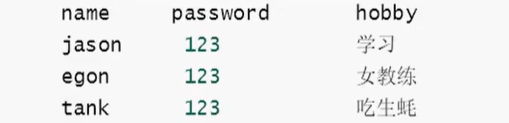
数据库

数据库是用于存储、管理和检索大量数据的系统。它允许用户存储、检索和修改存储在其中的数据。以下是数据库的一些基本简介：

1. **数据库类型**：
   * **关系型数据库**：基于关系模型，使用表格、行和列来组织数据。常见的关系型数据库管理系统（RDBMS）包括MySQL、PostgreSQL、Oracle和Microsoft SQL Server。
   * 
   * **非关系型数据库**：也称为NoSQL数据库，它们不依赖于表格模型，而是使用其他存储数据的方式，如文档、键值对、宽列存储和图形数据库。例如MongoDB、Cassandra和Redis。
   * 
2. **数据库模型**：
   * **层次模型**：数据以树状结构组织。
   * **网络模型**：数据以有向图的形式组织。
   * **关系模型**：数据以表格的形式组织，通过关系（如外键）连接不同的表格。
3. **SQL和NoSQL**：
   * **SQL**：结构化查询语言（SQL）是用于管理关系型数据库的标准编程语言。
   * **NoSQL**：非关系型数据库通常不需要使用SQL，而是使用特定的查询语言或API。
4. **数据库操作**：
   * **CRUD操作**：创建（Create）、读取（Read）、更新（Update）和删除（Delete）是数据库中的基本操作。
   * **事务**：确保数据的完整性和一致性。事务是一组原子性的数据库操作，要么全部成功，要么全部失败。
   * **并发控制**：管理多个用户同时访问数据库的能力，以防止数据冲突和不一致。
5. **数据库索引**：提高数据库查询性能的数据库结构。
6. **数据库安全**：包括用户认证、权限控制、数据加密和备份等，以保护数据不被未授权访问和丢失。
7. **数据库备份和恢复**：定期备份数据库，以便在数据丢失或损坏时可以恢复。
8. **数据库设计**：设计高效的数据库模式，以优化查询性能和存储效率。

数据库技术是信息技术领域的一个重要组成部分，广泛应用于各种业务和应用中，从简单的个人应用程序到大型企业系统。

SQL（Structured Query Language，结构化查询语言）是一种专门用来与数据库通信的编程语言。它被广泛用于数据库管理和数据操作，包括数据的查询、插入、更新和删除等操作。SQL的主要功能可以分为以下几个方面：

1. **数据查询**：使用SELECT语句从数据库中检索数据。
2. **数据操作**：使用INSERT、UPDATE和DELETE语句来添加、修改和删除数据库中的数据。
3. **数据定义**：使用CREATE和DROP语句来创建和删除数据库和表。
4. **数据控制**：使用GRANT和REVOKE语句来控制用户对数据库的访问权限。

SQL语言具有高度的结构化和标准化特点，不同的数据库管理系统（DBMS）如MySQL、PostgreSQL、Oracle、SQL Server等都支持SQL，尽管它们在语法和功能上可能存在一些差异

mysql

windows系统MySQL常用操作

windows系统mysql重要文件存放位置

客户端启动文件C:\Program Files\MySQL\MySQL Server 8.0\bin\mysql.exe，

服务的启动文件C:\Program Files\MySQL\MySQL Server 8.0\bin\mysqld.exe

data文件C:\ProgramData\MySQL\MySQL Server 8.0\Data

配置文件C:\ProgramData\MySQL\MySQL Server 8.0\ my.ini

Windows启动mysql服务端

方法1

1. 以管理员打开cmd

2. 切换到该C:\Program Files\MySQL\MySQL Server 8.0\bin\

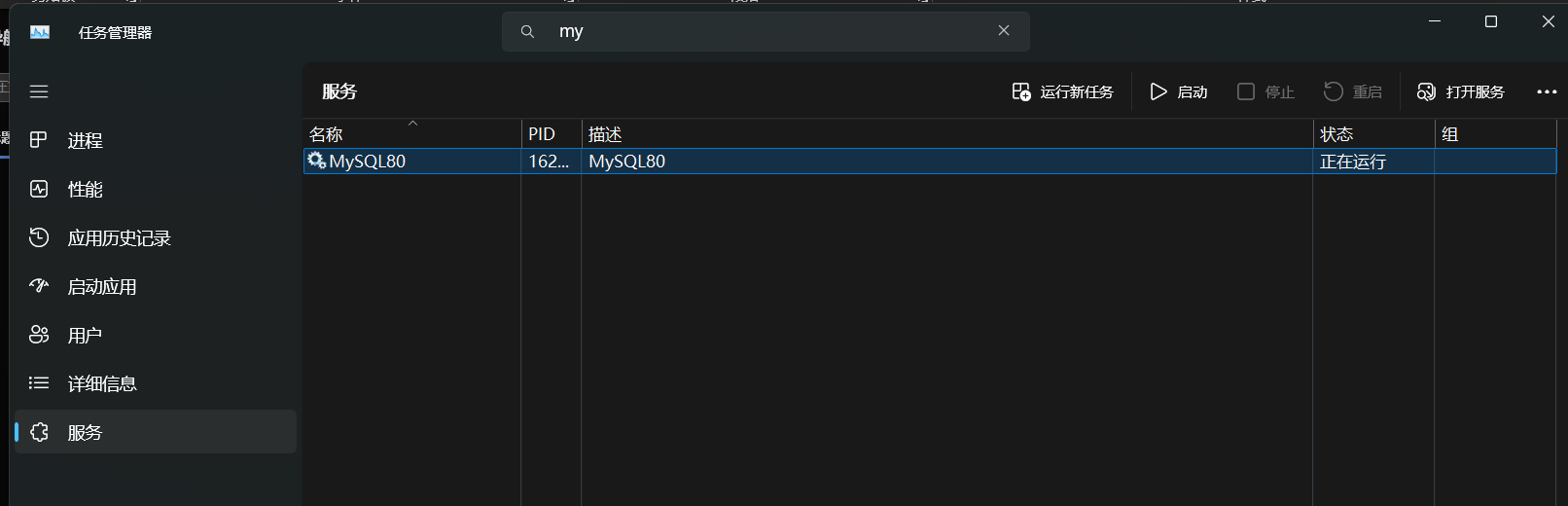
c:

