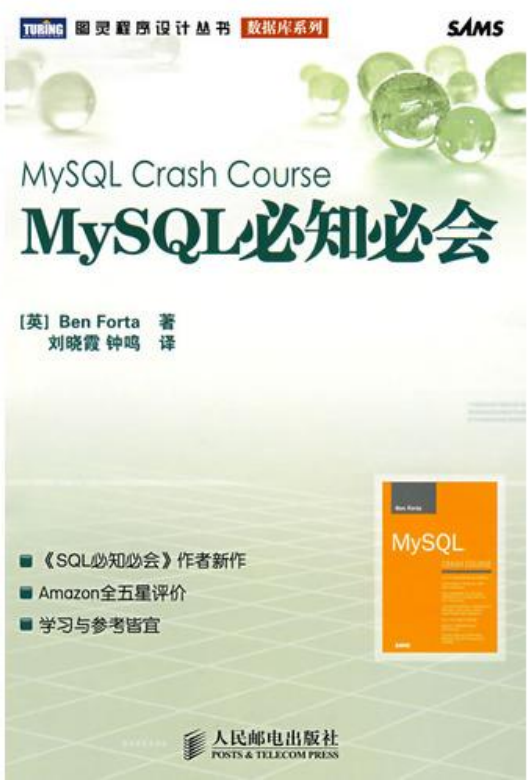
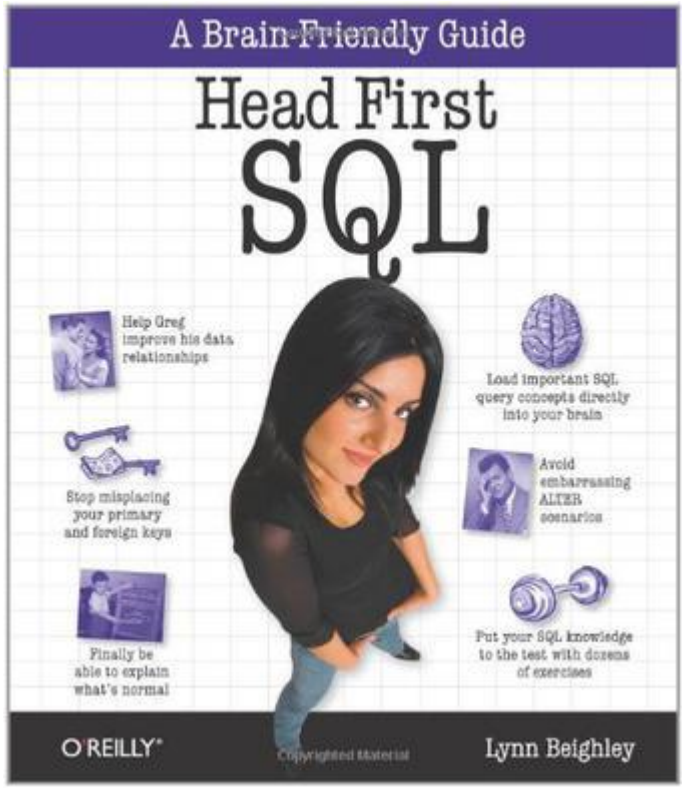
MySQL basic

凯盛软件

推荐书籍

凯盛软件


TURING 图灵程序设计丛书 数据库系列
SAMS
MySQL Crash Course
MySQL必知必会
[英] Ben Forta 著
刘晓霞 钟鸣 译
《SQL必知必会》作者新作
Amazon全五星评价
学习与参考皆宜
MySQL
人民邮电出版社
POSTS & TELECOM PRESS


A Brain-Friendly Guide
Head First
SQL
Help Greg improve his data relationships
Load important SQL query concepts directly into your brain
Stop misplacing your primary and foreign keys
Avoid embarrassing ALTER scenarios
Finally be able to explain what's normal
Put your SQL knowledge to the test with dozens of exercises
O'REILLY
Lynn Beighley

