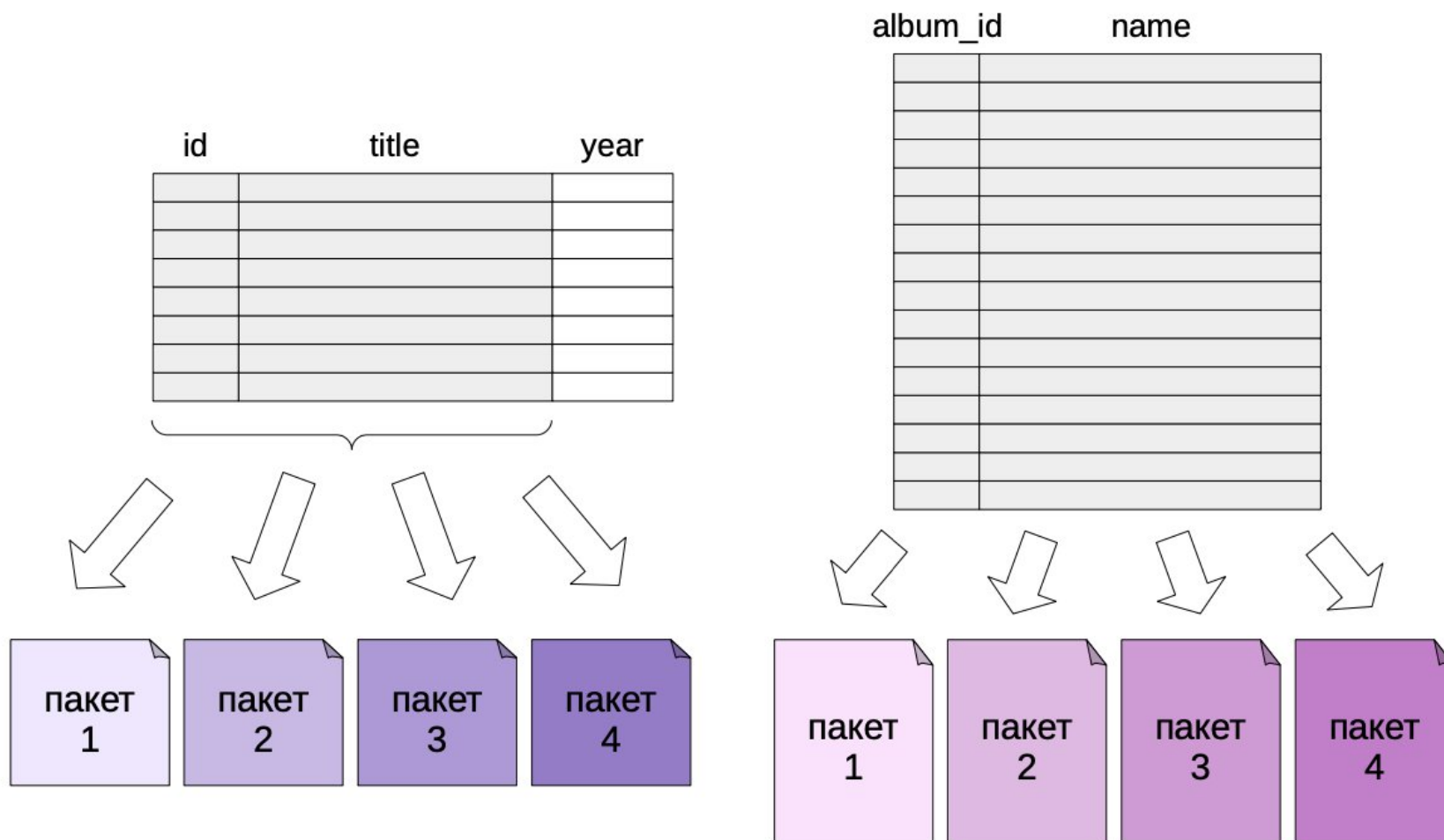
Сопоставление



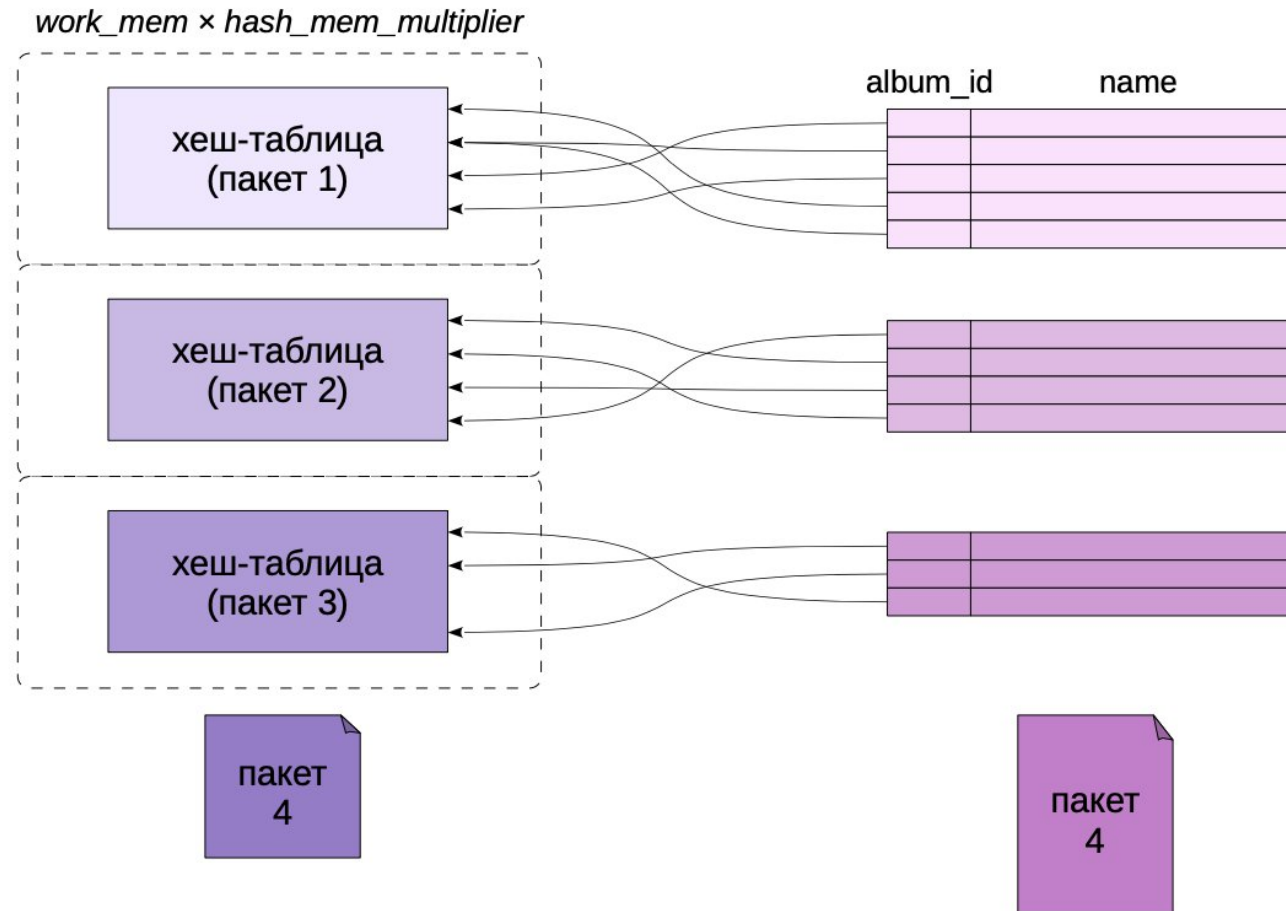
Параллельно, два прохода

- Наборы строк разбиваются на пакеты, которые затем параллельно обрабатываются рабочими процессами

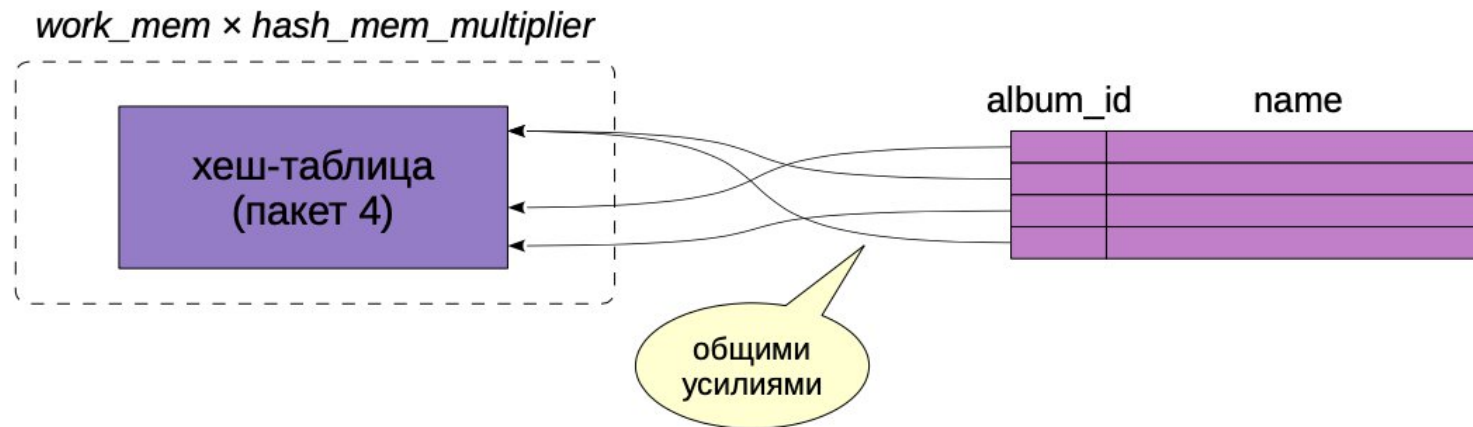
Разбиение на пакеты



Сопоставление



Сопоставление



Оптимизация запросов

- Соединение слиянием

Соединение слиянием

- Алгоритм соединения слиянием
- Вычислительная сложность
- Соединение слиянием в параллельных планах

Сортировка

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.a_id;
```

id	title	year
1	Yellow Submarine	1969
3	Let It Be	1970
4	The Beatles	1968
6	Abbey Road	1969

album_id	name
1	All Together Now
1	All You Need Is Love
2	Another Girl
2	Act Naturally
3	Across the Universe
5	A Day in the Life

либо сортировка,
либо получение отсортированных данных от нижестоящего узла плана

Слияние

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.a_id;
```

id	title	year	album_id	name
1	Yellow Submarine	1969	1	All Together Now
3	Let It Be	1970	1	All You Need Is Love
4	The Beatles	1968	2	Another Girl
6	Abbey Road	1969	2	Act Naturally
			3	Across the Universe
			5	A Day in the Life

результат соединения автоматически отсортирован

Слияние

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.a_id;
```

id	title	year
1	Yellow Submarine	1969
3	Let It Be	1970
4	The Beatles	1968
6	Abbey Road	1969


album_id	name
1	All Together Now
1	All You Need Is Love
2	Another Girl
2	Act Naturally
3	Across the Universe
5	A Day in the Life



Слияние