cd C:\Program Files\MySQL\MySQL Server 8.0\bin\

3. 输入命令mysqld启动，启动后窗口不能关闭。

方法2

打开资源管理器-》选择服务选项-》搜索mysql并启动



windows连接mysql服务器

1. 以管理员打开cmd

2. 切换到该C:\Program Files\MySQL\MySQL Server 8.0\bin\

c:

cd C:\Program Files\MySQL\MySQL Server 8.0\bin\

3. 输入命令：mysql -h <hostname> -P <port> -u <username> -p

* <hostname> 是远程MySQL服务器的地址。
* <-P port> 是MySQL服务监听的端口号。
* <username> 是你的MySQL用户名。
* -p 表示会提示你输入密码。

示例：

远程机mysql -h 192.168.1.100 -P 3306 -u root -p

本机mysql -h localhost -u root -p

退出命令

quit

eixt

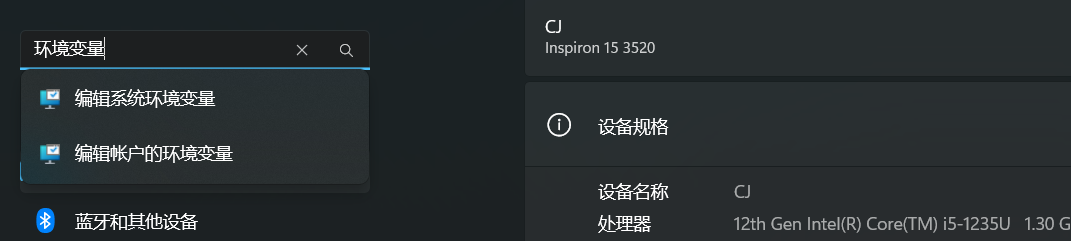
取消操作

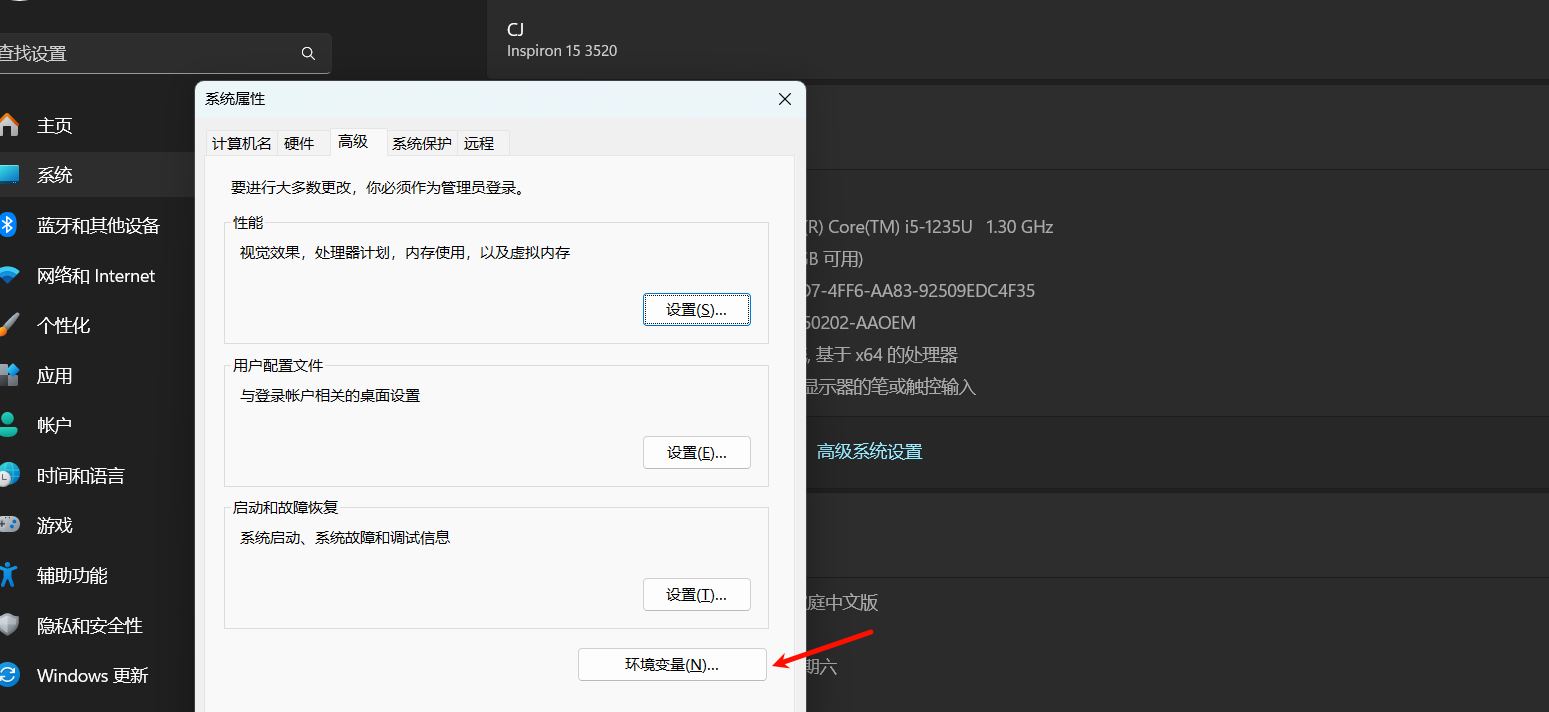
\c

