

## DBMS

### DBMS (DBMS)

#### DBMS

- DBMS의 역할
- DBMS의 구성 요소
- DBMS의 데이터 모델

#### DBMS

DBMS의 역할은 데이터를 저장, 관리, 검색하는 것입니다. DBMS는 데이터를 저장하고, 데이터를 검색하고, 데이터를 관리하는 역할을 합니다.

#### DBMS

- DBMS의 데이터 모델
- DBMS의 데이터 구조
- DBMS의 데이터 관리
- DBMS의 데이터 검색
- DBMS의 데이터 관리 (primary key)

#### DBMS

- DBMS

DBMS	DBMS (bytes)	DBMS	DBMS
TINYINT	1	-128	-127
SMALLINT	2	-32768	32767
MEDIUMINT	3	-8388608	8388607
INT	4	-2147483648	2147483647
BIGINT	8	DBMS	DBMS

- DBMS
  - decimal: decimal(5,2)은 5자리 정수, 2자리 소수
- DBMS
  - char: 고정 길이 문자열
  - varchar: 가변 길이 문자열 + 1 (문자열 '\0')
  - text: 긴 문자열
- DBMS (enum)
  - DBMS gender enum('M', 'F', 'O')
- DBMS
  - data: 날짜
  - datetime: 날짜와 시간
  - timestamp: 날짜와 시간

SQL

数据类型

数据类型	说明
NOT NULL	非空
PRIMARY KEY	主键
UNIQUE KEY	唯一键
DEFAULT	默认值
FOREIGN KEY	外键(关联表)
AUTO_INCREMENT	自增

```
create table student(  
  id int unsigned PRIMARY KEY AUTO_INCREMENT COMMENT 'id',  
  name varchar(10) NOT NULL COMMENT '姓名',  
  age int unsigned COMMENT '年龄',  
  class int unsigned COMMENT '班级',  
  gender enum("男","女") COMMENT '性别'  
  status char(1) DEFAULT '1' COMMENT '状态'  
);COMMENT '学生表'  
  
-- 插入数据  
insert into student (name,age,class,gender) values('张三',18,3,"男") # status=1  
insert into student (name,age,class,gender) values('李四',17,2,"女") # id=4
```

命令

==>mysql->mysql== 数据库

命令	说明
mysql -uroot -p	登录mysql
exit/quit/ctrl+d	退出mysql
select version();	查看mysql版本
select now();	查看当前时间

数据库操作

命令	说明
show databases;	显示数据库
select database();	显示当前数据库
create database 数据库名 charset = utf8;	创建数据库

use 数据库;	数据库
drop database 数据库;	数据库

```
-- 数据库
show databases;
-- 数据库
select database();
-- 数据库
create database itcast charset=utf8;
-- 数据库
use itcast;
-- 数据库
drop database itcast
```

数据库

命令	说明
show tables;	显示数据库中的表
desc 表;	显示表结构
show create table 表;	显示表的创建语句
create table 表;	创建表
comment '数据库'	数据库注释

```
-- 数据库
create table xxx(
    id int unsigned primary key auto_increment not null,
    name varchar(20),
    age int unsigned default 0,
    high decimal(5,2),
    gender enum("男","女"),
    cls_id int unsigned
);
```

数据库

命令	说明
alter table 表 add 列 列;	添加列
alter table 表 change 列 列 数据类型;	修改列名和数据类型
alter table 表 modify 列 数据类型;	修改列数据类型
alter table 表 drop 列;	删除列
drop table 表;	删除表

```
-- 修改
alter table student modify name varchar(20);
-- 添加
alter table student add gender enum("男","女");
-- 删除
alter table student drop gender
-- 添加注释
alter table student modify name COMMENT '姓名'
```

插入语句

SQL语句	说明
insert into 表 values(...);	插入一行数据
insert into 表(列1,...)values();	插入一行数据，指定列名
insert into 表 values(...),(...);	插入多行数据
insert into 表(列1,...) values(列1...)(列1...);	插入多行数据，指定列名

```
-- 插入
insert into student values(0,"张三",18,166.66,"男",2);
# 批量插入
insert into student (name,age,height,gender,class) values("张三",18,166.66,"男",2); # 插入一行
insert into student values(0,"张三",18,166.66,"男",2),(0,"李四",17,154.43,"男",1); # 插入两行
```

