



Базы данных

Лекция 3

Представления. Индексы



Андрей Каледин

В предыдущих сериях...

В предыдущих сериях...

- Создание таблиц в БД
- Добавление и изменение данных
- Получение данных
- Оконные функции
- JOIN

План лекции

1. View
2. Materialized View
3. Индексы: структуры данных
4. Индексы: назначение

VIEW

VIEW

View (Представление) – виртуальная таблица, создаваемая на основе SQL-запроса.

VIEW

View (Представление) – виртуальная таблица, создаваемая на основе SQL-запроса.

Сценарии использования:

1. Упрощение запросов
2. Ограничение доступа к части данных
3. Изменение структуры БД не требует изменений в коде (но точно ли вы этого хотите?)

VIEW

View (Представление) – виртуальная таблица, создаваемая на основе SQL-запроса.

Сценарии использования:

1. Упрощение запросов
2. Ограничение доступа к части данных
3. Изменение структуры БД не требует изменений в коде (но точно ли вы этого хотите?)

```
CREATE VIEW active_users AS  
SELECT id, company_id, username  
FROM users  
WHERE status = 'active';
```


VIEW

View (Представление) – виртуальная таблица, создаваемая на основе SQL-запроса.

Сценарии использования:

1. Упрощение запросов
2. Ограничение доступа к части данных
3. Изменение структуры БД не требует изменений в коде (но точно ли вы этого хотите?)

```
CREATE VIEW active_users AS  
SELECT id, company_id, username  
FROM users  
WHERE status = 'active';
```

```
SELECT * FROM pg_views WHERE schemaname = 'public';
```

VIEW vs MATERIALIZED VIEW

VIEW	MATERIALIZED VIEW

VIEW vs MATERIALIZED VIEW

VIEW	MATERIALIZED VIEW
Не хранит копию данных, хранит только запрос к настоящей таблице	Хранит копию данных

VIEW vs MATERIALIZED VIEW

VIEW	MATERIALIZED VIEW
Не хранит копию данных, хранит только запрос к настоящей таблице	Хранит копию данных
Возвращает самые актуальные данные, через запрос в реальную таблицу	Возвращает устаревшие данные из своей копии

VIEW vs MATERIALIZED VIEW

VIEW	MATERIALIZED VIEW
Не хранит копию данных, хранит только запрос к настоящей таблице	Хранит копию данных
Возвращает самые актуальные данные, через запрос в реальную таблицу	Возвращает устаревшие данные из своей копии
Ответ отдает медленнее	Ответ отдает быстрее (данные уже сохранены в нужном формате)

VIEW vs MATERIALIZED VIEW

VIEW	MATERIALIZED VIEW
Не хранит копию данных, хранит только запрос к настоящей таблице	Хранит копию данных
Возвращает самые актуальные данные, через запрос в реальную таблицу	Возвращает устаревшие данные из своей копии
Ответ отдает медленнее	Ответ отдает быстрее (данные уже сохранены в нужном формате)
Самые актуальные данные отдает автоматически	Требует явного обновления через REFRESH

VIEW vs MATERIALIZED VIEW

VIEW	MATERIALIZED VIEW
Не хранит копию данных, хранит только запрос к настоящей таблице	Хранит копию данных
Возвращает самые актуальные данные, через запрос в реальную таблицу	Возвращает устаревшие данные из своей копии
Ответ отдает медленнее	Ответ отдает быстрее (данные уже сохранены в нужном формате)
Самые актуальные данные отдает автоматически	Требует явного обновления через REFRESH
Подходит для часто меняющихся данных	Подходит для данных с редкими изменениями

VIEW vs MATERIALIZED VIEW

VIEW	MATERIALIZED VIEW
<pre>CREATE VIEW active_users AS SELECT id, company_id, username FROM users WHERE status = 'active';</pre>	<pre>CREATE MATERIALIZED VIEW active_users AS SELECT id, company_id, username FROM users WHERE status = 'active';</pre>

Индексы

Индексы

Индекс - структура данных, которая ускоряет поиск, сортировку и фильтрацию записей в таблице.

