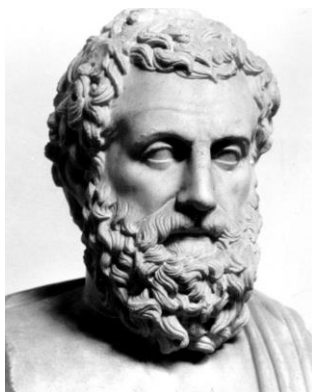


用SQL开启你的数据检索之旅

Lesson-04

陈旻 & 开课吧人工智能学院课程组
2021.1



- Thinking: behind the theory, original from the real problem
- Action: solve problems by tools, present the results

>> 今天的学习目标

SQL语言使用

- SQL执行流程
- DDL创建数据库
- SQL规范
- SELECT执行的顺序
- SQL数据过滤
- SQL函数
- SQL92标准
- SQL99标准

SQL Action

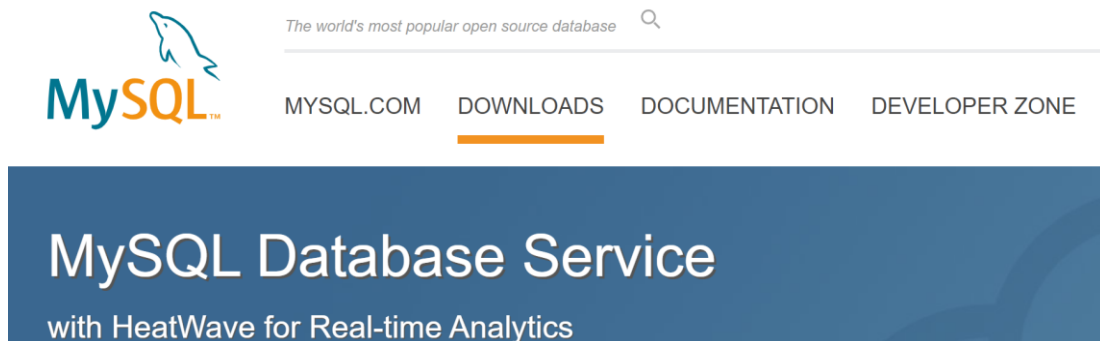
- Action1 查找上升的温度
- Action2 两张表的连接
- Action3 查找网约车每天的取消率
- Action4 查找 超过经理收入的员工
- Action5 找到从不订购的用户
- Action6 查找部门工资最高的员工
- Action7 查找第二高的薪水
- Action8 查找至少有3名直接下属的经理
- Action9 出行流量统计

1/2 SQL语言使用

MySQL安装

- Step1, 下载安装包

<https://www.mysql.com/downloads>



选择 MySQL Community(GPL) Downloads （社区开源版本）

<https://dev.mysql.com/downloads/mysql/>

- 可以选择MySQL 5.7或8.0

8.0 功能和性能上都是目前最好的MySQL

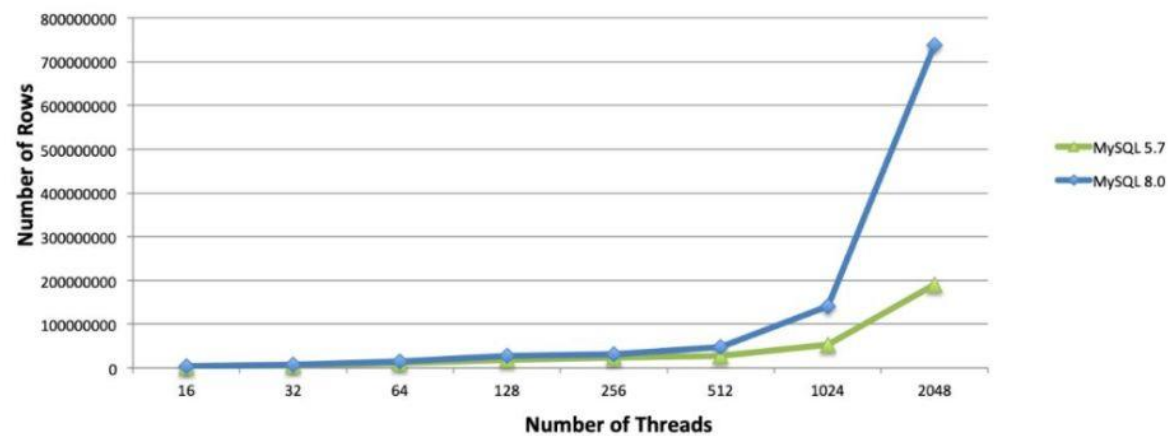
1) 性能上, 增删改查效率高

2) 功能上, 新增多种功能, 比如MySQL 被吐槽最多的特性之一就是缺少 `rank()` 函数, 当需要在查询当中实现排名时, 必须手写 `@` 变量。但是从 8.0 开始, MySQL 新增“窗口函数”