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.a_id;
```

id	title	year	album_id	name
1	Yellow Submarine	1969	1	All Together Now
3	Let It Be	1970	1	All You Need Is Love
4	The Beatles	1968	2	Another Girl
6	Abbey Road	1969	2	Act Naturally
			3	Across the Universe
			5	A Day in the Life



Слияние

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.a_id;
```

id	title	year
1	Yellow Submarine	1969
3	Let It Be	1970
4	The Beatles	1968
6	Abbey Road	1969

album_id	name
1	All Together Now
1	All You Need Is Love
2	Another Girl
2	Act Naturally
3	Across the Universe
5	A Day in the Life



Слияние

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.a_id;
```

id	title	year
1	Yellow Submarine	1969
3	Let It Be	1970
4	The Beatles	1968
6	Abbey Road	1969

album_id	name
1	All Together Now
1	All You Need Is Love
2	Another Girl
2	Act Naturally
3	Across the Universe
5	A Day in the Life

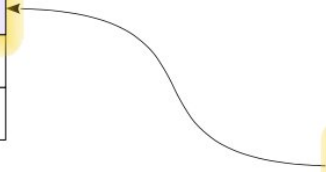


Слияние

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.a_id;
```

id	title	year
1	Yellow Submarine	1969
3	Let It Be	1970
4	The Beatles	1968
6	Abbey Road	1969

album_id	name
1	All Together Now
1	All You Need Is Love
2	Another Girl
2	Act Naturally
3	Across the Universe
5	A Day in the Life

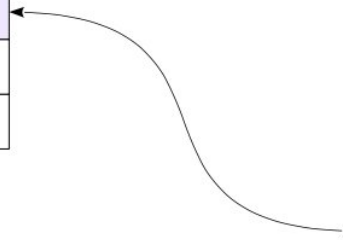


Слияние

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.a_id;
```

id	title	year
1	Yellow Submarine	1969
3	Let It Be	1970
4	The Beatles	1968
6	Abbey Road	1969

album_id	name
1	All Together Now
1	All You Need Is Love
2	Another Girl
2	Act Naturally
3	Across the Universe
5	A Day in the Life