更新语句

SQL语句	说明
select *from 表;	查询所有数据
select 列1,列2... from 表;	查询指定列
update 表 set 列1 = 值1,列2 = 值2 ...where 条件;	更新数据
select distinct from 表;	查询不重复数据

```
-- 更新
update students set gender = '男'; # 更新所有数据
update students set gender = '男' where id = 2; # 更新id=2的数据

-- 查询
select * from students; # 查询所有数据
select * from students where id = 2; # 查询id=2的数据
select name,age from students; # 查询姓名和年龄
select name as '姓名',age as '年龄' from students; # 查询姓名和年龄并起别名
```

```
select age as '나이', name as '이름', from students; # 나이와 이름
select distinct gender from students; # 성별
```

테이블

구문	설명
delete from 테이블 where 조건;	테이블

```
-- 테이블 삭제
delete from students where id = 4;

-- 테이블 수정
alter table students add is_delete bit default 0; # is_delete 컬럼 추가
update students set is_delete = 1 where id = 4; # is_delete = 1로 설정
```

## where

- 비교 연산자
  - = (같다)
  - > (보다 크다)
  - < (보다 작다)
  - != <> (다름)
- 논리 연산자
  - and (둘 다 참일 때)
  - or (둘 중 하나 참일 때)
  - not (반대)

테이블

테이블 like

- % (모든 문자)
- \_ (한 글자)

```
-- 이름이 '이'로 시작하는 학생 찾기
select * from students where name like '이%';

-- 이름이 '이'로 끝나는 학생 찾기
select * from students where name like '%이';

-- 이름이 '이'인 학생 찾기
select * from students where name like '이';

-- 이름이 '이'로 시작하고 '이'로 끝나는 학생 찾기
select * from students where name like '이%이';
```

테이블

- between and (A와 B 사이) between A and B (A와 B 포함)
- in (A, B, C 중 하나)

```
-- 18세 이상 30세 이하의 학생 찾기
select name from students where age in (18, 30);
```

```
-- 18과 30 사이의 학생
select * from students where age not in (18,30);
-- 18과 30 사이의 학생
select * from students where age between 18 and 30;
-- 18과 30 사이의 학생
select * from students where age not between 18 and 30;
```

NULL

- **NULL is null**
- **NULL is not null**

```
-- NULL 값 찾기
select * from students where height is null;
```

**order**

```
select * from 테이블 order by 1 asc|desc[, 2asc|desc,...]
```

- 1번째 컬럼을 기준으로 1번째 컬럼을 2번째 컬럼
- asc: 오름차순
- desc: 내림차순
- NULL 처리

```
-- 18과 34 사이의 학생을 키순으로 정렬
select * from students where age between 18 and 34 and gender = 'M' order by height asc;
-- 18과 34 사이의 학생을 키순으로 정렬, 나이순으로 정렬
select * from students where age between 18 and 34 and gender = 'M' order by height desc, age desc;
```

집계함수

함수	설명
count(컬럼)	행의 수
max(컬럼)	최대값
min(컬럼)	최소값
sum(컬럼)	합계
avg(컬럼)	평균

```
-- 학생 수
select count(*) from students where gender = 'M';
-- 최대 나이
select max(age) from students;
-- 최소 키
select min(height) from students;
```

```
-- 合計
select sum(age) from students;

-- 平均
select round(avg(age),2) from students;
```

**group**□□□□

- **group by**: 1
- 

関数	説明
group concat(関数)	グループごとに指定した関数を適用し、結果をコンカテナートした文字列で返す。
having 条件	グループごとに条件を満たすかどうかを判定し、条件を満たすグループのみを返す。
with roll up	指定した列をロールアップして、結果を階層的に表示する。

```
-- 性别分组
select gender from students group by gender

-- 性别和人数
select gender, count(*) from students group by gender;

-- 姓名和性别
select group concat(name), gender group by gender;

-- 30岁以上
select group concat(name), gender group by gender having avg(age) > 30;

-- 性别和人数(带rollup)
select gender, count(*) from students group by gender with rollup;
```

**limit**□□□□

- `limit(limit, offset, sql)`
- `limit` : 쿼리 결과의 개수
- `offset`
  - 쿼리 결과의 시작 위치 (0부터 시작)
  - 쿼리 결과의 끝 위치 (0부터 시작)

```
-- 5명만
select * from students limit 5;

-- 0명부터 2명까지
select * from students limit 0,2;

-- 2명부터 2명까지
select * from students limit 2,2;

-- 2명부터 4명까지
select * from students order by age asc limit 6,2;
```