数据库

数据库是以一定组织方式储存在一起的，能为多个用户共享的，具有尽可能小的冗余度的、与应用彼此独立的相互关联的数据集合。

数据库管理系统 DBMS

凯盛软件

关系型数据库

NoSQL数据库

MySQL

MySQL AB → SUN → Oracle

开源关系型数据库

LAMP组合成员

http://www.mysql.com

http://zh.wikipedia.org/wiki/MySQL

MySQL下载与安装

凯盛软件

下载地址 http://www.mysql.com/downloads/mysql/

端口号：3306

用户名：root

密码：自定义

连接到MySQL服务器


```
>mysql -uroot -proot [-h127.0.0.1]
```

断开连接

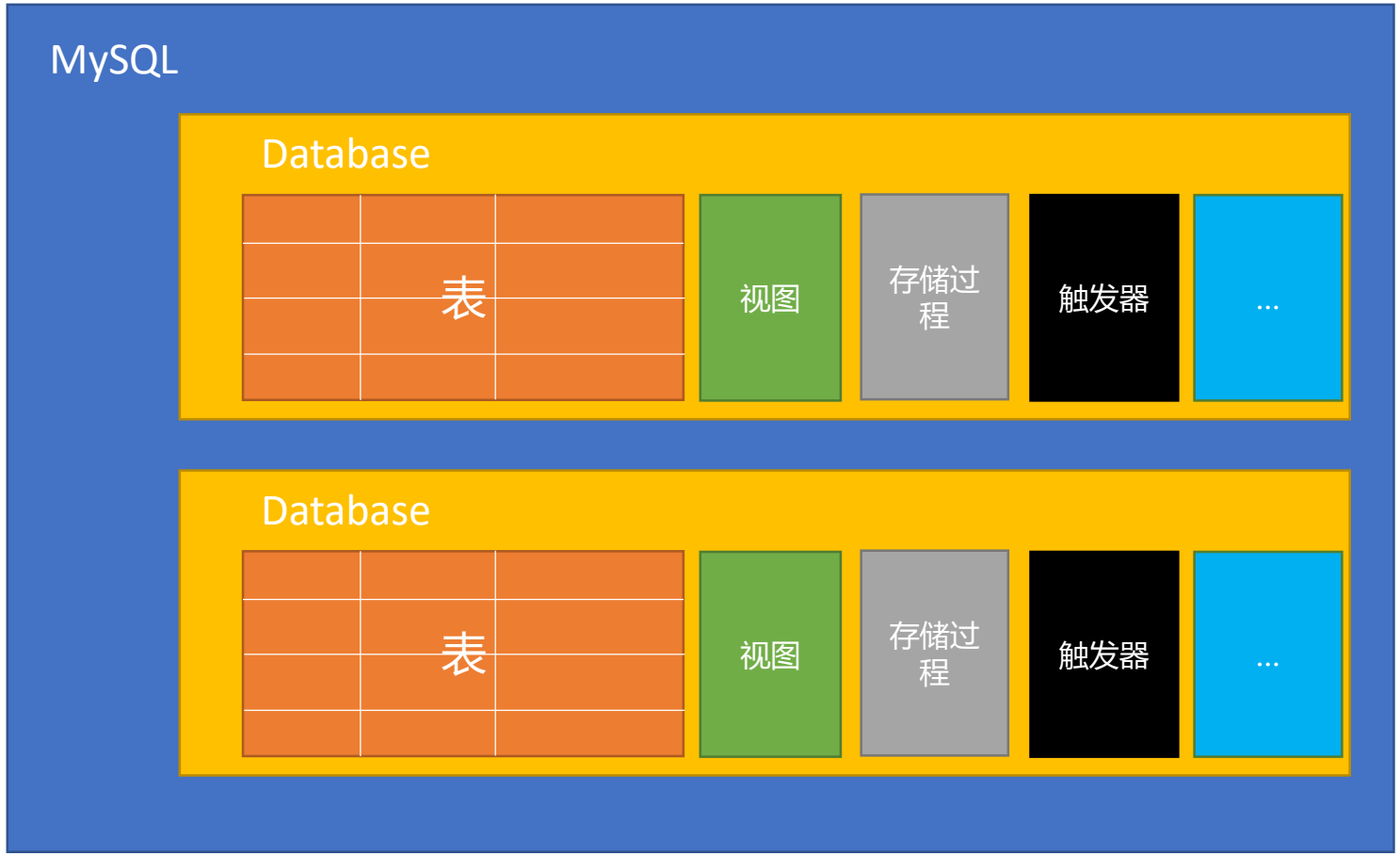
>exit

or

>quit

MySQL

凯盛软件



命令

显示MySQL中所有的数据库

```
> show databases;
```

切换到mydb数据库

```
> use mydb;
```

查看数据库中所有的表

```
> show tables;
```

查看表结构

```
> desc t_user;
```

查看数据库版本和时间

```
> select version(),now();
```

创建数据库


```
>create database mydb;
```

创建表

姓名	年龄	家庭住址
Tom	23	郑州
Jerry	24	洛阳
Rose	23	焦作
Alex	22	开封
Tom	22	郑州
Kate	22	郑州
Tom	22	郑州

数据类型（整型）

凯盛软件

数据类型	无符号范围	有符号范围
TINYINT	0~255	-128~127
SMALLINT	0~65535	-32768~32767
MEDIUMINT	0~16777215	-8388608~8388607
INT(Integer)	0~4294967295	-2147483648~2147483647
BIGINT	0~18446744073709551615	-9223372036854775808~9223372036854775807

数据类型（浮点型）

凯盛软件

数据类型	无符号范围	有符号范围
FLOAT	0，(1.175 494 351 E-38，3.402 823 466 E+38)	-3.402 823 466 E+38，1.175 494 351 E-38)，0，(1.175 494 351 E-38，3.402 823 466 351 E+38)
DOUBLE	(1.797 693 134 862 315 7 E+308，2.225 073 858 507 201 4 E-308)，0，(2.225 073 858 507 201 4 E-308，1.797 693 134 862 315 7 E+308)	0，(2.225 073 858 507 201 4 E-308，1.797 693 134 862 315 7 E+308)
DECIMAL(M,D)		

数据类型（字符型）

凯盛软件

数据类型	大小	用途
CHAR	0-255字节	定长字符串
VARCHAR	0-255字节	变长字符串
TINYTEXT	0-255字节	短文本字符串
TEXT	0-65 535字节	长文本数据
MEDIUMTEXT	0-16 777 215字节	中等长度文本数据
LONGTEXT	0-4 294 967 295字节	极大文本数据

数据类型（日期时间型）

凯盛软件

类型	范围	格式
DATE	1000-01-01~9999-12-31	YYYY-MM-DD
TIME	-838:59:59~838:59:59	HH:MM:SS
DATETIME	1000-01-01 00:00:00~9999-12-31 23:59:59	YYYY-MM-DD HH:MM:SS
TIMESTAMP	1970-01-01 00:00:00~2037年	YYYY-MM-DD HH:MM:SS

创建表


```
>create table t_student (
    stuname  varchar(20),
    stuage  int,
    stuaddress varchar(100)
);
```

插入数据


```
> insert into t_student
(stuname,stuage,stuaddress)
values
('tom',23,'郑州');
```

查询表中数据


```
>select * from t_student;
```

主键

姓名	年龄	家庭住址
Tom	23	郑州
Jerry	24	洛阳
Rose	23	焦作
Alex	22	开封
Tom	22	郑州
Kate	22	郑州
Tom	22	郑州

主键

ID	姓名	年龄	家庭住址
1	Tom	23	郑州
2	Jerry	24	洛阳
3	Rose	23	焦作
4	Alex	22	开封
5	Tom	22	郑州
6	Kate	22	郑州
7	Tom	22	郑州

主键

- 在设计表时总是要定义表的主键
- 表的主键设计策略
    - 任意两行都不具备相同的主键值
    - 每行都必须具有一个主键值（主键不允许Null列）
    - 主键和业务无关，不更改，不重用
- 主键可以是一个列或者是多个列的组合
- 使用**PRIMARY KEY（XXX）**来声明一个主键列
- 如果使用多个列作为主键则需要如下声明：**PRIMARY KEY(XXX,XXX)**

创建表时添加主键


```
>create table t_student (
    id  int,
    stuname  varchar(20),
    stuage  int,
    stuaddress varchar(100),
    primary key(id)
);
```

主键自动增长

凯盛软件

AUTO_INCREMENT

- 用来标示一个自动增长列
- 一个表中只允许有一个自动增长列

```
>create table t_student (
    id  int auto_increment,
    ...
);
```

删除表


```
> drop table t_student;
```

非null约束

stuname varchar(20) not null

默认约束


```
stuaddress varchar(100) default '郑州'
```

唯一约束

stuname varchar(20) not null unique

更改表

添加一列
```
>alter table t_student add tel char(20);
```

删除一列
```
>alter table t_student drop column tel;
```

添加唯一约束
```
>alter table t_student add constraint uk_username unique(usercode);
```

