- В чем отличие первичного ключа и уникального индекса?
    - Первичный ключ уже автоматически создает уникальный индекс
    - Уникальный индекс можно назначить не только уникальным и первичным столбцам
    - уникальный индекс может содержать NULL
- В каких случаях имеет смысл создавать индексы? Какие колонки следует включать в индекс и почему?
    - Если часто происходит поиск с условиями. Колонки которые мы чаще всего пишем в условии надо заметить что составные индексы менее эффективны поскольку (x,y) нельзя применить к y, а к x он применяется медленнее чем одинарный. Итак есть часто используем условие x сравнение AND y сравнение то лучше y.
    - Также это имеет смысл создавать когда данных много. Поскольку иногда простой последовательный выбор с проверкой условия может быть эффективней
    - Там ниже еще три преимущества частичных индексов
- Какие существуют способы внутренней организации индексов?
    - составные
    - по выражениям
    - уникальные
    - частичные
    - ---
    - B-дерево
    - gist
    - hash
    - SP-GiST
    - GIN
    - BRIN-индексы

- Рассказать о проблеме фрагментации индексов.
    - https://postgrespro.ru/docs/postgrespro/9.6/sql-reindex
- Как бороться с фрагментацией?
    - Использовать REINDEX
- Имеет ли значение порядок указания колонок при создании индекса?
    - Да. например индекс (x,y) может быть применим (хотя и менее эффективно что (x)) к условию только для x. К условию по y не может.
- В чем разница между Index Scan и Index Seek?
- Да сначала надо указывать те которых больше селективност то етсь которые больше отсеивают потому что поиск идет последовательно
    - https://www.sql.ru/articles/mssql/2006/102501scansvsseeks.shtml
    - Если смотреть на рисунки на плане исполнения, то scan - стрелка сквозь дерево индекса, seek - по листьям.
    - BOL:
    - The Index Scan logical and physical operator retrieves all rows from the nonclustered index specified in the Argument column.

    - The Index Seek logical and physical operator uses the seeking ability of indexes to retrieve rows from a nonclustered index.
- В чем разница между секционированием и наследованием?
    - при секционировании происходит урезанное наслоедорвание. в дочерних таблицах нельзя менять столбцы удалять добавлять там
- Зачем нужен ANALYZE?
    - Чтобы узнать актуальную статистику о проведённом запросе (укажет секунды выполнения. Причём реально выполнится а не предположит)
- Могут ли индексы ухудшить производительность? Если да, то продемонстрировать это.
    - Да могут. Если мало данных например. Но зачастую это сложно ведь планировщик просто перестанет его использовать.
- На что влияет порядок сортировки (ASC\DESC) при создании индекса? Продемонстрировать это.
    - По умолчанию элементы B-дерева хранятся в порядке возрастания, при этом значения NULL идут в конце. Это означает, что при прямом сканировании индекса по столбцу x порядок оказывается соответствующим указанию ORDER BY x (или точнее, ORDER BY x ASC NULLS LAST). Индекс также может сканироваться в обратную сторону, и тогда порядок соответствует указанию ORDER BY x DESC (или точнее, ORDER BY x DESC NULLS FIRST, так как для ORDER BY DESC подразумевается NULLS FIRST).
    - У вас может возникнуть вопрос, зачем нужны все четыре варианта при создании индексов, когда и два варианта с учётом обратного просмотра покрывают все виды ORDER BY. Для индексов по одному столбцу это и в самом деле излишне, но для индексов по многим столбцам это может быть полезно. Рассмотрим индекс по двум столбцам (x, y): он может удовлетворять указанию ORDER BY x, y при прямом сканировании или ORDER BY x DESC, y DESC при обратном. Но вполне возможно, что приложение будет часто выполнять ORDER BY x ASC, y DESC. В этом случае получить такую сортировку от простого индекса нельзя, но можно получить подходящий индекс, определив его как (x ASC, y DESC) или (x DESC, y ASC).
    - https://postgrespro.ru/docs/postgrespro/11/indexes-ordering
- Продемонстрировать полезность индекса по выражению.
    - индексы по выражениям можно использовать ещё и для обеспечения ограничений, которые нельзя записать как простые ограничения уникальности.
    - И для индексирования полнотекстового поиска
- Продемонстрировать полезность частичного индекса.
    - Частичные индексы могут быть полезны, во-первых, тем, что позволяют избежать индексирования распространённых значений. Так как при поиске распространенного значения (такого, которое содержится в значительном проценте всех строк) индекс всё равно не будет использоваться, хранить эти строки в индексе нет смысла. Исключив их из индекса, можно уменьшить

его размер, а значит и ускорить запросы, использующие этот индекс. Это также может ускорить операции изменения данных в таблице, так как индекс будет обновляться не всегда. Возможное применение этой идеи проиллюстрировано в Примере 11.1.
    - Во-вторых, частичные индексы могут быть полезны тем, что позволяют исключить из индекса значения, которые обычно не представляют интереса; это проиллюстрировано в Примере 11.2. При этом вы получаете те же преимущества, что и в предыдущем случае, но система не сможет извлечь «неинтересные» значения по этому индексу, даже если сканирование индекса может быть эффективным. Очевидно, настройка частичных индексов в таких случаях требует тщательного анализа и тестирования.
    - Третье возможное применение частичных индексов вообще не связано с использованием индекса в запросах. Идея заключается в том, чтобы создать уникальный индекс по подмножеству строк таблицы, как в Примере 11.3. Это обеспечит уникальность среди строк, удовлетворяющих условию предиката, но никак не будет ограничивать остальные.
- Отличие первичного ключа и уникального индекса.
    - Когда для таблицы определяется ограничение уникальности или первичный ключ, Postgres Pro автоматически создаёт уникальный индекс по всем столбцам, составляющим это ограничение или первичный ключ (индекс может быть составным). Такой индекс и является механизмом, который обеспечивает выполнение ограничения.
    - Можно использовать null
    - При этом значения NULL считаются не равными друг другу.
- Спрашивали про то, зачем по уникальному(не первичному) ключу постгрес строит свой индекс
    - Это и есть механизм реализации уникальность в postgres (т.е. через уникальный индекс)
- Кластеризованные индексы
    - Кластеризованный индекс хранит реальные строки данных в листьях индекса. Возвращаясь к предыдущему примеру, это означает что строка данных, связанная со значение ключа, равного 123 будет храниться в самом индексе. Важной характеристикой кластеризованного индекса является то, что все значения отсортированы в определенном порядке либо возрастания, либо убывания. Таким образом, таблица или представление может иметь только один кластеризованный индекс. В дополнение следует отметить, что данные в таблице хранятся в отсортированном виде только в случае если создан кластеризованный индекс у этой таблицы.

Таблица не имеющая кластеризованного индекса называется кучей.
В отличие от кластеризованного индекса, листья некластеризованного индекса содержат только те столбцы (*ключевые*), по которым определен данный индекс, а также содержит указатель на строки с реальными данными в таблице. Это означает, что системе подзапросов необходима дополнительная операция для обнаружения и получения требуемых данных.

- Как B дерево устроено
    - Корень с метаданными
    - Ниже главный корень от которого может быть много потомков.
    - Потомки между собой ходить могут
    - В листьях ссылка на строки
    - Сбалансированное дерево (т.е. ходим до листьев за одно и то же время)
- Где в планировщике видна польза от секционирования
    - В параллельности.