MySql

DataBase

表的本质仍是文件，表的一行称为一条记录，在Java中常常对应一个对象

SQL语言分类

- DDL：数据定义语句，create
- DML：数据管理语句，insert、update、delete
- DQL：数据查询语句，select
- DCL：数据控制语句，grant、revoke

sql语言运行的本质是客户端进程（Java程序、DataGrip、Navicat）将一条语句（`select * from table`）通过网络传送到服务器进程（MySql端口号为3306），使用进程的DBMS（数据库管理系统）进行分析，将结果返回给客户端进程

备份数据库

使用语句备份数据库（仅限于DOS界面，本质是执行MySql组件中的`mysqldump.exe`，可以同时备份多个数据库）：`mysqldump -u userName -p -B DataBaseName > （目录）文件名.sql`

备份文件就是对应的sql语句，导入时执行语句创建一个备份时的数据库

还原数据库只能在MySql命令行执行：`source （目录）文件名.sql`

备份可以对表单独进行（一个数据库的多个表）：`mysqldump -u userName -p密码 DataBaseName tableName1··· > （目录）文件名.sql`

Table

create table

SQL

```sql
create table 表名
(
    字段名0    数据类型  (UNSINGED),
    字段名1    数据类型,
    字段名2    数据类型
)<character set 字符集(utf8)
  collate 校验规则(utf8_bin)>

# 创建示例
create table sjx_user
(
    id       int auto_increment primary key ,
    name     varchar(255) not null ,
    password varchar(32) default '123456',
    birthday date check (brithday < current_date)
) character set utf8
  collate utf8_bin
  -- CHECK的限定功能在MySql中只支持校验但不生效（一般使用程序控制或触发器实现）
  -- auto_increment可以使用`alter`进行修改默认开始值
  alter table tavleName auto_increment = 100;

# 可以使用`LIKE`关键字进行结构的复制
create table tableName1 like tableName2
```

基本数据类型

分类	数据类型	含义
位类型	BIT	位类型，根据指定的位数进行存储
字符串、文本类型	CHAR(n)	长度为n的字符串，最大255个字符
	VARCHAR(n)	最大长度为65535个字节的变长字符串（预留1~3个字节存放长度）
	TEXT	[0 ~ $2^{16}-1$] 多用来存放文本，不能有默认值
	LONGTEXT	[0 ~ $2^{32}-1$]
二进制型	BLOB	[0 ~ $2^{16}-1$]
	LONGBLOB	[0 ~ $2^{32}-1$]

数值型	SMALLINT	短整数（2 字节）
	INT	整数（4字节）
	BIGINT	大整数（8 字节）
	FLOAT	单精度（4字节）
	DOUBLE	双精度（8字节）
	DECIMAL(p, d)	总共p位数字，其中小数的位数是d位，默认为 $10:0$ ，最大为 $65:30$
时间日期	DATE	日期，YYYY-MM-DD
	TIME	时间，HH:mm:SS
	DATATIME	YYYY-MM-DD HH:mm:SS
	TIMESTAMP	时间戳

update table

SQL

```sql
# 追加字段
alter table tableName
add
(
columnName dataType
......
);

# 修改表
alter table tableName
modify
(
columnName dataType
......
);

# 删除列
alter table tableName
drop
(
columnName
);

# 查看表结构
desc tableName;

# 修改表名
rename oldName to newName

# 修改列名
alter table tableNamame change oldName newName <属性>
```

CRUD

insert

1. 插入数据时可以使用逗号分隔多条元组 `insert tableName (列名) values (),(),()...`
2. 日期和字符串使用单引号包围必须符合对应格式，可用 `curdate()` 自动插入当前时间
3. 如果字段允许为空可以插入空值

4. 默认值：非空情况下不添加某一字段且没有默认值会报错，有则自动添加默认值

SQL

```sql
insert into sjx_user(name, birthday)
VALUES ('Alice', curdate());
-- 没有指定id和password的值，id会自增，password使用默认值

insert into sjx_user()
VALUES (001, 'Aidan', '123456', curdate()),
       (002, 'Edward', '666666', '2000-03-04');
-- 以上为两种日期插入格式
insert into sjx_user(name, password, birthday)
VALUES ('Aidan', '123456', current_timestamp)
-- 插入时间戳时，会对相应日期类型进行更改适配
```