添加主键约束
```
>alter table t_user add constraint pk_t_user_id primary key t_user(id);
```

添加默认约束
```
>alter table t_user alter password set default '123456';
```

添加非null约束
```
>alter table t_teacher modify column uname varchar(20) not null;
```

重命名表


```
>rename table t_student to t_stu
```

导出数据库


```
>mysqldump -hlocalhost -uroot -proot mydb>C:/a.sql
```

批量导入SQL脚本


```
>source C:/a.sql
```

一次性插入多条语句

```
>insert into t_student(stuname,stuage,stuaddress)
  values
  ('tom','23','焦作'),
  ('jerry','25','南阳'),
  ('hanks','21','韩国')
```

insert

凯盛软件

- insert语句中列的数量和值的数量必须相同
- 每个值的数据类型、精度和小数位数必须和列的要求相匹配
- 列的值要符合列的约束
- 如果列有默认值,可以使用关键字default来插入默认值

运算符

逻辑运算符

- = 等于
- <>,!= 不等于
- < 小于
- > 大于
- <= 小于等于
- >= 大于等于
- between 在指定的两个值之间

关系运算符

- and
- or
- not

update


```
>update t_student
  set
  stuname = 'Alex',age = '26'
  [where id = 1];
```

where

凯盛软件

- where stuname = 'tom'
- where stuname = 'tom' or stuname = 'alex'
- where id > 1 and id < 3
- where id != 23
- where id = 12 or id = 34
- where id in (12,34)
- where id between 12 and 34
- where password is null
- where password is not null

delete


```
>delete from t_student [where id = 1];
```

truncate

凯盛软件

>truncate table t_student;

TRUNCATE TABLE用于删除表中的所有记录，但该语句不能包含WHERE语句，该操作运行速度比DELETE语句快

表设计

凯盛软件

ID	姓名	年龄	学院	学院电话
1	Tom	23	计算机学院	3546677
2	Rose	22	计算机学院	3546677
3	Alex	23	软件学院	8788990
4	Jerry	21	计算机学院	3546677
5	Jack	24	软件学院	8788990

三大范式

1. 确保每列的原子性
2. 在第一范式的基础上，确保每列都和主键相关
3. 在第二范式的基础上，确保每列都和主键直接相关，而不是间接相关

拆表

ID	姓名	年龄	学院ID
1	Tom	23	x1
2	Rose	22	x1
3	Alex	23	x2
4	Jerry	21	x1
5	Jack	24	x2

外键(foreign key)列

子表

优势：降低数据冗余程度

劣势：添加查询难度

主表

ID	学院	学院电话
x1	计算机学院	3546677
x2	软件学院	8788990

创建外键


```
>alter table t_user add schoolid int;

>alter table t_user
  add constraint fk_student_cus
  foreign key(schoolid)
  references
  t_school(id);
```

删除外键


```
> alter table t_user
  drop foreign key fk_student_cus
```

外键

- 子表中外键列中添加的数据必须在主表的主键中存在
- 外键列的数据类型及长度必须和主表的主键的数据类型及长度相同
- 删除主表数据时，如果有子表引用，则删除失败

查询

查询会产生一个虚拟的表，看到的是以表的形式显示的结果，但结果并不真正的存储，每次执行查询只是从现有表中提取数据，并按照表的形式显示出来。

基本查询

查询所有的列

>SELECT * FROM vendors;

查询指定的列

>SELECT vend_id,vend_name,vend_address,vend_city FROM vendors;

如果查询时需要显示表中的所有列，尽量避免使用通配符(*)，而要采用写出所有列名的方式进行查询，因为采用通配符查询会降低程序的查询性能。

基本查询

去除重复记录

```
>SELECT DISTINCT vend_id FROM products;
```

分页

```
>SELECT * FROM products LIMIT 5;
```

```
>SELECT * FROM products LIMIT 0,5;
```

```
>SELECT * FROM products LIMIT 5,5;
```

基本查询

排序(降序)

```
>SELECT * FROM products ORDER BY prod_price DESC;
```

排序(升序)

```
>SELECT * FROM products ORDER BY prod_price [ASC];
```

多列排序

```
>SELECT * FROM products ORDER BY prod_price ASC,prod_name ASC;
```

过滤查询

查询产品价格在2到10之间的产品

```
>SELECT * FROM products WHERE prod_price >= 2 AND prod_price <= 10;
>SELECT * FROM products WHERE prod_price BETWEEN 2 AND 10;
```

查询产品价格不等于2.5的所有产品

```
>SELECT * FROM products WHERE prod_price <> 2.5;
>SELECT * FROM products WHERE prod_price != 2.5;
```

查询没有电子邮件信息的客户

```
>SELECT * FROM customers WHERE cust_email IS NULL;
```

查询有电子邮件信息的客户

```
>SELECT * FROM customers WHERE cust_email IS NOT NULL;
```

过滤查询

查询由供应商1001和1003制造并且价格在10元以上的产品

>SELECT * FROM products WHERE vend_id = '1001' OR vend_id = '1003' AND prod_price > 10;

>SELECT * FROM products WHERE (vend_id = '1001' OR vend_id = '1003') AND prod_price > 10;

>SELECT * FROM products WHERE vend_id IN('1001','1003') AND prod_price > 10;

查询不是由供应商1001和1003制造的产品

> SELECT * FROM products WHERE vend_id NOT IN(‘1001’ , ‘1003’) ;

模糊查询

“_”通配符代表一个字符

“%”通配符代表0个或一个或任意多个字符

查询产品名称中以jet开头的产品

```
> SELECT * FROM products WHERE prod_name LIKE 'jet%';
```

查询_ ton anvil产品

```
> SELECT * FROM products WHERE prod_name LIKE '_ ton anvil'
```

- 不要过度使用LIKE通配符，如果其他操作符可以完成就使用其他操作符
- 通配符搜索使用的时间比其他搜索的时间长
- 如果确实需要使用通配符，除非绝对有必要，否则不要把通配符放到WHERE子句的开始处，把通配符放到搜索模式的开始处，搜索起来是最慢的

更多基本查询

凯盛软件

列的别名

> SELECT vend_id AS '供应商编号' FROM products;

算数运算

>SELECT quantity,item_price,quantity * item_price AS '总价' FROM orderitems;

