# Лабораторная работа №3

## По предмету СУБД Вариант 10 Продажа подержанных автомобилей

Валентюкевич Олеся   
группа 3312

Оглавление

[Контрольные вопросы для допуска к работе 3](#_Toc212637563)

[1. Что такое настройка производительности SQL? 3](#_Toc212637564)

[2. Что такое настройка производительности базы данных? 3](#_Toc212637565)

[3. На чем сосредоточена большая часть действий по настройке производительности и почему она существует? 3](#_Toc212637566)

[4. Что такое статистика базы данных и почему она важна? 3](#_Toc212637567)

[5. Какие статистические измерения базы данных типичны для таблиц, индексов и ресурсов? 3](#_Toc212637568)

[6. Чем отличается обработка операторов SQL DDL от обработки, требуемой операторами DML? 4](#_Toc212637569)

[7. Если индексы так важны, почему бы не проиндексировать каждый столбец в каждой таблице? 4](#_Toc212637570)

[8. В чем разница между оптимизатором на основе правил и оптимизатором на основе затрат? 4](#_Toc212637571)

[Задание для выполнения 5](#_Toc212637572)

[1. Определить столбцы для создания кластерных индексов 5](#_Toc212637573)

[2. Определить столбцы для создания некластерных индексов. 6](#_Toc212637574)

[3. Сравнить результаты выполнения запросов до и после создания индексов 7](#_Toc212637575)

# Контрольные вопросы для допуска к работе

## 1. Что такое настройка производительности SQL?

Настройка производительности SQL - это процесс оптимизации SQL-запросов для уменьшения времени выполнения и потребления ресурсов, который включает: Рефактор запросов, отпимизатор запросов и подзапросов, использование.

## 2. Что такое настройка производительности базы данных?

Настройка производительности БД - более широкое понятие, включающее проектирование схемы базы данных, создание и оптимизацию индексов, настройку параметров сервера (память, кэши, параллелизм) и т.д

## 3. На чем сосредоточена большая часть действий по настройке производительности и почему она существует?

Основной фокус - на индексах и запросах, потому что: 80% проблем производительности связаны с неправильной индексацией, плохие запросы могут потреблять в 100-1000 раз больше ресурсов, индексы напрямую влияют на скорость поиска и соединения данных

Причина существования: разрыв между теоретическим проектированием и реальными паттернами доступа к данным.

## 4. Что такое статистика базы данных и почему она важна?

Статистика БД - это метаданные о распределении данных в таблицах: Количество строк, плотность данных, распределение значений в столбцах (гистограммы), количество уникальных значений

Важность: Оптимизатор использует статистику для выбора оптимального плана выполнения

## 5. Какие статистические измерения базы данных типичны для таблиц, индексов и ресурсов?

Для таблиц:

* num\_rows -количество строк
* avg\_row\_len -средняя длина строки
* blocks -количество блоков данных
* empty\_blocks -пустые блоки

Для индексов:

* leaf\_blocks -количество листовых блоков
* distinct\_keys -количество уникальных ключей
* avg\_leaf\_blocks\_per\_key -среднее количество листовых блоков на ключ
* clustering\_factor -фактор кластеризации (как хорошо данные отсортированы)

Для ресурсов:

* buffer cache hit ratio -процент попаданий в кэш
* memory grants -выделения памяти
* lock waits -ожидания блокировок
* I/O statistics -статистика ввода/вывода

## 6. Чем отличается обработка операторов SQL DDL от обработки, требуемой операторами DML?

DDL (Data Definition Language):CREATE, ALTER, DROP - определение структуры, выполняются относительно быстро (метаданные)

DML (Data Manipulation Language): SELECT, INSERT, UPDATE, DELETE -манипуляция данными, производительность зависит от объема данных и индексов

## 7. Если индексы так важны, почему бы не проиндексировать каждый столбец в каждой таблице?

Проблемы избыточной индексации: Замедление операций DML -каждый INSERT/UPDATE/DELETE должен обновлять все индексы  
Увеличение размера БД -индексы занимают значительное место (30-50% от данных)  
Фрагментация -частое обновление приводит к фрагментации индексов

Правило: создавать индексы только для столбцов, используемых в WHERE, JOIN, ORDER BY.

