Ora le读书笔记

# 1．SQL部分

## 1.select部分

1. select … from …where …;
2. select选择部分能使用表达式（数值，日期类型）

例：SELECT las t\_name, salary, salary + 300, SALARY \*2 FROM employees;

1. NULL的概念：不可用，未赋值，不知道，不适用；

它不是代表0或者空格

因此我们判空的时候 是IS NULL 而不是 =NULL；

并且，一个数值和NULL进行四则运算的，其结果是NULL；不是错误。

1. 列别名：例：AS name和 AS “Name”区别就是前者列名会大写，后者列名根据引号中大小写区分
2. 字符串的连接 ||，也可以连接常量字符串
3. DISTINCT使用与作用：DISTINCT的作用是去除重复查列

使用：select DISTINCT department\_id…;

## 2.Oracle的配置

1. 安装虚拟机
2. 安装虚拟机管理软件及虚拟机
3. 打开管理软件 点击打开虚拟机 D:\资料\MAS虚拟机\Oracle Linux 6.5\...
4. 如果显示失败，重启电脑，并且设置开启虚拟机技术。(Esc)
5. 重启后，按照第一步启动即可。
6. 软件—安装Oracle 11g数据库（管理员）
7. 将MAS虚拟机中的tnsnames.ora文件拷贝到app/username/produce/11.2.0/client\_1/network/admin/tnsnames.ora;（设置成虚拟机IP地址）
8. 在虚拟机中 ifconfig获取本机ip地址
9. 在虚拟机中，开启数据库 cd .. cd .. cd root pws ls ./startdb.sh(关闭数据库 。/stopdb.sh)
10. 开启plsqldeb.exe
11. 如果没有用的话（打开虚拟机 Computer/FileSystem/etc/hosts修改第二个ip设置为本机IP）

## 3.条件限制和排序

1. between…and… 前小后大，包括前后
2. In(set)集合范围,一个值得时候，可以不加括号。

例：select \* from table where id in(50,60);

1. LIKE 模糊查询 %为通配符 一个或多个字符； \_为一个字符；还有在字符串中的确有%的话，需要使用escape ‘/’设置通配符 /%或/\_。 escape制定转义字符.

代码：select \* from dept\_temp1

wherednamelike'%/%%'escape'/';

(4)NULL 只能 IS NULL 不能=NULL;

（5）ORDER BY 排序 默认升序 ASC ；降序DESC；

## 4.单行函数

（1）包括：字符函数，数值函数，日期函数，转换函数

（2）LOWER() UPPER() INITCAP()(小写，大写，首字母大写)

（3）dual是Oracle中的一个虚拟表，只有一条记录

（4）Oracle数据库中的数据是大小写敏感的（区分大小写），可以利用字符函数来查询。

（5）SQL中字符串使用 单引号。

（6）字符串操作函数：CONCAT(str1，str2)连接,SUBSTR(‘helloworld’,1,5)(hello 1开始 数5个),INSTR(‘world’,’w’)查询第一个w字符，1,LPAD(salary,10,\*)结果：\*\*\*\*\*24000,RPAD()(左右补齐)；

（7）数字操作函数：ROUND(45.9899,2) 45.99（四舍五入）TRUNC(45.98712,2) 45.98

MOD(1600,300) 模，取余 ROUND(45.923,-1)小数点前一位 50;

1. 日期类型：‘YYYY/MM/DD’ ‘YYYY/MM/DD HH24:MI:SS’ ‘YYYY/MM/DD HH:MI:SS’

‘MM-DD-YYYY’ ‘DD-MMM-RR’ ‘DD-MMM-YY’

两个日期相减得：相差天数，小数点部分为相差小时换算成的天数。

1. 日期操作函数：MONTHS\_BETWEEN(’01-SEP-95’,’11-JAN-94’)结果：19.67；相差月份ADD\_MONTHS() to

ALTER SESSION SET NLS\_DATE\_LANGUAGE=’AMERICAN’;