Назначение:

1. Ускорение WHERE, ORDER BY и GROUP BY
2. Оптимизация JOIN
3. Обеспечение уникальности значений

```
SELECT * FROM pg_indexes WHERE tablename = 'ваша_таблица';
```

Виды индексов

По структуре данных:

1. В-дерево
2. Hash-индекс
3. Bitmap-индекс
4. R-дерево
5. Generalized Inverted Index

Виды индексов

По структуре данных:

1. B-дерево
2. Hash-индекс
3. Bitmap-индекс
4. R-дерево
5. Generalized Inverted Index

По назначению:

1. Уникальный
2. Составной
3. Частичный
4. Покрывающий
5. Полнотекстовый
6. Кластеризованный

Виды индексов

По структуре данных:

1. В-дерево
2. Hash-индекс
3. Bitmap-индекс
4. R-дерево
5. Generalized Inverted Index

По назначению:

1. Уникальный
2. Составной
3. Частичный
4. Покрывающий
5. Полнотекстовый
6. Кластеризованный

В-дерево (B-tree)

Сбалансированное дерево, где каждый узел содержит ключи и ссылки на дочерние узлы. Все листья находятся на одном уровне.

Назначение:

1. Диапазонные запросы ($>$, $<$)
2. Сортировки
3. Поиск по префиксу (LIKE 'lol%')

В-дерево (B-tree)

Сбалансированное дерево, где каждый узел содержит ключи и ссылки на дочерние узлы. Все листья находятся на одном уровне.

Назначение:

1. Диапазонные запросы ($>$, $<$)
2. Сортировки
3. Поиск по префиксу (LIKE 'lol%')

Плюс: поддержка разных запросов

Минус: занимает много места

В-дерево (B-tree)

Сбалансированное дерево, где каждый узел содержит ключи и ссылки на дочерние узлы. Все листья находятся на одном уровне.

Назначение:

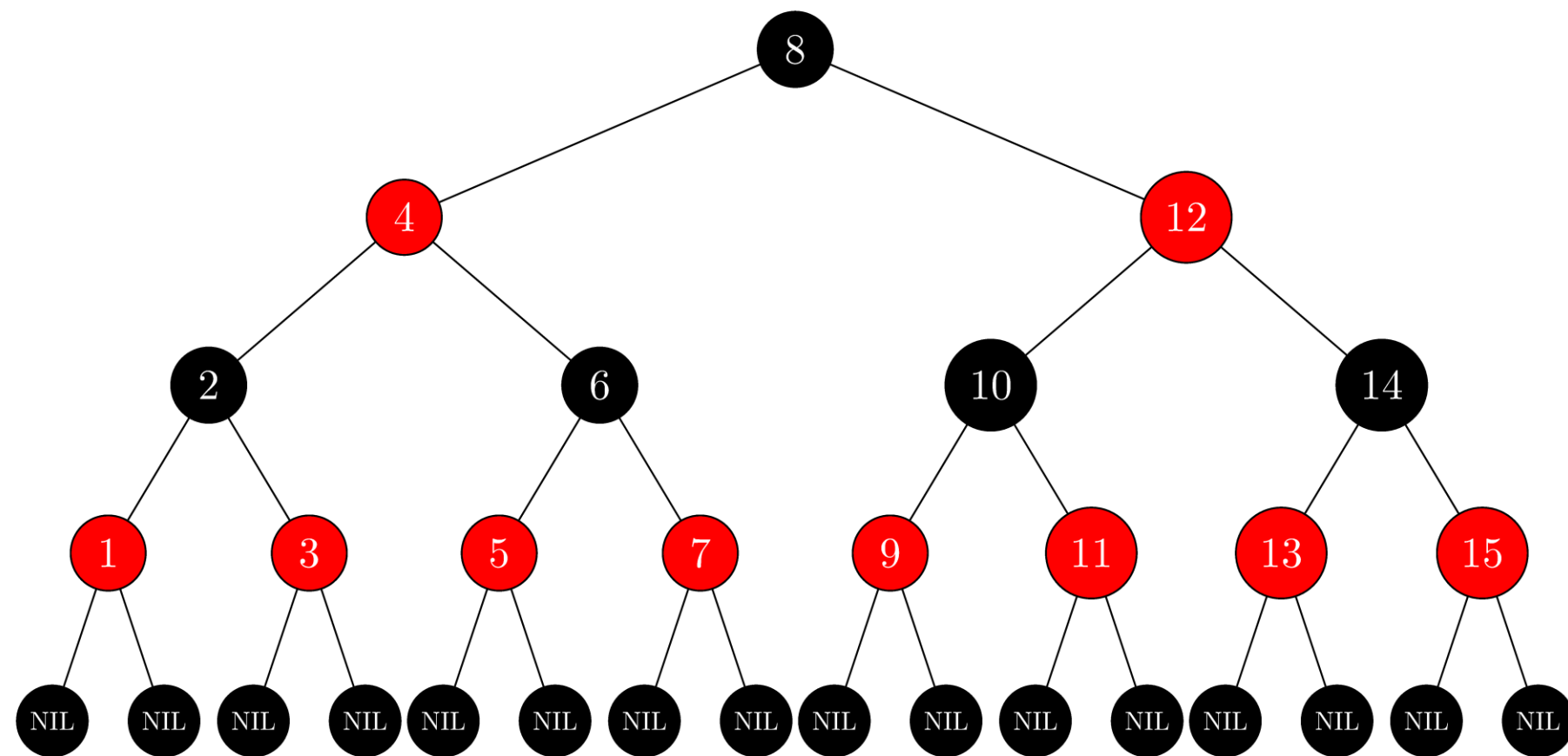
1. Диапазонные запросы ($>$, $<$)
2. Сортировки
3. Поиск по префиксу (LIKE 'lol%')