ctrl+c

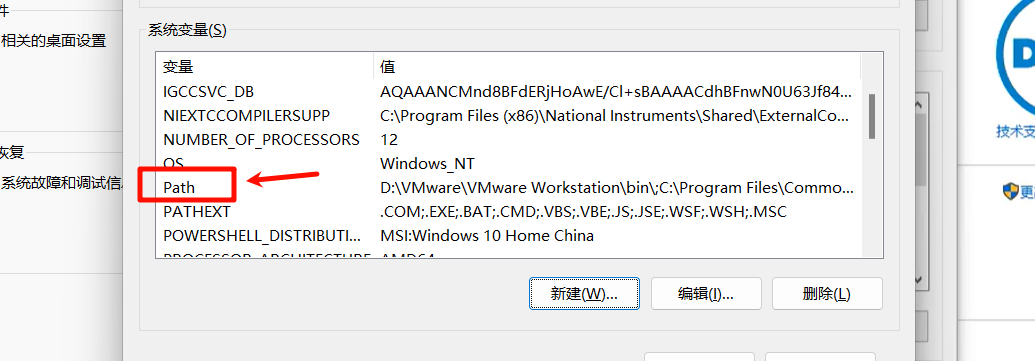
配置环境变量

1. 打开系统设置输入：环境变量

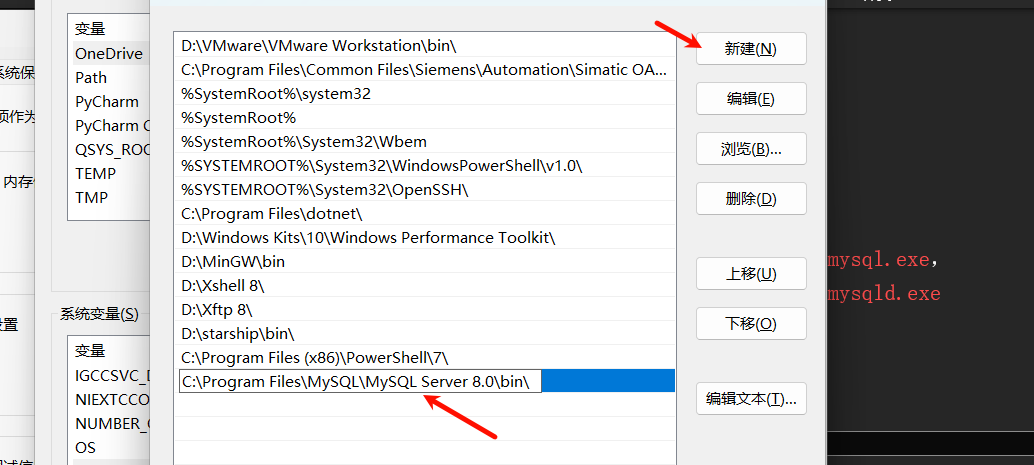




2. 点击Path



3. 添加MySQL.exe和MySQLd.exe的路径



命令行查看杀死进程

tasklist #查看所有进程

tasklist | findstr mysql #查看mysql进程

taskkill /F /PID PID号 #杀死进程

修改密码

命令行输入：mysqladmin -u root -p 新密码 #无须进入客户端

示例：

E:\Desktop>mysqladmin -u root -p password 123

Enter password: \*\*\*\*\*\*\*\*\* #输入原来的密码

mysqladmin: [Warning] Using a password on the command line interface can be insecure.

Warning: Since password will be sent to server in plain text, use ssl connection to ensure password safety.

