MySQL

Go链接SQL

增

```
func main() {
    db, err := sql.Open("mysql", "root:1234@tcp(127.0.0.1:3306)/first")
    if err != nil {
       panic(err)
    }
    defer db.Close()

    stmt, err := db.Prepare("INSERT INTO user(username,password) values(?,?)")
    if err != nil {
       panic(err)
    }
    defer stmt.Close()

    _, err = stmt.Exec("gaojiyuan", "123456dwadawdaw")
    if err != nil {
       panic(err)
    }
    fmt.Println("输入成功")
}
```

改

```
func main() {
    db, err := sql.Open("mysql", "root:1234@tcp(127.0.0.1:3306)/first")
    if err != nil {
       fmt.Println(err)
    }
    defer db.Close()

    stmt, err := db.Prepare("UPDATE user SET username=?,password=? where id=?")
    if err != nil {
       fmt.Println(err)
    }
    defer stmt.Close()
```

```
13
14      r, _ := stmt.Exec("liu", "1234", 1)
15      fmt.Println("succfesse")
16  }
```

删

```
1  func main() {
2      db, err := sql.Open("mysql", "root:1234@tcp(127.0.0.1:3306)/first")
3      if err != nil {
4          fmt.Println(err)
5      }
6      defer db.Close()
7
8      stmt, err := db.Prepare("delete from user where id=?")
9      if err != nil {
10         fmt.Println(err)
11     }
12     defer stmt.Close()
13
14     _, _ = stmt.Exec(1)
15     if err != nil {
16         fmt.Println(err)
17     }
18     fmt.Println("ok")
19 }
```

查

```
1  func main() {
2      db, err := sql.Open("mysql", "root:1234@tcp(127.0.0.1:3306)/first")
3      if err != nil {
4          fmt.Println(err)
5      }
6      defer db.Close()
7
8      stmt, err := db.Prepare("select * from user")
9      if err != nil {
10         fmt.Println(err)
11     }
12     defer stmt.Close()
13
14     rows, err := stmt.Query()
```

```
    for rows.Next() {
        var id int
        var username string
        var password string
        rows.Scan(&id, &username, &password)
        if username == "gaojiyuan" {
            fmt.Println(id, username, password)
        }
    }
}
```

SQL(数据查询语言)

https://blog.csdn.net/qq_45173404/article/details/115712758?ops_request_misc=%257B%2522request%255Fid%2522%253A%252217216963101680017856616969%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=172169631016800178566169&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-4-115712758-null-null.142^v100^pc_search_result_base1&utm_term=mysql&spm=1018.2226.3001.4187

关系型数据库：SQL（Structured Query(查询) Language）

- MySQL、Oracle、Sql Server、DB2、SQLlite
- 通过表和表之间，行和列之间的关系进行数据的存储
- 通过外键关联来建立表与表之间的关系

常用术语

- **数据库:** 数据库是一些关联表的集合。
- **数据表:** 表是数据的矩阵。在一个数据库中的表看起来像一个简单的电子表格。
- **列:** 一列(数据元素) 包含了相同类型的数据, 例如邮政编码的数据。
- **行：**一行（元组，或记录）是一组相关的数据，例如一条用户订阅的数据。
- **冗余**：存储两倍数据，冗余降低了性能，但提高了数据的安全性。
- **主键**：主键是唯一的。一个数据表中只能包含一个主键。你可以使用主键来查询数据。
- **外键：**外键用于关联两个表。
- **复合键**：复合键（组合键）将多个列作为一个索引键，一般用于复合索引。

- **索引：**使用索引可快速访问数据库表中的特定信息。索引是对数据库表中一列或多列的值进行排序的一种结构。类似于书籍的目录。
- **参照完整性:** 参照的完整性要求关系中不允许引用不存在的实体。与实体完整性是关系模型必须满足的完整性约束条件，目的是保证数据的一致性。

Database(db)

先建立数据库,然后使用,创立数据库的常用语句

创造数据库:CREATE DATABASE [IF NOT EXISTS] 数据库名;

删除数据库:DROP DATABASE [if EXISTS] 数据库名;

使用数据库:use 数据库名;

查看数据库:SHOW DATABASES;

数据库的列类型

数据存储的数据的相关范围:

数值类型:

数据类型	描述	大小
tinyint	十分小的数据	1个字节
smallint	较小的数据	2个字节
mediumint	中等大小的数据	3个字节
int	标准的整数	4个字节
bigint	较大的数据	8个字节
float	浮点数	4个字节
double	浮点数	8个字节
decimal	字符串形式的浮点数，一般用于金融计算	