NEXT\_DAY ('01-SEP-95','FRIDAY')下个星期五,

NEXT\_DAY('01-SEP-95',1)周日下一周的第一天

1. to\_date(‘1995-09-01’，’YYYY-MM-DD’);字符串到日期的转化
2. 设置 01-SEP-95 日期类型:tools/Prefences/NLS options中修改（缺陷：YY和年份前两位相同；RR类型：下半叶世纪，取上半叶世纪的年份，其会自动+1，上半叶世纪，取下半叶实际的年份，其会自动-1）
3. to\_char(date,’YYYY-MM-DD’);日期转字符串（转换格式）
4. 数据转换函数：1.隐式 2.显式
5. TRIM(‘H’ from ‘Helloworld’)结果为：elloworld;去除字符串前后的空格，这里是去除前后的H字符。
6. Select \* from test1 where column1=1234；报错，隐式转换出错
7. TO\_NUMBER() TO\_CHAR(),TO\_DATE(字符串转数字，转换字符串，转日期)
8. TO\_CHAR(number,’format\_model’);9表示数字.小数点 ，千分位 L 本地货币符号
9. Alter session set NLS\_CURRENCY=’$’;//修改语言环境
10. To\_number()中掩码一定要等于或者大于转化数字要长。
11. NVL（exp1,exp2）若 1为空，返回2
12. NVL2(1，2，3) 若1为空返回3，否则返回2
13. NULLIF(1,2)相等 为空，不相等返回第一个值 coalesce(null.null,1,null,2) 1第一个不为空的参数
14. 条件表达式：1.casejob\_id when ‘it’ then 1.1\*sal

When ‘st’ then 1.2\*sal

Else sal

END ‘end\_sal’(别名)

(24)DECODE(job\_id,’it’,1.1\*sal,

’st’,1.2\*sal

,sal);(只能判断离散的值，连续的无法进行)

(25)在别名中有单引号的时候，整体必须使用双引号。

(26) Oracle 数据类型 隐式转换

赋值：varchar可以和date,number互相转换

表达式比较 var char转 date,number，不可逆

(27)查找年内第几天，TO\_CHAR(hire\_date, 'DDD')

## 5.多表关联查询

（1）在工程中，表大部分符合第三范式，没有传递依赖，无部分依赖（2NF）。

（2）等值连接中where后有等值条件，笛卡儿积是没有条件的，笛卡儿积在实际的项目中是没有意义的。Cross join(笛卡儿积连接)

（3）不等连接：><<>!= between…and…

（4）外连接：左外连接，右外连接（Oracle）

左外连接：selecte.last\_name

,e.department\_id

,d.department\_name

From employees e,department d

Where e.department\_id=d.department\_id(+)

下列一样的，不同语法的外连接：

select \* fromemp,deptwhereemp.deptno(+)=dept.deptno;

select \* fromemprightouterjoindeptonemp.deptno=dept.deptno;

（如果想出现所有的员工名称，另外一张表后面使用（+）符号）

右外连接：其实这种（+）号没有具体分左右外连接，他（+）随便放置，只是可以表示两种连接（相对的 ）

（5）标准的SQL中的外连接：

左外连接：table1LEFT JOIN table2table1为主表，table2补空

右外连接：table1 RIGHT JOIN table2table2为主表，table1补空

全外连接：table1 FULL JOIN table2 table1,table2中不符合连接的均显示，无记录补空。

1. 自连接，将同一个表设置两个别名，通过别名的两个不同列连接，例：课程的先行课的先行课 course c,course c1 where c.course\_pre\_id=c1.course\_id.;
2. 自然连接，系统自动自己去找两张表中字段相同的字段(可能多个相同字段)，从而连接NATURAL JOIN
3. Using子句：自然连接的补充，做限制…natural Join…Using(id)使用Using中的连接。
4. 内连接 等于连接table1 inner join table2 on table1.id=table2.id where…;

## 6.分组计算函数和GROUP BY子句

（1）分组函数：sum,avg,count,max,min（后者任何数据类型）

（2）count(\*) 满足条件所有行（包括空）

Count(expr)返回满足条件的且表达式不为空的行，

Count(distinct expr)上述条件下，且不重复的行数

1. 当分组函数遇到NULL,count(\*) 计算sum()不计算。AVG(NVL(salary,0))将空值转化为0.如果不转化其会忽略空值。函数会忽略空值，不进入计算。
2. 分组函数可以按照某个字段分组，也可以按照多个字段组合分组。
3. 除了分组函数以外的其他字段，我们都需要写在GROUP BY后面，否则不合法。

错误代码：ORA-00979not a GROUP BY expression.

(6)注意：使用GROUP BY子句分组：不能在WHERE条件中使用分组计算函数表达式，当出现这种需求的时候我们可以使用HAVING子句。

错误代码：ORA-00933:SQL command not properly ended;

Select AVG(salary) ,department\_id FROM employees GROUP BY department\_idhavingavg(salary) >10000.

