安装mysql

- 解压`mysql-5.7.29-winx64.zip`到`C:\lab\common`
- 进入`C:\lab\common\mysql-5.7.29-winx64`，创建`data`文件夹，并创建`my.ini`

```
[mysqld]
#端口号
port = 3306
#mysql-5.7.29-winx64的路径
basedir=C:\\lab\\common\\mysql-5.7.29-winx64
#mysql-5.7.29-winx64的路径+\data
datadir=C:\\lab\\common\\mysql-5.7.29-winx64\\data
#最大连接数
max_connections=200
#编码
character-set-server=utf8

default-storage-engine=INNODB

sql_mode=NO_ENGINE_SUBSTITUTION,STRICT_TRANS_TABLES

[mysql]
#编码
default-character-set=utf8
```

- 管理员身份打开cmd，初始化mysql

```
cd C:\lab\common\mysql-5.7.29-winx64\bin

mysqld --initialize-insecure
```

- 注册mysql服务

```
mysqld -install

```

```
Service successfully installed
```

- 启动mysql服务

```
net start mysql

MySQL 服务正在启动 .
MySQL 服务已经启动成功。
```

- 修改默认账户密码

```
mysqladmin -u root password dnc.2009

mysqladmin: [Warning] Using a password on the command line interface can
Warning: Since password will be sent to server in plain text, use ssl cor
```

- 登录并添加远程访问权限，重启生效

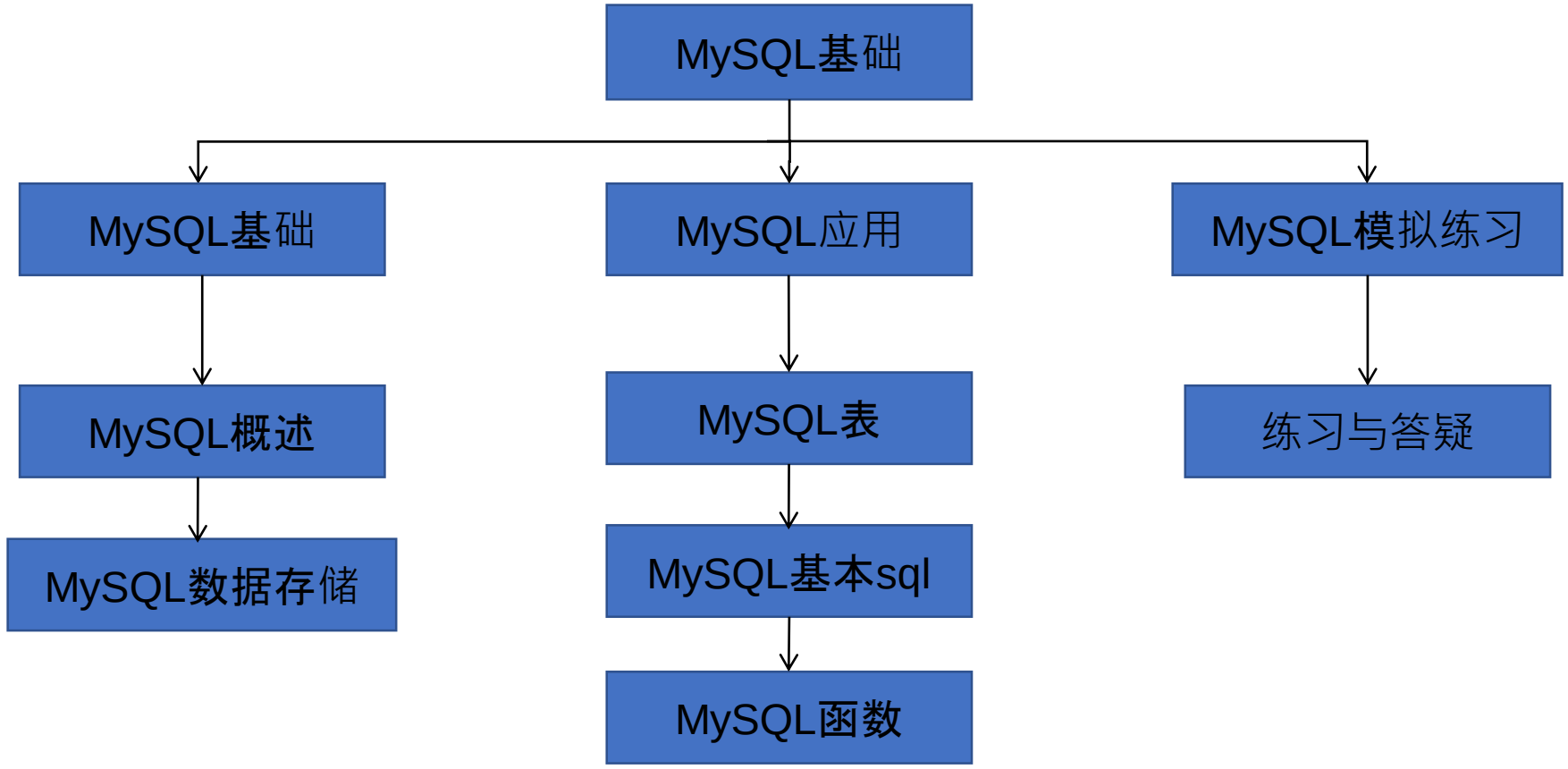
```
mysql -uroot -p

use mysql
update user set host='%' where user='root';
exit;
net stop mysql

MySQL 服务正在停止.
MySQL 服务已成功停止。

net start mysql

MySQL 服务正在启动 .
MySQL 服务已经启动成功。
```