Плюс: поддержка разных запросов

Минус: занимает много места

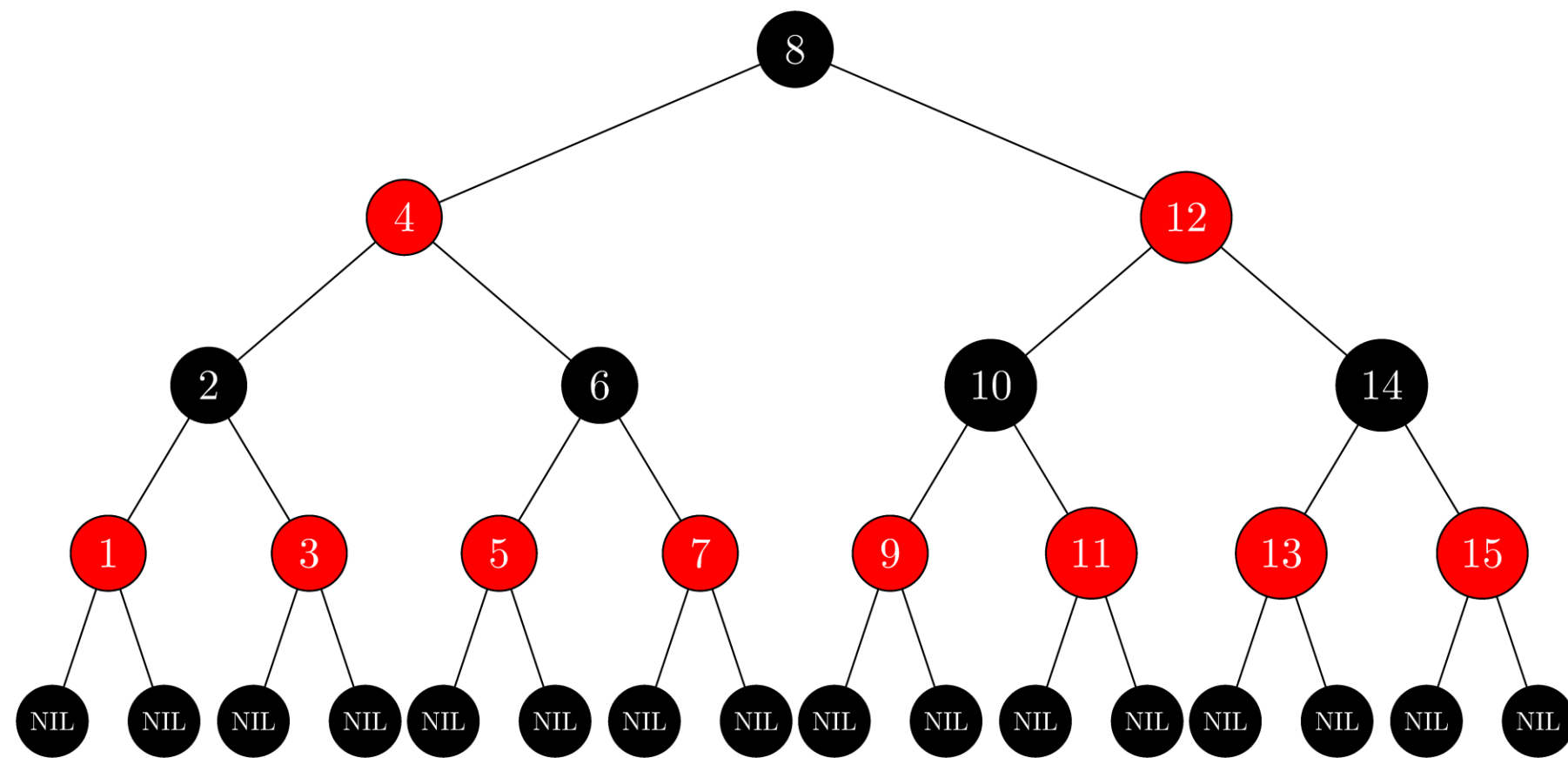
```
CREATE INDEX idx_users_name ON users (name);
```


В-дерево (B-tree)