(7)分组函数可以嵌套使用。

## 6.子查询

（1）子查询：单行比较必须对应单行子查询（返回单一结果值的查询）：比如;=,>

多行比较必须对应多行子查询（返回一个数据集合的查询） ；比如：IN ,>ANY,>ALL;

（2）ANY() 小于任一值

## 7.DML语句

（1）SQL中的增删改语句，

（2）INSERT INTO…VALUES(…,…,…);标准插入

（3）从其他表中copy一行或多行。

INSERT INTO S(id,name)select emp\_id,last\_name from emp where job\_id=‘rep\_ad’;

（4）insert intotable\_name（后面直接跟子查询 with check option 检验要插入的内容，是否符合目标子查询的where条件）values（）；

（5）UPDATE语句：

Update … set … where …

1. 使用子查询结果作为更新的值。
2. UPDATE 当存在约束的时候，某些更新可能会失败。
3. DELETE语句

DELETE FROM…WHERE…

(9)删除表中所有记录 就是不加限制条件即可。

（10）TRUNCATE TABLE …;不能回滚，所以谨慎使用。

（11）当存在约束的时候，可能会删除失败。（外键）

（12）MERGE语句

MERGE INTO target\_emp t USING source\_emp s

ON (t,employee\_id=s.employee\_id)

WHEN MATCHED THEN

UPDATE SET

t.last\_name=s.last\_name,t.department\_id=s.department\_id

WHEN NOT MATCHED THEN

INSERT

VALUES(s.employee\_id,s.last\_name,s.department\_id);

(如果源表中的数据与目标表中数据匹配，更新，如果不匹配（不存在），则插入)；

## 8.事务控制

（1）在数据该已经没有更改时，但没有Commit前，本地session能够看到，其他用户看不到。

（2）Oracle实现了一致性。

（3）Oracle事务的隔离级别：提交读，串行化

## 9.锁

（1）作用：防止并发事务对相同资源（所谓资源是指 表 /行/共享数据结构/数据字典行等）进行更改的时候，相互破坏。

（2）锁既有隐式的；也有显式的；但某用户对某一批数据进行更改，而未提交之前，Oracle会隐式进行加锁，

显式枷锁：select …from tableA WHERE…for uodateNOwait;

（3）查询死锁发生语句：

selectdecode(request,0,'Holder:','Waiter:')||sidsess,

id1,

id2,

lmode,

request,

TYPE

FROMgv$lock

WHERE (id1,id2,TYPE) IN (SELECT id1,id2 ,TYPE

FROMgv$lock

where request>0

andtype!='HW')

ORDERBY id1,request;

前阻塞后。

（4）创建表后在一个session中插入一条记录，但不提交，在另一个session中插入一条记录，该记录插入在等待之中，此时提交第二条记录失败，出现错误代码:ORA-00001违反唯一约束条件。

插入其实成功，但是还是会报错误。

（5）更新记录的时候也会发生死锁。第一个session未提交，第二个session中更新也会处于阻塞。

避免方式：在更新记录后面：select \* from testtab3 where pk1=1 for updateNOwait

## 10.数据库对象—表

(1)表的命名：1）必须以字母开头， 2）长度不能超过30个字符 3）只能包含A-Z,a-z 0-9,$,and# 4）不能与数据库中已有的对象重名 5）不能使用保留字。

(2)建表的语法: create table …(col1,col2…);

(3)从子查询中快速建表;CREATE TABLE as select \* from tableb.

(4)如果只想留表结构，但不想要数据，可以：

CREATE TABLE as select \* from tableb where 1=2;（1=2永远不成立）

(5)ALTER TABLE test ADD (test\_code varchar2(30));--在test表上添加一列。

更改列 ADD 变成MODIFY

(6)更改表名：RENAME oldtablename to newtableName;

(7)TRUNCATE和DELETE FROM TABLE的区别：1）没有Rollback的机会2）HWM标记复位(TRUNCATE TABLE …)

(8)修改列名:

altertableemp\_testRENAMECOLUMNdepartment\_nametoname;

## 11.约束

（1）约束概念：Oracle数据库使用“约束”来阻止对数据库表中的数据的不合法的“增删改”动作，常用的约束有如下几种：

NOT NULL(非空约束)PRIMARY KEY(主键约束) CHECK(自定义约束)

