МИНОБРНАУКИ РОССИИ

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ

ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)

Кафедра САПР

ОТВЕТЫ НА ВОПРОСЫ

по дисциплине «Базы данных»

Студент гр. 1335 Максимов Ю.Е.

Преподаватель Новакова Н.Е.

Санкт-Петербург

2024

Вопрос 24: Типы данных в SQL. Символьные, числовые, бинарные типы данных, даты и времени и т.п. Предикат LIKE.

Типы данных в SQL

SQL поддерживает различные типы данных для хранения информации в таблицах базы данных. Они делятся на несколько основных категорий:

1. Символьные (текстовые) типы данных

Используются для хранения строк текста, таких как имена, адреса, описания и т. д.

Тип	Описание
CHAR(n)	Хранит строку фиксированной длины n (до 255 символов). Если строка короче, добавляются пробелы.
VARCHAR(n)	Хранит строку переменной длины до n символов. Экономит место, так как пробелы не добавляются.
TEXT	Используется для хранения длинных текстовых данных (до 2 ГБ в MySQL, 2^31-1 байт в PostgreSQL).
CLOB (Character Large Object)	Большие текстовые данные, аналог TEXT в некоторых СУБД.

Различия CHAR и VARCHAR:

- CHAR(n) занимает фиксированное место в памяти. Подходит для хранения строк одинаковой длины (например, кодов стран, ИНН).

- VARCHAR(n) экономит место, так как использует только необходимый объем памяти.

2. Числовые типы данных

Используются для хранения целых чисел, чисел с плавающей запятой и точных значений.

Целочисленные типы

Тип	Диапазон значений	Описание
TINYINT	-128 до 127 (SIGNED) / 0 до 255 (UNSIGNED)	1 байт, используется для хранения небольших чисел
SMALLINT	-32,768 до 32,767 / 0 до 65,535	2 байта
MEDIUMINT	-8,388,608 до 8,388,607	3 байта
INT (INTEGER)	-2,147,483,648 до 2,147,483,647	4 байта, стандартный тип для целых чисел
BIGINT	±9,223,372,036,854,775,807	8 байт, используется для больших чисел

SIGNED vs. UNSIGNED:

- SIGNED (со знаком) включает отрицательные числа.
- UNSIGNED (без знака) увеличивает диапазон за счет отказа от отрицательных чисел.

Числа с плавающей запятой

Тип	Диапазон значений	Размер	Описание
FLOAT(m, d)	$\pm 1.5 \times 10^{-45}$ до $\pm 3.4 \times 10^{38}$	4 байта	Приблизительная точность, быстрое вычисление
DOUBLE (FLOAT8, REAL)	$\pm 5.0 \times 10^{-324}$ до $\pm 1.8 \times 10^{308}$	8 байт	Более высокая точность
DECIMAL(m, d) / NUMERIC(m, d)	Точное значение	До 65 знаков	Используется для финансовых расчетов

DECIMAL vs FLOAT/DOUBLE:

- DECIMAL сохраняет точность (важно для денег).
- FLOAT и DOUBLE допускают небольшие ошибки округления.

3. Дата и время

Хранят временные значения, даты и интервалы.

Тип	Формат	Описание
DATE	YYYY-MM-DD	Дата без времени (0001-01-01 – 9999-12-31)
TIME	HH:MI:SS	Время без даты (-838:59:59 до 838:59:59)
DATETIME	YYYY-MM-DD HH:MI:SS	Дата + время (0001-01-01 00:00:00 – 9999-12-31 23:59:59)
TIMESTAMP	YYYY-MM-DD HH:MI:SS	Как DATETIME, но учитывает часовой пояс

Тип	Формат	Описание
INTERVAL	Временной интервал	Разница между датами (например, INTERVAL '2 years 3 months')

Различия TIMESTAMP и DATETIME:

- TIMESTAMP автоматически обновляется при изменении записи, полезен для логирования.
- DATETIME сохраняет данные без учета часового пояса.

4. Булевы (логические) типы

Используются для хранения значений TRUE и FALSE.

Тип	Описание
BOOLEAN	В MySQL хранится как TINYINT(1) (0 = FALSE, 1 = TRUE). В PostgreSQL — полноценный тип BOOLEAN.

5. Бинарные типы данных

Хранят двоичные (не текстовые) данные, такие как изображения, видео, зашифрованные пароли.