文本处理函数

left()返回左边指定长度的字符

```
> SELECT prod_name,LEFT(prod_name,2) FROM products;
```

right()返回右边指定长度的字符

```
>SELECT prod_name,RIGHT(prod_name,5) FROM products;
```

length()返回字符串的长度

```
> SELECT prod_name,LENGTH(prod_name) FROM products;
```

lower()将字符串转换为小写

```
> SELECT prod_name,LOWER(prod_name) FROM products;
```

upper()将字符串转换为大写

```
> SELECT prod_name,UPPER(prod_name) FROM products;
```

文本处理函数

ltrim()去掉字符串左边的空格

```
> SELECT prod_name,LTRIM(prod_name) FROM products;
```

rtrim()去掉串右边的空格

```
>SELECT prod_name,RTRIM(prod_name) FROM products;
```

trim()去掉左右两边的空格

```
>SELECT prod_name,TRIM(prod_name) FROM products;
```

字符串连接

```
>SELECT CONCAT('I love ',cust_name) AS 'Message' FROM customers;
```

日期时间函数

凯盛软件

函数	用途	函数	用途
curDate()	返回当前日期	curTime()	返回当前时间
now()	返回当前日期和时间	date()	返回日期时间的的日期部分
time()	返回日期时间的时间部分	day()	返回日期的天数部分
dayofweek()	返回一个日期对应星期数	hour()	返回时间的小时部分
minute()	返回时间的分钟部分	month()	返回日期的月份部分
second()	返回时间的秒部分	year()	返回日期的年份部分
datediff()	计算两个日期之差	addDate()	添加一个日期（天数）

日期和时间函数

获取2005-9-1日的订单

```
>SELECT * FROM orders WHERE order_date = '2005-09-01';
```

```
>SELECT * FROM orders WHERE DATE(order_date) = '2005-09-01';
```

获取2005年9月的订单

```
>SELECT * FROM orders WHERE order_date >= '2005-09-01' AND order_date <= '2005-09-30';
```

```
>SELECT * FROM orders WHERE YEAR(order_date) = '2005' AND MONTH(order_date) = '9';
```

聚合函数

- min()
- max()
- count()
- sum()
- avg()

聚合函数常用于统计数据使用

聚合函数统计时忽略值为NULL的记录

聚合函数

查询商品价格最高的产品

```
> SELECT MAX(prod_price) FROM products;
```

查询商品价格最低的产品

```
> SELECT MIN(prod_price) FROM products;
```

查询商品价格总和

```
> SELECT SUM(prod_price) FROM products;
```

查询商品平均价格

```
> SELECT AVG(prod_price) FROM products;
```

查询客户数量

```
>SELECT COUNT(*) FROM customers;
```

```
>SELECT COUNT(cust_email) FROM customers;
```

分组统计

获取每个供应商提供的产品数量

> SELECT vend_id,COUNT(*) FROM products GROUP BY vend_id;

获取提供产品数量大于2的供应商

> SELECT vend_id,COUNT(*) FROM products GROUP BY vend_id HAVING COUNT(*) > 2;

HAVING语句用于GROUP BY的过滤

WHERE用于分组前过滤

获取产品提供产品数量大于等于2并产品价格大于10的供应商

> SELECT vend_id,COUNT(*) FROM products WHERE prod_price > 10 GROUP BY vend_id HAVING COUNT(*) >= 2;

查询语句顺序

1. SELECT
2. FROM
3. WHERE
4. GROUP BY
5. HAVING
6. ORDER BY
7. LIMIT

子查询

子查询指的是嵌套在查询中的查询

获取订购商品编号为TNT2的客户名

1.从订单详情表中获取订单编号：

```
> SELECT order_num FROM orderitems WHERE prod_id = "TNT2";
```

2.根据订单编号获取下订单的客户ID：

```
> SELECT cust_id FROM orders WHERE order_num IN ('20005','20007');
```

3.根据客户ID获取客户的姓名：

```
> SELECT cust_name FROM customers WHERE cust_id IN ('10001','10004');
```

子查询


```
SELECT cust_name FROM customers WHERE cust_id
IN (SELECT cust_id FROM orders WHERE order_num
IN(SELECT order_num FROM orderitems WHERE prod_id = "TNT2")
);
```

子查询

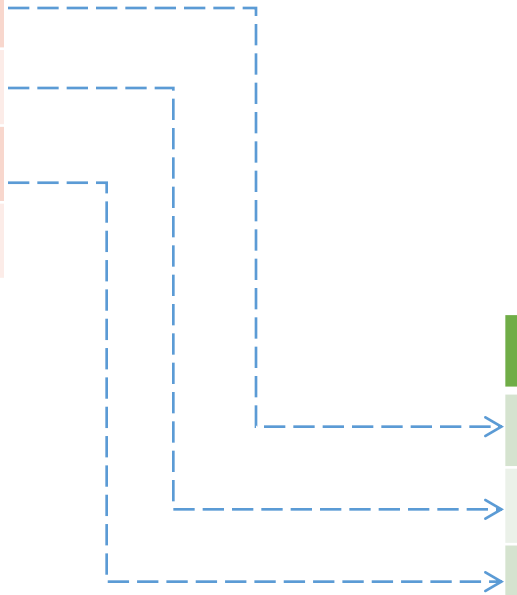
获取每个客户下的订单数量

```
> SELECT
  cust_id,cust_name,
  (SELECT COUNT(*) FROM orders WHERE orders.cust_id = customers.cust_id)
  FROM
  customers ;
```

联接查询

学生表

ID	stu_name	class_id
1	tom	1
2	jerry	2
3	rose	3
4	alex	7



班级表

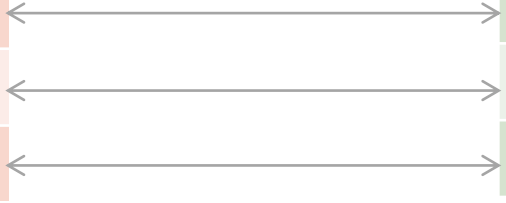
ID	class_name
1	JAVA-01
2	PHP-01
3	DB-01
4	JAVA-02
5	WEB-01

等值查询和内联接查询

凯盛软件

查询学员及对应班级信息，班级没有学员或学员班级信息错误的不显示。

ID	stu_name	class_id
1	tom	1
2	jerry	2
3	rose	3



ID	class_name
1	JAVA-01
2	PHP-01
3	DB-01

等值查询

```
> SELECT ts.id AS 'stuid',stu_name,tc.id AS 'class_id',class_name
  FROM t_student AS ts,t_class AS tc
  WHERE ts.class_id = tc.id
```