Ubuntu安装配置mysql

安装

1. **更新软件包列表**： 打开终端，首先更新你的软件包列表以确保你安装的是最新版本的软件包。

sudo apt update

1. **安装MySQL**： 使用以下命令安装MySQL：

sudo apt install mysql-server #默认安装8.0版本

安装过程中，系统会提示你设置MySQL的root用户密码。

1. **安全配置**： 安装完成后，运行安全脚本以增强MySQL的安全性。

sudo mysql\_secure\_installation

这个脚本会提示你设置root密码（如果之前没有设置），移除匿名用户，禁止root用户远程登录，移除测试数据库，并重新加载权限表。

1. **验证MySQL服务状态**： 检查MySQL服务是否正在运行。

sudo systemctl status mysql

1. **登录MySQL**： 使用以下命令登录MySQL：

sudo mysql -u root -p

首次登录没有密码，直接按回车。

1. **退出MySQL**： 使用以下命令退出MySQL：

exit

重要文件位置

配置文件

/etc/mysql/my.cnf

/etc/mysql/mysql.conf.d/mysqld.cnf

数据库文件

/var/lib/mysql/

日志文件

/var/log/mysql/error.log

可执行文件

/usr/bin/mysql

/usr/sbin/mysqld

ssl证书

/var/lib/mysql/

默认安装目录

/usr/bin/mysql

/usr/sbin/mysqld

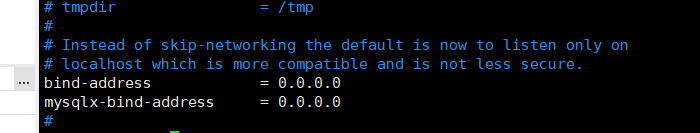
配置root远程访问（如果需要）：

默认情况下，MySQL不允许远程访问。如果你需要远程访问MySQL，需要编配置文件并重新启动服务。

某些 MySQL 安装（如 Ubuntu 上的 MySQL）默认使用 auth\_socket 插件，该插件仅允许本地连接，而不需要密码。

1. MySQL默认只监听本地127.0.0.1，需改变策略

打开配置文件：sudo vim /etc/mysql/mysql.conf.d/mysqld.cnf



把这2个地址改为0.0.0.0，允许所有主机连接

bind-address = 0.0.0.0

mysqlx-bind-address = 0.0.0.0

2. 重启MySQL服务

sudo systemctl restart mysql

3. 验证监听端口

sudo netstat -tuln | grep 3306

tcp 0 0 0.0.0.0:3306 0.0.0.0:\* LISTEN

红色为0.0.0.0则配置成功

4. 开放防火墙

sudo ufw allow 3306

sudo ufw reload

5. 检查MySQL用户权限

确保 MySQL 用户（如 root）具有远程连接的权限。

* 登录到 MySQL：

mysql -u root -p

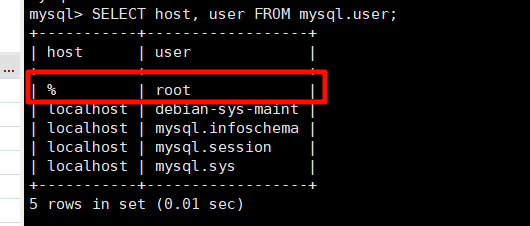
* 检查用户权限：

SELECT host, user FROM mysql.user;

* 如果 root 用户的 host 不是 %（表示允许所有 IP 连接），需要更新权限：

UPDATE mysql.user SET host='%' WHERE user='root';

FLUSH PRIVILEGES;



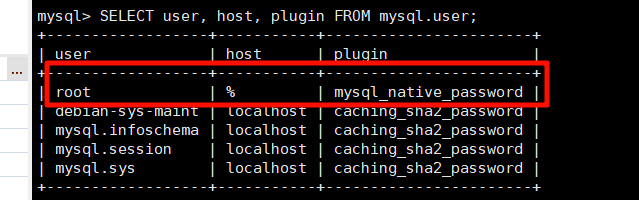
6. 检查 MySQL 认证插件

某些 MySQL 安装（如 Ubuntu 上的 MySQL）默认使用 auth\_socket 插件，该插件仅允许本地连接，而不需要密码。

* 检查 root 用户的认证插件：

SELECT user, host, plugin FROM mysql.user;

如果 plugin 是 auth\_socket，需要将其更改为 mysql\_native\_password 或 caching\_sha2\_password：



ALTER USER 'root'@'%' IDENTIFIED WITH mysql\_native\_password BY 'your\_password';

FLUSH PRIVILEGES;

将 your\_password 替换为你的密码。

例如：

ALTER USER 'root'@'%' IDENTIFIED WITH mysql\_native\_password BY 'p-0p-0p-0';

FLUSH PRIVILEGES;

查看和更改密码策略

1. 查看当前密码策略

首先，查看 MySQL 的密码策略配置。

* 登录到 MySQL：

mysql -u root -p

* 查看密码策略：

SHOW VARIABLES LIKE 'validate\_password%';