字符串:

数据类型	描述	大小
char	字符串固定大小	0~255
varchar	可变字符串	0~65535
tinytext	微型文本	2^8-1
text	文本串	2^16-1

时间日期:

数据类型	描述	格式
date	日期格式	YYYY-MM-DD
time	时间格式	HH：mm：ss
datetime	最常用的时间格式	YYYY-MM-DD HH：mm：ss
timestamp	时间戳，1970.1.1到现在的毫秒数	
year	年份表示	

Null:

没有值,未知,不参与计算

数据库字段属性

unsigned:无符号的,不能成为负数;

zerofill:由0填充的;

Aoto_InCrement:自增的

NULL:空;

NOT NULL:必须有值;

DEFAULT:默认的.用于设置默认值;

扩展:每一个表必有的五个字段:

拓展：每一个表，都必须存在以下五个字段：

名称	描述
id	主键
version	乐观锁
is_delete	伪删除
gmt_create	创建时间
gmt_update	修改时间

创建数据库表:

```
CREATE TABLE IF NOT EXISTS `student`(
        `id` INT(4)         NOT NULL AUTO_INCREMENT COMMENT '学号',
        `name` VARCHAR(30) NOT NULL DEFAULT '匿名' COMMENT '姓名',
        `pwd` VARCHAR(20) NOT NULL DEFAULT '123456' COMMENT '密码',
        `sex` VARCHAR(2) NOT NULL DEFAULT '女' COMMENT '性别',
        `birthday` DATETIME DEFAULT NULL COMMENT '出生日期',
        `address` VARCHAR(100) DEFAULT NULL COMMENT '家庭住址',
        `email` VARCHAR(50) DEFAULT NULL COMMENT '邮箱',
        PRIMARY KEY (`id`)
)ENGINE=INNODB DEFAULT CHARSET=utf8
```

展示数据库表:

SHOW CREATE DATABASE 数据库名;*-- 查看创建数据库的语句*
SHOW CREATE TABLE 表名;*-- 查看表的定义语句*
DESC 表名;*-- 显示表的具体结构*

数据库存储位置:

MySQL数据表以文件方式存放在磁盘中

- 包括表文件 , 数据文件 , 以及数据库的选项文件
- 位置 : `Mysql安装目录\data\`

修改数据库:

```
-- 修改表名
-- ALTER TABLE 旧表名 RENAME AS 新表名
ALTER TABLE teacher RENAME AS teachers;
```

```sql

-- 增加表的字段
-- ALTER TABLE 表名 ADD 字段名 列属性
ALTER TABLE teachers ADD age INT(11);

-- 修改表的字段(重命名，修改约束)
-- ALTER TABLE 表名 MODIFY 字段名 [列属性];
ALTER TABLE teachers MODIFY age VARCHAR(11);-- 修改约束
-- ALTER TABLE 表名 CHANGE 旧名字 新名字 [列属性];
ALTER TABLE teachers CHANGE age age1 INT(1);-- 字段重命名

-- 删除表的字段
-- ALTER TABLE 表名 DROP 字段名
ALTER TABLE teachers DROP age1;

```

约束表

外键:约束:CONSTRAINT 关联:REFERENCES

增添外键的方法

直接在创建的时候添加

后补

```sql
alter table emp add constraint fk_emp_dept_id foreign key (dept_id)
references dept(id)


ALTER TABLE child_table_name
ADD CONSTRAINT fk_constraint_name
FOREIGN KEY (child_column_name)
REFERENCES parent_table_name(parent_column_name);
```