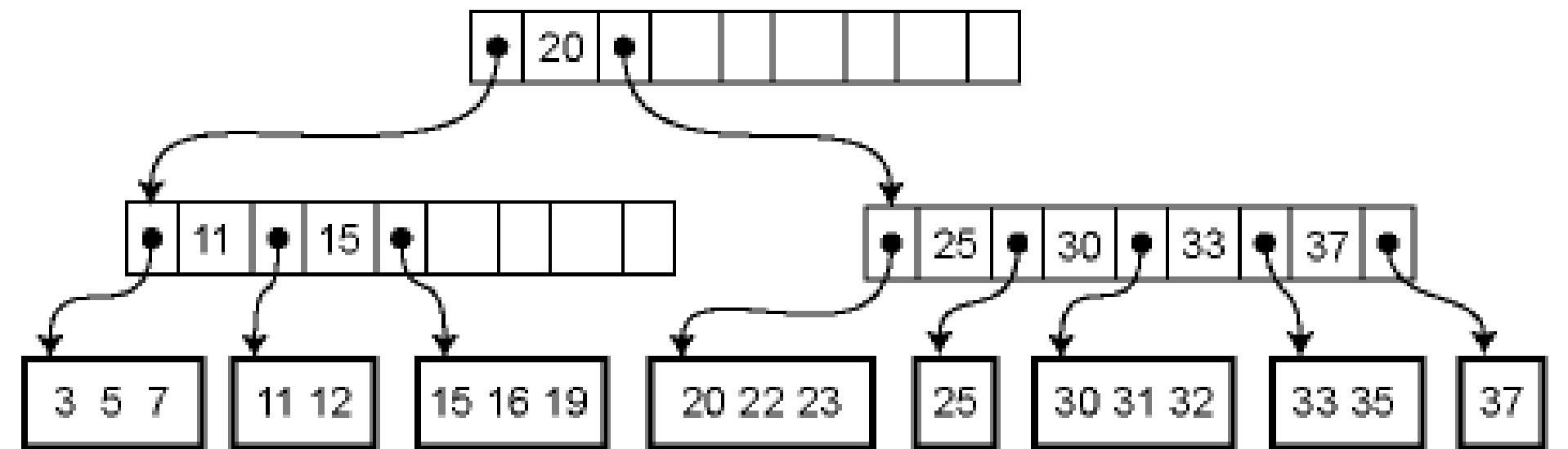


Red-Black tree

В-дерево (B-tree)



Red-Black tree



B-tree

Хэш-индекс (Hash-index)

Хеш-таблица, где ключ — результат хеш-функции от значения столбца.

Назначение:

1. Точные совпадения (=)

Хэш-индекс (Hash-index)

Хеш-таблица, где ключ — результат хеш-функции от значения столбца.

Назначение:

1. Точные совпадения (=)

Плюс: поиск за $O(1)$

Минус: только точные совпадения

Хэш-индекс (Hash-index)

Хеш-таблица, где ключ — результат хеш-функции от значения столбца.

Назначение:

1. Точные совпадения (=)

Плюс: поиск за $O(1)$

Минус: только точные совпадения

```
CREATE INDEX idx_users_email_hash ON users USING HASH (email);
```

Bitmap-индекс

Битовые маски для каждого уникального значения, где каждый бит соответствует строке в таблице

Назначение:

1. Поиск по enum'ам

Bitmap-индекс

Битовые маски для каждого уникального значения, где каждый бит соответствует строке в таблице

Назначение:

1. Поиск по enum'ам

Плюс: минимальный размер

Минус: плохо работает с большим количеством уникальных значений

Bitmap-индекс

Битовые маски для каждого уникального значения, где каждый бит соответствует строке в таблице

Назначение:

1. Поиск по enum'ам

Плюс: минимальный размер

Минус: плохо работает с большим количеством уникальных значений

Oracle:

```
CREATE BITMAP INDEX idx_users_status ON users (status);
```


R-tree

Структура, группирующая пространственные объекты в минимальные ограничивающие прямоугольники (minimum bounding rectangle)

Назначение:

1. Географические данные

R-tree

Структура, группирующая пространственные объекты в минимальные ограничивающие прямоугольники (minimum bounding rectangle)

Назначение:

1. Географические данные

Плюс: оптимизирован для многомерных данных

Минус: дорого обновлять

R-tree

Структура, группирующая пространственные объекты в минимальные ограничивающие прямоугольники (minimum bounding rectangle)

Назначение:

1. Географические данные

Плюс: оптимизирован для многомерных данных

Минус: дорого обновлять

```
CREATE INDEX idx_geo_data ON geo_objects USING GIST (geom);
```

PostgreSQL + PostGIS

GIN (Generalized Inverted Index)

Инвертированный индекс, хранящий соответствие элементов (например, слов, ключей, JSON) и строк.

Назначение:

1. Полнотекстовый поиск

GIN (Generalized Inverted Index)

Инвертированный индекс, хранящий соответствие элементов (например, слов, ключей, JSON) и строк.

Назначение:

1. Полнотекстовый поиск

Плюс: полнотекстовый поиск 🤖

Минус: дорого обновлять и хранить

GIN (Generalized Inverted Index)

Инвертированный индекс, хранящий соответствие элементов (например, слов, ключей, JSON) и строк.