□□□□

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □

111

INNER JOIN 内连接

SQL: select \* from t1 inner join t2 on t1.col1 = t2.col2

- on 连接

```
-- 内连接
select * from students inner join classes;
-- 内连接
select * from students inner join classes on students.cls_id = classes.id;
-- 内连接, 别名
select s.name, c.name from students as s inner join classes as c on s.cls_id = c.id;
```

SQL

SQL: select \* from t1 left join t2 on t1.col1 = t2.col2

- 左连接 left join
- 右连接 right join

```
-- 左连接
select * from students left join classes on students.cls_id = classes.id;
-- 右连接
select * from students right join classes on students.cls_id = classes.id;
```

SQL

SQL: select \* from t1 right join t2 on t1.col1 = t2.col2

SQL

image-20241007190605047

SQL

1. 查询aid
2. 查询pid和title

==SQL: select \* from t1 where t1.col1 = t2.col2==

```
-- SQL
select * from areas as city inner join areas as province on city.pid = province.aid;
```

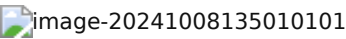
SQL

SQL: select \* from t1 where t1.col1 > (select avg(col1) from t1)

```
-- SQL
select avg(height) from students;
-- SQL
select * from students where height > (select avg(height) from students);
```



00



```
-- cate_name '000' name price
select name,price from goods where cate_name = '000';

-- 
## 
select brand_name from goods group by brand_name;
## 
select distinct brand_name from goods;

-- avg
select round(avg(price),2) from goods;

-- cate_name
select avg(price),cate_name from goods group by cate_name;

-- 
select cate_name,max(price),min(price),avg(price),count(*) from goods group by
cate_name;

-- 
## 
select avg(price) from goods;
## 
select * from goods where price > (select avg(price) from goods) order by price
desc;

-- 
## 
select cate_name,max(price) as max from goods group by cate_name;
## 
select * from goods inner join (select cate_name,max(price) as max from goods group
by cate_name ) as max_price on goods.cate_name = max_price.cate_name and goods.price
= max_price.max;
```

00

- A B A B
- 

atler table  add foreign key( ) references ( );	
atler table  drop foreign key ;	
show create table goods;	

```
-- 删除
alter table goods add foreign key(cate_id) references goods_cate(id);
-- 删除
## 删除
show create table goods;
## 删除
alter table 删除 drop foreign key goods_ibfk_1;

-- 创建表
alter table goods{
    id int primary key auto_increment not null,
    name varchar(20),
    price decimal(5,2),
    cate_id int unsigned,
    brand_id int unsigned

    foreign key (cate_id) references goods_cate(id),
    foreign key (brand_id) references goods_brand(id)
};
```

SQL

- 数据库系统概论SQL
- 数据库SQL
- 数据库系统概论SQL

SQL	SQL
create view 视图 as select 语句;	视图
show tables;	视图
select * from v_goods_info;	视图
drop view 视图;	视图

```
-- 删除id
select s.id,s.name,s.age,s.gender,c.name as cls_name from students as s inner join
classes as c on s.id = c.id;

-- 删除
create view v_students as select s.id,s.name,s.age,s.gender,c.name as cls_name from
students as s inner join classes as c on s.id = c.id;

-- 删除
select * from v_students;

-- 删除
drop view v_students;
```

SQL

- 数据库事务的英文是**sql**数据库事务的英文是**sql**
- 数据库**ACID**
  - 原子性
  - 一致性
  - 隔离性
  - 持久性

命令	说明
begin;或start transaction;	开始事务
commit;	提交事务
rollback;	回滚事务

```
-- 开始事务
begin;
update students set age = 100 where id = 1;
-- 提交事务
commit;
```

索引

==数据库索引的英文是**index**==

命令	说明
show index from 表;	显示表的索引
alter table 表 add index 索引名 (列名);	创建索引
drop index 索引名 on 表;	删除索引

```
-- 创建索引
set profiling = 1 ;
## 查看索引创建时间
select * from test_index where title = 'ha-99999';
show profiles;
## 删除索引
alter table test_index add index(title);
## 查看索引创建时间
select * from test_index where title = 'ha-99999';
show profiles;
```