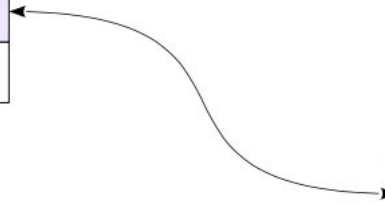


Слияние

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.a_id;
```

id	title	year
1	Yellow Submarine	1969
3	Let It Be	1970
4	The Beatles	1968
6	Abbey Road	1969


album_id	name
1	All Together Now
1	All You Need Is Love
2	Another Girl
2	Act Naturally
3	Across the Universe
5	A Day in the Life



Слияние

```
SELECT a.title, s.name  
FROM albums a JOIN songs s ON a.id = s.a_id;
```

id	title	year	album_id	name
1	Yellow Submarine	1969	1	All Together Now
3	Let It Be	1970	1	All You Need Is Love
4	The Beatles	1968	2	Another Girl
6	Abbey Road	1969	2	Act Naturally
			3	Across the Universe
			5	A Day in the Life



Вычислительная сложность

$\sim N+M$, где

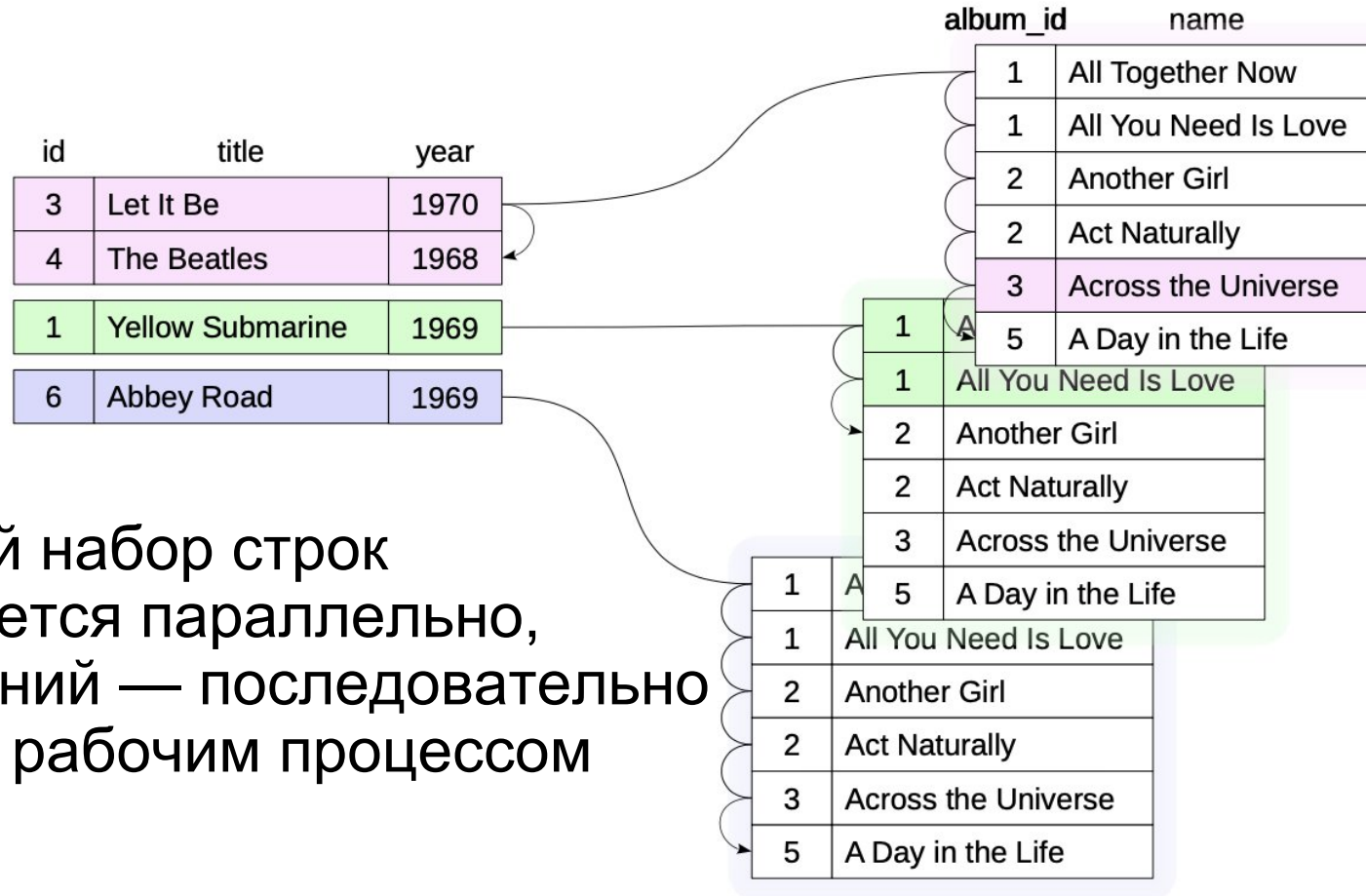
N и M — число строк в первом и втором наборах данных,
если не требуется сортировка

$\sim N \log N + M \log M$,

если сортировка нужна

- Возможные начальные затраты на сортировку
- Эффективно для большого числа строк

В параллельных планах



внешний набор строк
сканируется параллельно,
внутренний — последовательно
каждым рабочим процессом

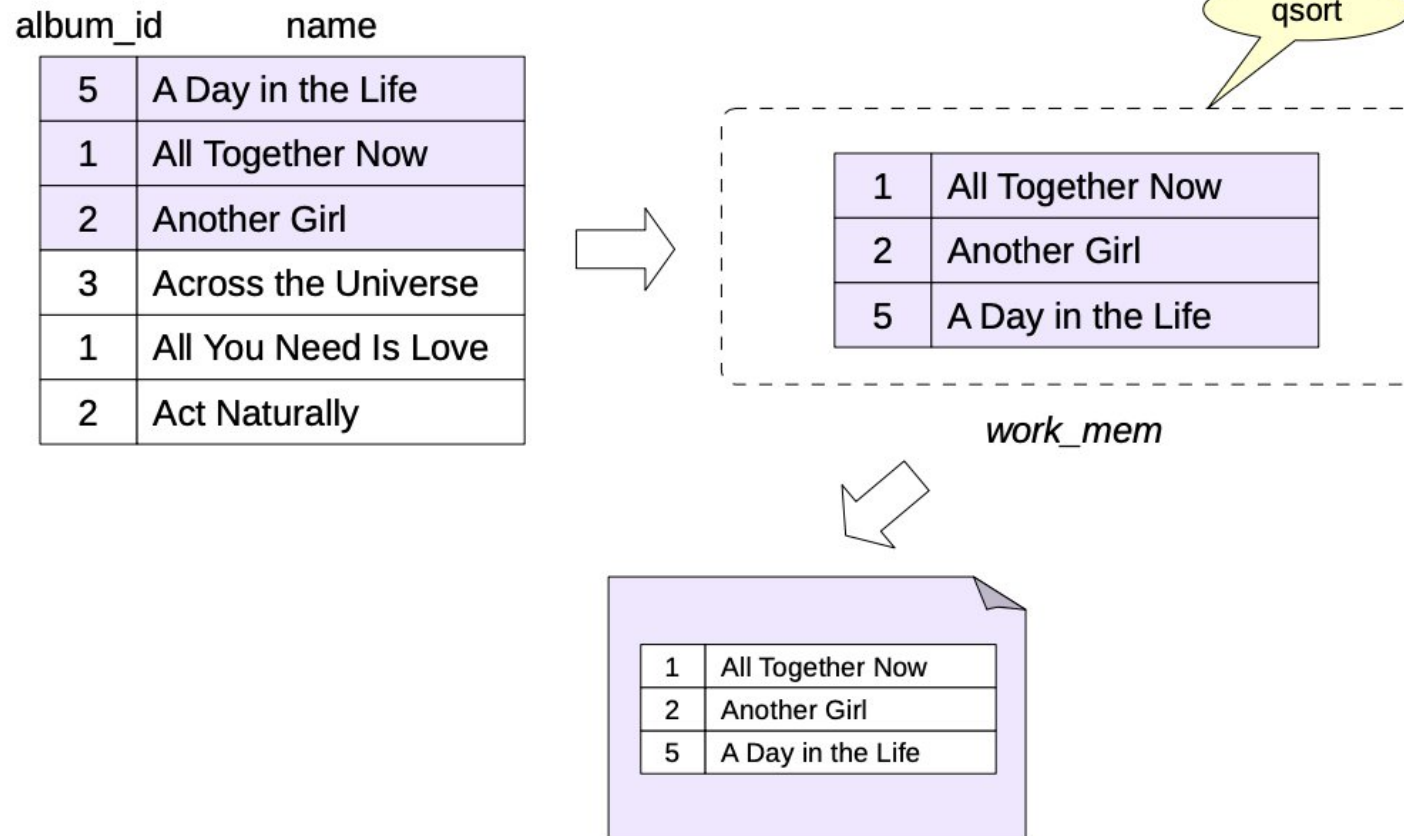
Сортировка

- Сортировка в памяти
- Внешняя сортировка
- Группировка с помощью сортировки
- Сортировка в параллельных планах

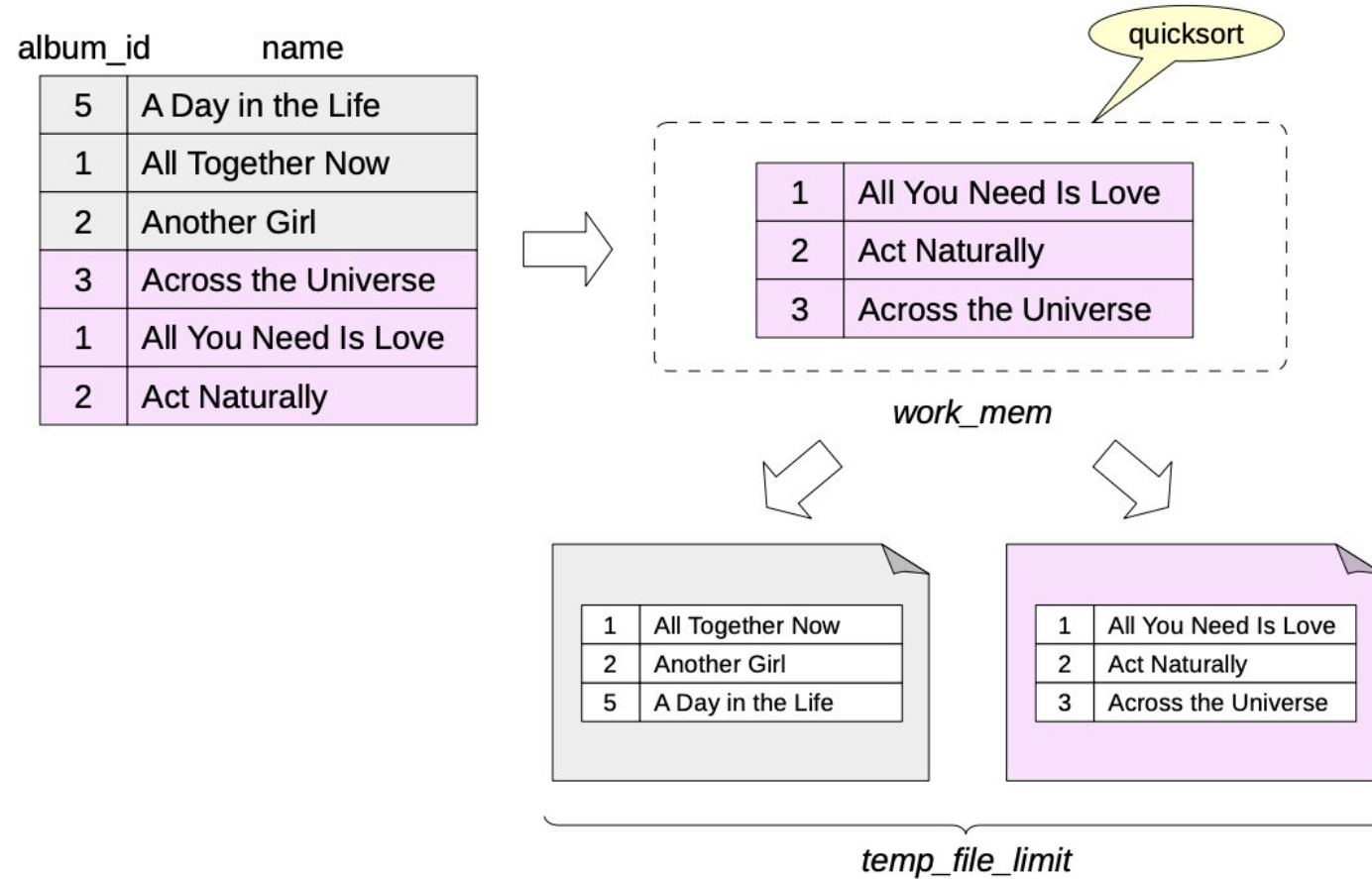
Сортировка в памяти

- Быстрая сортировка (quick sort)
- Частичная пирамидальная сортировка (top-N heapsort)
 - когда нужна только часть значений
- Инкрементальная сортировка
 - когда данные уже отсортированы, но не по всем ключам

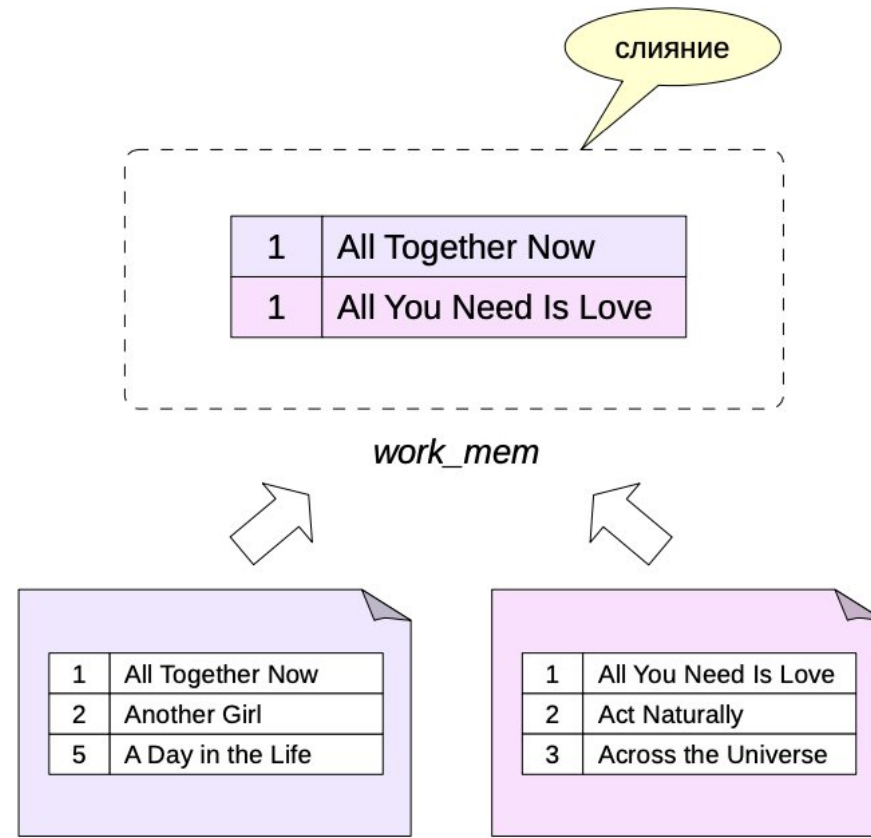
Внешняя сортировка



Внешняя сортировка

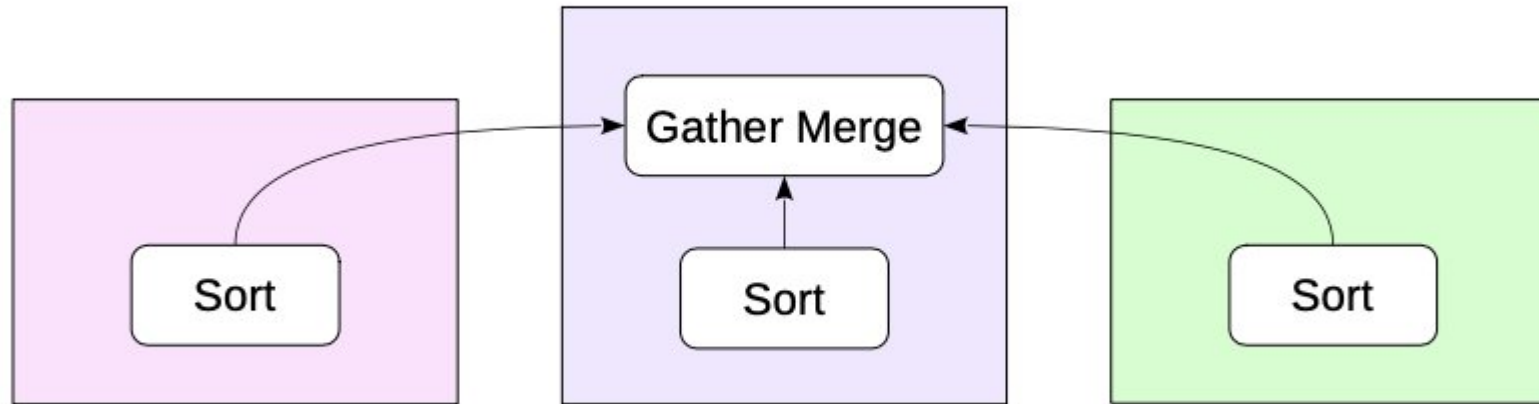


Внешняя сортировка



Параллельная сортировка

- Узел Gather Merge сохраняет порядок сортировки