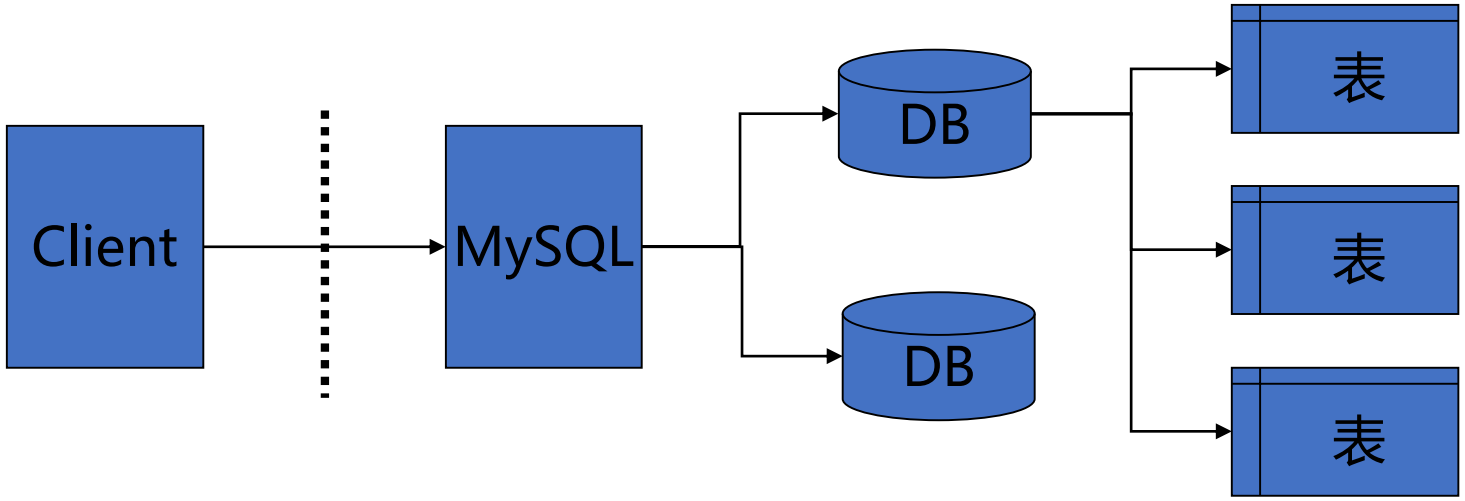
激活旧知

一步一练

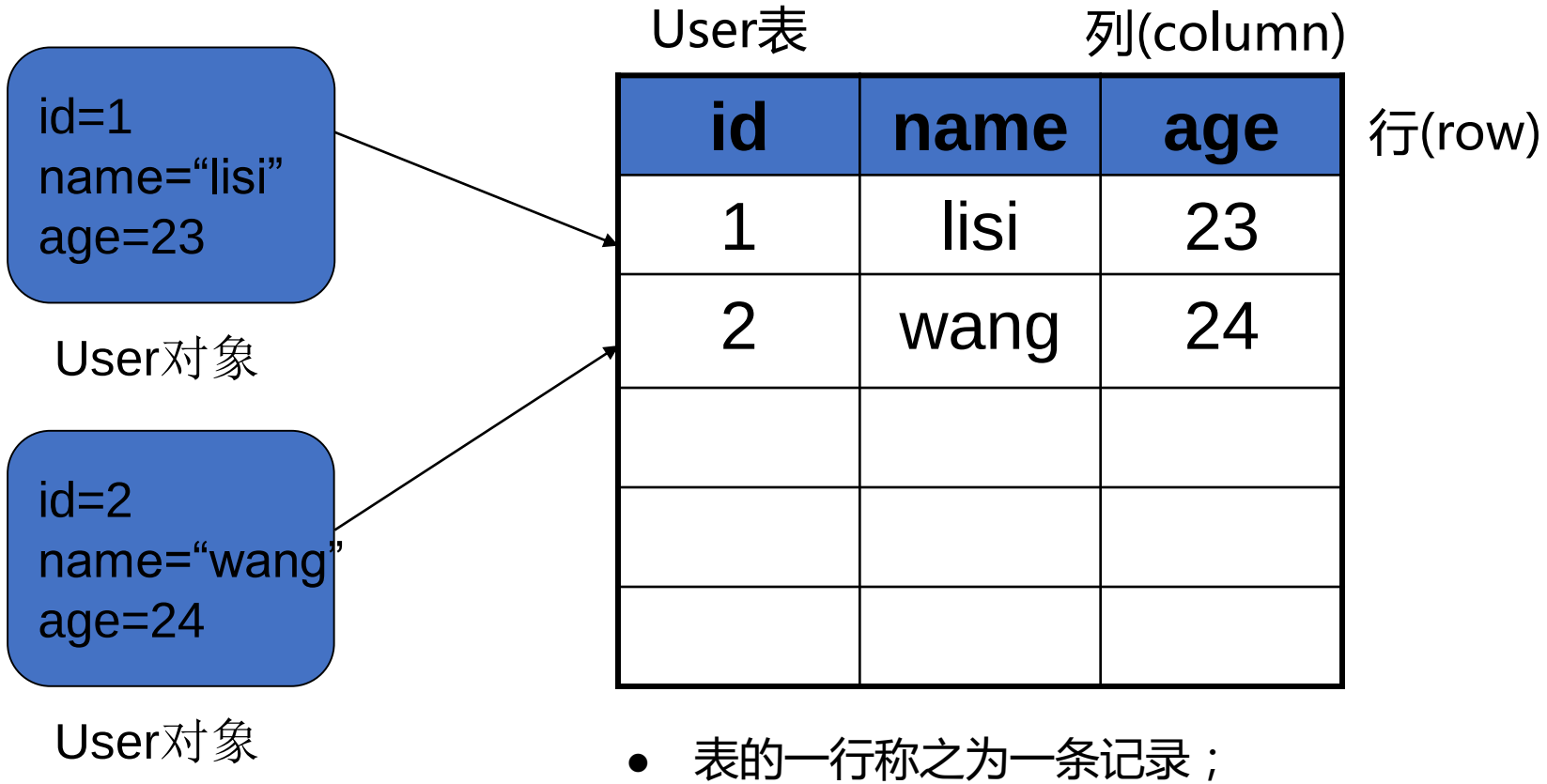
模拟实战

第一章 MySQL基础--概述

- 所谓安装数据库服务器，只是在机器上装了一个数据库管理程序，这个管理程序可以管理多个数据库，一般开发人员会针对每一个应用创建一个数据库;
- 为保存应用中实体的数据，一般会在数据库创建多个表，以保存程序中实体的数据;
- 数据库服务器、数据库和表的关系如图所示：