索引

- 数据库索引的英文是**index**
  - 数据库索引的英文是**index**
1. 索引
    - 数据库索引的英文是**index**
  2. 索引**(索引)**

- 数据库引擎
- 数据库引擎的驱动程序
- 数据库

### 3. 连接(数据库)

- 数据库引擎的驱动程序
- 数据库引擎的驱动程序A和数据库引擎B的驱动程序B

## SQL

数据库引擎的驱动程序SQL和数据库引擎的驱动程序SQL

```
from pymysql import Connection
## 连接数据库
conn = Connection(host='localhost',port=3306,user='root',password='zxb050818')
print(conn.get_server_info())
## 创建游标
cursor = conn.cursor()
## 选择数据库
conn.select_db('world')
## 执行SQL语句
find_name = input() # or 1 or
sql = "select * from students where name = '%s'" % find_name
cursor.execute(sql)
## 获取结果
content = cursor.fetchall() # 返回结果
```

==数据库==

- 数据库
- 数据库引擎

```
params = [find_name]
sql = 'select * from students where name = %s'
cursor.execute(sql,params)
```

## SQL

数据库引擎的驱动程序

### SQL引擎

- 数据库DDL
  - 数据库引擎的驱动程序
- 数据库DML
  - 数据库引擎的驱动程序
- 数据库DCL
  - 数据库引擎的驱动程序
- 数据库DQL
  - 数据库引擎的驱动程序

数据库

- □□□□□
- □□□□□□□□□□
- □□□□□
  - □□□□□--□□□□(□□□□□□□□□□)
  - □□□□□#□□□□
  - □□□□□/\* □□□□ \*/

## DDL□□□□□□

- `mysql`
  - `mysql> show databases ;`
  - `mysql> use mysql ;`
  - `mysql> create database testdb [CHARSET UTF8] ;`
  - `mysql> drop database testdb ;`
  - `mysql> select database() ;`
- `mysql`
  - `mysql> show tables ;`
  - `mysql> drop table test ;` `mysql> drop table if exists test ;`
  - `mysql> create table test( id int, name varchar(255), age int, sex char(1), birthday timestamp(6), ... )`
    - `int` `float` `data(type)` `timestamp(type)` `varchar(length):utf8`

**DML**□□□□□□

- `insert into [(10020...N)] values(10020.....N),(10020.....N),(10020.....N),...(10020.....N)`
- `delete from 表 [where 条件] (py)where`
- `update 表 set 列 = 值 [where 条件] where`

## DQL□□□□□□

- `SELECT * FROM table WHERE condition`
- `SELECT column1/column2(column3) FROM table WHERE condition GROUP BY column1`
  - `GROUP BY column1 SELECT column1`
  - `AGGREGATE FUNCTION`
    - `SUM`
    - `AVG`
    - `MIN`
    - `MAX`
    - `COUNT`
- `ORDER BY column1 SELECT column1/column2(column3) FROM table WHERE condition GROUP BY column1 ORDER BY column1 [ASC|DESC]`
- `LIMIT n, m` `LIMIT n, m` `LIMIT n + 1`

# pythonSQL

- `commit()` 提交数据到数据库
- `autocommit` 是否自动提交

1111

1. pymysql
2. pymysql
3. pymysql
4. pymysql
5. pymysql

```

from pymysql import Connection
## 连接数据库
conn = Connection(host='localhost',port=3306,user='root',password='zxb050818')
print(conn.get_server_info())
## 创建游标
cursor = conn.cursor()
## 选择数据库
conn.select_db('world')
## 编写SQL语句
sql = 'select * from students'
cursor.execute(sql)
## 获取数据
content = cursor.fetchone()
## 获取所有数据
content = cursor.fetchall()
print(content)
## 关闭游标
cursor.close()
conn.close()

```

数据库

==数据库事务==

```

from pymysql import Connection
## 连接数据库
conn = Connection(host='localhost',port=3306,user='root',password='zxb050818')
## 创建游标
cursor = conn.cursor()
## 选择数据库
conn.select_db('world')
## 编写SQL语句
sql = "insert into students(name) values('')"
cursor.execute(sql)
## 提交事务
sql = 'select * from students;'
cursor.execute(sql)
content = cursor.fetchall()
for i in content:
    print(i)
## 提交事务
conn.commit()
## 关闭游标
cursor.close()
conn.close()

```