**Конспект по SQL**

**1. Типы баз данных**

* **Реляционные (SQL)**: Таблицы со строгой схемой (MySQL, PostgreSQL, **MSSQL**).
* **NoSQL**: Документные (MongoDB), ключ-значение (Redis), графовые (Neo4j), колоночные (Cassandra).
* **Иерархические**: Древовидная структура (IBM IMS).
* **Сетевые**: Сложные связи между записями (CODASYL).

**Отличия**: Реляционные БД строги к схеме, NoSQL гибкие, но менее строгие в ACID (**атомарность** – гарантирует, что каждая транзакция будет выполнена полностью либо не будет выполнена совсем; **согласованность** – каждая успешная транзакция по определению фиксирует только допустимые результаты; **изолированность** – во время выполнения транзакции параллельные транзакции не должны оказывать влияния на ее результат; **надежность** – уверенность в том, что если транзакция выполнена, то сделанные изменения не будут отменены из-за сбоя).

**2. Основные операторы**

* **SELECT**: Извлечение данных.



* **INSERT**: Добавление строк.



* **UPDATE**: Обновление данных.



* **DELETE**: Удаление строк.



**3. Соединения таблиц (JOIN)**

* **INNER JOIN**: возвращает совпадающие строки.
* **LEFT JOIN**: Все строки из левой таблицы + совпадения справа.
* **RIGHT JOIN**: Все строки из правой таблицы + совпадения слева.
* **FULL JOIN**: Все строки из обеих таблиц.
* **CROSS JOIN**: Декартово произведение строк.
* **SELF JOIN**: соединение таблицы с самой собой – используется когда данные в одной таблице связаны между собой (например, иерархия сотрудников)
* **SEMI JOIN**: возвращает строки из первой таблицы только если они имеют совпадения во второй таблице; в отличие от обычного JOIN строки не объединяются, а только проверяется наличие связи, реализуется через EXISTS, IN, INTERSECT
* **UNION**: объединяет ВЕРТИКАЛЬНО результаты двух или более запросов, удаляя дубликаты строк.
* **UNION ALL**: объединяет ВЕРТИКАЛЬНО результаты сохраняя все строки, включая дубликаты.

**4. Типы данных в MSSQL**

* **Числовые**: INT, DECIMAL (10,2) – 10: точность | 2: масштаб, FLOAT.
* **Строковые**: VARCHAR (255) – длина в символах, TEXT, NVARCHAR (Unicode).
* **Дата/время**: DATE, DATETIME, TIME.
* **Специальные**: UNIQUEIDENTIFIER (GUID), XML, JSON.

**5. Агрегатные и оконные функции**

* **Агрегатные**: SUM(), AVG(), COUNT(), MIN(), MAX().



* **Оконные**: Вычисления над окном строк (не сворачивают результат).

A screen shot of a computer

AI-generated content may be incorrect.

**Основные категории оконных функций**

1. **Ранжирующие**: ROW\_NUMBER() – уникальный номер строки, RANK() – ранг с пропусками (два одинаковый значения -> одинаковый ранг, но следующий ранг будет 3), DENSE\_RANK() – ранг без пропусков, NTILE() – разделение на группы.
2. **Смещения**: LEAD() – значение из следующей строки, LAG() – значение из предыдущей строки.
3. **Агрегатные**: SUM(), AVG(), COUNT(), MIN(), MAX() (используются с OVER()), например, для расчета средней зарплаты по отделу или накопительной суммы.A screen shot of a computer code

   AI-generated content may be incorrect.
4. **Доступа к значениям**: FIRST\_VALUE() – первое значение в окне, LAST\_VALUE() – последнее значение в окне.

**6. OLTP vs OLAP**

* **OLTP** (Online Transaction Processing):
  + Короткие транзакции (например, банковские операции, системы резерваций, цепочки поставок), упор на скорость записи, использование для обработки в реальном времени, низкая задержка.
  + Нормализованные схемы.
* **OLAP** (Online Analytical Processing):
  + Анализ больших данных (например, отчеты), упор на комплексную аналитику, объемные транзакции, но меньшие по количеству, оптимизированы на чтение данных, высокая задержка данных.
  + Денормализованные схемы (звезда, снежинка).