 - выполняет слияние данных, поступающих от нижестоящих узлов



Создание индекса

- Используется сортировка
 - сначала все строки сортируются
 - затем строки собираются в листовые индексные страницы
 - ссылки на них собираются в страницы следующего уровня
 - и так далее, пока не дойдем до корня
- Может выполняться параллельно
 - `max_parallel_maintenance_workers`
- Ограничение
 - `maintenance_work_mem`, так как операция не частая

Оптимизация запросов

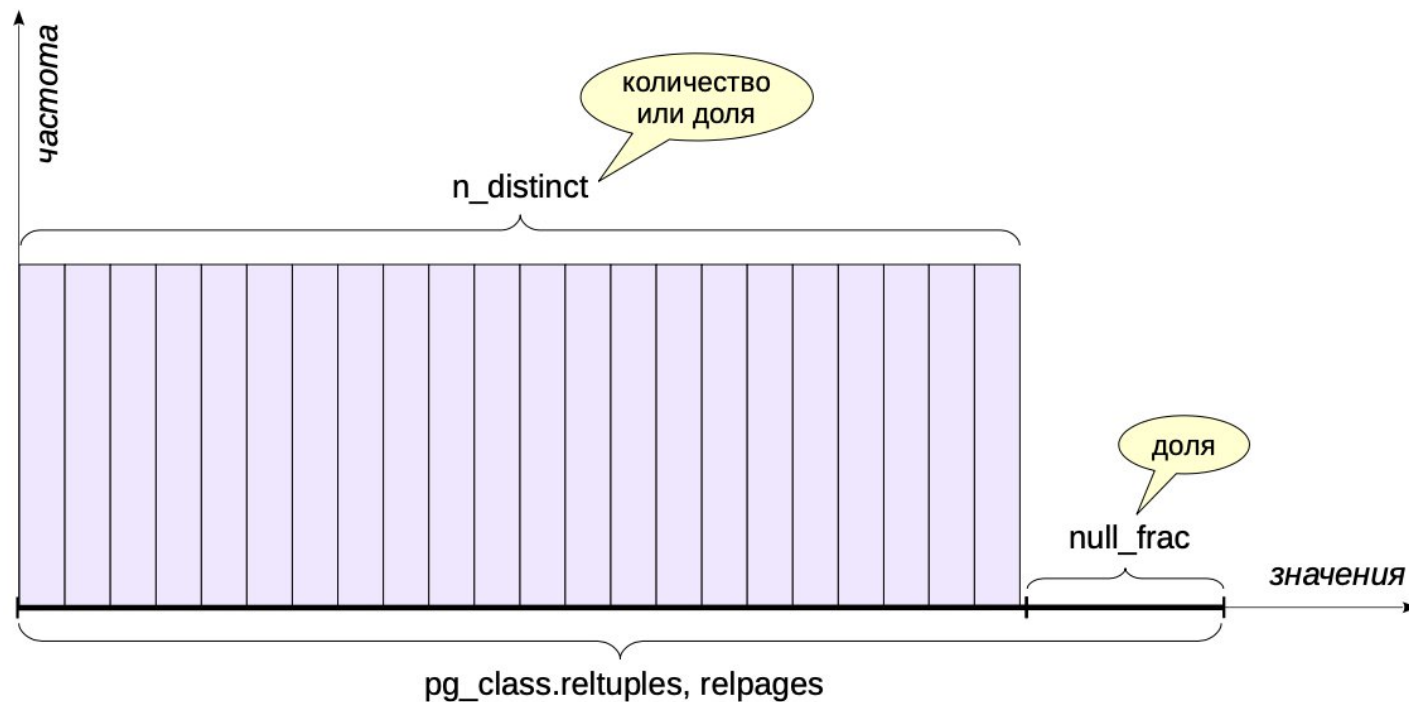
- Статистика

Базовая статистика

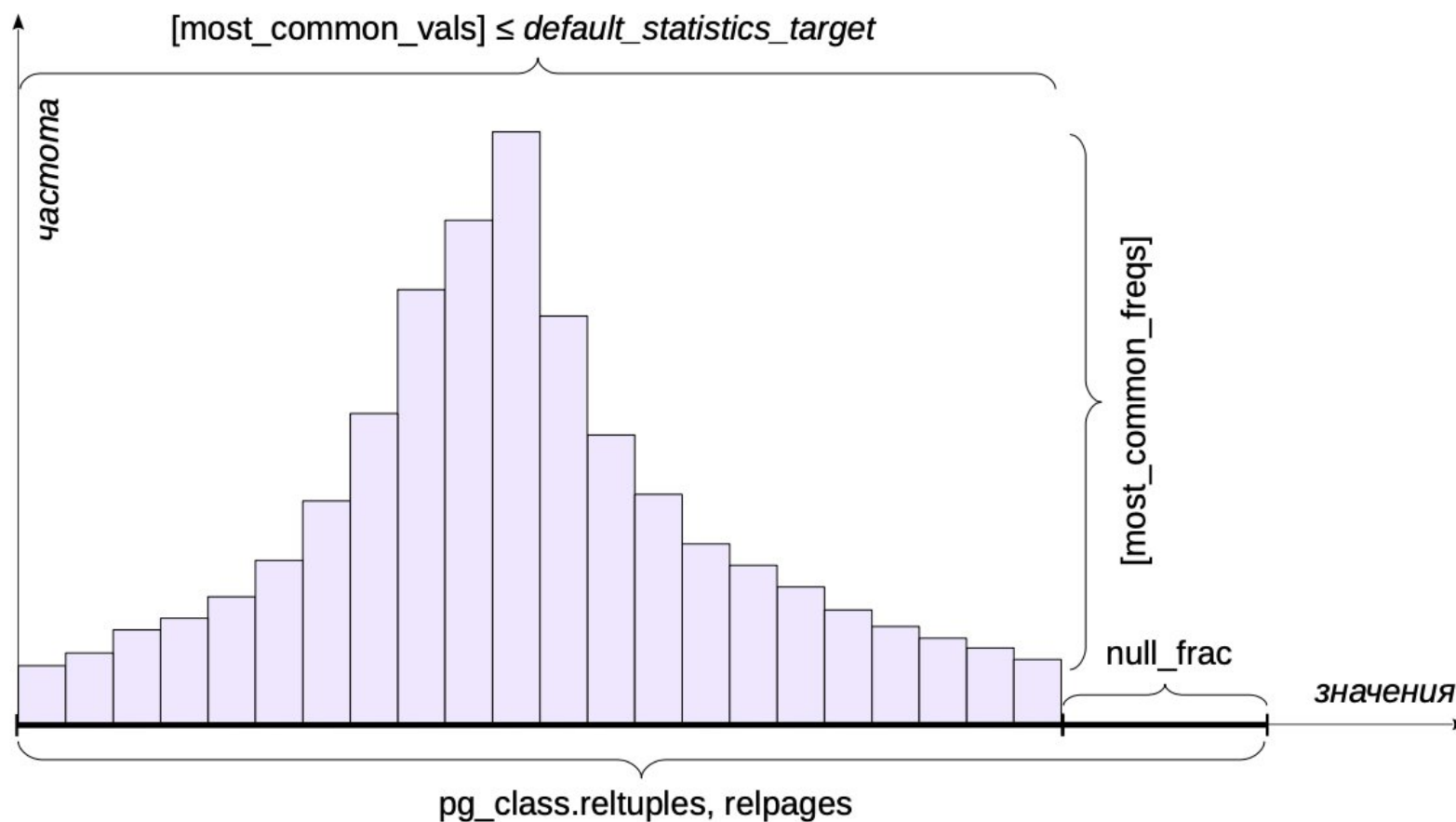
- Размер таблицы
 - строки (`pg_class.reltuples`) и страницы (`pg_class.relpages`)
- Собирается
 - операциями DDL
 - очисткой
 - анализом
- Настройка
 - `default_statistics_target = 100`

Базовая статистика

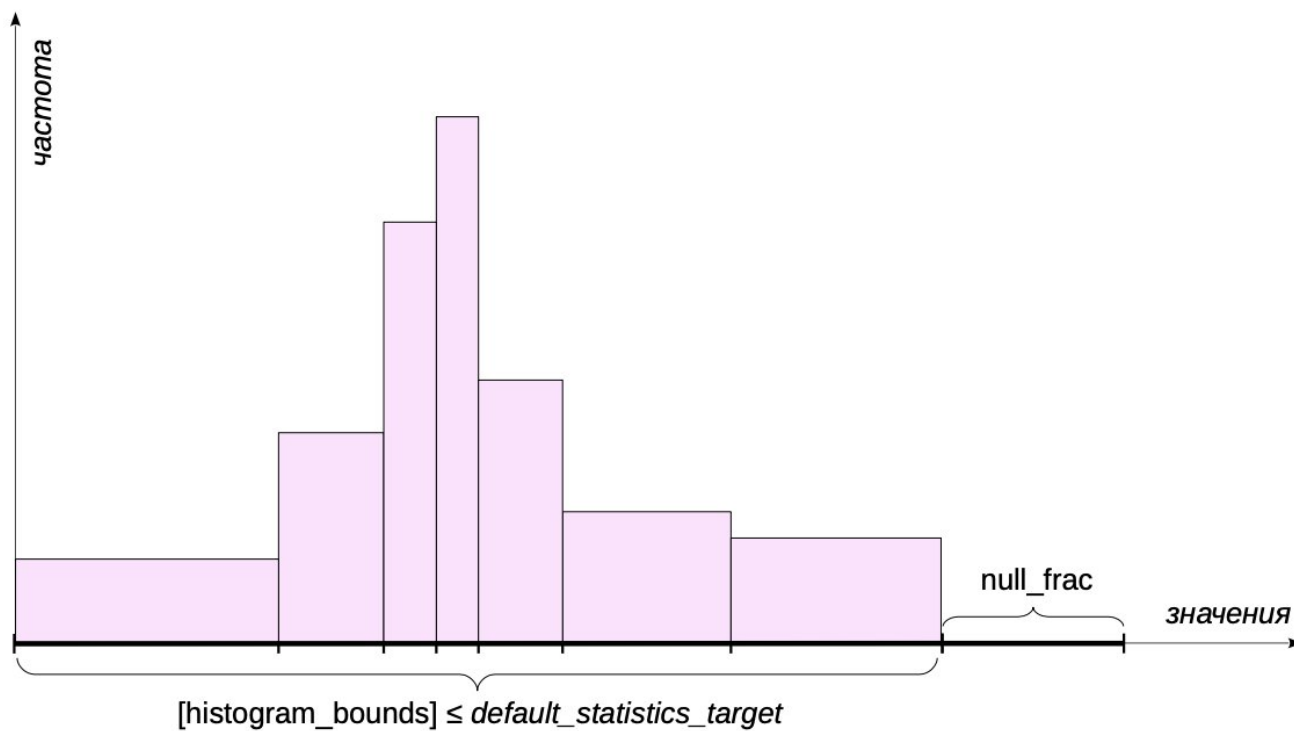
pg_statistic (pg_stats)



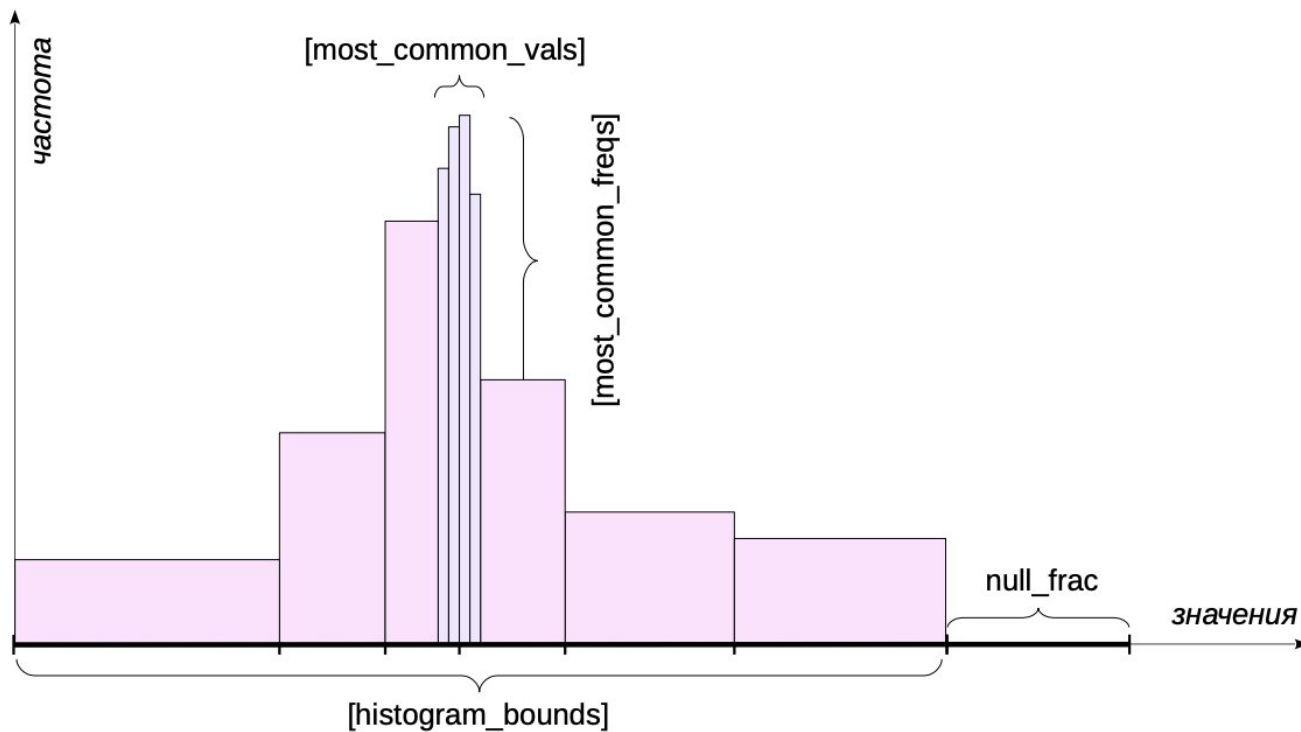
Наиболее частые значения



Гистограмма



Комбинация методов



Дополнительные поля

- Упорядоченность (использовать ли битовую карту?)
 - `pg_stats.correlation`
(1 — по возрастанию, 0 — хаотично, -1 — по убыванию)
- Видимость (использовать ли сканирование только индекса?)
 - `pg_class.relallvisible`
- Средний размер значения в байтах (оценка памяти)
 - `pg_stats.avg_width`
- Информация об элементах массивов, `tsvector` и т. п.
 - `pg_stats.most_common_elems`
 - `pg_stats.most_common_elem_freqs`
 - `pg_stats.elem_count_histogram`

Расширенная статистика

- CREATE STATISTICS
 - объект базы данных, создается вручную
 - после создания статистика собирается автоматически
 - pg_statistic_ext и pg_statistic_ext_data; представление pg_stats_ext
- Функциональные зависимости между столбцами и списки наиболее частых комбинаций значений
 - улучшают оценку селективности условий с коррелированными предикатами