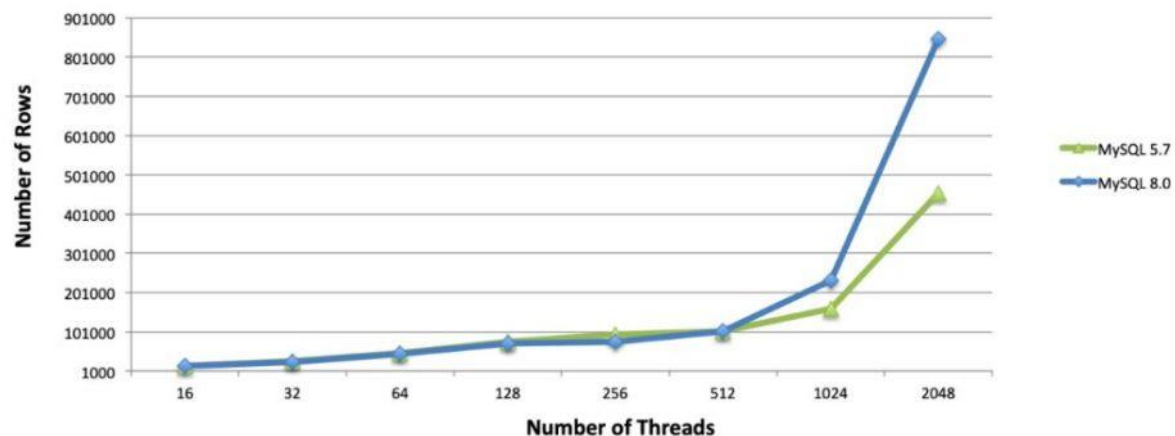
需要说明, 仍有很多项目采用MySQL5.7

MySQL安装

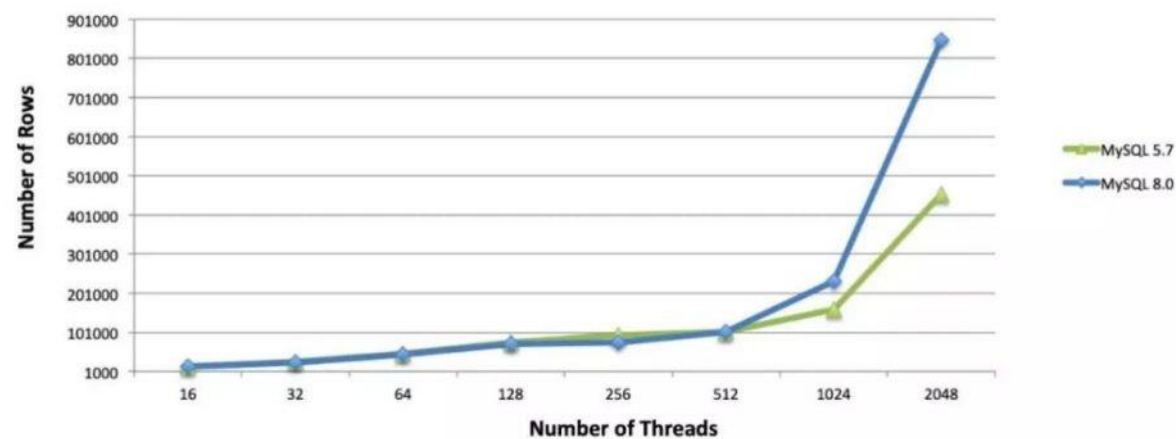
Innodb_rows_read



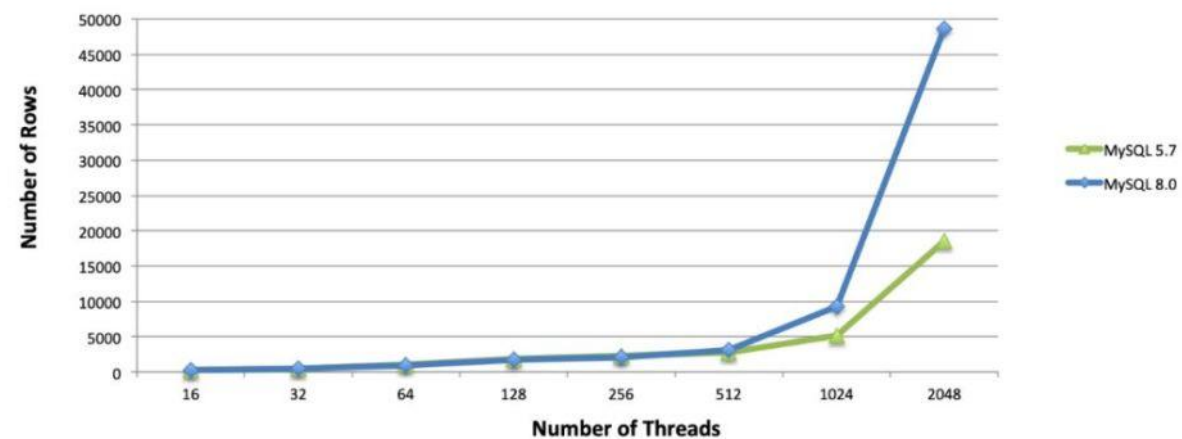
Innodb_rows_inserted



Innodb_rows_deleted



Innodb_rows_updated



SQL是一门语言

- Level1: 掌握SQL并不难

了解基础语法，通过例子进行训练

- Level2: 进阶SQL查询效率

如何有效的设计数据表，规范是什么

怎样查询效率更好

- Level3: 进阶数据库架构设计

高可用，负载均衡，读写分离

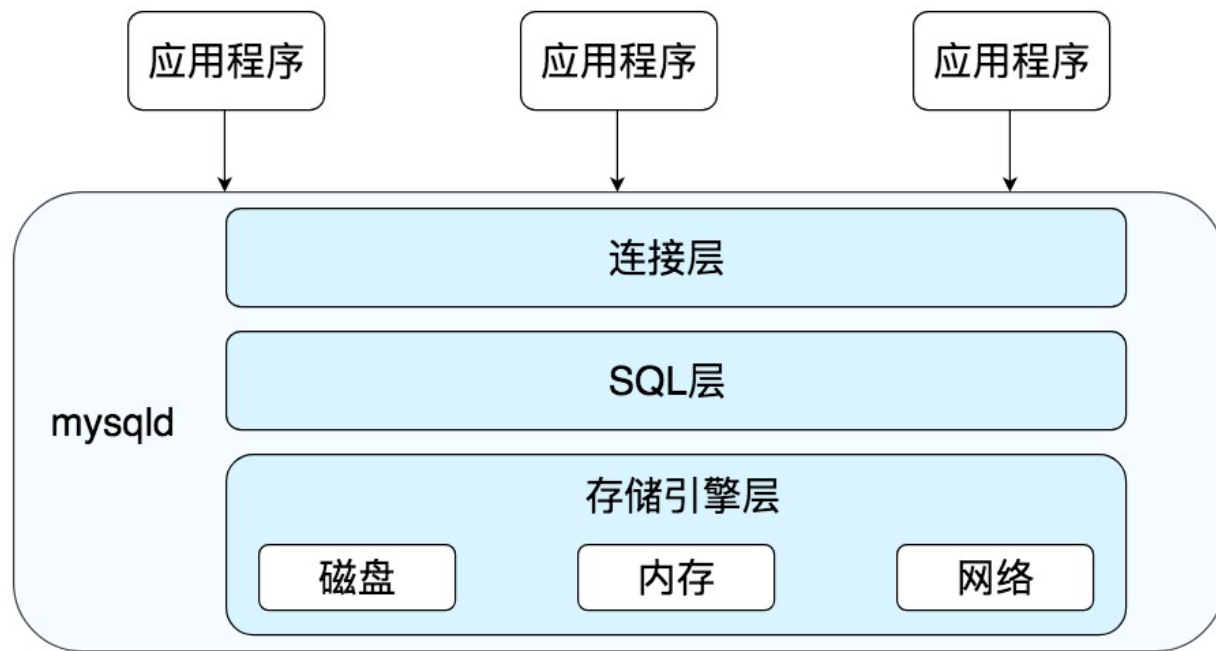
SQL执行流程

MySQL流程结构：

- 连接层：客户端和服务端建立连接，客户端发送SQL至服务器端
- SQL层：对SQL语句进行查询处理
- 存储引擎层：与数据库文件打交道，负责数据的存储和读取

其中SQL层与数据库文件的存储方式无关

MySQL是典型的C/S架构，即Client/Server架构，服务器端程序使用的mysqld（mysqld是服务，在linux中服务常以d结尾，比如httpd，d即daemon）

