**Системы управления базами данных**

**1. Полная структура SQL запроса**

**SQL (Structured Query Language)** запрос имеет определенную структуру, которая позволяет извлекать и манипулировать данными в реляционных базах данных. Полная структура оператора SELECT включает следующие компоненты в порядке их выполнения:

SELECT [DISTINCT | ALL] <список\_выбора>  
FROM <таблица\_или\_представление> [AS <псевдоним>]  
[JOIN <таблица> ON <условие\_соединения>]  
[WHERE <условие\_фильтрации>]  
[GROUP BY <список\_группировки>]  
[HAVING <условие\_для\_групп>]  
[ORDER BY <список\_сортировки> [ASC | DESC]]  
[LIMIT <количество\_строк> [OFFSET <смещение>]]

Каждая часть запроса выполняет определенную функцию:

**SELECT** — определяет, какие столбцы должны быть включены в результат

**FROM** — указывает таблицу или представление, из которого извлекаются данные

**JOIN** — соединяет две или более таблиц по определенному условию

**WHERE** — фильтрует строки по заданному условию

**GROUP BY** — группирует строки с одинаковыми значениями

**HAVING** — фильтрует группы по заданному условию

**ORDER BY** — сортирует результат по одному или нескольким столбцам

**LIMIT/OFFSET** — ограничивает количество возвращаемых строк и их смещение

**2. Индексы, их разновидности и принципы работы**

**Индексы** — это специальные структуры данных, которые улучшают производительность запросов за счет ускорения поиска, сортировки и группировки данных.

**Типы индексов:**

**Кластерные индексы** — определяют физический порядок хранения данных в таблице. Таблица может иметь только один кластерный индекс.

**Некластерные индексы** — создают отдельную структуру, которая указывает на строки данных. Таблица может иметь множество некластерных индексов.

**Уникальные индексы** — обеспечивают уникальность значений в индексированных столбцах.

**Составные индексы** — индексы по нескольким столбцам таблицы.

**Полнотекстовые индексы** — позволяют эффективно выполнять поиск по текстовым данным.

**Пространственные индексы** — оптимизированы для работы с географическими и пространственными данными.

**Индексы на основе хеширования** — используют хеш-функции для быстрого доступа к данным.

**B-Tree индексы** — наиболее распространенный тип индексов, использующий сбалансированное дерево.

**Принципы работы:**

Индексы работают по принципу упорядоченной структуры данных, которая содержит значения индексируемых столбцов и указатели на соответствующие строки таблицы. При выполнении запроса система сначала обращается к индексу, находит нужные указатели, а затем по ним извлекает данные из таблицы.

При создании индекса необходимо учитывать: - Селективность индекса (соотношение уникальных значений к общему количеству) - Частоту операций чтения и записи - Размер индекса и его влияние на производительность системы

**3. Триггеры, их особенности и отличия от процедур и функций**

**Триггеры** — это специальные хранимые процедуры, которые автоматически запускаются в ответ на определенные события в базе данных.

Особенности триггеров:

**Автоматическое выполнение** — триггеры срабатывают автоматически при наступлении определенного события.

**Связь с событиями** — триггеры привязаны к событиям DML (INSERT, UPDATE, DELETE) или DDL (CREATE, ALTER, DROP).

**Время срабатывания** — триггеры могут выполняться до (BEFORE) или после (AFTER) события.

**Уровень срабатывания** — триггеры могут выполняться для каждой строки (FOR EACH ROW) или один раз для всей операции (FOR EACH STATEMENT).

**Доступ к старым и новым значениям** — в триггерах уровня строки можно обращаться к старым (OLD) и новым (NEW) значениям данных.

**Ограничения целостности** — триггеры могут использоваться для реализации сложных правил целостности данных.

Отличия от процедур и функций:

| Характеристика | Триггеры | Хранимые процедуы | Функции |
| --- | --- | --- | --- |
| Вызов | Автоматический при событии | Явный вызов | Явный вызов |
| Параметры | Используют OLD/NEW | Могут иметь входные и выходные параметы | Имеют входные параметы |
| Возвращаемое значение | Нет | Может возвращать набор значений | Всегда возвращает одно значение |
| Использование в запросах | Нельзя | Нельзя | Можно |
| Транзакции | Нельзя управлять | Можно управлять | Обычно нельзя управлять |
| Вложенность | Ограничена | Высокая | Высокая |

**4. Работа с данными через курсоры**

**Курсор** — это объект базы данных, который позволяет обрабатывать результаты запроса построчно, предоставляя механизм итерации по набору данных.