内联接查询

```
> SELECT ts.id AS 'stuid',stu_name,tc.id AS 'class_id',class_name
    FROM t_student AS ts
    INNER JOIN t_class AS tc
    ON ts.class_id = tc.id
```

左(外)联接查询

查询所有学员及对应的班级信息

ID	stu_name	class_id
1	tom	1
2	jerry	2
3	rose	3
4	alex	7

左

ID	class_name
1	JAVA-01
2	PHP-01
3	DB-01

右

```
> SELECT ts.id AS 'stuid',stu_name,tc.id AS 'class_id',class_name
  FROM t_student AS ts
  LEFT JOIN t_class AS tc
  ON ts.class_id = tc.id
```

左外联接将会显示左表的所有记录

右(外)联接查询

凯盛软件

查询所有班级及对应学员信息

ID	stu_name	class_id
1	tom	1
2	jerry	2
3	rose	3

ID	class_name
1	JAVA-01
2	PHP-01
3	DB-01
4	JAVA-02
5	WEB-01

```
>SELECT ts.id AS 'stuid',stu_name,tc.id AS 'class_id',class_name
 FROM t_student AS ts
 RIGHT JOIN t_class AS tc
 ON ts.class_id = tc.id
```

右外联接将会显示右表的所有记录

组合查询

t_user表

id	name	createtime
1	tom	2012-10-29 11:42:25
2	jerry	2012-10-29 11:42:28
3	alex	2012-10-29 11:42:38

t_company表

id	name	createtime
1	Google	2012-10-29 11:42:49
2	Apple	2012-10-29 11:42:51
3	Microsoft	2012-10-29 11:42:55
4	FaceBook	2012-10-29 11:43:02
5	Twitter	2012-10-29 11:43:05
6	NetEasy	2012-10-29 11:43:08

组合查询

凯盛软件

查询所有的用户和公司，并在一个结果集中显示

```
> SELECT id,name,createtime FROM t_user
  UNION
  SELECT id,name,createtime FROM t_company;
```

id	name	createtime
1	tom	2012-10-29 11:42:25
2	jerry	2012-10-29 11:42:28
3	alex	2012-10-29 11:42:38
1	Google	2012-10-29 11:42:49
2	Apple	2012-10-29 11:42:51
3	Microsoft	2012-10-29 11:42:55
4	FaceBook	2012-10-29 11:43:02
5	Twitter	2012-10-29 11:43:05
6	NetEasy	2012-10-29 11:43:08

组合查询

凯盛软件

查询所有的用户和公司，并在一个结果集中按照创建时间(createtime)降序显示

```
> select id,name,createtime from t_user
  union
  select id,name,createtime from t_company
  order by createtime desc;
```

id	name	createtime
6	NetEasy	2012-10-29 11:43:08
5	Twitter	2012-10-29 11:43:05
4	FaceBook	2012-10-29 11:43:02
3	Microsoft	2012-10-29 11:42:55
2	Apple	2012-10-29 11:42:51
1	Google	2012-10-29 11:42:49
3	alex	2012-10-29 11:42:38
2	jerry	2012-10-29 11:42:28
1	tom	2012-10-29 11:42:25

组合查询

凯盛软件

- union必须由两条或两条以上的select语句组成，语句之间使用union分割
- union的每个查询必须包含相同的列，表达式或聚合函数
- 列的数据类型必须兼容：类型不必完全相同，但是必须是相互可以转换的
- union查询会自动去除重复的行，如果不需要此特性，可以使用union all
- 对union结果进行排序，order by语句必须在最后一条select语句之后

```
>SELECT vend_id FROM vendors
 UNION ALL
 SELECT vend_id FROM products;
```

MySQL其他知识

- **数据库引擎：**
  - InnoDB：可靠的事务处理引擎，不支持全文搜索
  - MyISAM：是一个性能极高的引擎，支持全文搜索，但不支持事务处理
  - MEMORY：功能等同于MyISAM引擎，但由于数据存储在内存中，所以速度快

```
> create table xxx (
    ...
  )engine=innodb;
```

根据查询记录添加到表：

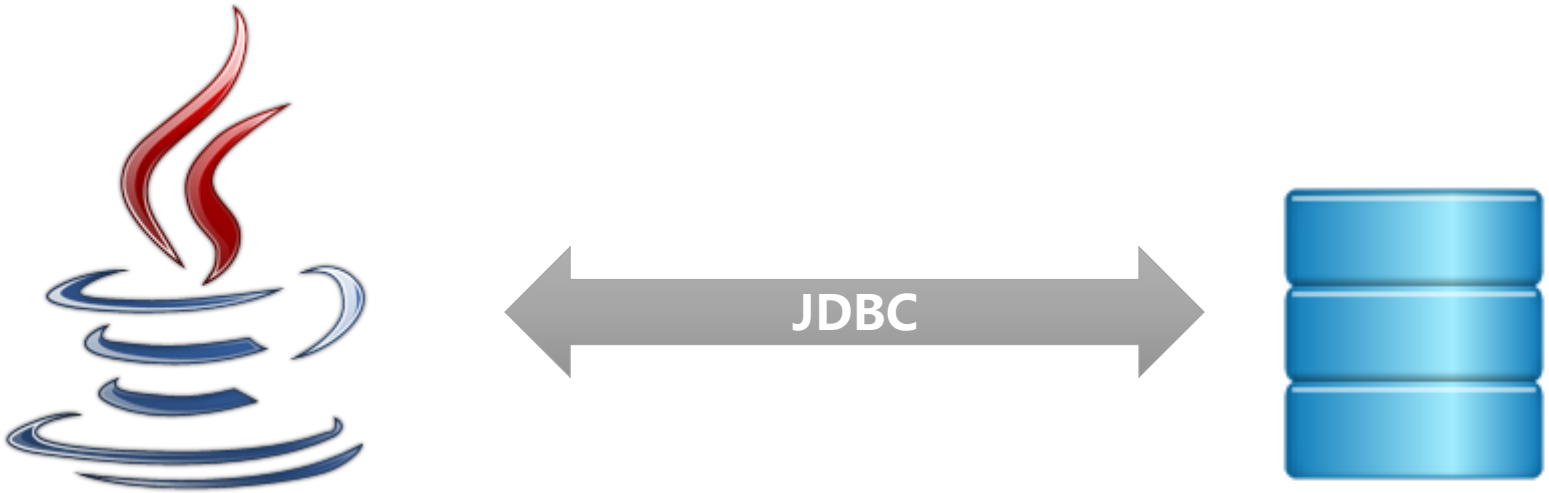
```
>insert into t_tableb(val)
  select val from t_tablea;
```