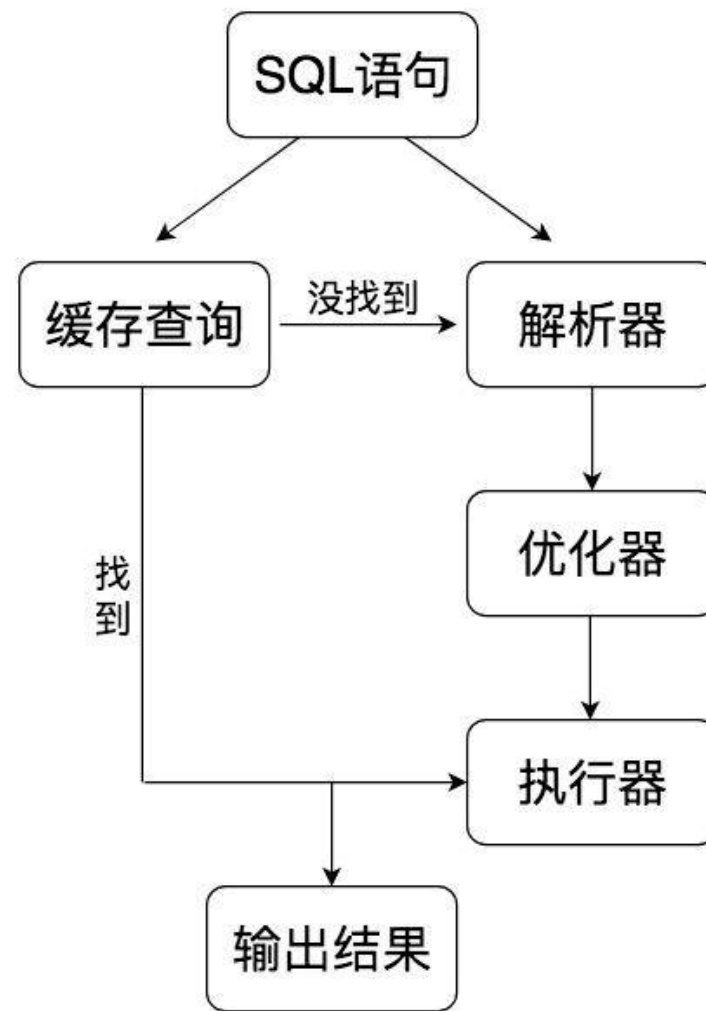


MySQL流程图

SQL执行流程

SQL查询流程：

- 查询缓存：Server如果在查询缓存中发现了这条SQL语句，就会直接将结果返回给客户端；如果没有，就进入到解析器阶段。（因为查询缓存往往效率不高，所以在MySQL8.0之后就抛弃了这个功能）
- 解析器：在解析器中对SQL语句进行语法分析、语义分析
- 优化器：在优化器中会确定SQL语句的执行路径，比如是根据全表检索，还是根据索引来检索等
- 执行器：在执行之前需要判断该用户是否具备权限，如果具备权限就执行SQL查询并返回结果（在MySQL8.0以下的版本，如果设置了查询缓存，此时会将查询结果进行缓存）
- SQL语句的执行流程是：SQL语句→缓存查询→解析器→优化器→执行器
（MySQL和Oracle执行SQL的原理是一样的）



SQL查询流程

DDL创建数据库

DDL使用:

- Data Definition Language, 定义了数据库以及数据表的结构
- 对数据库进行定义

```
CREATE DATABASE nba; // 创建一个名为nba的数据库
```

```
DROP DATABASE nba; // 删除一个名为nba的数据库
```

- 对数据表进行定义

```
CREATE TABLE [table_name](字段名 数据类型, .....)
```

- 创建表结构:

```
CREATE TABLE player (
```

```
player_id int(11) NOT NULL AUTO_INCREMENT,
```

```
player_name varchar(255) NOT NULL
```

```
);
```

语句最后以分号 ; 作为结束符

数据类型中int(11)代表整数类型, 这里的11代表最大有效显示长度, 和真实数值的宽度没有关系

varchar(255)代表的是最大长度为255的可变字符串类型

NOT NULL表明整个字段不能是空值, 是一种数据约束

AUTO_INCREMENT代表主键自动增长

DDL创建数据库

使用客户端创建数据表：

- Navicat 或者 dbeaver
- 可视化方式创建后，可以直接将数据表导出为DDL语句

```
DROP TABLE IF EXISTS `player`;
CREATE TABLE `player` (
  `player_id` int(11) NOT NULL AUTO_INCREMENT,
  `team_id` int(11) NOT NULL,
  `player_name` varchar(255) CHARACTER SET utf8 COLLATE
utf8_general_ci NOT NULL,
  `height` float(3, 2) NULL DEFAULT 0.00,
  PRIMARY KEY (`player_id`) USING BTREE,
  UNIQUE INDEX `player_name` (`player_name`) USING BTREE
) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8_general_ci
ROW_FORMAT = Dynamic;
```

- 修改表结构：

添加字段，比如数据表中添加一个age字段，类型为int(11)

```
ALTER TABLE player ADD (age int(11));
```

修改字段名，将age字段改成player_age

```
ALTER TABLE player RENAME COLUMN age to player_age
```

修改字段的数据类型，将player_age的数据类型设置为float(3,1)

```
ALTER TABLE player MODIFY (player_age float(3,1));
```

删除字段, 删除刚才添加的player_age字段

```
ALTER TABLE player DROP COLUMN player_age;
```



DDL语句生成，可以通过Navicat或dbeaver 可视化操作实现

数据表常见约束

- 1) 主键约束，UNIQUE+NOT NULL，主键可以是一个字段，也可以由多个字段复合组成
- 2) 外键约束，外键确保了表与表之间引用的完整性。一个表中的外键对应另一张表的主键
- 3) 字段的唯一性约束

Thinking: 唯一性约束和普通索引（Normal Index）的区别？

唯一性约束相当于创建了一个约束和普通索引，目的是保证字段的正确性，而普通索引只是提升数据检索的速度，并不对字段的唯一性进行约束

4) NOT NULL约束

5) CHECK约束

用来检查特定字段取值范围的有效性，比如可以对身高height的数值进行CHECK约束，必须 ≥ 0 ，且 < 3 ，即CHECK(height ≥ 0 AND height < 3)

SQL编写规范：（客户端一般自带美化功能）

- 表名、表别名、字段名、字段别名等都小写
- SQL保留字、函数名、绑定变量等都大写

```
SELECT name, hp_max FROM heros WHERE role_main = '战士'
```

数据表的设计原则：

- 数据表的个数越少越好

RDBMS的核心在于对实体和联系的定义，也就是E-R图（Entity Relationship Diagram），数据表越少，证明实体和联系设计得越简洁

- 数据表中的字段个数越少越好

字段个数越多，数据冗余的可能性越大

字段个数少是相对的，通常会在数据冗余和检索效率中进行平衡

- 数据表中联合主键的字段个数越少越好

设置主键是为了确定唯一性，当一个字段无法确定唯一性的时候，就需要采用联合主键的方式（也就是用多个字段来定义一个主键）

联合主键中的字段越多，占用的索引空间越大，不仅会加大理解难度，还会增加运行时间和索引空间，因此联合主键的字段个数越少越好







- 是否使用外键约束要根据情况而定

SQL数据源

- NBA数据表

https://github.com/cystanford/sql_nba_data


player表, player_score表, team表, team_score表,
height_grades表

cystanford Add files via upload		
	.gitattributes	Initial commit
	height_grades.sql	Add files via upload
	player.sql	Add files via upload
	player_score.sql	Add files via upload
	team.sql	Add files via upload
	team_score.sql	Add files via upload