第一章 MySQL基础--数据存储



- 表的一行称之为一条记录；
- 表中一条记录对应一个java对象的数据；

创建数据库

```
CREATE  DATABASE  [IF NOT EXISTS] db_name
            [create_specification [, create_specification] ...]

create_specification:

    [DEFAULT] CHARACTER SET charset_name
|   [DEFAULT] COLLATE collation_name
```

- CHARACTER SET：指定数据库采用的字符集
- COLLATE：指定数据库字符集的比较方式
- 练习：
  - 创建一个名称为mydb1的数据库。
  - 创建一个使用utf-8字符集的mydb2数据库。
  - 创建一个使用utf-8字符集，并带校对规则的mydb3数据库。

修改、备份、恢复数据库

```
ALTER  DATABASE  [IF NOT EXISTS] db_name
                [alter_specification [, alter_specification] ...]

alter_specification:

    [DEFAULT] CHARACTER SET charset_name
|   [DEFAULT] COLLATE collation_name
```

- 备份数据库表中的数据
  mysqldump -u 用户名 -p 数据库名 > 文件名.sql
- 恢复数据库
  Source 文件名.sql
- 练习
  - 查看服务器中的数据库，并把其中某一个库的字符集修改为utf8;
  - 备份test库中的数据，并恢复

创建表(基本语句)

```
CREATE TABLE table_name
(
        field1  datatype,
        field2  datatype,
        field3  datatype,
)character set 字符集 collate 校对规则
```

field：指定列名　**datatype**：指定列类型

- 注意：创建表前，要先使用use db语句使用库。

- 注意：创建表时，要根据需保存的数据创建相应的列，并根据数据的类型定义相应的列类型。例：user对象

id　　int

name string

password string

birthday date

Id	Name	Password	birthday

MySQL常用数据类型

分类	数据类型	说明
数值类型	BIT(M) TINYINT [UNSIGNED] [ZEROFILL] BOOL，BOOLEAN SMALLINT [UNSIGNED] [ZEROFILL] INT [UNSIGNED] [ZEROFILL] BIGINT [UNSIGNED] [ZEROFILL] FLOAT[(*M*,*D*)] [UNSIGNED] [ZEROFILL] DOUBLE[(*M*,*D*)] [UNSIGNED] [ZEROFILL]	位类型。M指定位数，默认值1，范围1-64 带符号的范围是-128到127。无符号0到255。 使用0或1表示真或假 2的16次方 2的32次方 2的64次方 M指定显示长度，d指定小数位数 表示比float精度更大的小数
文本、二进制类型	CHAR(size) char(20) VARCHAR(size) varchar(20) BLOB LONGBLOB TEXT(clob) LONGTEXT(longclob)	固定长度字符串 可变长度字符串 二进制数据 大文本
时间日期	DATE/DATETIME/TimeStamp	日期类型(YYYY-MM-DD) (YYYY-MM-DD HH:MM:SS)，TimeStamp表示时间戳，它可用于自动记录insert、update操作的时间

- VARCHAR、BLOB和TEXT类是变长类型。每个类型的存储需求取决于列值的实际长度。

创建表练习

- 创建一个员工表

字段	属性
Id	整形
name	字符型
sex	字符型或bit型
brithday	日期型
Entry_date	日期型
job	字符型
Salary	小数型
resume	大文本型

修改表

使用 ALTER TABLE 语句追加, 修改, 或删除列的语法.

```
ALTER TABLE table
ADD         (column datatype [DEFAULT expr]
            [, column datatype]...);
```

```
ALTER TABLE table
MODIFY      (column datatype [DEFAULT expr]
            [, column datatype]...);
```

```
ALTER TABLE table
DROP       (column);
```

修改表的名称：**Rename table** 表名 **to** 新表名
修改表的字符集：alter table student character set utf8;

修改表

- 练习
  - 在上面员工表的基本上增加一个image列。
  - 修改job列，使其长度为60。
  - 删除sex列。
  - 表名改为user。
  - 修改表的字符集为utf-8
  - 列名name修改为username
    - alter table user **change** column name username varchar(20);

数据库CRUD语句

- Insert语句　(增加数据)
- Update语句　(更新数据)
- Delete语句　(删除数据)
- Select语句　(查找数据)

Insert语句

- 使用 INSERT 语句向表中插入数据。

```
INSERT INTO      table [(column [, column...])]
VALUES           (value [, value...]);
```

- 插入的数据应与字段的数据类型相同。
- 数据的大小应在列的规定范围内，例如：不能将一个长度为80的字符串加入到长度为40的列中。
- 在values中列出的数据位置必须与被加入的列的排列位置相对应。
- **字符和日期型数据应包含在单引号中。**
- 插入空值，不指定或insert into table value(null)

Insert语句练习

- 练习：使用insert语句向表中插入三个员工的信息。

字段名	字段类型
id	整形
name	字符串型
sex	字符或整数类型
birthday	日期型
salary	浮点型
entry_date	日期型
resume	大文本型

- 注意：字符和日期要包含在单引号中。
- show variables like 'character%';
- set character_set_results=gbk;

Update语句

- 使用 update语句修改表中数据。

```
UPDATE    tbl_name
          SET col_name1=expr1 [, col_name2=expr2 ...]
          [WHERE where_definition]
```

- UPDATE语法可以用新值更新原有表行中的各列。
- SET子句指示要修改哪些列和要给予哪些值。
- WHERE子句指定应更新哪些行。如没有WHERE子句，则更新所有的行。

Update语句练习

- 练习：在上面创建的employee表中修改表中的纪录。

id	name	gender	birthday	salary	entry_date	resume
1	zs	☺	1985-10-08	2000.1	2009-10-10	good employee
2	ls	☺	1976-10-08	5000.1	2007-10-10	good jobs
3	wu		1996-08-18	3000.1	2008-07-10	good job