复制：可以将其他表或自身的数据进行插入

SQL

```sql
# 自我复制
insert into tableName1
select *
from tableName1

# 其他表复制时两表中的域类型与数据必须对应
insert into tableName1 (field1, ...)
select field1, ...
from tableName2
```

update

在更改时不对条件进行`where`限制，将会对所有的数据进行更改

SQL

```sql
update sjx_user
set password = 666666666;
-- 将所有用户的密码改写，一般软件会进行提示

update sjx_user
set password = 123456
where name = 'Alice';
-- 根据姓名将特定用户的密码改写

update sjx_user
set password = password+123
where name = 'Alice';
-- set语句可以对数值型进行运算，select也可以
```

delete

删除时不对条件进行`where`限制，将会对所有的数据进行删除

SQL

```sql
delete
from sjx_user
where password = '666666666'

delete
from sjx_user
-- 删除所有记录，与drop table的区别就是留一个空表
```

select

SQL

```sql
select distinct name, birthday
from sjx_user
-- 去重查询姓名与生日（姓名与生日都重复时），不写distant默认为all

select password - 123 as Fakecode
from sjx_user
-- 用Fakecode作为列名显示运算后的密码，不写别名默认将字段名用'+'相连

select *
from sjx_user
order by (password + savings) desc
-- 使用算术组合列的方式降序查询所有信息，desc为降序，默认为升序

select *
from sjx_user
where score > 60
  and id > 1;
-- 查询成绩大于60且id大于1的用户信息，or表示或

select *
from sjx_user
where name not like 'A%';
-- 查询姓名首字母不为'A'的所有信息，'%'通配多个字符，'_'通配单个字符

select *
from sjx_user
where score between 60 and 80;
-- 按成绩范围查询，都是开区间（包含60和80）

select *
from sjx_user
where score in (60, 80)
-- 查询符合条件在集合中的信息，本质上是`score=60 or score=80`

select avg(score), max(score), city
from sjx_user
group by city, school
having avg(score) > 60;
-- 查询各个城市的各个学校的平均成绩大于'60'的平均分数和最高分数
-- group by 不能使用'where'来进行条件的限制，可以使用'having'
```

函数

统计函数

SQL

```sql
select COUNT(*) as total
from sjx_user
-- 查询总共有多少条数据，包括空数据
select count(birthday)
from sjx_user
-- 根据列名统计数据，不包括'null'

select sum(distinct savings) as total_savings
from sjx_user
-- 统计列数据总和（仅限数值型），可同时操纵多列

select avg(score)
from sjx_user
-- 计算列数据平均值（仅限数值型），可同时操纵多列，结果为浮点型

select max(savings), min(score)
from sjx_user
-- 求列最大最小值
select max(savings + score)
from sjx_user
-- 可对多列算术进行操作
```

字符串函数

SQL

```sql
select charset(name)
from sjx_user
-- 查询表的字符编码类型

select concat('name is:', name) as name
from sjx_user
-- 使用字符串连接字段数据

select instr(name, 'A')
from sjx_user;
-- 查找子串在字符串的位置，从1开始，0为没有
```

```sql
select instr('name', 'A')
from dual;
-- dual称为亚元表，可以作为默认测试表使用

select ucase(name), lcase(name)
from sjx_user;
-- ucase将数据全部转换为大写，lcase为小写

select left(name, 2), right(name, 2)
from sjx_user
-- 从左（右）截取列数据的两个字符

select length('ccc')
from dual;
select length(name)
from sjx_user;
-- 获取字符串长度

select name as oldName, replace(name, 'Alice', 'Amy') as newName
from sjx_user;
-- 替换字符串

select strcmp('sjx', 'szdan')
from dual
-- 根据字符序比较两字符串大小，<为-1，=为0，>为1

select substr(name, 1, 2)
from sjx_user;
-- 从'name'字段的第1个位置开始截取2个字符，不指定长度默认全截取

select ltrim(name), rtrim(name)
from sjx_user;
-- 分别去掉左右两侧的空格，'trim'表示左右两边都去掉
```

SQL

```sql
# 将字段的首字母小写输出，其他不变
# 先截取第一个字符改为小写，后边的字符也截取，原样截取后拼接两个字符串
select concat(lcase(substr(name, 1, 1)), substr(name, 2)) as newName
from sjx_user;
```