Тип	Описание
BLOB (Binary Large Object)	Двоичные данные до 2 ГБ
TINYBLOB, MEDIUMBLOB, LONGBLOB	Варианты BLOB с разным размером
BINARY(n)	Фиксированная длина двоичных данных

Тип	Описание
VARBINARY(n)	Переменная длина двоичных данных

Различие BLOB и TEXT:

- BLOB хранит бинарные данные.
- TEXT хранит строковые данные.

6. Пространственные (географические) типы

Используются для хранения географических данных, например, точек, линий, полигонов.

Тип	Описание
GEOMETRY	Хранит любую геометрическую фигуру
POINT	Координата (X, Y)
LINESTRING	Линия из точек
POLYGON	Полигон

Используется в ГИС (геоинформационных системах).

Оператор LIKE

LIKE используется для поиска строк по шаблону с помощью **подстановочных символов**.

Символ	Значение
%	Любое количество символов
_	Один любой символ

Примеры использования LIKE:

sql

```
-- Найти всех пользователей, чьи имена начинаются с "A"
SELECT * FROM users WHERE name LIKE 'A%';

-- Найти все email с ".com" в конце
SELECT * FROM users WHERE email LIKE '%.com';

-- Найти слова с "abc" в середине
SELECT * FROM words WHERE word LIKE '%abc%';

-- Найти телефоны, у которых третий символ — 5
SELECT * FROM contacts WHERE phone LIKE '__5%';
```

LIKE vs =

- LIKE ищет **приблизительное** совпадение.
- = ищет **точное** совпадение.

Оптимизация:

- Использование LIKE '%...%' может быть **медленным**.

- В больших базах лучше применять **FULLTEXT-индексы**.

Вывод

SQL предлагает мощный набор типов данных для различных задач:

- **Текст** (VARCHAR, TEXT)
- **Числа** (INT, DECIMAL, FLOAT)
- **Дата и время** (DATETIME, TIMESTAMP)
- **Логика** (BOOLEAN)
- **Бинарные данные** (BLOB)
- **Геоданные** (GEOMETRY)

Оператор LIKE помогает искать текстовые данные по шаблону, но для больших объемов данных лучше использовать индексированные или полнотекстовые поисковые механизмы.

Вопрос 43: Резервное копирование БД. Процесс резервного копирования. Выполнение резервного копирования. Команда BACKUP DATABASE.

Резервное копирование (backup) базы данных (БД) — это процесс создания копии данных, которая может быть использована для восстановления информации в случае сбоя, потери данных или кибератаки. Это важнейшая часть стратегии обеспечения безопасности и целостности данных в любой системе управления базами данных (СУБД).

1. Зачем нужно резервное копирование?

Резервное копирование предотвращает потерю данных в следующих ситуациях:

- **Сбой оборудования** (жесткие диски, серверы, блоки питания)
- **Атаки вирусов и кибератаки** (вирусы-шифровальщики, SQL-инъекции)
- **Ошибки пользователей** (удаление данных, изменение конфигурации)
- **Программные ошибки** (баги в ПО, повреждение файлов БД)
- **Стихийные бедствия** (пожары, наводнения, землетрясения)

Основная цель: минимизировать время простоя и быстро восстановить систему после сбоя.

2. Типы резервного копирования

Существует несколько способов резервного копирования БД:

1) Полное (Full Backup)

- Создается полная копия базы данных.
- Самый надежный, но занимает много места и времени.
- Пример (SQL Server):

sql

```
BACKUP DATABASE my_database TO DISK = 'C:\backups\my_database.bak' WITH FORMAT;
```

- Используется в критически важных системах.

2) Инкрементное (Incremental Backup)

- Копирует только **изменения** с момента последнего **полного** или **инкрементного** бэкапа.
- Требует меньше места и выполняется быстрее.
- Недостаток: для восстановления нужны все предыдущие копии.

Пример стратегии:

1. В воскресенье — **полный бэкап**
2. С понедельника по субботу — **инкрементные бэкапы**

3) Дифференциальное (Differential Backup)

- Сохраняет изменения с момента последнего **полного** бэкапа.
- Быстрее, чем полный, но требует больше места, чем инкрементный.

Пример (SQL Server):

sql

BACKUP DATABASE my_database TO DISK =
'C:\backups\my_database_diff.bak' WITH DIFFERENTIAL;

- В отличие от инкрементного, **всегда ссылается на последний полный бэкап**.