- 要求
  - 将所有员工薪水修改为5000元。
  - 将姓名为’zs’的员工薪水修改为3000元。
  - 将姓名为’aaa’的员工薪水修改为4000元,job改为ccc。
  - 将wu的薪水在原有基础上增加1000元。

Delete语句

- 使用 delete语句删除表中数据。

```
delete from tbl_name
            [WHERE where_definition]
```

- 如果不使用where子句，将删除表中所有数据。
- Delete语句不能删除某一列的值（可使用update）
- 使用delete语句仅删除记录，不删除表本身。如要删除表，使用drop table语句。
- 同insert和update一样，从一个表中删除记录将引起其它表的参照完整性问题，在修改数据库数据时，头脑中应该始终不要忘记这个潜在的问题。
- 删除表中数据也可使用TRUNCATE TABLE 语句，它和delete有所不同，参看mysql文档。

Delete语句练习

- 删除表中名称为’zs’的记录。
- 删除表中所有记录。
- 使用truncate删除表中记录。

Select语句(1)

- 基本select语句

```
SELECT [DISTINCT] *|{column1, column2. column3..}
                  FROM    table;
```

- Select 指定查询哪些列的数据。
- column指定列名。
- *号代表查询所有列。
- From指定查询哪张表。
- DISTINCT可选，指显示结果时，是否剔除重复数据

Select语句(1)

- 练习：
  - 查询表中所有学生的姓名。
  - 查询表中所有学生的姓名和对应的英语成绩。
  - 过滤表中重复数据。

Select语句(2)

- 在select语句中可使用表达式对查询的列进行运算

```
SELECT *|{column1 | expression, column2 | expression , ..}
                    FROM    table;
```

- 在select语句中可使用as语句

```
SELECT column as 别名 from 表名;
```

Select语句(2)

- 练习
  - 在所有学生分数上加10分特长分。
  - 统计每个学生的总分。
  - 使用别名表示学生总分。

Select语句(3)

- 使用where子句，进行过滤查询。练习：
  - 查询姓名为wu的学生成绩
  - 查询英语成绩大于90分的同学
  - 查询总分大于200分的所有同学

第二章 MySQL应用--CRUD语句

Select语句(4)

- 在where子句中经常使用的运算符

比较运算符	> < <= >= = <>	大于、小于、大于(小于)等于、不等于
	BETWEEN ...AND...	显示在某一区间的值
	IN(set)	显示在in列表中的值，例：in(100,200)
	LIKE '张pattern'	模糊查询
	IS NULL	判断是否为空
逻辑运算符	and	多个条件同时成立
	or	多个条件任一成立
	not	不成立，例：where not(salary>100);

Like语句中，% 代表零个或多个任意字符，_ 代表一个字符，例first_name like '_a%'；

Select语句(4)

- 查询英语分数在 80 - 90之间的同学。
- 查询数学分数为89,90,91的同学。
- 查询所有姓李的学生成绩。
- 查询数学分>80，语文分>80的同学。

第二章 MySQL应用--CRUD语句

Select语句(5)

- 使用order by 子句排序查询结果。

```
SELECT column1, column2. column3..
                FROM    table;
                order by column asc/desc
```

- Order by 指定排序的列，排序的列即可是表中的列名，也可以是select语句后指定的列名。
- Asc 升序、Desc 降序
- ORDER BY 子句应位于SELECT语句的结尾。
- 练习： 对数学成绩排序后输出。
  对总分排序后输出，然后再按从高到低的顺序输出
  对姓李的学生成绩排序输出

Select语句(6)

- 使用group by 子句对列进行分组

```
SELECT column1, column2. column3.. FROM table;
                    group by column
```

- 练习：对订单表中商品归类后，显示每一类商品的总价
- 使用having 子句过滤

```
SELECT column1, column2. column3..
                    FROM table;
                    group by column having ...
```

- 练习：查询购买了几类商品，并且每类总价大于100的商品
- **Having和where均可实现过滤，但在having可以使用合计函数,having通常跟在group by后，它作用于组。**

合计函数 - count

- Count(列名)返回某一列，行的总数

Select count(*)|count(列名) from tablename

[WHERE *where_definition*]

- 练习：
  - 统计一个班级共有多少学生？
  - 统计数学成绩大于90的学生有多少个？
  - 统计总分大于250的人数有多少？

合计函数 - SUM

- Sum函数返回满足where条件的行的和

```
Select sum(列名) {,sum(列名)…} from tablename
                [WHERE where_definition]
```

- 练习：
  - 统计一个班级数学总成绩？
  - 统计一个班级语文、英语、数学各科的总成绩
  - 统计一个班级语文、英语、数学的成绩总和
  - 统计一个班级语文成绩平均分
- 注意：sum仅对数值起作用，否则会报错。
- 注意：对多列求和，“，”号不能少。

合计函数 - AVG

- AVG函数返回满足where条件的一列的平均值

```
Select sum(列名) { ,sum(列名)... }  from tablename
                [WHERE where_definition]
```

- 练习：
  - 求一个班级数学平均分？
  - 求一个班级总分平均分

合计函数 - MAX/MIN

- Max/min函数返回满足where条件的一列的最大/最小值

Select max(列名) from tablename
[WHERE *where_definition*]

- 练习：
  - 求班级最高分和最低分（数值范围在统计中特别有用）

第二章 MySQL应用--常用函数

时间日期相关函数

ADDTIME (date2 ,time_interval)	将**time_interval**加到**date2**
CURRENT_DATE ( )	当前日期
CURRENT_TIME ( )	当前时间
CURRENT_TIMESTAMP ( )	当前时间戳
DATE (datetime)	返回**datetime**的日期部分
DATE_ADD (date2 , INTERVAL d_value d_type)	在**date2**中加上日期或时间
DATE_SUB (date2 , INTERVAL d_value d_type)	在**date2**上减去一个时间
DATEDIFF (date1 ,date2)	两个日期差
NOW ( )	当前时间
YEAR|Month|DATE (datetime)	年月日