数学函数

SQL

```sql
select abs(-10) as '绝对值'
from dual;
-- 转换为绝对值

select bin(10) as '二进制'
from dual;
-- 转换为二进制
select hex(100)
from dual;
-- 转换为十六进制
select conv(100, 10, 2)
from dual;
-- 将数值‘100’从‘10’进制转换为‘2’进制

select ceiling(3.4) as '上取整', floor(3.4) as '下取整'
from dual;
-- 求一个小数的上下取整

select format(3.14159, 3)
from dual;
-- 将‘3.14159’保留‘3’位小数（四舍五入）

select least(10, -10, 20, -20)
from dual;
-- 求多个数值中的最小值

select mod(10, 3)
from dual;
-- ‘10’对‘3’取余求得的数

select rand(), rand(3), rand(rand())
from dual;
-- 返回0<x<1.0的随机数，指定[seed]之后的随机数始终一致
```

时间日期

SQL

```sql
select current_date, current_time, current_timestamp
from dual;
-- 返回当前的日期、时间、时间戳

select now()
from dual;
-- 返回当前的时间日期（datetime）

select name, DATE(birthday)
from sjx_user;
-- 查询‘datetime'类型时只显示'date'
select year(birthday)
from sjx_user
-- 只显示'year'

select datediff('2011-11-11', '1990-1-1')
from dual;
-- 两日期的差，结果以天为单位，TIMEDIFF结果为时分秒

select *
from sjx_user
where date_add(birthday, interval 10 hour) >= now();
-- 十个小时前过生日的人
-- 日期相加（减）：DATE_ADD[SUB](dateValue,INTERVAL d_value D_TYPE)
select datediff(date_add('2000-03-04', interval 80 year), now())
from dual;
-- 活到80岁还要多少天

select from_unixtime(unix_timestamp())
from dual;
-- 'unix_timestamp()'获取时间戳，from_unixtime()将时间戳改为时间，可以定义格式
```

加密函数

SQL

```sql
select user()
from sjx_user;
-- 查询当前哪些用户登录

select database()
from sjx_user;
-- 查询当前使用的数据库

select md5('123456')
from dual;
-- 将字符串进行一个md5的32位加密字符串
select password('123456')
from dual;
-- password方式加密，也是MySql默认登录用户名的加密方式
```

流程控制

SQL

```sql
# IF(expr1,expr2,expr3)如果expr1为'true'，返回expr2，反之返回expr3
select IF(name like 'Aidan', '艾丹', name)
from sjx_user;
-- 将'name'为'Aidan'的返回'艾丹'其他正常返回

# IFNULL(expr1,expr2)如果expr1不为‘NULL'，返回expr1，反之返回expr2
select IFNULL(birthday, '未知')
from sjx_user;
-- 名字为'NULL'的返回未知

# CASE when expr1 then expr2 [when expr3 then expr4...] else expr5 END
select CASE
           when name like 'Aidan' then '班长'
           when name like 'Alice' then '团支书'
           else '学生' end
from sjx_user;
-- 'name'为'Aidan'返回班长，为'Ailce'返回团支书，其他返回学生
```

加强

分页查询

SQL

```sql
select ... limit start,rows;
-- 正常查询，对结果进行行数的限制，'start'从0开始，查询'rows'行（不包括0）

select *
from student
limit 0,2;
-- 查询前两行数据
```

运行顺序： `group by` > `having` > `order by` > `limit`

嵌套查询

SQL

```sql
# 子查询的结果也可以通过起别名的方式当作临时表的方式进行使用
select name1, field1Max
    from (select name1, max('field1') as field1Max
                from tableName1) as aliasName, tableName2
    where aliasName.name1 = tableName2.name1

# 同时子查询的查询返回也可以使用all或any进行一个限制
select *
    from tableName1
    where field1 > all (select field2
                            from tableName2
                            where ...
                        )

# 子查询可以使用多列匹配的方式（如：每科成绩都相同）对结果进行筛选具体语法格式
select *
    from Grade1
    where (English, Math) = (select English , Math
                                    from Grade2
                                    where name='Aidan'
                                ) and name != 'Aida
```

合并查询

SQL

```sql
# 可以使用`union`关键字对两个查询结果进行合并
select field1, field2
    from tableName1
    union [all]
    select field1, field2
        from tableName2
```