## 8. В чем разница между оптимизатором на основе правил и оптимизатором на основе затрат?

RBO (Rule-Based Optimizer): Использует предопределенные правила (например: "индекс лучше полного сканирования"), не учитывает объем данных и распределение значений, прост в реализации, но часто неоптимален

CBO (Cost-Based Optimizer): Оценивает "стоимость" каждого возможного плана в единицах CPU и I/O, использует статистику для реалистичной оценки, выбирает план с наименьшей расчетной стоимостью

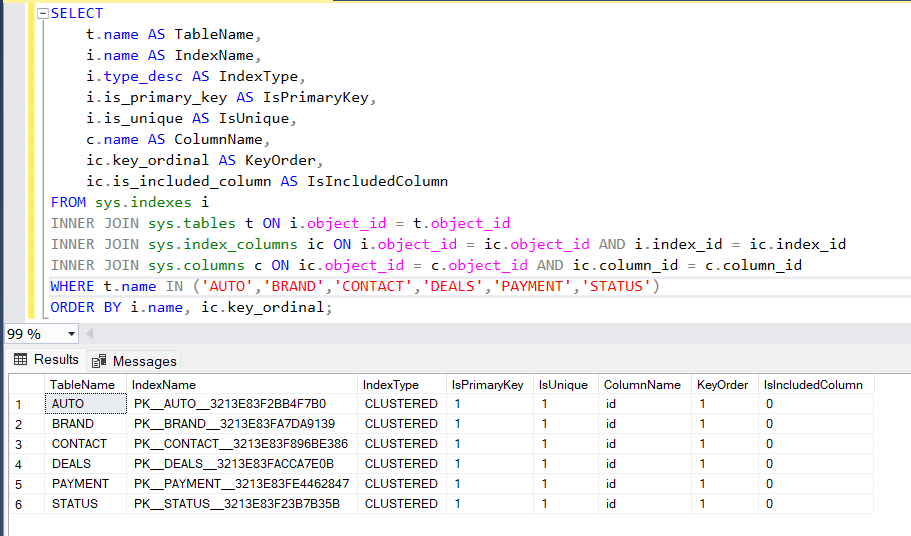
# Задание для выполнения

1. Определить столбцы для создания кластерных индексов. Обосновать выбор. Создать кластерные индексы для каждой таблицы в базе данных для заданной предметной области.

1. **Таблица AUTO**:  
   Обоснование: ID - первичный ключ, наиболее часто используется в JOIN и WHERE. Автомобили обычно запрашиваются по конкретному ID.
2. **Таблица BRAND**Обоснование: ID - первичный ключ, используется в связях с таблицей AUTO. Бренды редко меняются, последовательный ID обеспечивает хорошую производительность.
3. **Таблица CONTACT**:  
   Обоснование: ID - первичный ключ, используется в связях. Контакты обычно ищутся по ID при связывании с платежами.
4. **Таблица DEALS**:  
   Обоснование: Для таблицы сделок наиболее логично кластеризовать по ID , так как чаще всего требуется:

* Отчеты по периодам
* Анализ продаж за месяц/квартал/год
* Сортировка сделок по хронологии

1. **Таблица PAYMENT**Обоснование: ID - первичный ключ, обеспечивает уникальность записей о платежах.
2. **Таблица STATUS**:  
   Обоснование: ID - первичный ключ, таблица-справочник с небольшим количеством записей.



2. Определить столбцы для создания некластерных индексов. Обосновать выбор. Создать некластерные индексы.