UNIQUE (唯一性约束) FOREIGN KEY(外键约束)

（2）约束创建方式：1）建表的时候同时创建约束 2）另外单独创建约束(ADD)

（3）CONSTRAINT emp\_emp\_id\_pk PRIMARY KEY (EMPLOYEE\_ID);（约束关键字，约束名，约束类型，约束字段）

（4）只写NOT NULL 系统会指定约束名，也可以自己指定约束名 在NOT NULL之前，写CONSTRAINT constraint\_name NOT NULL;

（5）删除约束： ALTER TABLE emp DROP CONSTRAINT emp\_manager\_fk;

删除约束类型（也可以）：ALTER TABLE dept DROP PRIMARY KEY CASCADE(级联删除);dept\_id和外键均删除。

（6）失效/生效约束：

ALTER TABLE empDISABLE/ENABLE CONSTRAINT emp\_emp\_id\_pk CASCADE;

（7）CHECK(pk>0 and col1>0);

(8) 级联删除约束

ALTER TABLE test\_t1 DROP (pk) CASCADE CONSTAINTS;(含有S)

1. 查询系统中的约束

SELECT constraint\_name, constraint\_type,search\_condition from user\_constraints

Where table\_name=’emp’;

1. 添加约束：

Altertable emp\_copy add constraint emp\_copy\_sal check(sal>1000);

## 12.视图

（1）视图类似于表，操作也类型，他是将select查询语句封装起来。

（2）视图创建语句：

CREATE VIEW empvu80

AS SELECT emp\_id,last\_name,sal(AS+ 子查询)

FROM emp

Where dept\_id=80;

（3）删除视图：DROP VIEW view\_name;

（4）TOP-N查询， 例：假如我们需要了解工资最高的3名员工，我们先利用ORDER BY进行排序，然后通过 where ROWNUM<=N --(3);

SELECT …,…,ROWNUM

FROM (select e.salary,e.last\_name

FROM e

ORDER BY e.salary )

WHERE ROWNUM<=3;

实际操作：

selectROWNUM,empno,sal,row\_inner

from (selectempno,sal,ROWNUMrow\_innerfromemporderbysaldesc)

whereROWNUM<=3;

## 13.数据库对象-序列，索引，同义词

（1）序列，Oracle数据库提供序列对象，自动生成目前序列的下一个。满足我们主键自增的要求，并且其每次是唯一的。

（2）语法：CREATE SEQUENCE empno\_seq START WITH 100 MAXVALUE 99999INCREMENT BY 1 NOCACHE NOCYCLE;--不缓存不循环

(Cache 20缓冲区20个，数据已经生成，在缓存区中取值)（修改序列 ALTER）

（3）序列的使用：从序列中取值：CURRVAL取当前值，NEXTVAL取下个值（插入的时候使用NEXTVAL,一次未使用的时候，该序列未生成，不能使用CURRVAL获取当前值。）

（4）序列名.NEXTVAL使用。

（5）删除序列，DROP SEQUENCE …;

（6）索引作用：能够更快的查询，提高查询速度。

（7）CREATE INDEX emp\_last\_name\_idx ON emp(last\_name);--方便以后按员工姓名查询

（8）什么情况下有利于查询：查询条件中使用的列，而且列上的数字跨度很大，而大多数情况下我们要获取的数据量占整个表的4%。不利于：表很小的时候。

（9）在有UPPER()查询的时候，普通索引无效，必须在索引时也加上函数。简称函数索引。

（10）同义词：我们在一个模式下面访问另外一个模式下面的表，直接访问是无效的，我们可以使用模式名.表名来访问，也可以创建一个同义词在该模式下访问，

（11）同义词语法：(synonym)

CREATE SYNONYM cux\_emp for cux.cux\_emp;

(12)我们要通过DB-LINK访问另外一个数据库中的某张表的时候，我们需要加@后缀

Select \* from table1@db-link-name;

## 14.集合操作

（1）UNION/UNION ALL（去重/不去重）并集，两个具有相同列的查询的并集

（2）INTERASECT 交集 两个相同列查询的交部分。

（3）MINUS 差集 A-B A去掉A,B相交部分就是差。

## 15.控制用户权限

（1）用户权限：用户能进行的每一个操作，都与用户所拥有的权限有关。

(2)权限管理：分为 对象权限 和系统权限。

(3)角色：是权限的集合,然后将角色赋于用户

(4)系统权限语法：GRANTCREATE SESSION,CREATE TABLE,CREATE VIEWto user1 WITH ADMIN OPTION(传播权限);REVOKE 权限/create table from 用户（回收权限）