第三方GUI工具：

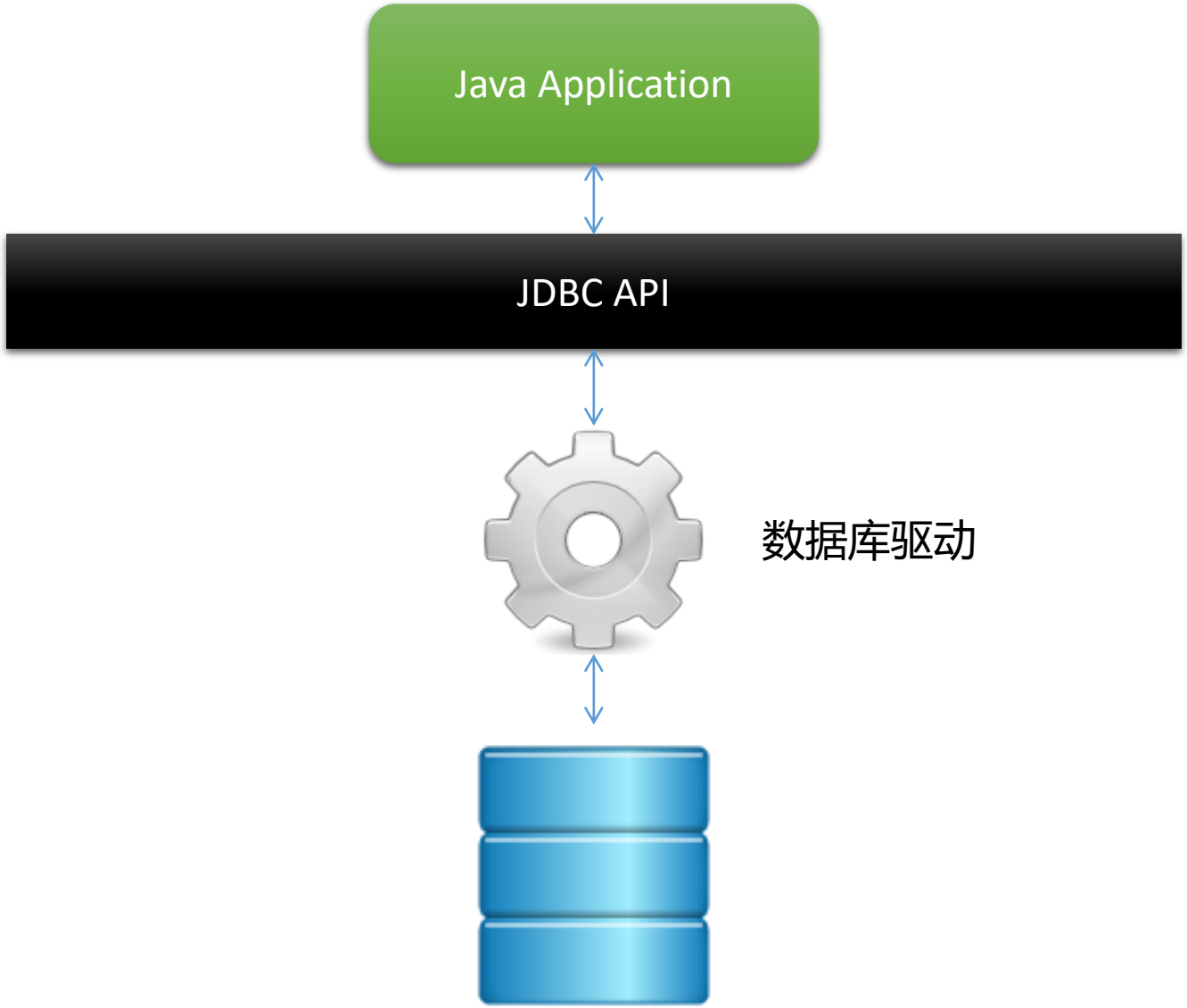
- SQLyog
- MySQLWorkBench

JDBC

JDBC：Java Database Connectivity



JDBC



数据库驱动

由数据库厂商提供，例如Oracle、MySQL、SQLServer。数据库驱动大部分以jar包的形式提供。

获取数据库驱动的jar包后，需要添加入Eclipse的java builder path中。

使用JDBC-加载数据库驱动

凯盛软件

```
try {
        Class.forName("com.mysql.jdbc.Driver");


} catch (Exception e) {
        e.printStackTrace();
}
```

使用JDBC-获取数据库连接

凯盛软件

```
try {
        Class.forName("com.mysql.jdbc.Driver");

        Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb", "root", "root");

} catch (Exception e) {
        e.printStackTrace();
}
```

使用JDBC-执行insert update delete语句

凯盛软件

```
try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb", "root", "root");

        String sql = "delete from t_student where id = 1";
        Statement stat = conn.createStatement();
        stat.executeUpdate(sql);



} catch (Exception e) {
        e.printStackTrace();
}
```

使用JDBC-执行select语句

凯盛软件

```
String sql = "select id,name from t_student";
Statement stat = conn.createStatement();
ResultSet rs = stat.executeQuery(sql);

while(rs.next()) {
        int id = rs.getInt("id");
        String name = rs.getString("name");
        System.out.println(id + " : " + name);
}
```

释放资源

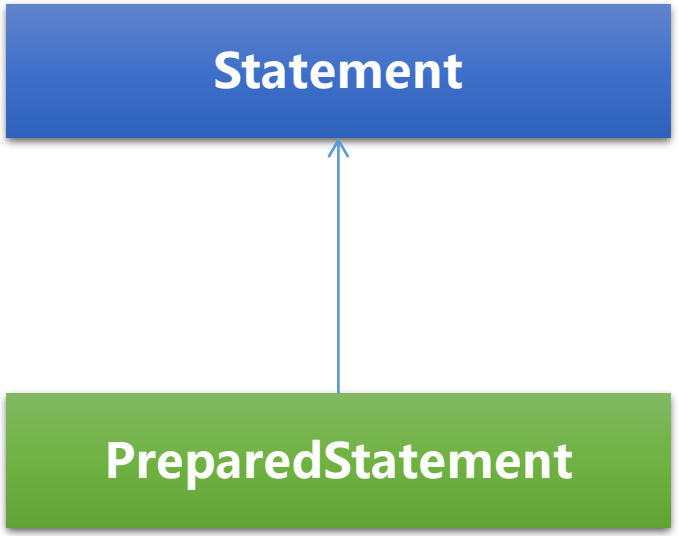

```
finally {
    rs.close();
    stat.close();
    conn.close();
}
```