- Step1, 下载数据表SQL文件
- Step2, 使用Navicat或dbeaver将数据表导入

height_grades
height_level: varchar(255)
height_lowest: float(3, 2)
height_highest: float(3, 2)

player_score
game_id: int(0)
player_id: int(0)
is_first: tinyint(1)
playing_time: int(0)
rebound: int(0)
rebound_o: int(0)
rebound_d: int(0)
更多 (12 列) ...

team_score
 game_id: int(0)
h_team_id: int(0)
v_team_id: int(0)
h_team_score: int(0)
v_team_score: int(0)
game_date: date

player
 player_id: int(0)
team_id: int(0)
 player_name: varchar(255)
height: float(3, 2)

team
 team_id: int(0)
team_name: varchar(255)

SELECT *使用:

- 在做数据探索的时候，SELECT *还是很有用的，这样就不需要写很长的SELECT语句了

Thinking: 在生产环境时要尽量避免使用SELECT *

会增加数据库的负担

如果我们不需要把所有列都检索出来，生产环境中需要SELECT指定的列名，减少数据表查询的网络传输量

- ORDER BY排序

ORDER BY可以使用非选择列进行排序（即使在SELECT后面没有这个列名）

```
SELECT name, hp_max FROM heros ORDER BY mp_max,  
hp_max DESC
```

- 约束返回结果的数量

```
SELECT name, hp_max FROM heros ORDER BY hp_max  
DESC LIMIT 5
```

ORACLE需要基于ROWNUM来统计行数

```
SELECT name, hp_max FROM heros WHERE ROWNUM  
<=5 ORDER BY hp_max DESC
```


SELECT执行的顺序:

SELECT DISTINCT player_id, player_name, count(*) AS
num #顺序5

FROM player JOIN team ON player.team_id =
team.team_id #顺序1

WHERE height > 1.80 #顺序2

GROUP BY player.team_id #顺序3

HAVING num > 2 #顺序4

ORDER BY num DESC #顺序6

LIMIT 2 #顺序7

- SQL语法顺序不能颠倒

SQL是一门类似英语的结构化查询语言，所以在写SELECT语句的时候，还要注意相应的关键字顺序 => SQL语句写法

- SQL底层运行流程，与语法顺序不同

在MySQL和Oracle中，SELECT执行顺序基本相同



SELECT是SQL的基础，不同阶段看SELECT会有不同体会

Step1，掌握语法

Step2，关注SELECT查询效率

SELECT查询

Thinking: 当你进行SELECT查询时，知道只有1条记录满足要求，可以怎样提升查询效率？

使用LIMIT 1来进行约束

WHERE子句:

- WHERE子句的基本格式是: SELECT(列名)
FROM(表名) WHERE(子句条件)
- 在WHERE子句中使用比较运算符进行判断

不同DBMS写法会有差异, 比如Access不支持!=, 不等于应该使用<>。在MySQL中, 不支持!>, !<等

含义	运算符
等于	=
不等于	<>或!=
小于	<
小于等于 (大不予)	<=或!>
大于	>
大于等于 (不小于)	>=或!<
不小于	!<
在指定的两个数值之间	BETWEEN
为空值	IS NULL

常用比较运算符

SQL数据过滤

使用通配符进行过滤：

- 通配符就是用来匹配值的一部分的特殊字符，需要使用到LIKE操作符
- 想要匹配任意字符串出现的任意次数，需要使用%通配符

Thinking: 想要查找英雄名中包含“太”字的英雄都有哪些？

```
SELECT name FROM heros WHERE name LIKE '%太%'
```

name
东皇太一
太乙真人

使用通配符进行过滤：

如果想要匹配单个字符，就需要使用下划线 _

Thinking: %和_的区别？

% 代表零个或多个字符

_ 只代表一个字符

Thinking: 想要查找英雄名除了第一个字以外，包含‘太’字的英雄有哪些？

```
SELECT name FROM heros WHERE name LIKE '_%太%'
```

name
东皇太一

- SQL函数

函数对于语言来说很重要， $\text{function}(x) \Rightarrow y$

SQL函数，对于SQL数据分析来说很重要

封装好了常用的函数，使用方便

- SQL聚集函数

函数	说明
COUNT()	总行数
MAX()	最大值
MIN()	最小值
SUM()	求和
AVG()	平均值

- 算数函数

函数名	定义
ABS()	取绝对值
MOD()	取余
ROUND()	四舍五入为指定的小数位数，需要有两个参数，分别为字段名称、小数位数

Thinking: `SELECT ROUND(37.25,1)`，运行结果=?

37.3

- 字符串函数

函数名	定义
CONCAT()	将多个字符串拼接起来
LENGTH()	计算字段的长度，一个汉字算三个字符，一个数字或字母算一个字符
CHAR_LENGTH()	计算字段的长度，汉字、数字、字母都算一个字符
LOWER()	将字符串中的字符转化为小写
UPPER()	将字符串中的字符转化为大写
REPLACE()	替换函数，有3个参数：要替换的表达式或字段名、要查找的被替换字符串、替换成哪个字符串
SUBSTRING()	截取字符串，有3个参数：待截取的表达式或字段名、开始截取的位置、想要截取的字符串长度

Thinking: `SELECT REPLACE('fabcd', 'abc', 123)`，运行结果=?

f123d

- 日期函数

函数名	定义
CURRENT_DATE()	系统当前日期
CURRENT_TIME()	系统当前时间, 没有具体的日期
CURRENT_TIMESTAMP()	系统当前时间, 包括具体的日期+时间
EXTRACT() DATE() YEAR() MONTH() DAY() HOUR() MINUTE() SECOND()	抽取具体的年、月、日 返回时间的日期部分 返回时间的年份部分 返回时间的月份部分 返回时间的天数部分 返回时间的小时部分 返回时间的分钟部分 返回时间的秒部分

Thinking: `SELECT DATE('2021-01-10 15:04:05')`, 运行结果=?
2021-01-10

- 转换函数

函数名	定义
CAST()	数据类型转换, 参数是一个表达式, 表达式通过AS关键词分割了2个参数, 分别是原始数据和目标数据类型
COALESCE()	返回第一个非空数值

`SELECT CAST(222.222 AS DECIMAL(8,2))`, 运行结果=?