示例：select addtime(‘02:30:30’ , ‘01:01:01’); 注意：字符串、时间日期的引号问题

select date_add(entry_date,INTERVAL 2 year) from student; / / 增加两年

select addtime(time, ‘1 1-1 10:09:09’) from student; / / 时间戳上增加，注意年后没有-

第二章 MySQL应用--常用函数

字符串相关函数

CHARSET(str)	返回字串字符集
CONCAT (string2 [,...])	连接字串
INSTR (string ,substring)	返回**substring**在**string**中出现的位置,没有返回**0**
UCASE (string2)	转换成大写
LCASE (string2)	转换成小写
LEFT (string2 ,length)	从**string2**中的左边起取**length**个字符
LENGTH (string)	**string**长度
REPLACE (str ,search_str ,replace_str)	在**str**中用**replace_str**替换**search_str**
STRCMP (string1 ,string2)	逐字符比较两字串大小,
SUBSTRING (str , position [,length])	从**str**的**position**开始,取**length**个字符
LTRIM (string2) RTRIM (string2) trim	去除前端空格或后端空格

第二章 MySQL应用--常用函数

数学相关函数

ABS (number2)	绝对值
BIN (decimal_number)	十进制转二进制
CEILING (number2)	向上取整
CONV(number2,from_base,to_base)	进制转换
FLOOR (number2)	向下取整
FORMAT (number,decimal_places)	保留小数位数
HEX (DecimalNumber)	转十六进制
LEAST (number , number2 [,..])	求最小值
MOD (numerator ,denominator)	求余
RAND([seed])	**RAND([seed])**

第二章 MySQL应用--注意点

Tip：定义表的约束

- 定义主键约束
  - primary key:不允许为空，不允许重复
    - ●删除主键：alter table tablename drop primary key ;
- 定义主键自动增长
  - auto_increment
- 定义唯一约束
  - unique
- 定义非空约束
  - not null
- 定义外键约束
  - ●constraint ordersid_FK foreign key(ordersid) references orders(id),

第二章 MySQL应用--注意点

Tip：mysql中文乱码

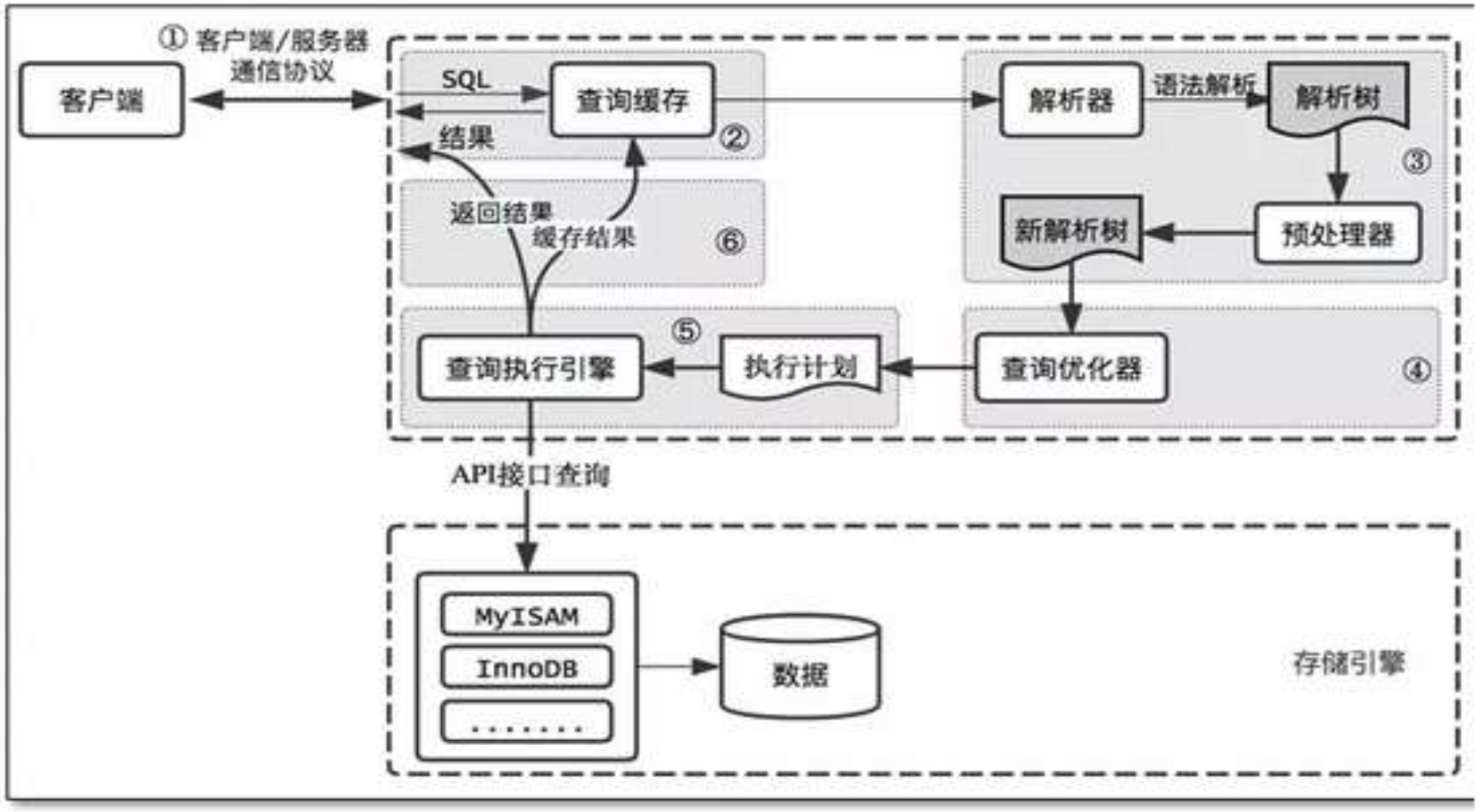
- mysql有六处使用了字符集，分别为：client 、connection、database、results、server 、system。
  - client是客户端使用的字符集。
  - connection是连接数据库的字符集设置类型，如果程序没有指明连接数据库使用的字符集类型就按照服务器端默认的字符集设置。
  - database是数据库服务器中某个库使用的字符集设定，如果建库时没有指明，将使用服务器安装时指定的字符集设置。
  - results是数据库给客户端返回时使用的字符集设定，如果没有指明，使用服务器默认的字符集。
  - server是服务器安装时指定的默认字符集设定。
  - system是数据库系统使用的字符集设定。