- `child_table_name`：要添加外键的子表格。
- `fk_constraint_name`：外键约束的名称，用于标识这个约束。
- `child_column_name`：子表格中将用作外键的列。
- `parent_table_name`：父表格的名称，即被参考的表格。
- `parent_column_name`：父表格中作为主键的列，子表格的外键将与此列进行关联

```
CONSTRAINT(约束) `FK_gradeid` FOREIGN KEY (`gradeid`) REFERENCES(关联)
`grade`(`gradeid`)
```

```
CREATE TABLE IF NOT EXISTS `student`(
        `id` INT(4) NOT NULL AUTO_INCREMENT COMMENT '学号',
        `name` VARCHAR(30) NOT NULL DEFAULT '匿名' COMMENT '姓名',
        `pwd` VARCHAR(20) NOT NULL DEFAULT '123456' COMMENT '密码',
        `sex` VARCHAR(2)        NOT NULL DEFAULT '女' COMMENT '性别',
        `birthday` DATETIME DEFAULT NULL COMMENT '出生日期',
        `address` VARCHAR(100) DEFAULT NULL COMMENT '家庭住址',
        `email` VARCHAR(50) DEFAULT NULL COMMENT '邮箱',
        `gradeid` INT(10) NOT NULL COMMENT '学生的年级',
        PRIMARY KEY (`id`),
        KEY `FK_gradeid` (`gradeid`),
        CONSTRAINT `FK_gradeid` FOREIGN KEY (`gradeid`) REFERENCES
`grade`(`gradeid`)
)ENGINE=INNODB DEFAULT CHARSET=utf8

-- 创建年级表
CREATE TABLE `grade`(
        `gradeid` INT(10) NOT NULL COMMENT '年级id',
        `gradename` VARCHAR(50) NOT NULL COMMENT '年纪名称',
        PRIMARY KEY (`gradeid`)
)ENGINE=INNODB DEFAULT CHARSET=utf8
```

删除外键

```
alter table 表名 drop foreign key 外键名称
```

删除有外键关系的表的时候，必须要先删除引用别人的表（从表），再删除被引用的表（主表）

DML语句

`Data Manipulation Luaguge`：数据操作语言

Insert

插入语句语法:

```
INSERT INTO 表名([字段1,字段2..])VALUES('值1','值2'..),[('值1','值2'..)..];
```

```
2
```

eg.

```
-- 普通用法
INSERT INTO `student`(`name`) VALUES ('zsr');

-- 插入多条数据
INSERT INTO `student`(`name`,`pwd`,`sex`) VALUES ('zsr','200024','男'),
('gcc','000421','女');

-- 省略字段
INSERT INTO `student` VALUES (5,'Bareth','123456','男','2000-02-04','武
汉','1412@qq.com',1);

```

updata

更新语句语法:

```
UPDATE 表名 SET 字段1=值1,[字段2=值2...] WHERE 条件[];
```

实例:

```
-- 修改学员名字,指定条件
UPDATE `student` SET `name`='zsr204' WHERE id=1;

-- 不指定条件的情况,会改动所有表
UPDATE `student` SET `name`='zsr204';

-- 修改多个属性
UPDATE `student` SET `name`='zsr',`address`='湖北' WHERE id=1;

-- 通过多个条件定位数据
UPDATE `student` SET `name`='zsr204' WHERE `name`='zsr' AND `pwd`='200024';

```

Delete

删除语句:

```
DELETE FROM 表名 [WHERE 条件]
```

TRUNCATE

删除整个表

where条件语句

操作符	含义
=	等于
<>或!=	不等于
>	大于
<	小于
<=	小于等于
>=	大于等于
BETWEEN... AND...	闭合区间
AND	和
OR	或

DQL

`Data QueryLanguage` 数据查询语言

数据库查询语句实例:

```
SELECT [ALL | DISTINCT]
{* | table.* | [table.field1[as alias1][,table.field2[as alias2]][,...]]}
FROM table_name [as table_alias]
  [left | right | inner join table_name2]  -- 联合查询
  [WHERE ...]  -- 指定结果需满足的条件
  [GROUP BY ...]  -- 指定结果按照哪几个字段来分组
  [HAVING]  -- 过滤分组的记录必须满足的次要条件
  [ORDER BY ...]  -- 指定查询记录按一个或多个条件排序
  [LIMIT {[offset,]row_count | row_countOFFSET offset}]; -- 指定查询的记录从哪条至哪条
```

基础查询

基础语法;

```
SELECT 查询列表 FROM 表名;
```

实例:

```
-- 查询全部学生
SELECT * FROM student;

-- 查询指定的字段
SELECT `LoginPwd`,`StudentName` FROM student;

-- 别名 AS(可以给字段起别名,也可以给表起别名)
SELECT `StudentNo` AS 学号,`StudentName` AS 学生姓名 FROM student AS 学生表;

-- 函数 CONCAT(str1,str2,...)
SELECT CONCAT('姓名',`StudentName`) AS 新名字 FROM student;

-- 查询系统版本(函数)
SELECT VERSION();

-- 用来计算(计算表达式)
SELECT 100*53-90 AS 计算结果;

-- 查询自增步长(变量)
SELECT @@auto_increment_increment;

-- 查询有哪写同学参加了考试,重复数据要去重
SELECT DISTINCT `StudentNo` FROM result;

```

条件查询

基础语法:

```
select 查询列表 from 表名 where 筛选条件;