Основные операции с курсорами:

**Объявление курсора** — определение запроса, результаты которого будут обрабатываться:

DECLARE cursor\_name CURSOR FOR SELECT column1, column2 FROM table;

**Открытие курсора** — выполнение запроса и подготовка к обработке результатов:

OPEN cursor\_name;

**Получение данных** — чтение текущей строки и перемещение к следующей:

FETCH cursor\_name INTO variable1, variable2;

**Закрытие курсора** — освобождение ресурсов:

CLOSE cursor\_name;

**Освобождение курсора** — удаление определения курсора:

DEALLOCATE cursor\_name;

Типы курсоров:

**Статические** — создают копию данных при открытии курсора, изменения в исходной таблице не видны

**Динамические** — отражают все изменения, происходящие с данными во время работы курсора

**Только для чтения** — не позволяют изменять данные через курсор

**Прокручиваемые** — позволяют перемещаться по результатам в произвольном порядке

**Непрокручиваемые** — позволяют перемещаться только вперед

**Применение курсоров:**

Курсоры применяются в ситуациях, когда необходима построчная обработка данных, особенно: - При выполнении сложных вычислений для каждой строки - При необходимости принятия решений на основе значений строки - При обработке иерархических данных - При миграции или преобразовании больших объемов данных

**5. Транзакции и их основные свойства**

**Транзакция** — логическая единица работы, включающая одну или несколько операций с базой данных, которая должна быть выполнена полностью или не выполнена вообще.

Основные свойства транзакций (ACID):

**Атомарность (Atomicity)** — транзакция выполняется полностью или не выполняется вообще. Если какая-либо операция внутри транзакции не может быть выполнена, вся транзакция откатывается.

**Согласованность (Consistency)** — транзакция переводит базу данных из одного согласованного состояния в другое, сохраняя все правила целостности.

**Изолированность (Isolation)** — выполнение одной транзакции не влияет на выполнение других транзакций. Результаты незавершенной транзакции не видны другим транзакциям.

**Долговечность (Durability)** — после завершения транзакции, её результаты сохраняются в базе данных даже в случае сбоя системы.

Управление транзакциями:

**BEGIN TRANSACTION** — начало транзакции

**COMMIT** — успешное завершение транзакции, сохранение изменений

**ROLLBACK** — отмена транзакции, отмена всех изменений

**SAVEPOINT** — установка точки сохранения внутри транзакции

**ROLLBACK TO SAVEPOINT** — откат к определенной точке сохранения

**6. Уровни изоляции транзакций**

Уровни изоляции транзакций определяют, насколько одна транзакция может видеть изменения, выполняемые другими транзакциями. Стандарт SQL определяет **четыре уровня изоляции**:

**1. READ UNCOMMITTED** (Чтение незафиксированных данных)

Транзакция может видеть незафиксированные изменения, сделанные другими транзакциями

Наименее ограничительный уровень

Возможны проблемы: “грязное чтение”, “неповторяющееся чтение”, “фантомное чтение”

**2. READ COMMITTED** (Чтение зафиксированных данных)

Транзакция видит только зафиксированные изменения, сделанные другими транзакциями

Предотвращает “грязное чтение”

Возможны проблемы: “неповторяющееся чтение”, “фантомное чтение”

**3. REPEATABLE READ** (Повторяемое чтение)

Транзакция гарантирует, что если она прочитала строку однажды, при последующих чтениях она увидит те же данные

Предотвращает “грязное чтение” и “неповторяющееся чтение”

Возможны проблемы: “фантомное чтение”

**4. SERIALIZABLE** (Сериализуемость)

Наиболее строгий уровень изоляции

Транзакции выполняются так, как если бы они выполнялись последовательно

Предотвращает все проблемы параллельного выполнения транзакций

Может существенно снижать производительность системы

Проблемы параллельного выполнения транзакций:

**Грязное чтение (Dirty Read)** — транзакция читает данные, измененные другой незавершенной транзакцией

**Неповторяющееся чтение (Non-repeatable Read)** — транзакция повторно читает те же данные и обнаруживает, что они были изменены другой транзакцией

**Фантомное чтение (Phantom Read)** — транзакция повторно выполняет запрос, который возвращает набор строк, удовлетворяющих условию, и обнаруживает, что набор строк изменился из-за другой транзакции

**Потерянное обновление (Lost Update)** — две транзакции читают и обновляют одни и те же данные, в результате чего одно из обновлений теряется

Выбор уровня изоляции зависит от требований приложения к целостности данных и производительности. Более высокие уровни изоляции обеспечивают лучшую защиту данных, но могут снижать производительность из-за блокировок и возможных конфликтов между транзакциями.