实战演练

①回顾并消化知识。

②完成员工表中数据的增删改查操作。

SQL查询过程



MySQL 查询过程

优化维度

数据库优化维度有如下四个：

- **硬件**
- **系统配置**
- **数据库表结构**
- **SQL 及索引**

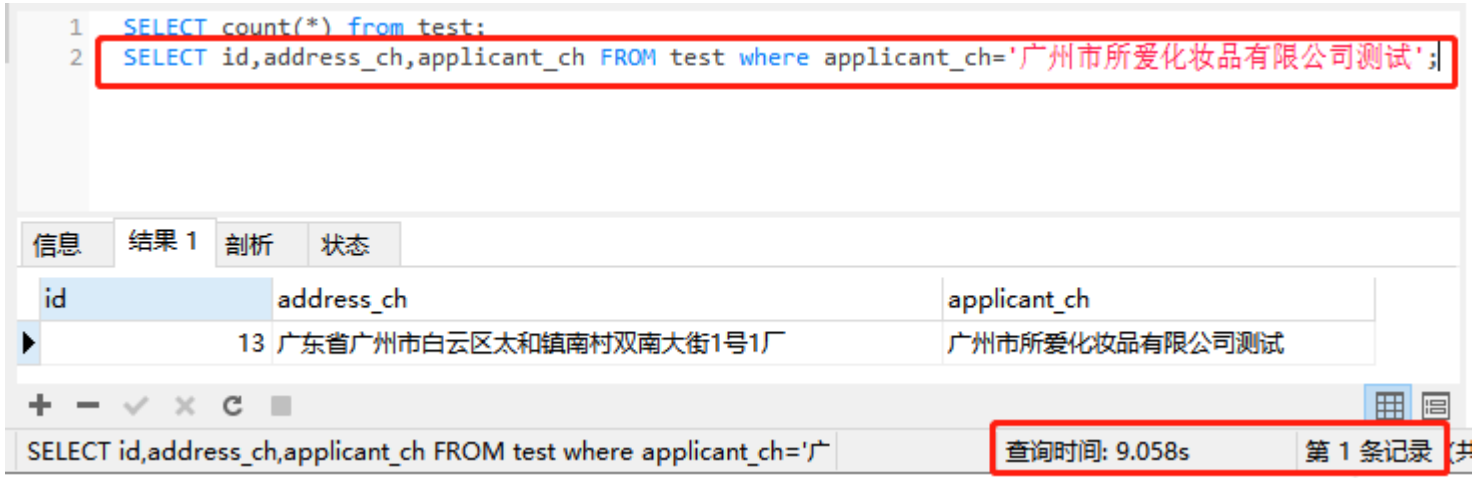
优化选择：

- **优化成本**：硬件>系统配置>数据库表结构>SQL 及索引。
- **优化效果**：硬件<系统配置<数据库表结构<SQL 及索引。

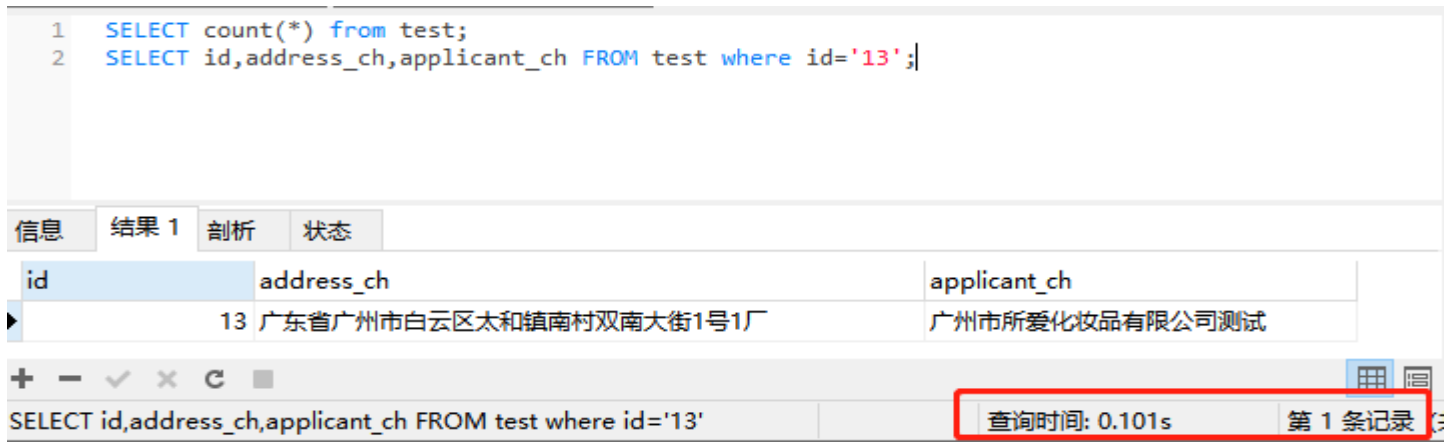
优化前后查询结果耗时对比

注意：表内数据量为300万条

优化之前：



优化后：



学会分析SQL执行计划

通过EXPLAIN分析慢SQL

```
17
18    EXPLAIN      SELECT id,address_ch,applicant_ch FROM test WHERE applicant_ch ='广州市所爱化妆品有限公司测试'
```

信息 | 结果 1 | 剖析 | 状态

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	test	(Null)	ALL	(Null)	(Null)	(Null)	(Null)	2720963	10.00	Using where

```
20
21    EXPLAIN      SELECT id,address_ch,applicant_ch FROM test WHERE id='13'
```