- Число уникальных комбинаций значений
 - улучшает оценку кардинальности для группировки

Оптимизация запросов

- Профилирование

Инструмент

- Профилирование
 - выделение подзадач
 - продолжительность
 - количество выполнений
- Что оптимизировать?
 - чем больше доля подзадачи в общем времени выполнения, тем больше потенциальный выигрыш
 - необходимо учитывать затраты на оптимизацию
 - полезно взглянуть на задачу шире

Что профилировать

- Всю активность системы
 - полезно администратору для поиска ресурсоемких задач
 - мониторинг, расширение pg_profile
- Отдельную задачу, вызывающую нарекания
 - полезно для решения конкретной проблемы
 - чем точнее, тем лучше: широкий охват размывает картину

Что измерять

- Время выполнения
 - имеет смысл для пользователя
 - крайне нестабильный показатель
- Страничный ввод-вывод
 - стабилен по отношению ко внешним факторам
 - мало что говорит пользователям

Профиль приложения

- Подзадачи
 - клиентская часть
 - сервер приложений
 - сервер баз данных ← проблема часто, но не всегда, именно здесь
 - сеть
- Как профилировать
 - технически трудно, нужны разнообразные средства мониторинга
 - подтвердить предположение обычно несложно

Профиль PL/pgSQL

- Расширение PL Profiler
 - стороннее расширение
 - предназначено только для функций на PL/pgSQL
 - профилирование отдельного скрипта или сеанса
 - отчет о работе в html, включая flame graph

Профиль запросов

- Журнал сообщений сервера
 - включается конфигурационными параметрами:
 - `log_min_duration_statements=0` — время и текст всех запросов
 - `log_line_prefix` — идентифицирующая информация
 - сложно включить для отдельного сеанса
 - большой объем; при увеличении порога времени теряем информацию