多表查询

多表查询的条件必须不能少于表的数量-1，否则会出现笛卡儿积集（如：三个表需要两条主键链接）

自连接

可以使用别名的方式对同一张表起两个别名进行连接，将相同的字段（如：先选课、上级编号）进行链接以此查询需要的结果字段（如：先选课课程名，上级姓名）

SQL

```sql
# 通过上级编号和员工编号的链接，查询每名员工的上级
select worker.name as '员工名', boss.name as '上级名'
from employ worker join employ boss
               on worker.mangeNo = boss.no;
```

外连接

SQL

```sql
select *
    from tableName1 [left|right] join tableName2
    on tableName1.field = tableName2.field
-- left或right会保存某一侧的悬浮数据
```

index

在数据结构（表变为树）上进行优化提高数据库的性能，相当于添加了一个目录，面对海量数据时可以快速查询，索引本身也会占用空间

SQL

```sql
# 对某个表的某一列创建索引，查询时的约束条件只对创建索引的这一列有效
create [unique] index indexName on tableName (columnName)

# 更改表的方式添加索引
alter table tableName add index indexName (columnName)

# 查询表的索引
show index from tableName
show indexes from tableName
show keys from tableName
desc tableName

# 删除索引
drop index indexName on tableName

# 更改表的方式删除主键索引
alter alter table tableName drop primary key
```

- 主键索引：逐渐自动为主索引
- 唯一索引：某个域的值不会重复，优先使用唯一索引
- 普通索引
- 全文索引：MySql字节的全文索引一本使用Solr或ES框架

transaction

将一组SQL语句（针对增删改）当作整体进行执行，必须全部成功或失败

SQL

```sql
start transaction     -- 开始一个事务
savepoint pointName     -- 设置保存点
rollback to pointName     -- 回滚事务
rollback     -- 回滚全部事务
commit     -- 提交事务（生效后不能回退）
```

隔离级别

多个连接开始事务操作数据时需要考虑隔离性，不考虑会导致脏读、不可重复读、幻读

- 脏读：一个事务还未提交的修改被另一个事务读取到

- 不可重复读：当前事务查询的结果集被另一事务的更改所影响
- 幻读：当前事务查询的结果集被另一事务的插入所影响

隔离级别	脏读	不可重复读	幻读	加锁?
Read uncommited	√	√	√	×
Read commited	×	√	√	×
Repeatable read	×	×	×	×
Serializable（可串行化）	×	×	×	√

SQL

```sql
# 设置隔离级别
set session transaction isolation level levelName

# 查询隔离级别（V5.7之后，旧版使用'tx_isolation'）
show variables like 'transaction_isolation';
select @@transaction_isolation;
-- 可以使用'global'或'session'查询全局或会话的隔离级别
```

存储引擎

SQL

```sql
# 查询所有存储引擎
show engines
-- 不同的存储引擎有着不同的特点和使用场景

# 指定引擎
create table tableName(
    field1 datatype,
    ......
)engine engineName;

# 修改表引擎
alter table tableName engine = engineName;
```

视图

视图中的数据来自对基表的查询，通过给用户的不同视图查询权限来对数据隐私和安全性进行控制，可以看作一个虚拟表的实际存在，可以进行查询和数据更改等操作，视图和基表的数据更改是相互影响的，视图可以继续创建视图（数据还是来自基表）

SQL

```sql
# 基本操作
create view viewName as select...
alter view viewName as select...
show create view viewName
-- 查看创建视图的指令
drop view viewName1,viewName2
```

用户

用户信息存储在'Mysql'数据库下的'user'表中

SQL

```sql
# 创建用户并指定密码
create user userName @'允许登录位置' identified by '密码'

# 删除用户
drop user userName @'允许登录位置'

# 修改自己的密码
set password = password('123456')

# 修改其他人密码（有权限的情况下）
set password for userName @'允许登录位置' = password('123456')
```

权限

通过root用户对其他用户进行权限的赋予和收回

SQL

```sql
# 权限授予语法
grant 权限列表 on 库.对象名 to userName@'登录位置' [identified by '密码']
-- 多个权限用逗号隔开
-- *.*表示所有数据库的所有对象
-- 库.*表示某个库的所有对象
-- 'identified by'如果用户存在修改密码，不存在创建用户

# 回收权限
revoke 权限列表 on 库.对象名 to userName@'登录位置'

# 没有即时生效的权限可以刷新
flush privileges;
```

- 如何不指定`host`，则为`%`，表示所有IP都有权限连接
- 可以使用`XXX@192.168.1.%`表示`192.168.1.*`的ip可以登录
- 删除用户时的位置不是`%`，必须指定明确的用户位置