(5)对象权限语法：（对于某一张表的一些操作权限）

GRANT select ,insert ,update on human.empto user1;

（回收）REVOKE 权限/ selectOnhuman.emp from 用户（回收权限）

(6)自定义角色:

CREATE ROLE manager ;

GRANT create table,create view to manager;

GRANT manager to 用户a,b,c；

（回收）REVOKE manager from 用户；

(6)数据库连接的概念：访问另外一个数据库中的表，我们可以通过DB-LINK访问另一数据库中的表：

语句：

CREATE PUBLIC DATABASE LINKhq.acme.com(链接名)

USING ‘sales’（B中的sales用户）

--A数据库中查询

Select\*from[emp@hq.acme.com](mailto:emp@hq.acme.com);

16.GROUP BY 子句的增强

（1）ROLLUP作用：

例：

SELECT department\_id,job\_id,sum(sal)

FROM employees

Where department\_id<60

GROUP BY ROLLUP(department\_id,job\_id);

无ROLLUP 普通分组就是根据部门id和工作id分组

有ROLLUP 先通过job\_id和部门id进行汇总，然后 减去job\_id进行整个部门所有工资汇总。

（2）CUBE作用

在ROLLUP部分全用还加上了第二个参数的工作汇总。最后有一个总汇总

（3）group by grouping sets((department\_id,job\_id)，（manager\_id，department\_id）);就类似多个group by进行 union操作所得出来的结果（4）GROUPING(department\_id) 意思是：说明当前这列是否是在通过GROUPING(何种汇总)所得出来的。

## 16. 子查询进阶

(1)from子句子查询（将其看成一张表来操作。：称为 内嵌视图）

select \* from empe1,(select \* from emp)e2…(有点牵强)

(2)select 子句子查询（利用外表的字段控制。）

例：通过外表emp e2控制该子查询中数据的显示。（相关子查询 该子查询中参考外部住查询的表）

select e2.\*,(selectenamefromempwhere e2.mgr=emp.empno ) MGR\_NAME fromemp e2;

(3)where子句子查询，当作条件。

例：工资大于所在部门工资员工的员工资料(无关子查询 子查询中未参考外部主查询中的表)

Select \* from empwhere sal>(selectavg(sal) from empgroup by deptno);

(4)Exists 操作；其效率高于两个正常的子查询。（NOT EXISTS）

例：是上司的员工信息，（通过EXISTS看他是否有上司，如果不为空，则EXISTS返回TRUE，将主外查询显示，FALSE反之。）

Select \*

From employees outer

Where EXISTS(SELECT ‘x’

FROM employees

WHERE manager\_id=outer.employee\_id);

(5)UPDATE中使用相关子查询：

Update employees e

SET department\_name =

(SELECT department\_name FROM department d WHERE e.department =d.department );

先添加一行，department\_name在employees中，然后通过emp中的部门号，对应dept表中部门号，查询出部门名称，从而更新至emp中

（6）使用WITH子句.(1)如果在后面多次使用则可以简化SQL,2)适当提高性能)

例：查询部门人数最多的部门（红字为 别名）

WITHdeptinfo AS

(SELECT department\_id,count(\*)num FROM employees GROUP BY department\_id)

SELECT \*FROM departments where department\_id IN (select department\_id from deptinfowhere num=(select max(num) from deptinfo))

## 17.添加虚拟机磁盘

1. 编辑虚拟机，并且添加一个20G的硬盘
2. 开启虚拟机 ，进入dos窗口
3. 使用 ： fdisk -l (查询磁盘，找到20G磁盘的位置 /dev/sdv)
4. 扩充主磁盘：pvcreate /dev/sdv
5. 查看当前主机名称： vgs（查的：vg\_mas218）
6. 扩充磁盘组：vgextend vg\_mas218 /dev/sdv
7. 查看当前所进文件夹位置:df -lh(查的：/dev/vg\_mas218/v\_root)
8. 扩充逻辑磁盘：lvextend -L +19G -n /dev/vg\_mas218/v\_root
9. 重新刷新磁盘大小： resize2fs /dev/vg\_mas218/v\_root
10. 查看当前磁盘大小:df -hl