222.22

DECIMAL(8,2)代表的是精度为8位（整数加小数位数最多为8位），小数位数为2位的数据类型

SQL Action (日期函数)

- Action1 查找上升的温度

Weather表，记录了特定日期的温度

Column Name	Type	
id	int	
recordDate	date	
temperature	int	

id	recordDate	Temperature
1	2015-01-01	10
2	2015-01-02	25
3	2015-01-03	20
4	2015-01-04	30

查询：比昨天温度高的所有日期（id）

Result table:

id
2
4

2015-01-02 的温度比前一天高（10 -> 25）

2015-01-04 的温度比前一天高（20 -> 30）

DATEDIFF函数

SELECT DATEDIFF('2021-01-11','2021-01-10') AS diffdate

运行结果： 1

Thinking: 如何使用DATEDIFF函数进行判断

SELECT a.id

FROM Weather AS a

INNER JOIN Weather AS b

WHERE DATEDIFF(a.recordDate, b.recordDate) = 1

AND a.temperature > b.temperature;

SQL标准:

- SQL有两个主要的标准，分别是SQL92和SQL99

92和99代表了标准提出的时间，SQL92就是92年提出的标准规范

- SQL92中的表连接

需要使用NBA数据表（player, team）

1) 两张表的笛卡尔积

```
SELECT * FROM player, team
```

player_id	team_id	player_name	height	team_id(1)	team_name
10001	1001	韦恩·艾灵顿	1.93	1001	底特律活塞
10001	1001	韦恩·艾灵顿	1.93	1002	印第安纳步行者
10001	1001	韦恩·艾灵顿	1.93	1003	亚特兰大老鹰
.....
10037	1002	伊凯·阿尼博古	2.08	1003	亚特兰大老鹰

2) 等值连接

两张表的等值连接就是用两张表中都存在的列进行连接。也可以对多张表进行等值连接

Thinking: player表和team表都存在team_id列，怎样写等值连接？

```
SELECT player_id, player.team_id, player_name, height, team_name
FROM player, team WHERE player.team_id = team.team_id
```

非等值连接:

- 连接多个表的条件是等号时, 就是等值连接, 其他的运算符连接就是非等值查询

这里使用了身高级别表height_grades

height_level	height_lowest	height_highest
A	2.00	2.50
B	1.90	1.99
C	1.80	1.89
D	1.60	1.79

Thinking: 如果要知道每个球员的身高的级别, 怎样写SQL语句

```
SELECT p.player_name, p.height, h.height_level
```

```
FROM player AS p, height_grades AS h
```

```
WHERE p.height BETWEEN h.height_lowest AND h.height_highest
```

player_name	height	height_level
韦恩·艾灵顿	1.93	B
雷吉·杰克逊	1.91	B
安德烈·德拉蒙德	2.11	A
.....
Ike Anigbogu	2.08	A

外连接:

- 除了查询满足条件的记录以外，外连接还可以查询某一方不满足条件的记录
- 两张表的外连接，会有一张是主表，另一张是从表。如果是多张表的外连接，那么第一张表是主表（显示全部的行）
- 在SQL92中采用+号代表从表所在的位置，在SQL92中，只有左外连接和右外连接，没有全外连接

Thinking: 什么是左外连接，什么是右外连接呢？

左外连接，就是指左边的表是主表，需要显示左边表的全部行，而右侧的表是从表，+号表示哪个是从表

Thinking: 下面那个是SQL92标准，哪个是SQL99标准？

- 1) `SELECT * FROM player, team where player.team_id = team.team_id(+)`
- 2) `SELECT * FROM player LEFT JOIN team on player.team_id = team.team_id`
- 3) `SELECT * FROM player, team where player.team_id(+) = team.team_id`
- 4) `SELECT * FROM player RIGHT JOIN team on player.team_id = team.team_id`

自连接:

- 自连接可以对多个表进行操作，也可以对同一个表进行操作（也就是查询条件使用了当前表的字段）

Thinking: 想要查看比布雷克·格里芬高的球员都有谁，以及他们的对应身高

```
SELECT b.player_name, b.height FROM player as a ,  
player as b WHERE a.player_name = '布雷克-格里芬' and  
a.height < b.height
```

Thinking:如果不用自连接会怎样做？ 几次SQL查询

```
SELECT height FROM player WHERE player_name = '布雷克-  
格里芬'
```

运行结果为 2.08

```
SELECT player_name, height FROM player WHERE height >  
2.08
```

分两次进行查询，结果与自连接一致

SQL99标准

SQL99标准:

- 交叉连接

交叉连接实际上就是SQL92中的笛卡尔乘积，只是这里采用的是CROSS JOIN

Thinking: 想要得到player和team两张表的笛卡尔积结果

SELECT * FROM player CROSS JOIN team

player_id	team_id	player_name	height	team_id(1)	team_name
10001	1001	韦恩-艾灵顿	1.93	1001	底特律活塞
10001	1001	韦恩-艾灵顿	1.93	1002	印第安纳步行者
10001	1001	韦恩-艾灵顿	1.93	1003	亚特兰大老鹰
.....
10037	1002	伊凯·阿尼博古	2.08	1003	亚特兰大老鹰

- 自然连接

自然连接的英文是NATURAL JOIN，你可以把它理解为SQL92中的等值连接。它会帮你自动查询两张连接表中所有相同的字段，然后进行等值连接

SQL92: SELECT player_id, a.team_id, player_name, height, team_name FROM player as a, team as b WHERE a.team_id = b.team_id

SQL99: SELECT player_id, team_id, player_name, height, team_name FROM player NATURAL JOIN team

- ON连接

ON连接用来指定我们想要的连接条件

Thinking: 刚才的NATURAL JOIN, 用ON连接怎样写?

```
SELECT player_id, player.team_id, player_name, height,  
team_name FROM player JOIN team ON player.team_id =  
team.team_id
```

player_id	team_id	player_name	height	team_name
10001	1001	韦恩-艾灵顿	1.93	底特律活塞
10002	1001	雷吉-杰克逊	1.91	底特律活塞
10003	1001	安德烈-德拉蒙德	2.11	底特律活塞
.....
10037	1002	伊凯·阿尼博古	2.08	印第安纳步行者

Thinking: ON连接能否做非等值查询?

可以进行非等值连接, 比如我们想要查询球员的身高等级, 需要用player和height_grades两张表

```
SELECT p.player_name, p.height, h.height_level
```

```
FROM player as p JOIN height_grades as h
```

```
ON height BETWEEN h.height_lowest AND
```

```
h.height_highest
```


- USING连接

可以用USING指定数据表里的同名字段进行等值连接

比如: `SELECT player_id, team_id, player_name, height,
team_name FROM player JOIN team USING(team_id)`

1) 与自然连接NATURAL JOIN不同的是, USING指定了具体的相同的字段名称, 你需要在USING的括号()中填入要指定的同名字段

2) JOIN ... USING可以简化JOIN ON的等值连接, 相当于

`SELECT player_id, player.team_id, player_name, height,
team_name FROM player JOIN team ON player.team_id =
team.team_id`

SQL Action (表联查)

- Action2 两张表的连接

<https://leetcode-cn.com/problems/combine-two-tables/>

Person表

列名	类型
PersonId	int
FirstName	varchar
LastName	varchar
PersonId 是上表主键	

Address表

列名	类型
AddressId	int
PersonId	int
City	varchar
State	varchar
AddressId 是上表主键	

编写一个 SQL 查询：无论 person 是否有地址信息，都需要基于上述两表提供 person 的以下信息：

FirstName, LastName, City, State

Thinking：

- 1) 哪张表是主表
- 2) 应该是怎样的表连接方式

```
SELECT FirstName, LastName, City, State
FROM Person LEFT JOIN Address
ON Person.PersonId = Address.PersonId
```

SQL Action (JOIN使用)