信息 | 结果 1 | 剖析 | 状态

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	test	(Null)	const	PRIMARY	PRIMAR	4	const	1	100.00	(Null)

type**列**，连接类型。一个好的SQL语句至少要达到range级别。杜绝出现all级别。
key**列**，**使用到的索引名。如果没有**选择索引，值是。可以采取强制索引方式。
key_len**列**，**索引**长度。
rows**列**，扫描行数。该值是个预估值。
extra**列**，详细说明。注意，常见的不太友好的值，如下：Using filesort，Using temporary。

查询类型择优

select_type：表示SELECT的类型，常见的取值有SIMPLE(简单表，即不使用表连接或者子查询)、PRIMARY(主查询，即外层的查询)、UNION(UNION中的第二个或者后面的查询语句)、SUBQUERY(子查询中的第一个SELECT)等。

type：表示MySQL在表中找到所需行的方式，或者叫访问类型

- 常见的有：all< index < range < ref < eq_ref < const/system < null，性能由最差到最好。
- type=ALL：全表扫描。
- type=index：索引全扫描，MySQL遍历整个索引来查询。
- type=range：索引范围扫描，常见于<、<=、>、 >=、 between。
- type=ref：使用非唯一索引扫描或唯一索引的前缀扫描，返回匹配某个单独值的记录。
- type=eq_ref：类似ref，区别就在使用的索引是唯一索引，对于每个索引键值，表中只有一条记录匹配，简单来说，就是多表连接中使用primary key或者unique index作为关联条件。
- type=const/system：单表中最多有一个匹配行，查询起来非常迅速，一般主键primary key或者唯一索引unique index进行的查询，通过唯一索引uk_email访问的时候，类型type为const；而从我们构造的仅有一条记录的a表中检索时，类型type为system。
- type=null：MySQL不用访问表或者索引，就能直接得到结果。

通过索引提高查询效率

什么是索引 ？

MySQL官方对索引的定义为：索引(Index)是帮助MySQL高效获取数据的数据结构。我们可以简单理解为：快速查找排好序的一种数据结构。

哪些字段适合建立索引：

1. 表的某个字段值得离散度越高，该字段越适合选作索引的关键
2. Where 子句中经常使用的字段应该创建索引
3. 查询中与其他表关联的字段，外键关系建立索引

哪些字段不适合建立索引:

1. 重复度高的字段，例如性别
2. 索引应该建在小字段上，对于大的文本字段甚至超长字段，不要建索引
3. 更新频繁的字段不适合创建索引

工作中几种常见的优化

1.SQL语句中IN包含的值不应过多

MySQL对于IN做了相应的优化，即将IN中的常量全部存储在一个数组里面，而且这个数组是排好序的。但是如果数值较多，产生的消耗也是比较大的。
再例如：select id from t where num in(1,2,3) 对于连续的数值，能用between就不要用in了；再或者使用连接来替换

2.SELECT语句务必指明字段名称
SELECT*增加很多不必要的消耗（CPU、IO、内存、网络带宽）；增加了使用覆盖索引的可能性；当表结构发生改变时，前断也需要更新。所以要求直接在select后面接上字段名。
从数据库里读出越多的数据，那么查询就会变得越慢。并且，如果你的数据库服务器和WEB服务器是两台独立的服务器的话，这还会增加网络传输的负载。所以，你应该养成一个需要什么就取什么的好的习惯。

工作中几种常见的优化

3.当只需要一条数据的时候，使用limit 1

这是为了使EXPLAIN中type列达到const类型

4.如果排序字段没有用到索引，就尽量少排序

5.如果限制条件中其他字段没有索引，尽量少用or

or两边的字段中，如果有一个不是索引字段，而其他条件也不是索引字段，会造成该查询不走索引的情况。很多时候使用union all或者是union（必要的时候）的方式来代替“or”会得到更好的效果。

6.尽量用union all代替union

union和union all的差异主要是前者需要将结果集合并后再进行唯一性过滤操作，这就会涉及到排序，增加大量的CPU运算，加大资源消耗及延迟。当然，union all的前提条件是两个结果集没有重复数据。

工作中几种常见的优化

7.不使用ORDER BY RAND

select id from `dynamic` order by rand limit 1000;

上面的SQL语句，可优化为：

select id from `dynamic` t1 join (select rand * (select max(id) from `dynamic`) as nid) t2 on t1.id > t2.nidlimit 1000;

工作中几种常见的优化

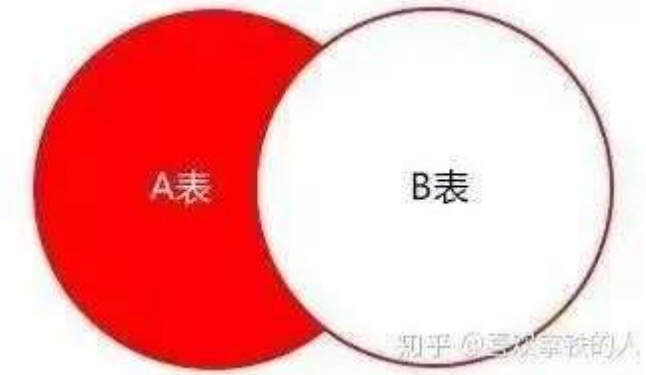
8.区分in和exists、not in和not existsselect * from 表A where id in (select id from 表B)上面SQL语句相当于select * from 表A where exists(select * from 表B where 表B.id=表A.id)区分in和exists主要是造成了驱动顺序的改变（这是性能变化的关键），如果是exists，那么以外层表为驱动表，先被访问，如果是IN，那么先执行子查询。所以IN适合于外表大而内表小的情况；EXISTS适合于外表小而内表大的情况。关于not in和not exists，推荐使用not exists，不仅仅是效率问题，not in可能存在逻辑问题。如何高效的写出一个替代not exists的SQL语句？

原SQL语句：select colname … from A表 where a.id not in (select b.id from B表)