1. **Таблица AUTO**:

* **Составной индекс** на BRANDNAMEKEY + STATUS - самые частые фильтры в автосалонах
* **INCLUDE столбцы** PRICE, MILEAGE - избегаем key lookups в результатах поиска
* **Отдельный индекс по цене** - для фильтрации по ценовому диапазону

1. **Таблица BRAND:**

* **Поиск по названию бренда** - частый сценарий
* **INCLUDE иконки** - данные сразу в индексе, не нужно обращаться к таблице

1. **Таблица CONTACT**:

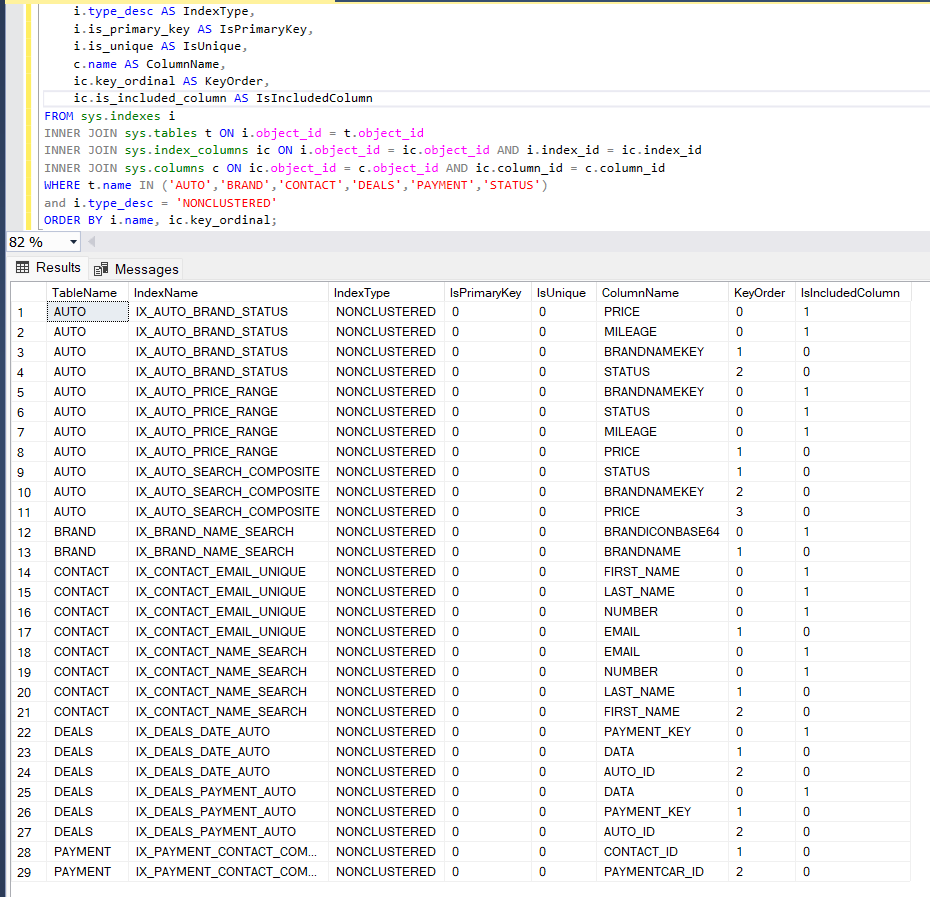
* **Email уникален** - быстрый поиск клиентов по email
* **Составной индекс по ФИО** - поиск по имени и фамилии
* **INCLUDE контактных данных** - покрывающие индексы для частых запросов

1. **Таблица DEALS**:

* **Индекс по дате + авто** - для отчетов "продажи по датам с привязкой к авто"
* **Индекс по платежу + авто** - отслеживание статусов сделок
* **Отдельный индекс по дате** - для быстрых временных отчетов

1. **Таблица PAYMENT**

* **Составной индекс** - поиск платежей по клиенту и карте
* **Индекс по карте** - анализ платежных инструментов



## 3. Сравнить результаты выполнения запросов до и после создания индексов

Серьезного буста в скорости замечено не было, т.к объём данных не является критичным для сборки данных. Отчет **ДО** создания кластерных и не кластерных индексов и **ПОСЛЕ**