拼SQL


```
String name = "tom";
String pwd = "123";

String sql = "insert into t_user(name,pwd) values('" + name + "','" + pwd + "')";
```

PreparedStatement

凯盛软件



优点：

- 对SQL语句进行预编译处理，执行速度快
- 防止SQL注入(SQL injection)，安全
- 代码阅读性提高

SQL注入攻击

SQL injection

凯盛软件

```
String sql = "SELECT id,username,PASSWORD,eanble,money FROM t_account WHERE
username = '"+name+"' AND PASSWORD = '"+pwd+"'";

Statement stat = conn.createStatement();

ResultSet rs = stat.executeQuery(sql);
```

```
请输入用户名:
tom
请输入密码:
123
1 : tom
```

```
请输入用户名:
tom'#
请输入密码:
xxxxxxxxxxx
1 : tom
```

PreparedStatement

凯盛软件

```
String sql = "select id,name from t_student";

PreparedStatement stat = conn.prepareStatement(sql);

ResultSet rs = stat.executeQuery();

-------------------------------------------------

String sql = "insert into t_user(username,age) valuse(?,?)";

PreparedStatement stat = conn.prepareStatement(sql);

stat.setString(1, "tom");

stat.setInt(2, 23);

stat.executeUpdate();
```

获取自动增长的主键值

凯盛软件

```
Class.forName("com.mysql.jdbc.Driver");
Connection conn = DriverManager.getConnection("jdbc:mysql:///kaishengit_db", "root", "root");

String sql = "insert into t_test(username,address) values(?,?)";
PreparedStatement stat = conn.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS);

stat.setString(1, "vivi");
stat.setString(2, "Jp");

stat.executeUpdate();

ResultSet rs = stat.getGeneratedKeys();
if(rs.next()) {
        System.out.println(rs.getObject(1));
}

rs.close();
stat.close();
conn.close();
```

批量执行SQL

凯盛软件

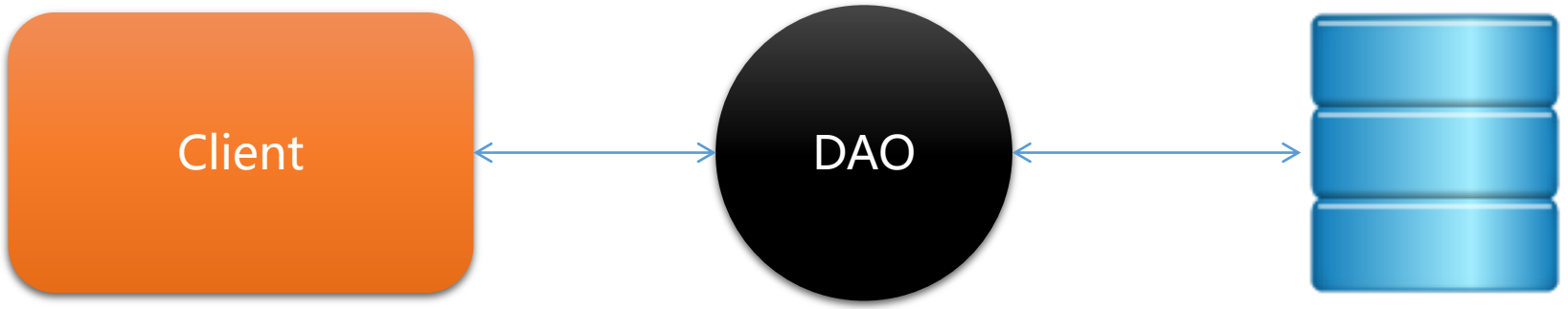
```
Statement stat = conn.createStatement();
long start = System.currentTimeMillis();

for (int i = 0; i < 1000; i++) {
    String sql = "INSERT INTO `user`(`username`,`password`) VALUES ('xx','xx')";

    stat.addBatch(sql);
    if(i%100 == 0) {
        stat.executeBatch();
    }
}
stat.executeBatch();

long end = System.currentTimeMillis();
System.out.println((end-start) + "ms");
```

```
String sql = "INSERT INTO `user`(`username`,`password`) VALUES (?,?)";

PreparedStatement stat = conn.prepareStatement(sql);
long start = System.currentTimeMillis();

for (int i = 0; i < 1000; i++) {
   stat.setString(1, "xxx");
   stat.setString(2, "123123");

   stat.addBatch();
   if(i%100 == 0) {
        stat.executeBatch();
    }
}
stat.executeBatch();

long end = System.currentTimeMillis();
System.out.println((end-start) + "ms");
```

DAO模式

DAO=Data Access Object

每个DAO中封装了对一个实体(Entity)类的CRUD操作



DBHelp类

内部类

在一个类中声明的类称为内部类

```
public class Outer {

    public void hi() {
        System.out.println("Hello,This is Outer Class");
    }

    public class Inner {
         public void sayHello() {
                System.out.println("Hello,This is Inner Class");
         }
    }
}
```

为什么有内部类

凯盛软件

- 内部类可以访问外部类中的私有实例变量
- 当内部类的访问修饰符声明为private时，内部类只能在外部类内部使用，使用外部类时不知道内部类的存在

内部类的实例化

凯盛软件

```
Outer.Inner inner = new Outer().new Inner();
inner.sayHello();
```

```
Outer o = new Outer();
Inner inner = o.new Inner();
inner.sayHello();
```

内部类中使用外部类实例变量

凯盛软件

```
public class Outer {

    private String name = "Tom";

    public void hi() {
        System.out.println("Hello,This is Outer Class");
    }

    public class Inner {
         public void sayHello() {
                System.out.println("Hello," + name);
         }
    }
}
```

```java
public class Outer {

    private String name = "Tom";

    public void hi() {
        System.out.println("Hello,This is Outer Class");
    }

    public class Inner {
         private String name = "Jerry";
         public void sayHello() {
             System.out.println("Hello," + Outer.this.name);
             System.out.println("Hello," + this.name);
         }
    }
}
```

在外部类内部创建内部类对象

凯盛软件

```
public class Outer {

    private String name = "Tom";

    public void hi() {
        Inner inner = new Inner();
        inner.age = 20;
        inner.sayHello();
    }
    private class Inner {
         private int age = 10;
         public void sayHello() {
                System.out.println("Hello," + Outer.this.name + "\t" + age);
         }
    }
}
```