```

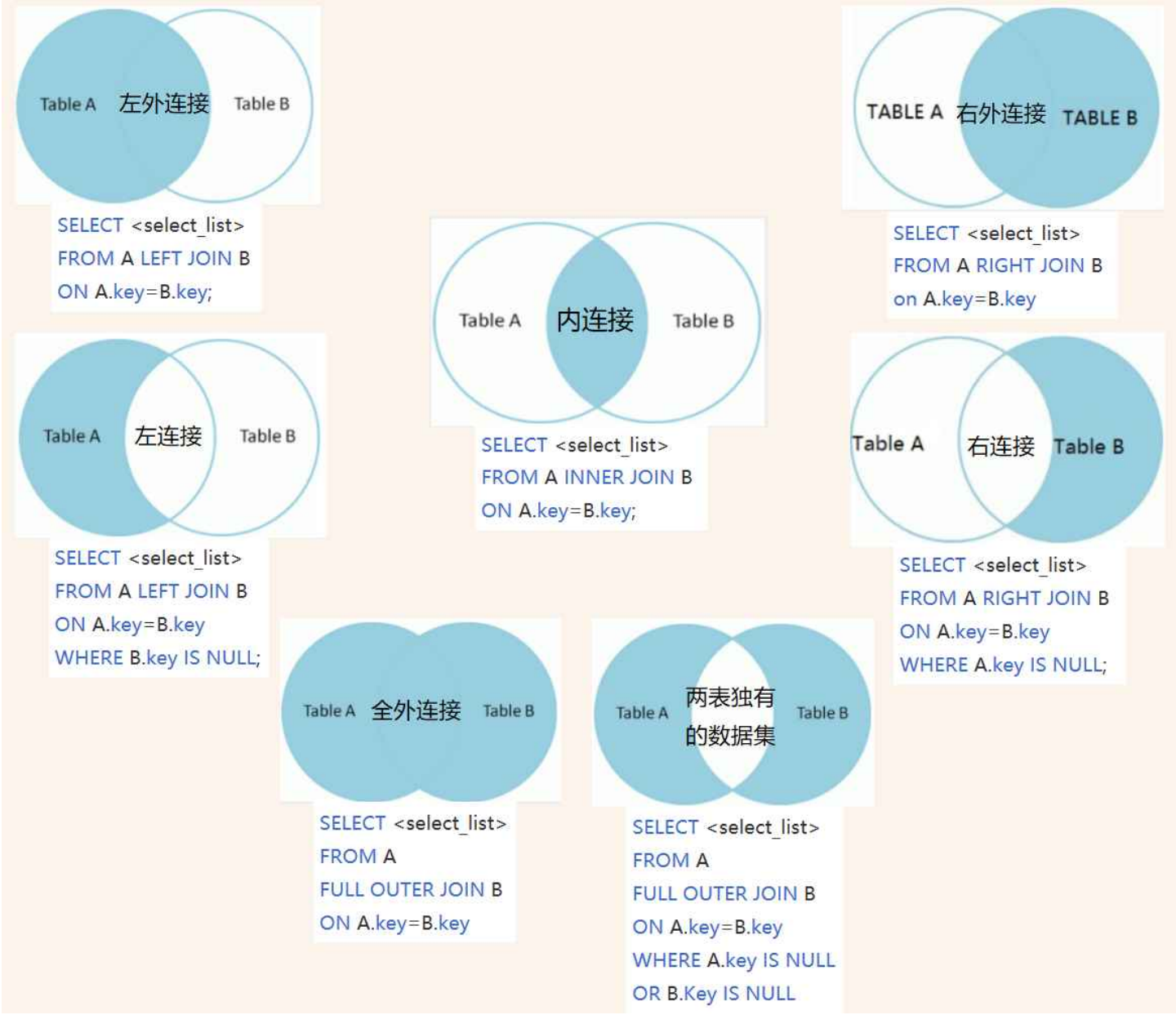
基础实例:

```sql
-- 查询考试成绩在95~100之间的
SELECT `StudentNo`,`StudentResult` FROM result
WHERE `StudentResult`>=95 AND `StudentResult`<=100;
-- &&
SELECT `StudentNo`,`StudentResult` FROM result
WHERE `StudentResult`>=95 && `StudentResult`<=100;
-- BETWEEN AND
SELECT `StudentNo`,`StudentResult` FROM result
WHERE `StudentResult`BETWEEN 95 AND 100;

-- 查询除了1000号以外的学生
SELECT `StudentNo`,`StudentResult` FROM result
WHERE `StudentNo`!=1000;
-- NOT
SELECT `StudentNo`,`StudentResult` FROM result
WHERE NOT `StudentNo`=1000;

-- 查询名字含d的同学
SELECT `StudentNo`,`StudentName` FROM student
WHERE `StudentName` LIKE '%d%';

-- 查询名字倒数第二个为d的同学
SELECT `StudentNo`,`StudentName` FROM student
WHERE `StudentName` LIKE '%d_';

-- 查询1000,1001学员
SELECT `StudentNo`,`StudentName` FROM student
WHERE `StudentNo` IN (1000,1001);