## 18.递归查询

（1）例：emp中的empno和 mgr，构成了递归层次关系（以下使用该关系）

（2）查询语句：

从上往下的各级员工

SELECT ename||’ r eportsto ’||

PRIOR ename“Walk Top Down”

FROM employees

START WITHename=’king’------最高级员工

CONNECT BY PRIORempno=mgr;该员工ID是下级员工的主管ID

从下往上的各级员工

Select empno,ename,job,mgr

From emp

Start with emp=101----最下级员工

Connect by PRIOR mgr =empno;下级员工的主管ID，是上级员工的ID

(3)递归查询的遍历方向

CONCECT BY PRIOR column1=column2

从上往下 column1=PARENT KEY

Column2=CHILD KEY

从下往上 column1=child key

Column2= parent key .

(4)显示树形层次 递归数据（利用LPAD 和LENGTH函数）

SELECT LPAD(ename,LENGTH(ename)+(LEVEL\*2)-2,’\_’)

AS org\_chart

FROM emp

START WITH ename=’KING’

CONNECT BY PRIOR empno=mgr;

## 19.INSERT增强\*

（1）从一个来源插入多个目标表（无条件,有条件 类似）--自动对应着下列相同列名的数据。

INSERT ALL

INTO sal\_history VALUES(EMPID,HIREDATE,SAL)

INTO mgr\_history VALUES(EMPID,MGR,SAL)

SELECT employee\_idEMPID,hiredate,salarysal,manager\_idmgr

FROM employees

WHERE employee\_id>200;

(2)一个来源插入多个目标表（有条件，首次匹配即下一条）。

INSERT FRIST WHEN sal>25000 THEN INTO special\_sal

VALUES (deptid,sal)

WHEN hiredate LIKE (‘%00%’) THEN INTP hiredate\_history\_00

VALUES(deptid,hiredate)

…

Select department\_iddeptid,MAX(hire\_date) HIREDATE

FROM employees

GROUP BY department\_id;

查询出来的数据进行判断，第一条匹配的，成功插入本表，下面判断不再继续。

未匹配的，往下一个判断进行匹配。

（3）列转行，通过

INSERT ALL

INTO sales\_info VALUES (…,…,first)

INTO sales\_info VALUES(…,…,second)

…

Select empno,first,second,…,…

From sales\_source\_data;

## 20.sql进阶功能

（1）分析函数：（import）

selectempno,ename, max(sal)over(partitionbydeptno) department\_max\_sal,

min(sal) over(partitionbydeptno) department\_max\_sal,

avg(sal) over(partitionbydeptno) department\_max\_sal,

sum(sal) over(partitionbydeptno) department\_max\_sal

fromemp;

该函数，可以不需要group by来实现 查出本人所在部门工资，比以前更加容易操作。

因为group by 后面加的约束过多的话，往往查出来的结果不是自己想要的，在后面会感觉更加的麻烦。

（2）分析函数-排序函数

Rank(),dense\_rank(),row\_number();

Rank()和dense\_rank() 会将相同的数据排名相同。Row\_number()根据数据所在行数来排，没有相同的排名

Rank()和dense\_rank()的区别：

Rank()函数会将相同的数据位置记录，例：100，300，300，400（排序 1，2，2，4）

Dense\_rank()函数会将相同的数据位置记录为相同，不会默认增长：例：100，300，300，400（排序： 1，2，2，3）

（3）闪回\*

deletefrom dept\_temp1

wheredeptno=20;----删除数据

commit;---提交

select \* from dept\_temp1 wheredeptno=20;

--查询

insertinto dept\_temp1

select \* from dept\_temp1 AS OF TIMESTAMPSYSDATE-5/(24\*60)

WHEREdeptno=20;

---查询5分钟之内删除的数据

--插入误操作数据进入原表之中

（4）临时表：\*

语法：create global temporary table table\_name (…,…,…) ON COMMIT PRESERVE ROWS;

（提交保留字段 会话范围有效 基于会话

Delete rows 基于事务）

（5）区别：基于会话的临时表，只能在本会话中有效，

基于事务的临时表，提交事务后无数据。

（6）物化视图（通过表中某种变化，自动或手动触发刷新）

creatematerializedviewlogon employees withPRIMARYKEY;

CREATEMATERIALIZEDVIEW employees\_90 *--创建物化视图*

BUILDIMMEDIATE*--在视图编写好后创建*

REFRESHFASTWITHPRIMARYKEY*--根据主表主键增量刷新（FAST，增量）*

ONDEMAND*--在用户需要时，由用户刷新*

ENABLEQUERYREWRITE*--查询重写*

AS

SELECT \* FROM employees wheredepartment\_id = 90;