静态内部类

凯盛软件

```
public class Outer {

    public static class Inner {
        public void say() {
                System.out.println("Hello");
        }
    }

}



Outer.Inner inner = new Outer.Inner();
inner.say();
```

```
public class Outer {

    public static String name = "Tom";
    public static void hi() {
        System.out.println("xixi");
    }

    public static class Inner {
        public String name = "Jerry";
        public void say() {
            Outer.hi();
            System.out.println("Hello"+Outer.name);
            System.out.println("Hi!" + name);
        }
    }
}
```

局部内部类

局部内部类创建在一个方法中，作用范围在这个方法中

```
public class Outer {

    public void hi() {

        class Inner {
             public void sayHello() {
                System.out.println("Hello");
             }
        }

        Inner inner = new Inner();
        inner.sayHello();
    }
}
```

为什么使用局部内部类

凯盛软件

- 解决一个复杂的问题，想创建一个类来辅助你的解决方案，但是又不希望这个类是公共可用的
- 在方法内部实现一个接口，并返回对其的引用

局部内部类使用方法中定义的变量，变量必须声明为final

```
public class Outer {

    public void hi() {
        final String name = "tom";
        class Inner {
            public void sayHello() {
                System.out.println("Hello" + name);
            }
        }

        Inner inner = new Inner();
        inner.sayHello();
    }

}
```

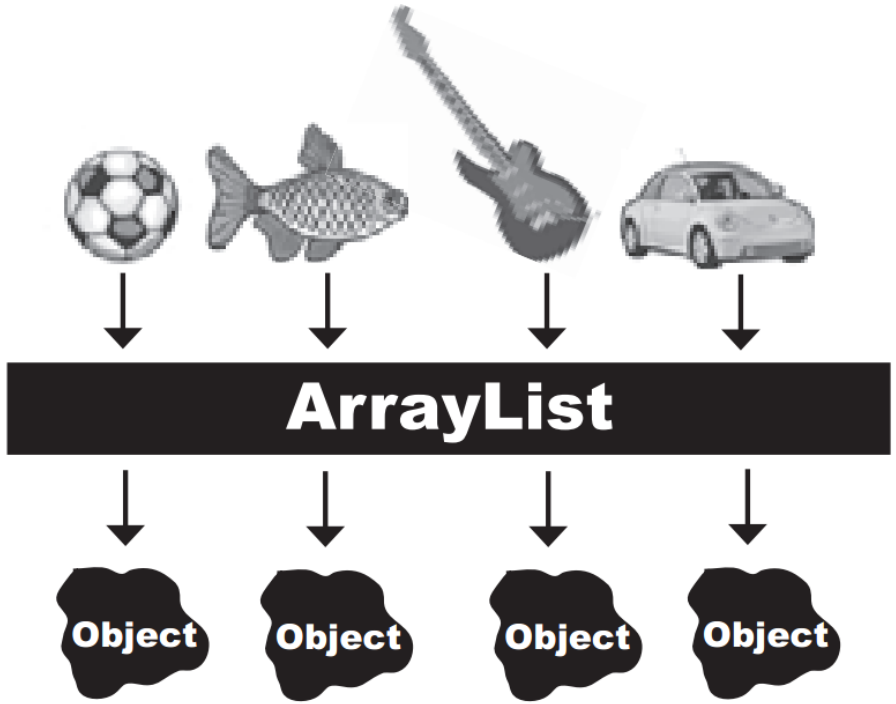
匿名局部内部类

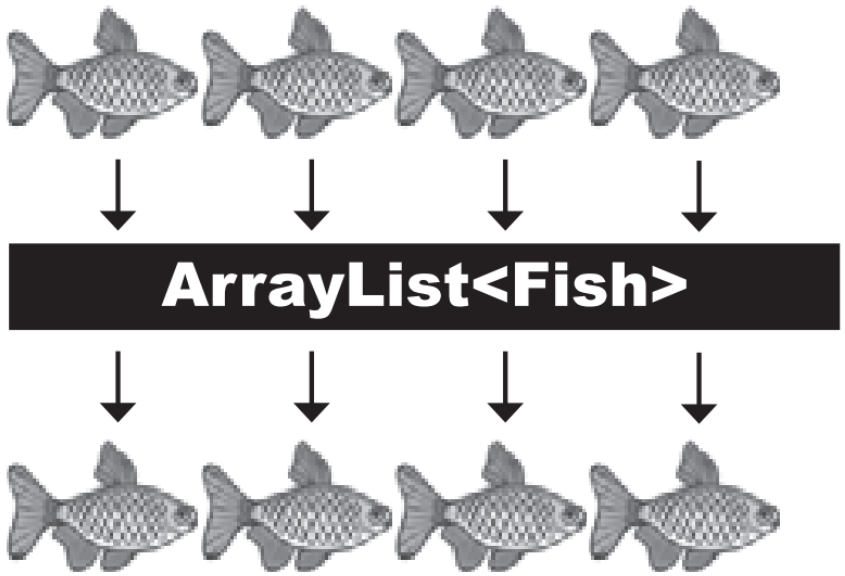
没有名字的局部内部类

```
public interface Usb {
        public void run();
}

public Usb getUsb() {
    return new Usb(){
        @Override
        public void run() {
                System.out.println("runing...");
        }
    };
}
```

```
public void method() {
    Object obj = new Object(){
        @Override
        public String toString() {
                return "hi";
        }
    };

    System.out.println(obj);
}
```

泛型<Generic>





泛型<Generic>

凯盛软件

```
public class DBHelp<T> {



    public T executeQueryForObject() {

    }

    public List<T> executeQueryForList()
    {

    }
}
```

```
DBHelp<User> db = new DBHelp<User>();
```

泛型<Generic>

凯盛软件

```
public class GenericClass<T extends Person> {

}

public class GenericClass<T extends Serializable> {

}
```

在泛型中extends代表继承(extends)和实现(implements)

泛型<Generic>

凯盛软件

```
public class GenericClass<T extends Serializable> {

    public void method(List<T> list) {

    }
}

public class GenericClass {

    public void method(List<? extends Serializable> list) {

    }
}

public class GenericClass {

    public <T extends Serializable> void method(List<T> list) {

    }
}
```

泛型<Generic>


```
public void method(List<? extends Serializable> list1,
                   List<? extends Serializable> list2) {

}


public <T extends Serializable> void method(List<T> list1,List<T> list2) {

}
```