输出示例：

复制

+--------------------------------------+--------+

| Variable\_name | Value |

+--------------------------------------+--------+

| validate\_password.check\_user\_name | ON |

| validate\_password.dictionary\_file | |

| validate\_password.length | 8 |

| validate\_password.mixed\_case\_count | 1 |

| validate\_password.number\_count | 1 |

| validate\_password.policy | MEDIUM |

| validate\_password.special\_char\_count | 1 |

+--------------------------------------+--------+

关键参数说明：

* + validate\_password.length：密码最小长度。
  + validate\_password.mixed\_case\_count：密码中至少包含的大写字母和小写字母数量。
  + validate\_password.number\_count：密码中至少包含的数字数量。
  + validate\_password.special\_char\_count：密码中至少包含的特殊字符数量。
  + validate\_password.policy：密码策略级别（LOW、MEDIUM、STRONG）。

2. 设置符合要求的密码

根据密码策略，设置一个符合要求的密码。例如：

* 至少 8 个字符。
* 包含至少 1 个大写字母、1 个小写字母、1 个数字和 1 个特殊字符。

示例密码：

P@ssw0rd

然后执行：

ALTER USER 'root'@'%' IDENTIFIED WITH mysql\_native\_password BY 'P@ssw0rd';

FLUSH PRIVILEGES;

3. 修改密码策略（可选）

如果不想设置复杂密码，可以临时降低密码策略级别或禁用密码验证插件。

**方法 1：降低密码策略级别**

将密码策略设置为 LOW，这样只需要满足最小长度要求。

* 修改密码策略：

SET GLOBAL validate\_password.policy = LOW;

* 然后设置简单密码：

ALTER USER 'root'@'%' IDENTIFIED WITH mysql\_native\_password BY 'p-0p-0p-0';

FLUSH PRIVILEGES;

**方法 2：禁用密码验证插件**

如果不需要密码策略，可以禁用 validate\_password 插件。

* 编辑 MySQL 配置文件（如 /etc/my.cnf 或 /etc/mysql/mysql.conf.d/mysqld.cnf）：

sudo nano /etc/my.cnf

* 在 [mysqld] 部分添加以下内容：

validate\_password = OFF

* 保存并退出，然后重启 MySQL 服务：

sudo systemctl restart mysql

* 登录 MySQL 并设置简单密码：

ALTER USER 'root'@'%' IDENTIFIED WITH mysql\_native\_password BY 'p-0p-0p-0';

FLUSH PRIVILEGES;

4. 验证更改

尝试使用新密码连接 MySQL，确保更改生效。

mysql -u root -h 192.168.75.10 -p

5. 安全性注意事项

* **避免使用简单密码**：即使降低了密码策略，也建议使用强密码以增强安全性。
* **限制远程访问**：仅允许受信任的 IP 地址访问 MySQL 服务器。
* **启用 SSL**：如果可能，配置 MySQL 使用 SSL 加密连接。

按照以上步骤操作后，问题应该可以解决。如果仍有问题，请提供更多信息，我会进一步帮助你排查！

设置root密码

sudo mysqladmin -u root password 'new\_password'

将new\_password替换为你想要设置的密码。

登录MySQL

sudo mysql -u root -p

输入root密码。

重置root密码

如果忘记密码，可以重置 root 用户密码：

* 1. 停止 MySQL 服务：

sudo systemctl stop mysql

* 1. 启动 MySQL 并跳过权限检查：

sudo mysqld\_safe --skip-grant-tables &

* 1. 登录到 MySQL：

mysql -u root

* 1. 重置密码：

ALTER USER 'root'@'%' IDENTIFIED BY 'new\_password';

FLUSH PRIVILEGES; #立刻将数据刷到硬盘

* 1. 退出 MySQL 并重启 MySQL 服务：

sudo systemctl restart mysql

MySQL-shell命令

mysql> show databases; //显示数据库

+--------------------+

| Database |

+--------------------+

| information\_schema |

| mysql |

| performance\_schema |

| sys |

+--------------------+

4 rows in set (0.01 sec)

> \status or \s # 显示当前MySQL Shell的状态

SQL

数据库关键字不区分大小写，不计入空白字符，语句后没有“；”会继续执行下面的语句。

USE store; // 使用store数据库

SELECT

first\_name,

last\_name,

points\*10 +10 as discount\_factor //as别名，如果要使用空格，就得加‘’或者“”

FROM customers //选择customers这张表

WHERE points > 3000 //查询条件

state = ‘VA’

birth\_date > ‘1990/01/01’

WHERE NOT ((birth\_date > '1990-01-01' OR points > 1000)

AND state = 'VA')

#运算符 <、 <= 、> 、>=、 =、 != or <>

#逻辑运算符AND OR NOT 搞不清楚优先级加括号

WHERE state IN ('VA', 'FL', 'GA') //state字段包含VA,FL,GA的信息

WHERE state NOT IN ('VA', 'FL', 'GA') //state字段不包含VA,FL,GA的信息

WHERE points BETWEEN 1000 AND 3000 //points字段1000到3000的信息

WHERE birth\_date BETWEEN '1990-01-01' AND '2000-01-01'

WHERE last\_name LIKE '\_\_\_\_\_y' //匹配5个任意字符+y，%表示匹配任意字符，\_匹配单个字符

WHERE last\_name LIKE '%y' //匹配以y结尾的字符

WHERE last\_name REGEXP '[a-h]e' //正则匹配

^ beginning

% end

| logical or

[abcd] 匹配abcd中的一个

[a-f] 匹配a到f中的一个

WHERE phone IS NOT NULL //查询phone字段非空信息

WHERE phone IS NULL //查询phone字段空信息

SELECT first\_name, last\_name, 10 AS points,state //增加10这列别名points

FROM customers

ORDER BY state,first\_name DESC //先按state升序排列，再按first\_name降序排列

LIMIT 1,3 //查询第2到4条记录，LIMIT放最后

LIMIT 3 //查询前3条记录

按这样的顺序写，不然会报错

SELECT \*

FROM customers

WHERE state in ('VA')