**7. DML, DDL, TCL**

* **DML** (Data Manipulation Language): SELECT, INSERT, UPDATE, DELETE.
* **DDL** (Data Definition Language): CREATE TABLE, ALTER, DROP, TRUNCATE.
* **TCL** (Transaction Control Language): COMMIT, ROLLBACK, SAVEPOINT.

**8. Условные выражения (CASE)**

A screenshot of a computer code

AI-generated content may be incorrect.

**9. CTE (Common Table Expressions)**

* Временный результат для сложных запросов.

A black background with white text

AI-generated content may be incorrect.

* **Рекурсивные CTE** (для иерархических данных).

**10. Оператор WHERE**

* **IS NULL**: Проверка на NULL.



* **BETWEEN**: Диапазон значений.



* **IN**: Проверка вхождения в список.



* **LIKE**: Поиск по шаблону.



* **Регулярные выражения**: Используется PATINDEX или LIKE с wildcards (%, \_, [a-z]).

**11. GROUP BY и HAVING**

* **GROUP BY**: Группировка строк для агрегации.
* **HAVING**: Фильтрация групп (после агрегации).

A black background with white text

AI-generated content may be incorrect.

**Отличие**: WHERE фильтрует строки до группировки, HAVING — после. Также HAVING возможно использовать с агрегатными функциями.

**12. ORDER BY**

Сортировка результатов:



**13. Представления (Views)**

Виртуальные таблицы для упрощения запросов:



**14. Индексы**

**Индексирование баз данных** — это техника, повышающая скорость и эффективность запросов к базе данных. Она создаёт отдельную структуру данных, сопоставляющую значения в одном или нескольких столбцах таблицы с соответствующими местоположениями на физическом накопителе, что позволяет базе данных быстро находить строки по конкретному запросу без необходимости сканирования всей таблицы.

* **Кластерный**: упорядочивает физическое хранение данных (только 1 на таблицу).
* **Некластерный**: отдельная структура для быстрого поиска.



**Рекомендации**

* Используйте индексы для часто запрашиваемых столбцов, но избегайте избыточности.
* Для анализа больших данных предпочитайте OLAP-системы (например, SQL Server Analysis Services).
* В MSSQL для ограничения выборки используйте TOP, а не LIMIT (как в MySQL):



**Пример регулярного выражения** (через LIKE):



**Порядок исполнения SQL-запроса:**

1. FROM: Определение источника данных.
2. JOIN: Присоединение таблиц.
3. WHERE: Фильтрация строк на основе заданных условий.
4. GROUP BY: Группировка строк по одному или нескольким столбцам.
5. HAVING: Фильтрация групп, созданных с помощью GROUP BY.
6. SELECT: Выбор столбцов и выражений для вывода.
7. ORDER BY: Сортировка результата.
8. LIMIT / OFFSET: Ограничение количества возвращаемых строк.

**Нормализация таблиц.**

**Нормализация таблиц в SQL —** это процесс реорганизации структуры базы данных таким образом, чтобы минимизировать избыточность данных и повысить целостность данных.

Основная цель нормализации заключается в уменьшении дублирования данных и обеспечении согласованности между различными элементами базы данных. Это помогает избежать аномалий обновления, вставки и удаления, возникающих вследствие излишней повторяемости данных.

**Первая нормальная форма (1NF)**

Таблица находится в первой нормальной форме, если каждое её значение является атомарным (неделимым). То есть, каждый столбец содержит только одно значение, а не список значений или массив.

**Вторая нормальная форма (2NF)**

Для достижения второй нормальной формы таблица должна находиться в первой нормальной форме, и все неключевые атрибуты должны зависеть от всего первичного ключа, а не его части.

**Третья нормальная форма (3NF)**

Третья нормальная форма требует устранения транзитивных зависимостей. Неключевой атрибут не должен зависеть от другого неключевого атрибута.

**Нормальная форма Бойса-Кодда (BCNF)**

* Каждый детерминант (атрибут, от которого зависит другой) **должен быть потенциальным ключом**.

A screenshot of a black screen

AI-generated content may be incorrect.

**Когда нарушать нормализацию?**  
➞ В аналитических базах (OLAP), где важнее скорость, чем целостность.  
➞ В кэшируемых таблицах (например, TotalOrders для быстрого доступа).