高效的SQL语句：

select colname … from A表 Left join B表

on where a.id = b.id where b.id is



工作中几种常见的优化

9.使用合理的分页方式以提高分页的效率

select id,name from product limit 866613, 20

使用上述SQL语句做分页的时候，可能有人会发现，随着表数据量的增加，直接使用limit分页查询会越来越慢。

优化的方法如下：可以取前一页的最大行数的id，然后根据这个最大的id来限制下一页的起点。比如此列中，上一页最大的id是866612。SQL可以采用如下的写法：

select id,name from product where id> 866612 limit 20

10.分段查询

在一些用户选择页面中，可能一些用户选择的时间范围过大，造成查询缓慢。主要的原因是扫描行数过多。这个时候可以通过程序，分段进行查询，循环遍历，将结果合并处理进行展示。

如下图这个SQL语句，扫描的行数成百万级以上的时候就可以使用分段查询：

工作中几种常见的优化

11.避免在where子句中对字段进行值判断

对于的判断会导致引擎放弃使用索引而进行全表扫描。

不建议使用%前缀模糊查询

例如LIKE“%name”或者LIKE“%name%”，这种查询会导致索引失效而进行全表扫描。但是可以使用LIKE“name%”。

12.避免在where子句中对字段进行表达式操作

比如：

```
select user_id,user_project from user_base where age*2=36;
```

中对字段就行了算术运算，这会造成引擎放弃使用索引，建议改成：

```
select user_id,user_project from user_base where age=36/2;
```

如何避免低效率SQL

- 使用连接（JOIN）来代替子查询(Sub-Queries)

```sql
SELECT
  CustomerID
FROM
  customerinfo
  LEFT JOIN salesinfo ON customerinfo.CustomerID = salesinfo.CustomerID
WHERE
  salesinfo.CustomerID IS NULL
```

```sql
SELECT
  CustomerID
FROM
  customerinfo
WHERE
  CustomerID NOT IN (SELECT CustomerID FROM salesinfo )
```

写SQL时尽量不要对字段进行运算操作

- 请看下面SQL的执行效率：

```
select * from iroomtypeprice where amount/30< 1000
```
（11秒）

- 当改写成下面的SQL语句时效率明显提高了：

```
select * from iroomtypeprice where amount< 1000*30
```
（<1秒）

- **尽可能的使用 NOT NULL**
  除非你有一个很特别的原因去使用 NULL 值，你应该总是让你的字段保持 NOT NULL。这看起来好像有点争议，请往下看。首先，问问你自己“Empty”和“NULL”有多大的区别（如果是INT，那就是0和NULL）？如果你觉得它们之间没有什么区别，那么你就不要使用NULL。（你知道吗？在 Oracle 里，NULL 和 Empty 的字符串是一样的！）
- 不要以为 NULL 不需要空间，其需要额外的空间，并且，在你进行比较的时候，你的程序会更复杂。 当然，这里并不是说你就不能使用NULL了，现实情况是很复杂的，依然会有些情况下，你需要使用NULL值。
- 下面摘自MySQL自己的文档:
- NULL columns require additional space in the row to record whether their values are NULL. For MyISAM tables, each NULL column takes one bit extra, rounded up to the nearest byte

注意索引被弃用

- **like语句优化**
  SELECT id FROM A WHERE name like '%abc% ‘
- 由于abc前面用了“%”，因此该查询必然走全表查询，除非必要，否则不要在关键词前加%，优化成如下
- **SELECT id FROM A WHERE name like 'abc% ‘**
- 在where子句中使用 ！= 或 <>操作符，索引将被放弃使用，会进行全表查询。
- **如SQL:SELECT id FROM A WHERE ID != 5**
-
- **优化成：SELECT id FROM A WHERE ID>5 OR ID<5**

- 很多时候使用union all 或 nuin(必要的时候)的方式替换“or”会得到更好的效果。where子句中使用了or,索引将被放弃使用。
- 如SQL:SELECT id FROM A WHERE num =10 or num = 20
-
- 优化成：SELECT id FROM A WHERE num = 10 union all SELECT id FROM A WHERE num=20
- in和not in 也要慎用，否则也会导致全表扫描
- 如SQL:SELECT id FROM A WHERE num in(1,2,3)
-
- 优化成：SELECT id FROM A WHERE num between 1 and 3

实际工作中遇到问题解决思路

- **一般应急调优的思路：**针对突然的业务办理卡顿，无法进行正常的业务处理，需要马上解决的场景。
- 1、show processlist
- 2、explain select id ,name from stu where name='clsn'; # ALL id name age sex
- select id,name from stu where id=2-1 函数 结果集>30;
- show index from table;
- 3、通过执行计划判断，索引问题（有没有、合不合理）或者语句本身问题
- 4、show status like '%lock%'; # 查询锁状态
- kill SESSION_ID; # 杀掉有问题的session

实际工作中遇到问题解决思路

- **常规调优思路：**针对业务周期性的卡顿，例如在每天 10-11 点业务特别慢，但是还能够使用，过了这段时间就好了。
- 1）查看slowlog，分析slowlog，分析出查询慢的语句；
- 2）按照一定优先级，一个一个排查所有慢语句；
- 3）分析top SQL，进行explain调试，查看语句执行时间；
- 4）调整索引或语句本身。

- 造成查询慢的一些因素
- 1、没有索引或者没有用到索引(这是查询慢最常见的问题，是程序设计的缺陷)
- 2、I/O吞吐量小，形成了瓶颈效应。
- 3、没有创建计算列导致查询不优化。
- 4、内存不足
- 5、网络速度慢
- 6、查询出的数据量过大（可以采用多次查询，其他的方法降低数据量）
- 7、锁或者死锁(这也是查询慢最常见的问题，是程序设计的缺陷)
- 8、sp_lock,sp_who,活动的用户查看,原因是读写竞争资源。
- 9、返回了不必要的行和列
- 10、查询语句不好，没有优化

总结

- 了解MYSQL优化的四个方向
- 避免写出低效SQL
- 遇到特备慢的SQL 如何分析优化