ORDER BY first\_name

LIMIT 1,3

连接

内连接

内连接:INNER JOIN 或JOIN ,只返回匹配的结果

SELECT order\_id, o.customer\_id, first\_name, last\_name

FROM orders o //给orders起别名为o，其它要引用的地方可以用o代替

JOIN customers ON o.customer\_id = customers.customer\_id //内连接customers这张表，连接条件是o.customer\_id = customers.customer\_id

不同库里的表连接

USE sql\_store;

SELECT \*

FROM products

JOIN sql\_inventory.products on sql\_store.products.product\_id = sql\_inventory.products.product\_id

自己与自己连接，要给自己起别名

USE sql\_hr;

SELECT

e.employee\_id,

e.first\_name,

m.first\_name AS manager

FROM employees e

JOIN employees m ON e.reports\_to = m.employee\_id

多表连接

USE sql\_store;

SELECT

o.order\_id,

o.order\_date,

c.first\_name,

c.last\_name,

os.name AS status

FROM orders o

JOIN customers c

ON o.customer\_id = c.customer\_id

JOIN order\_statuses os

ON o.status = os.order\_status\_id

复合连接

有2个主键唯一标识记录

SELECT \*

FROM order\_items oi

JOIN order\_item\_notes oin

ON oi.order\_id = oin.order\_id

AND oi.product\_id = oin.product\_id

隐式连接

USE sql\_store;

SELECT \*

FROM customers c, orders o

WHERE c.customer\_id = o.customer\_id

外连接

尽量使用LEFT连接，结构清晰

INNER JOIN内连接，只选择匹配的项。

OUTER JOIN外连接：

LEFT JOIN:选择所有FROM的项，不管有没有匹配

RIGHT JOIN:选择所有JOIN的项，不管有没有匹配

SELECT

p.product\_id,

p.name,

oi.quantity

FROM products p

LEFT JOIN order\_items oi //可以写成LEFT OUTER JOIN，OUTER可以省略

ON p.product\_id = oi.product\_id

ORDER BY p.product\_id

SELECT

o.order\_date,

o.order\_id,

c.first\_name,

sh.name AS shipper,

os.name AS status

FROM orders o

LEFT JOIN customers c

ON o.customer\_id = c.customer\_id

LEFT JOIN order\_statuses os

ON o.status = os.order\_status\_id

LEFT JOIN shippers sh

ON o.shipper\_id = sh.shipper\_id

order by shipper

USING连接

连接的列名要一样

USING (status, customer\_id) <==> ON o.status = c.status

USING (status, customer\_id) <==> ON o.status = c.status AND o.customer\_id = c.customer\_id

自然连接

NATURAL JOIN会自己选择相同列名进行连接,不建议使用

SELECT \*

FROM orders

NATURAL JOIN customers

交叉连接

交叉连接会把,2张表所有的结果都匹配一边,类型a\*a

SELECT

order\_id,

first\_name

FROM orders

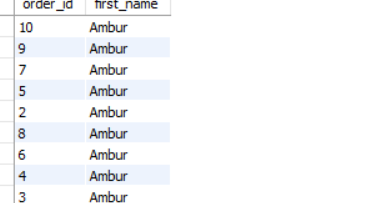
CROSS JOIN customers //显式写法

SELECT

order\_id,

first\_name

FROM orders , customers //隐式写法



联合查询

UNION可以连接2个不同的查询,但SELECT的列要一样

SELECT first\_name

FROM customers

UNION

SELECT order\_id

FROM orders

插入信息

插入单行

INSERT INTO customers ( // customer后不加括号,就要填入所有的信息,加了括号只填入括号内的信息,信息可以填default和NULL.

last\_name,

first\_name,

birth\_date,

address,

city,

state)

VALUES (

'Smith',

'John',

'1990-01-01',

'address',

'city',

'CA')

插入多行

INSERT INTO shippers (name)

VALUES

('shipper1'),

('shipper2'),

('shipper3')

LAST\_INSERT\_ID()获取最新插入id

不同的表相互管理,在这张插入后,在另一张表采用相关联的id插入

INSERT INTO orders (customer\_id, order\_date, status)

VALUES (1, '2019-01-01',1);

INSERT INTO order\_items

VALUES (LAST\_INSERT\_ID(), 1, 1, 2.95)

创建表和复制行

复制orders这张表,但不会复制主键,和AI(自动递增)

CREATE TABLE orders\_acrchived AS

SELECT \* FROM orders

复制orders带条件的行到orders\_acrchived

INSERT INTO orders\_acrchived

SELECT \*

FROM orders

WHERE order\_date < '2019-01-01'

更新单行

USE invoicing;