Назначение:

1. Полнотекстовый поиск

Плюс: полнотекстовый поиск 🤖

Минус: дорого обновлять и хранить

```
CREATE INDEX idx_users_tags_gin ON users USING GIN (tags);
```

Индексы: выводы

- B-tree: Сортировка, поиск по префиксу
- Hash-индекс: Точные совпадения
- Bitmap-индекс: Индекс по enum'ам
- R-tree: Геоданные
- GIN-индекс: полнотекстовый поиск

Виды индексов

По структуре данных:

1. В-дерево
2. Hash-индекс
3. Bitmap-индекс
4. R-дерево
5. Generalized Inverted Index

По назначению:

1. Уникальный
2. Составной
3. Частичный
4. Покрывающий
5. Полнотекстовый
6. Кластеризованный

Уникальный индекс (UNIQUE)

Гарантирует, что значения в индексируемом столбце (или группе столбцов) не повторяются

Особенности:

1. Автоматически создается для PK и UNIQUE-ограничения
2. Нельзя создать, если в данных уже есть повторы

```
CREATE UNIQUE INDEX idx_users_email ON users (email)
```

Составной индекс (Composite Index)

Создается на несколько столбцов.

!!! Порядок важен !!!

Особенности:

1. Подходит для WHERE, JOIN, ORDER BY
2. (last_name, first_name) != (first_name, last_name)

```
CREATE INDEX idx_users_name ON users (last_name, first_name);
```

Частичный индекс (Partial Index)

Индексирует только часть строк, отфильтрованных по условию.

Особенности:

1. Меньше полного индекса

```
CREATE INDEX idx_users_active ON users (id) WHERE status = 'active';
```

Покрывающий индекс (Covering Index)

Содержит все поля, необходимые для запроса, чтобы избежать обращения к таблице

Особенности:

1. Увеличивает размер индекса
2. Ускоряет обработку запросов

```
CREATE INDEX idx_users_covering ON users (id) INCLUDE (email, name);
```

Полнотекстовый индекс (Full-Text Index)

Оптимизирован для поиска по тексту

Особенности:

1. Может ранжировать результаты по релевантности
2. Работает быстрее LIKE

Полнотекстовый индекс (Full-Text Index)

Оптимизирован для поиска по тексту

Особенности:

1. Может ранжировать результаты по релевантности
2. Работает быстрее LIKE

```
ALTER TABLE articles ADD COLUMN content_tsvector tsvector; --text_search_vector
```

```
UPDATE articles SET content_tsvector = to_tsvector('russian', content);
```

```
CREATE INDEX idx_articles_content_gin ON articles USING GIN (content_tsvector);
```

Полнотекстовый индекс (Full-Text Index)

Оптимизирован для поиска по тексту

Особенности:

1. Может ранжировать результаты по релевантности
2. Работает быстрее LIKE

```
ALTER TABLE articles ADD COLUMN content_tsvector tsvector; --text_search_vector
```

```
UPDATE articles SET content_tsvector = to_tsvector('russian', content);
```

```
CREATE INDEX idx_articles_content_gin ON articles USING GIN (content_tsvector);
```

```
SELECT * FROM articles  
WHERE content_tsvector @@ to_tsquery('russian', 'кошка & собака');
```

Кластеризованный индекс (Clustered Index)


Задаёт физический порядок данных в таблице. Для каждой таблицы может быть только один.

Особенности:

1. Данные хранятся в определенном порядке
2. Ускоряет запросы с ORDER BY

```
CREATE CLUSTERED INDEX idx_users_created ON users (created_at);
```


Индексы: выводы

- Уникальный: ограничивает уникальность значений
- Составной: ускоряет запросы с AND/OR
- Частичный: индексирует только часть данных
- Покрывающий: можно не обращаться к таблице
- Полнотекстовый: поиск по тексту 
- Кластеризованный: определяет физический порядок хранения данных

Что мы сегодня узнали?

Что мы сегодня узнали?

1. Узнали, чем различаются View и Materialized View
2. Поняли, на какие группы можно разделить индексы и зачем нужен каждый из них

Что будет на следующей лекции?

1. Транзакции
2. ACID
3. Уровни изоляции
4. Аномалии

Спасибо за внимание!