```

分组查询

```sql
select 分组函数，分组后的字段
from 表
【where 筛选条件】
group by 分组的字段
【having 分组后的筛选】
【order by 排序列表】
```

多表查询(连接查询)



多表关系

一对多

在多的一方建立外键,只想一的一方主键

多对多

创立中间表,至少关联两个外键,分别关联两方主键

一对一

笛卡尔积

在多表查询中需要消除不必要的笛卡尔积

链接查询

内连接

查询ab交集部分数据

隐式内连接

```
select 字段 from 表1 表2 where 条件
```

显示内连接

```
select 字段 from 表1 inner join 表2 on 条件
```

外连接

左外连接

查询左表所有数据,以及交集部分

右外连接

查询右表所有数据,以及交集部分

自链接

当前表与自身的连接查询,子链接必须使用表别名

排序ORDER BY

语法:

```
select 查询列表
from 表
where 筛选条件
order by 排序列表 asc/desc

```

Orderby 放置于limit之前

例如:

```
SELECT `StudentNo`,`StudentName`,`GradeName`
FROM student s
```

```
INNER JOIN grade g
ON s.GradeID=g.GradeID
ORDER BY `StudentNo` DESC;

```

分页limit

语法:

```
select 查询列表
from 表
limit offset,pagesize;

SELECT 列表 FROM 表 LIMIT OFFSET(起始索引) PAGESIZE(查询页数)

offset:(n-1)*pagesize
```

offset:(n-1)*pagesize

子查询

where中嵌套查询句子

```
-- 查询‘课程设计’的所有考试结果（学号，科目编号，成绩）降序排列

-- 方式一:使用连接查询
SELECT `StudentNo`,r.`SubjectNo`,`StudentResult`
FROM result r
INNER JOIN `subject` s
on r.StudentNo=s.SubjectNo
WHERE SubjectName='课程设计'
ORDER BY StudentResult DESC;
-- 方式二:使用子查询（由里到外）
SELECT StudentNo,SubjectNo,StudentResult
from result
WHERE SubjectNo=(
        SELECT SubjectNo FROM `subject`
        WHERE SubjectName='课程设计'
)

```

SQL函数

常用函数大全

```sql
-- 数学运算
SELECT ABS(-8); -- 绝对值
SELECT CEIL(5.1); -- 向上取整
SELECT CEILING(5.1); -- 向上取整
SELECT RAND(); -- 返回0~1之间的一个随机数
SELECT SIGN(-10); -- 返回一个数的符号;0返回0;正数返回1;负数返回-1

-- 字符串函数
SELECT CHAR_LENGTH('我喜欢你'); -- 字符串长度
SELECT CONCAT('我','喜欢','你'); -- 拼接字符串
SELECT INSERT('我喜欢',1,1,'超级') -- INSERT(str,pos,len,newstr) 从str的pos位置开始替换为长度为len的newstr
SELECT UPPER('zsr'); -- 转大写
SELECT LOWER('ZSR'); -- 转小写
SELECT INSTR('zsrs','s'); -- 返回第一次出现字串索引的位置
SELECT REPLACE('加油就能胜利','加油','坚持'); -- 替换出现的指定字符串
SELECT SUBSTR('坚持就是胜利',3,6); -- 返回指定的字符串(源字符串,截取位置,截取长度)
SELECT REVERSE('rsz'); -- 反转字符串

-- 时间日期函数
SELECT CURRENT_DATE(); -- 获取当前日期
SELECT CURDATE(); -- 获取当前日期
SELECT now(); -- 获取当前时间
SELECT LOCALTIME(); -- 本地时间
SELECT SYSDATE(); -- 系统时间

SELECT YEAR(NOW());
SELECT MONTH(NOW());
SELECT DAY(NOW());
SELECT HOUR(NOW());
SELECT MINUTE(NOW());
SELECT SECOND(NOW());

-- 系统信息
SELECT SYSTEM_USER();
SELECT USER();
SELECT VERSION();
```

```
37
```