UPDATE invoices //要更新的表

SET payment\_total = 10 , payment\_date = '2019-03-01' //要更新的字段

WHERE invoice\_id = 1 //更新的条件

USE invoicing;

UPDATE invoices

SET

payment\_total = invoice\_total \* 0.5 ,

payment\_date = due\_date

WHERE invoice\_id = 1

更新多行

USE invoicing;

UPDATE invoices

SET

payment\_total = invoice\_total \* 0.5 ,

payment\_date = due\_date

WHERE client\_id IN (3,1)

更新多行(子查询)

加括号把子查询括起来

USE invoicing;

UPDATE invoices

SET

payment\_total = invoice\_total \* 0.5 ,

payment\_date = due\_date

WHERE client\_id IN (

SELECT client\_id

FROM clients

WHERE state IN ('CA', 'NY'))

删除行

DELETE FROM invoicing

WHERE client\_id = 1 //注意不加WHERE条件会删除所有行

postgreSql

Ubuntu安装postgresql

1. 安装

sudo apt install postgresql-[版本号] postgresql-client-[版本号]

2. 启动postgres服务

sudo systemctl start postgresql

3.设置 PostgreSQL 服务开机自启

sudo systemctl enable postgresql

4. 登录

sudo -u postgres psql #未设置密码不需要密码，这是 PostgreSQL 的默认超级用户

5. 设置 postgres 用户密码

\password postgres

6. 退出 psql

输入\q并按回车键退出。

7. 测试连接

psql -U postgres -h localhost

输入之前设置的 postgres 用户密码，如果能够成功连接到数据库，说明安装和配置成功。

Ubuntu配置postgresql远程连接

一、修改 pg\_hba.conf 文件

1. **找到文件位置**
   * pg\_hba.conf 文件通常位于 PostgreSQL 数据目录下。可以通过以下命令查找数据目录：
     + 在 PostgreSQL 中执行 SHOW data\_directory; 命令（需先连接到数据库，如 psql -U postgres）。
   * 常见的位置可能是 /var/lib/pgsql/[版本号]/data（CentOS/RHEL 系统）或 /etc/postgresql/[版本号]/main（Debian/Ubuntu 系统）。
2. **编辑文件**
   * 使用文本编辑器（如 sudo nano /var/lib/pgsql/[版本号]/data/pg\_hba.conf）打开 pg\_hba.conf 文件。
   * 在文件末尾添加以下内容（允许所有 IP 地址远程连接，示例中使用 md5 加密方式验证密码，你可以根据需要选择其他加密方式）：

host all all 0.0.0.0/0 md5

* + 如果你只想允许特定 IP 地址远程连接，将 0.0.0.0/0 替换为具体的 IP 地址或子网掩码，例如 192.168.1.0/24 表示允许 192.168.1.0 网段的主机连接。

二、修改 postgresql.conf 文件

1. **找到文件位置**
   * 该文件与 pg\_hba.conf 文件在同一目录下。
2. **编辑文件**
   * 打开 postgresql.conf 文件（如 sudo nano /var/lib/pgsql/[版本号]/data/postgresql.conf）。
   * 找到 listen\_addresses 参数，将其值从 localhost 修改为 \*，表示监听所有网络接口。
   * 保存并关闭文件。

三、重启 PostgreSQL 服务

1. **根据操作系统重启服务**
   * **Debian/Ubuntu 系统**：

sudo systemctl restart postgresql

* + **CentOS/RHEL 系统**：

sudo systemctl restart postgresql-[版本号].service

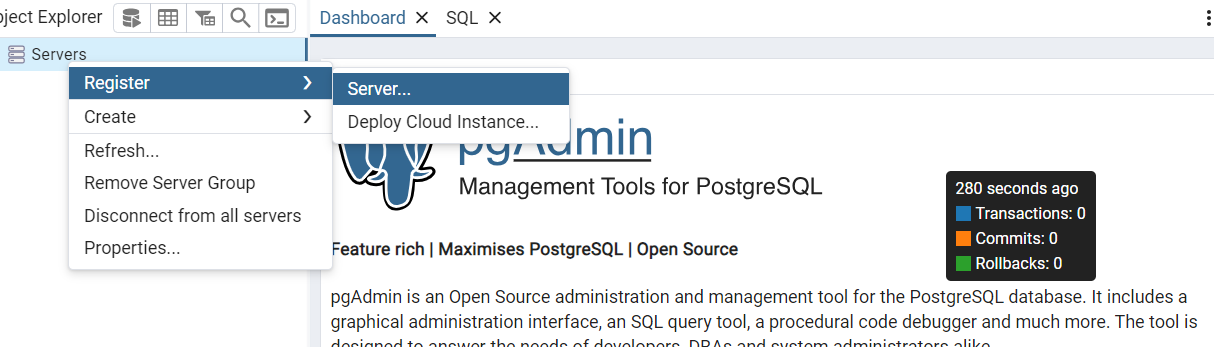
四、测试远程连接

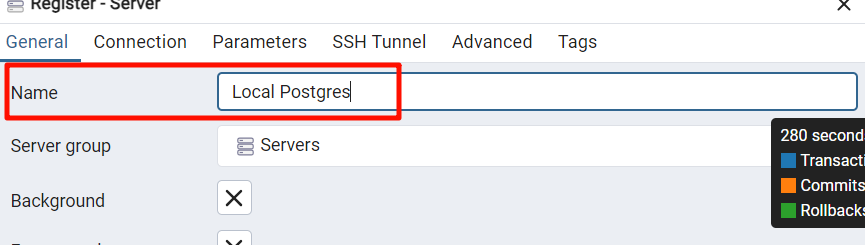
1. **在远程主机上使用 psql 命令连接**
   * 例如：psql -U [用户名] -h [服务器IP地址] -d [数据库名]
   * 输入密码后，如果能够成功连接到 PostgreSQL 数据库，说明远程登录已开启成功。