- Action3 查找网约车每天的取消率

Users 表，存储所有用户信息

字段	说明
User_Id	用户ID
Banned	这个用户是否被禁止
Role	角色 (client, driver, partner)

Users_Id	Banned	Role
1	No	client
2	Yes	client
3	No	client
4	No	client
10	No	driver
11	No	driver
12	No	driver
13	No	driver

Trips 表中存所有出租车的行程信息

字段	说明
Client_Id	用户ID
Driver_Id	司机ID
City_Id	城市ID
Status	订单状态 (completed, cancelled_by_driver, cancelled_by_client)
Request_at	订单发起时间

Id	Client_Id	Driver_Id	City_Id	Status	Request_at
1	1	10	1	completed	2013-10-01
2	2	11	1	cancelled_by_driver	2013-10-01
3	3	12	6	completed	2013-10-01
4	4	13	6	cancelled_by_client	2013-10-01
5	1	10	1	completed	2013-10-02
6	2	11	6	completed	2013-10-02
7	3	12	6	completed	2013-10-02
8	2	12	12	completed	2013-10-03
9	3	10	12	completed	2013-10-03
10	4	13	12	cancelled_by_driver	2013-10-03

SQL Action (JOIN使用)

- Action3 查找网约车每天的取消率

SQL查询: 2013年10月1日至2013年10月3日期间非禁止用户的取消率 (Cancellation Rate保留两位小数)

Day	Cancellation Rate
2013-10-01	0.33
2013-10-02	0.00
2013-10-03	0.50

取消率的计算方式:

(被司机或乘客取消的非禁止用户生成的订单数量) /
(非禁止用户生成的订单总数)

- Step1, 筛选订单

- 1) 司机和乘客都是非禁止用户
- 2) 日期在2013-10-01到2013-10-03之间

```
SELECT *
```

```
FROM trips t
```

```
JOIN users u1 ON (t.client_id=u1.users_id AND  
u1.banned='No')
```

```
JOIN users u2 ON (t.driver_id=u2.users_id AND  
u2.banned='No')
```

```
WHERE request_at BETWEEN '2013-10-01' AND '2013-10-03'
```

SQL Action (JOIN使用)

- Step2, 用日期进行分组

GROUP BY request_at

- Step3, 计算统计结果

分别统计所有订单数和被取消的订单数

取消订单数用一个bool条件得到0或1, 再用avg求均值

对订单取消率保留两位小数

```
SELECT request_at AS 'Day',
ROUND(AVG(Status!='completed'), 2) AS 'Cancellation
Rate'
```

Day	Cancellation Rate
2013-10-01	0.33
2013-10-02	0.00
2013-10-03	0.50

```
SELECT request_at AS 'Day', ROUND(AVG(Status!='completed'), 2
) AS 'Cancellation Rate'

FROM trips t

JOIN users u1 ON (t.client_id=u1.users_id AND u1.banned='No')

JOIN users u2 ON (t.driver_id=u2.users_id AND u2.banned='No')

WHERE request_at BETWEEN '2013-10-01' AND '2013-10-03'

GROUP BY request_at
```

SQL Action (表联查)

Action4: 查找 超过经理收入的员工

Employee 表包含所有员工, 他们的经理也属于员工。
每个员工都有一个 Id, 还有一列对应员工的经理的 Id

Employee表

Id	Name	Salary	ManagerId
1	Joe	70000	3
2	Henry	80000	4
3	Sam	60000	NULL
4	Max	90000	NULL

编写SQL 查询, 获取收入超过他们经理的员工的姓名。
上面数据中, Joe 是唯一一个收入超过他的经理的员工

两张表联查:

```
SELECT b. NAME AS Employee
```

```
FROM Employee a JOIN Employee b
```

```
WHERE a.Id = b.ManagerId AND b.salary > a.salary;
```

SQL Action (子查询)

Action5：找到从不订购的用户

某网站包含两个表，Customers 表和 Orders 表。编写 SQL 查询，找出所有从不订购任何东西的客户

Customers表

Id	Name
1	Joe
2	Henry
3	Sam
4	Max

Orders表

Id	CustomerId
1	3
2	1

查询结果：

Customers
Henry
Max

使用子查询：

```
SELECT Customers.Name AS Customers
FROM Customers
WHERE Customers.Id NOT IN
(
    SELECT CustomerId FROM Orders
);
```


SQL Action (子查询)

- Action6 查找部门工资最高的员工

Employee表 (员工信息)

Id	Name	Salary	DepartmentId
1	Joe	70000	1
2	Jim	90000	1
3	Henry	80000	2
4	Sam	60000	2
5	Max	90000	1

Department表 (部门信息)

Id	Name
1	IT
2	Sales

编写 SQL 查询，找出每个部门工资最高的员工

Department	Employee	Salary
IT	Max	90000
IT	Jim	90000
Sales	Henry	80000

- 如何使用子查询

子查询：找到每个部门最高的工资

主查询：找到员工中，符合子查询条件的员工

```
SELECT d.Name as Department, e.Name as Employee, Salary
from Employee e JOIN Department d ON e.DepartmentId =
d.Id
WHERE (e.DepartmentId, e.Salary) IN (
    SELECT DepartmentId, MAX(Salary)
    FROM Employee
    GROUP BY DepartmentId
);
```

SQL Action (临时表, 子查询)

- Action7 查询第二高的薪水

<https://leetcode-cn.com/problems/second-highest-salary>

Employee表

+-----+-----+		
Id	Salary	
+-----+-----+		
1	100	
2	200	
3	300	
+-----+-----+		

查询结果

+-----+	
SecondHighestSalary	
+-----+	
200	
+-----+	

SQL查询应该返回 200 作为第二高的薪水。如果不存在第二高的薪水，那么查询应返回 null

- 临时表

当数据量很大时，可以让SQL每次找出所需的少数记录，将记录存放到一个临时表，这样方便后续的查询，速度更快

- 创建临时表：

```
CREATE TEMPORARY TABLE tmp_table (  
  
    name VARCHAR(10) NOT NULL,  
  
    value INTEGER NOT NULL  
  
)
```

临时表在连接MySQL时都存在，断开连接时会自动释放

将查询结果导入临时表

```
CREATE TEMPORARY TABLE tmp_table SELECT * FROM  
table_name
```

```
CREATE TABLE tmp AS
```

```
(SELECT column1 AS field1, column2 AS field2...)
```

SQL Action (临时表, 子查询)

方法1: 使用临时表

Step1, 找到第二高的薪水

```
SELECT Salary AS SecondHighestSalary FROM Employee  
  
ORDER BY Salary DESC LIMIT 1, 1
```

已完成 执行用时: 96 ms

输入	<pre>{"headers": {"Employee": ["Id", "Salary"]}, "rows": {"Employee": [[1, 100], [2, 200], [3, 300]]}}</pre>
输出	<pre>{"headers": ["SecondHighestSalary"], "values": [[200]]}</pre>
预期结果	<pre>{"headers": ["SecondHighestSalary"], "values": [[200]]}</pre>

执行结果: 解答错误 显示详情 >

输入:

<pre>{"headers": {"Employee": ["Id", "Salary"]}, "rows": {"Employee": [[1, 100]]}}</pre>	
输出:	
<pre>{"headers": ["SecondHighestSalary"], "values": []}</pre>	

预期结果:

<pre>{"headers": ["SecondHighestSalary"], "values": [[null]]}</pre>

Step2, 找到第二高的薪水

```
SELECT  
  
(SELECT DISTINCT Salary  
  
FROM Employee  
  
ORDER BY Salary DESC  
  
LIMIT 1,1) SecondHighestSalary
```

方法2: 使用子查询

```
SELECT MAX(Salary) AS SecondHighestSalary  
  
FROM Employee  
  
WHERE Salary != (SELECT MAX(Salary) FROM Employee)
```

SQL Action (出行流量统计)

- Action: 出行流量统计

trip表, 记录每日出行流量信息 (出行日期, 人数)

Column Name	Type	id	visit_date	people
id	int	1	2017-01-01	10
visit_date	date	2	2017-01-02	109
people	int	3	2017-01-03	150
		4	2017-01-04	99
		5	2017-01-05	145
		6	2017-01-06	1455
		7	2017-01-07	199
		8	2017-01-09	188

Result table:

id	visit_date	people
5	2017-01-05	145
6	2017-01-06	1455
7	2017-01-07	199
8	2017-01-09	188



id 为 5、6、7、8 的四行 id 连续, 并且每行都有 ≥ 100 的人数记录

SQL 查询: 找出每行的人数大于或等于 100 且 id 连续的三行或更多行记录 (返回按 visit_date 升序排列)

不输出 id 为 2 和 3 的行, 因为需要三条及以上 id 连续

- Rank函数

MySQL8.0推出Rank函数（排名函数）

1) 创建分数表

```
CREATE TABLE `hero_score` (  
  
  `id` int NOT NULL AUTO_INCREMENT,  
  
  `score` int NOT NULL DEFAULT 0,  
  
  `name` varchar(20) CHARACTER SET utf8mb4 NULL,  
  
  PRIMARY KEY (`id`)  
  
);
```

2) 插入数据

```
INSERT INTO `hero_score` (`name`, `score`) VALUES  
  
('张飞', 80),  
  
('关羽', 95),  
  
('刘备', 76),  
  
('曹操', 92),  
  
('典韦', 89),  
  
('貂蝉', 87),  
  
('诸葛亮', 99),  
  
('赵云', 93);
```



使用Navicat或dbeaver生成数据表

Rank函数



3) 按照成绩从高到低进行排名

```
SELECT *, RANK() OVER(ORDER BY score DESC) 名次 FROM  
hero_score;
```

使用RANK函数必须用ORDER BY参数，排序字段就是排名
字段

id	score	name	名次
7	99	诸葛亮	1
2	95	关羽	2
8	93	赵云	3
4	92	曹操	4
5	89	典韦	5
6	87	貂蝉	6
1	80	张飞	7
3	76	刘备	8

Rank函数

4) 如何两个人的成绩相同, RANK排名会怎样?

修改刘备的score = 92

```
UPDATE hero_score SET score = 92 WHERE name = '刘备';
```

按照成绩从高到低进行排名

```
SELECT *, RANK() OVER(ORDER BY score DESC) 名次 FROM  
hero_score;
```

发现92分的英雄, 并列第4, 下一名是第6名

Thinking: 如果想要排名连续该怎样操作?

```
SELECT *, DENSE_RANK() OVER(ORDER BY score DESC) 名次  
FROM hero_score;
```

id	score	name	名次
7	99	诸葛亮	1
2	95	关羽	2
8	93	赵云	3
3	92	刘备	4
4	92	曹操	4
5	89	典韦	6
6	87	貂蝉	7
1	80	张飞	8

SQL Action（出行流量统计）

- Action: 出行流量统计

Step1, 创建trip表

Column Name	Type
id	int
visit_date	date
people	int

id 为主键，递增


id, visit_date, people均不为空

将people设置默认值为0

Thinking:

1) 通过Navicat/dbbeaver进行创建

2) 使用T-SQL进行创建

字段	索引	外键	触发器	选项	注释	SQL 预览				
名			类型			长度	小数点	不是 null	虚拟	键
id			int			0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	 1
visit_date			date			0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
people			int			0	0	<input checked="" type="checkbox"/>	<input type="checkbox"/>	

默认:

☒ 自动递增

☐ 无符号

☐ 填充零

默认:

☐ 自动递增

☐ 无符号

☐ 填充零

SQL Action (出行流量统计)

转储SQL文件（仅结构）

```
DROP TABLE IF EXISTS `trip`;
```

```
CREATE TABLE `trip` (
```

```
  `id` int(0) NOT NULL AUTO_INCREMENT,
```

```
  `visit_date` date NOT NULL,
```

```
  `people` int(0) NOT NULL DEFAULT 0,
```

```
  PRIMARY KEY (`id`) USING BTREE
```

```
) ENGINE = InnoDB CHARACTER SET = utf8mb4 COLLATE =
```

```
utf8mb4_0900_ai_ci ROW_FORMAT = Dynamic;
```

添加数据到trip表

```
INSERT INTO `trip` VALUES (1, '2021-01-01', 10);
```

```
INSERT INTO `trip` VALUES (2, '2021-01-02', 109);
```

```
INSERT INTO `trip` VALUES (3, '2021-01-03', 150);
```

```
INSERT INTO `trip` VALUES (4, '2021-01-04', 99);
```

```
INSERT INTO `trip` VALUES (5, '2021-01-05', 145);
```

```
INSERT INTO `trip` VALUES (6, '2021-01-06', 1455);
```

```
INSERT INTO `trip` VALUES (7, '2021-01-07', 199);
```

```
INSERT INTO `trip` VALUES (8, '2021-01-09', 188);
```

SQL Action (出行流量统计)

Step1, 筛选出people>100, 按照id进行排序

SELECT

id,

visit_date,

people,

id-RANK() OVER(ORDER BY id) rk

FROM trip

WHERE people >= 100

id	visit_date	people	rk
2	2021-01-02	109	1
3	2021-01-03	150	2
5	2021-01-05	145	3
6	2021-01-06	1455	4
7	2021-01-07	199	5
8	2021-01-09	188	6

我们要找的是连续的天数, 可以通过id-rk得到分组

(id连续情况, 与rk连续情况应该是相同的)

SQL Action (出行流量统计)

Step2, 连续的为一组, 通过id-RANK()进行统计

使用WITH ... AS ... 定义SQL片段, 方便后续将SQL片段用于SQL语句中

WITH t1 AS (

SELECT

id, visit_date, people,

id-RANK() OVER(ORDER BY id) rk

FROM trip

WHERE people >= 100

)

id	visit_date	people	rk
2	2021-01-02	109	1
3	2021-01-03	150	1
5	2021-01-05	145	2
6	2021-01-06	1455	2
7	2021-01-07	199	2
8	2021-01-09	188	2

筛选条数>=3的

SELECT rk FROM t1 GROUP BY rk HAVING COUNT(*) >= 3

SQL Action (出行流量统计)