4) Логический (Logical Backup)

- Сохраняет структуру БД и данные в **SQL-скрипт** (.sql).
- Используется для миграции или частичного восстановления.

Пример (MySQL):

bash

mysqldump -u root -p my_database > backup.sql

Пример (PostgreSQL):

bash

pg_dump -U postgres -F c my_database > backup.dump

5) Физическое копирование (Physical Backup)

- Копирует файлы базы данных на диске.
- Используется для восстановления всей системы.

Пример (PostgreSQL):

bash

cp -r /var/lib/postgresql/13/main /backup/

6) Горячее (Hot Backup) и Холодное (Cold Backup)

Тип	Описание
Горячее (Hot Backup)	Копирование **во время работы** базы. Требует поддержки со стороны СУБД (например, MySQL InnoDB, PostgreSQL).
Холодное (Cold Backup)	Копирование при **остановленной** базе. Более безопасный вариант.

3. Процесс резервного копирования

1. Определение стратегии

Перед настройкой резервного копирования нужно ответить на вопросы:

- Как часто делать бэкап? (ежедневно, ежечасно, раз в неделю)
- Где хранить бэкапы? (локальный диск, облако, удаленный сервер)
- Как долго хранить бэкапы? (30 дней, 6 месяцев, 1 год)
- Какие данные критически важны? (вся база или только определенные таблицы)

2. Выбор типа бэкапа

- **Критические системы → Полный + Инкрементный**
- **Частые обновления → Дифференциальный**
- **Минимум места → Инкрементный**
- **Быстрое восстановление → Полный**

3. Настройка автоматического резервного копирования

Можно настроить автоматический бэкап с помощью **планировщика задач (cron, Task Scheduler)**.

Пример для MySQL (cron-job, ежедневный бэкап в 2:00):

bash

```
0 2 * * * mysqldump -u root -p my_database > /backups/my_database_$(date +\%F).sql
```

4. Команда BACKUP DATABASE

В SQL Server команда BACKUP DATABASE используется для резервного копирования.

Синтаксис:

sql

```
BACKUP DATABASE database_name
TO DISK = 'backup_path'
[WITH options]
```

Примеры:

1. Полный бэкап:

sql

```
BACKUP DATABASE my_database
TO DISK = 'C:\backups\my_database_full.bak'
WITH FORMAT, INIT, NAME = 'Full Backup';
```

2. Дифференциальный бэкап:

sql

```
BACKUP DATABASE my_database
```

TO DISK = 'C:\backups\my_database_diff.bak'

WITH DIFFERENTIAL;

3. Инкрементный бэкап (журнал транзакций):

sql

BACKUP LOG my_database

TO DISK = 'C:\backups\my_database_log.trn';

📌 **Важно!**

Инкрементный бэкап доступен только при включенном **полном режиме восстановления** (FULL RECOVERY).

5. Восстановление базы данных (Restore Database)

Бэкап сам по себе не полезен без процедуры восстановления.

Синтаксис:

sql

RESTORE DATABASE database_name

FROM DISK = 'backup_path'

Примеры восстановления

1. Полное восстановление:

sql

RESTORE DATABASE my_database

FROM DISK = 'C:\backups\my_database_full.bak'

WITH REPLACE, RECOVERY;

2. Восстановление дифференциального бэкапа:

sql

```
RESTORE DATABASE my_database
FROM DISK = 'C:\backups\my_database_full.bak'
WITH NORECOVERY;

RESTORE DATABASE my_database
FROM DISK = 'C:\backups\my_database_diff.bak'
WITH RECOVERY;
```

3. Восстановление журнала транзакций:

sql

```
RESTORE LOG my_database
FROM DISK = 'C:\backups\my_database_log.trn'
WITH RECOVERY;
```

6. Где хранить резервные копии?

1. **Локально** (сервер или отдельный диск)
2. **Удаленный сервер** (SFTP, NAS, SAN)
3. **Облачные сервисы** (AWS S3, Google Cloud, Dropbox)
4. **Ленточные носители** (для долгосрочного хранения)

Лучший вариант — хранить копии в нескольких местах.

Вывод

- Резервное копирование **необходимо** для защиты данных.
- Используются **разные типы бэкапов**: полный, инкрементный, дифференциальный, логический.
- **Автоматизация** через cron или SQL Server Agent повышает надежность.
- **Хранение на удаленных серверах** или в облаке предотвращает потерю данных.