 - не отслеживаются вложенные запросы (можно использовать расширение `auto_explain`)
 - анализ внешними средствами, такими, как `pgBadger`

Профиль запросов

- Расширение `pg_stat_statements`
 - подробная информация о запросах в представлении (в том числе о вложенных)
 - ограниченный размер хранилища
 - запросы считаются одинаковыми «с точностью до констант», даже если имеют разные планы выполнения
 - идентификация только по имени пользователя и базе данных

Профиль одного запроса

- EXPLAIN ANALYZE

- подзадачи — узлы плана
- продолжительность — actual time
или страничный ввод-вывод — buffers
- количество выполнений — loops

- Особенности

- помимо наиболее ресурсоемких узлов, кандидаты на оптимизацию — узлы с большой ошибкой прогноза кардинальности
- любое вмешательство может привести к полной перестройке плана
- иногда приходится довольствоваться простым EXPLAIN

Оптимизация запросов

- Приемы оптимизации

Пути оптимизации

- Цель оптимизации — получить адекватный план
- Исправление неэффективностей
 - каким-то образом найти и исправить узкое место
 - бывает сложно определить, в чем проблема
 - часто приводит к борьбе с планировщиком
- Правильный расчет кардинальности
 - добиться правильного расчета кардинальности в каждом узле и положиться на планировщик
 - если план все еще неадекватный, настраивать глобальные параметры

Статистика

- Актуальность
 - настройка автоочистки и автоанализа
 - `autovacuum_max_workers`, `autovacuum_analyze_scale_factor`, ...
- Точность
 - `default_statistics_target` = 100
 - индекс по выражению
 - расширенная статистика (например, для коррелированных предикатов)
- Использование имеющейся статистики
 - планировщик не всегда может сделать правильные выводы из имеющихся данных
 - переформулирование запроса, временные таблицы

Настройки стоимости