Step3, 输出完整SQL查询

通过id-RANK(), 进行连续值分组统计

WITH t1 AS (

SELECT

id, visit_date, people,

id-RANK() OVER(ORDER BY id) rk

FROM trip

WHERE people >= 100

)

筛选条数>=3的

SELECT

id, visit_date, people

FROM t1

WHERE rk in (

SELECT rk FROM t1 GROUP BY rk HAVING COUNT(*) >= 3);

id	visit_date	people
5	2021-01-05	145
6	2021-01-06	1455
7	2021-01-07	199
8	2021-01-09	188

Summary (SQL92与SQL99)

SQL标准:

- 主流RDBMS，比如MySQL、Oracle、SQL Sever、DB2、PostgreSQL等都支持SQL语言，这些DBMS的使用符合大部分SQL标准，但很难完全符合（根据自身产品的特点进行了扩充）
- 在1992年，Windows3.1发布，SQL92标准也同时发布，如今我们早已不使用Windows3.1操作系统，而SQL92标准却一直持续至今
- Oracle对SQL92支持较好，而MySQL不支持SQL92的外连接

表连接方式:

- 内连接：将多个表之间满足连接条件的数据行查询出来。它包括了等值连接、非等值连接和自连接
- 外连接：会返回一个表中的所有记录，以及另一个表中匹配的行。它包括了左外连接、右外连接和全连接
- 交叉连接：也称为笛卡尔积，返回左表中每一行与右表中每一行的组合，在SQL99中使用的CROSS JOIN

Summary (SQL92与SQL99)

SQL92中的WHERE和SQL99中的JOIN:

- 在SQL92中, 会把所有需要连接的表都放到FROM之后, 然后在WHERE中写明连接的条件
- SQL99更灵活, 不需要一次性把所有需要连接的表都放到FROM之后, 而是采用JOIN的方式
- 多表连接时, 建议使用SQL99标准

SELECT ...

FROM table1

JOIN table2 ON table1和table2的连接条件

JOIN table3 ON table2和table3的连接条件

嵌套逻辑类似FOR循环

for t1 in table1:

for t2 in table2:

if condition1:

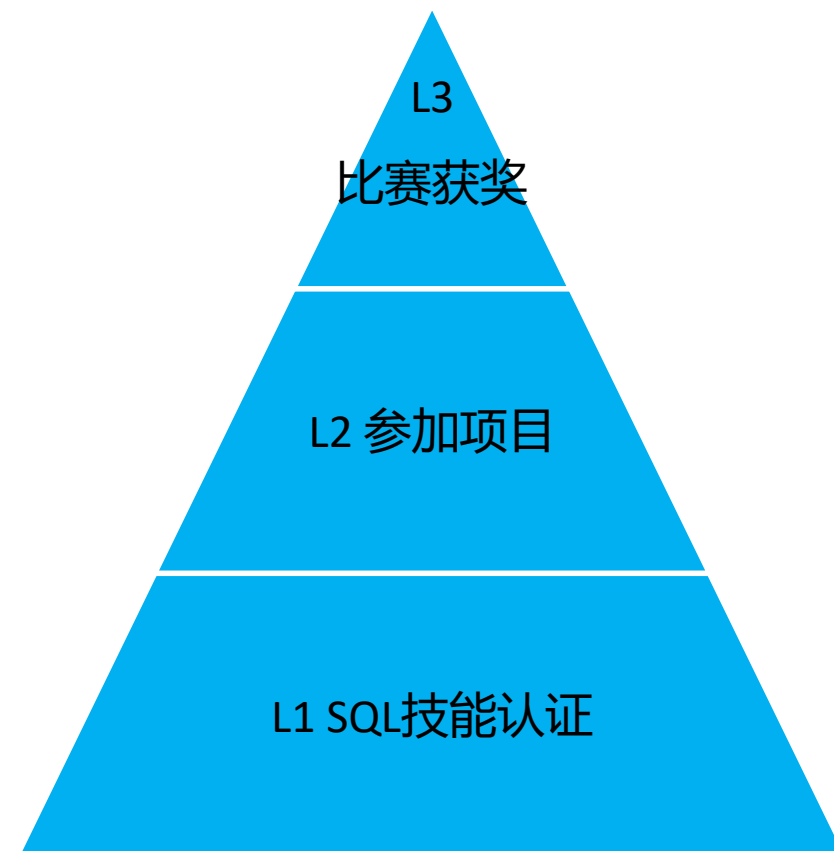
for t3 in table3:

if condition2:

output t1 + t2 + t3

SQL在数据分析中的作用越来越大

- 1) 必备工具：互联网，传统行业的产研岗必备工具
- 2) 更灵活：掌握SQL取数，不需要麻烦IT人员，就可以进行自助性数据分析
- 3) 学习成本较低：相比于其他语言，各种工具库
- 4) 通用性强：作为中台语言，和大数据直接相关，HiveQL, SparkSQL



Thinking1: 都有哪些常见的RDBMS?

Thinking2: 请简述SELECT 语句的写法顺序, 以及执行顺序

Thinking3: MySQL中定义DECIMAL类型的数, 如果不声明精度和标度, 系统会按哪个进行显示?

A、DECIMAL(6,0)

B、DECIMAL(8,0)

C、DECIMAL(10,0)

D、DECIMAL(12,0)



Thank You
Using data to solve problems



小红书招聘

商业经营分析师

20k-30k / 上海 / 经验1-3年 / 本科及以上 / 全职

数据分析

10:43 发布于拉勾网

职位诱惑:

扁平化管理、年底双薪、包三餐

职位描述:

工作职责:

- 1、基于销售团队业务整体框架、一线实际和管理需求，搭建常规报表体系；
- 2、根据业务重点，开展专题分析，给出业务认知迭代和策略建议；
- 3、推进数据底层（指标、库表等）、数据线上化及可视化的建设。

任职资格:

- 1、熟练使用Excel和SQL；
- 2、一年以上互联网公司商业分析、数据分析相关经验；
- 3、有销售思维，对数据有足够的敏感度和洞察力，优秀的学习能力；
- 4、有知名咨询公司经验者优先加分。

工作地址

上海 - 黄浦区 - 马当路388号SOHO复兴广场C座2楼

Hypers招聘

ETL开发工程师（SQL）

10k-15k·13薪 / 上海 / 经验不限 / 本科及以上 / 全职

ETL

Hive

2021-01-11 发布于拉勾网

职位诱惑:

做五休二 商业保险 定期体检 弹性工作

职位描述:

岗位职责:

- 1.根据业务逻辑，进行SQL、HIVE等开发
- 2.支持业务部门的数据分析及数据查询需求
- 3.了解数据仓库、建模等相关技术知识

任职资格:

- 1.熟悉SQL，有HIVE，Spark SQL的优化能力
- 2.了解或熟悉Linux系统的使用
- 3.了解一数据处理技术，如Hadoop、Spark者优先
- 4.有广告，美妆行业的项目经验者优先
- 5.积极、主动，良好沟通能力和团队协作能力

工作地址

上海 - 黄浦区 - 黄陂北路227号中区广场3楼