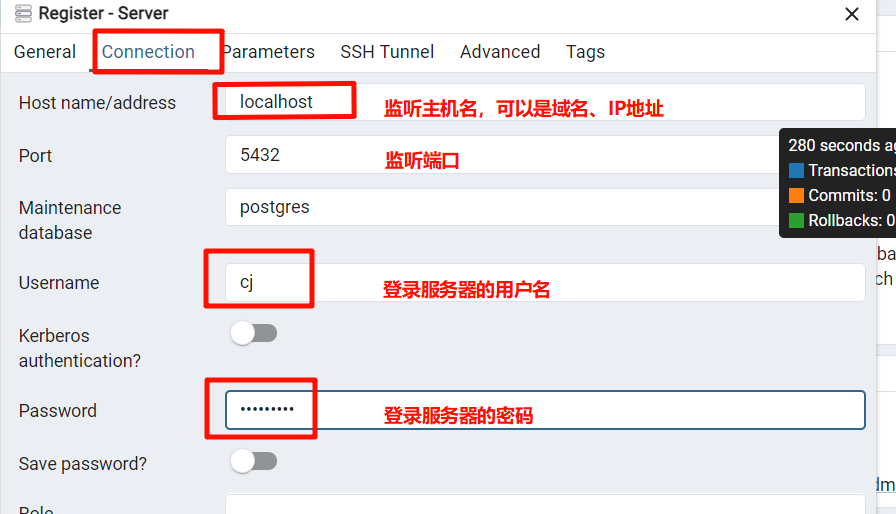
注意：开启远程登录可能会带来一定的安全风险，确保你的服务器和数据库有足够的安全措施，如设置强密码、配置防火墙等，只允许可信的主机进行远程连接。

图形化操作

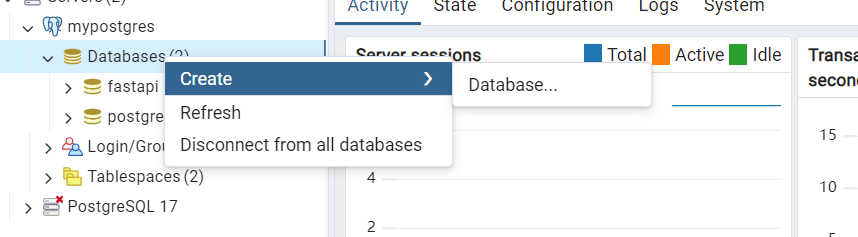
1. 创建要连接的服务器

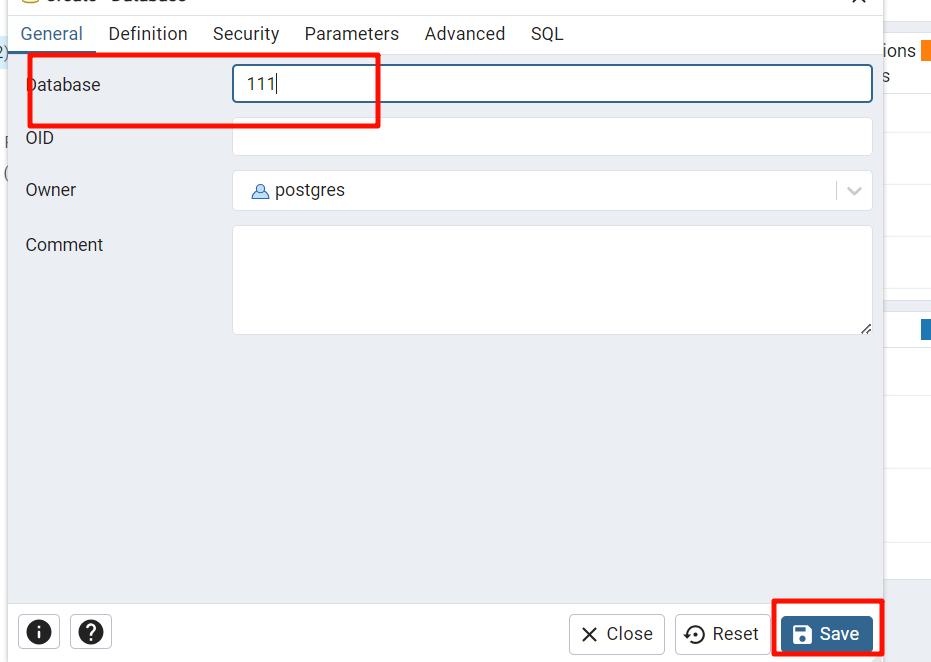






2. 创建数据库





3. 创建表