- Ввод-вывод
 - можно (и нужно) указывать на уровне табличных пространств
 - seq_page_cost = 1.0
 - random_page_cost = 4.0
 - effective_io_concurrency = 1

Настройки стоимости

- Время процессора
 - `cpu_tuple_cost` = 0.01
 - `cpu_index_tuple_cost` = 0.005
 - `cpu_operator_cost` = 0.0025
 - `CREATE FUNCTION ... COST` *стоимость*

Настройки стоимости

- Параллельное выполнение
 - `parallel_setup_cost` = 1000
 - `parallel_tuple_cost` = 0.1
- Курсоры
 - `cursor_tuple_fraction` = 0.1

Настройки памяти

- Память
 - work_mem = 4MB
 - hash_mem_multiplier = 1
 - maintenance_work_mem = 64MB
 - effective_cache_size = 4GB

Локализация настроек

- На уровне табличного пространства
 - ALTER TABLESPACE SET ...
- На уровне базы данных
 - ALTER DATABASE SET ...
- На уровне роли и базы данных
 - ALTER ROLE [IN DATABASE ...] SET ...
- На уровне процедуры или функции
 - ALTER ROUTINE SET ...
- На уровне сеанса или транзакции
 - SET [LOCAL] ...

Отладка

- Отключение определенных способов доступа
 - `enable_seqscan`
 - `enable_indexscan`, `enable_bitmapscan`, `enable_indexonlyscan`
- Отключение определенных методов соединений
 - `enable_nestloop`, `enable_hashjoin`, `enable_mergejoin`
- Отключение определенных операций `enable_hashagg`,
 - `enable_sort`, `enable_incremental_sort`
- Проверка возможности распараллеливания
 - `force_parallel_mode`

Схема данных

- Нормализация — устранение избыточности в данных
 - упрощает запросы и проверку согласованности
- Денормализация — привнесение избыточности
 - может повысить производительность, но требует синхронизации
 - индексы
 - предрассчитанные поля (генерируемые столбцы или триггеры)
 - материализованные представления
 - кеширование результатов в приложении

Схема данных

- Типы данных
 - выбор подходящих типов
 - составные типы (массивы, JSON) вместо отдельных таблиц
- Ограничения целостности
 - помимо обеспечения целостности данных, могут учитываться планировщиком для устранения ненужных соединений, улучшения оценок селективности и других оптимизаций
 - первичный ключ и уникальность — уникальный индекс
 - внешний ключ
 - отсутствие неопределенных значений
 - проверка CHECK (constraint_exclusion)

Физическое расположение

- Табличные пространства
 - разнесение данных по разным физическим устройствам
- Секционирование
 - разделение таблицы на отдельно управляемые части для упрощения администрирования и ускорения доступа
- Шардирование
 - размещение секций на разных серверах для масштабирования нагрузки на чтение и запись
 - секционирование и расширение postgres-fdw или сторонние решения

Порядок соединений

- Автоматический выбор порядка соединений
 - если число соединяемых таблиц не превышает `join_collapse_limit` — планировщик выбирает лучший порядок соединений
 - при большем количестве рассматриваются не все варианты
- Ручное управление порядком соединений
 - материализация подзапросов с помощью CTE или временных таблиц
 - явные соединения (JOIN) и `join_collapse_limit = 1`
 - подзапросы в предложении FROM и `from_collapse_limit = 1`

Изменение запросов

- Альтернативные способы выполнения
 - планировщик не всегда рассматривает все возможные трансформации
 - раскрытие коррелированных подзапросов
 - устранение лишних таблиц
 - замена UNION на OR и обратно
 - и т. п.
- Замена процедурного кода декларативным
 - чтобы избавиться от большого числа мелких запросов

Подсказки оптимизатору

- Отсутствуют в явном виде
 - хотя средства влияния имеются: конфигурационные параметры, материализация CTE и другие
- Сторонние расширения
 - `pg_hint_plan`