手动刷新：

PL/SQL语句：

Begin

DBMS\_MVIEW.REFRESH(‘employees\_90’，’f’);

END;

(7)物化视图，建立的时候先要建立相应的LOG日志，因此在删除的时候，我们也需要先删除LOG日志。

DROP MATERIALIZED VIEW LOG ON employees;

DROP MATERIALIZED VIEW employees\_90;

(8)length/length 和substr和substrb

前者按照字符个数，后者按照字节个数（一个中文字符2个字节（看数据库设置），英文字符一个字节）

（9）策略函数（进行一些查询方面的控制）\*

*1）创建策略函数 Function:*

CREATEORREPLACEFUNCTIONemp\_vpd\_test(p\_schemaVARCHAR2,p\_object VARCHAR2) RETURNVARCHAR2AS

BEGIN

RETURN'employee\_id>= 200';

END;

2）对数据库对象应用策略函数：

BEGIN

dbms\_rls.add\_policy(object\_schema =>'APPS'

,object\_name =>'EMPLOYEES'

,policy\_name =>'VPD\_TEST'

,function\_schema =>'APPS'

,policy\_function =>'EMP\_VPD\_TEST');

END;

3）通过查询数据字典，确认数据库对象上是否有策略函数：

SELECT \* FROMdba\_policies t WHEREt.object\_name = 'EMPLOYEES';

SELECT \* FROM EMPLOYEES;

4) 失效策略

BEGIN

dbms\_rls.drop\_policy(object\_schema =>'APPS'

,object\_name =>'EMPLOYEES'

,policy\_name =>'VPD\_TEST'

*--,function\_schema => 'APPS'*

*--,policy\_function => 'EMP\_VPD\_TEST'*

);

END;

(21)例题注意

1）列出矩阵类型薪资总和数据，横轴为工作类型，纵轴为部门代码

代码：

selectdeptno,

sum(decode(job, 'CLERK', sal)) SALESMAN,

sum(decode(job, 'SALESMAN', sal)) SALESMAN,

sum(decode(job, 'PRESIDENT', sal)) PRESIDENT,

sum(decode(job, 'MANAGER', sal)) MANAGER,

sum(decode(job, 'ANALYST', sal)) ANALYST

FROMemp

GROUPBY DEPTNO

ORDERBYdeptno;

# 2.PL/SQL部分

## 1.PLSQL概述

（1）PL/SQL是利用SQL来处理数据，使用控制结构来处理业务逻辑。

（2）PL/SQL好处：它可以通过减少来回交互减轻网络流量压力，节省时间。

（3）PLSQL 是一种类PASCAL语言，每段程序都是由BLOCK组成的。

（4）PL/SQL的块 包括3种，匿名块，存储过程，函数（有返回值）

## 2.PLSQL变量

（1）变量类型：1）系统内置的常规简单变量类型：

2）用户自定义复杂变量类型,比如记录类型

3）引用类型：保存了一个指针值

4）大对象类型（LOB）;保存一个指向大对象的地址。、

（2）PLSQL特有的%TYPE属性来声明与XX类型一致的变量类型：

%ROWTYPE属性声明和table表中的一条记录一样的类型。

语法：

Identifier table.column\_name%type;

(3)DBMS\_OUTPUT.PUT\_LINE();打印数据到界面上

(4)在PLSQL中 使用sql语句后面可以不需要；而PL过程化语言后面需要。

(5)PLSQL中注释：--，/\*…\*/

(6)有些SQL中的函数是不能在PL/SQL的过程化语言中直接被使用的。

DECODE函数，分组函数，（AVG,MIN,COUNT,SUM,…）

而字符串函数，数据转换函数，时间函数是可以的

（7）外部变量，<<outer>>,限定词outer表示外层，里层要访问外层的数据的时候，需要加上outer. 。

（8）可绑定变量：非PLSQL变量

声明：VARIABLE return\_code NUMBER

使用的时候 用：标识它

在SQL plus中使用，外部变量，一般不适用。

（9）不要以关键字作为变量名，有时候会产生问题（sum）

## 3.PLSQL中的SQL语句 \*

（1）SQL plus 需要执行PLSQL语句的输出语句时候

Set serveroutput on;

(2)

## 4.PLSQL中的控制语句

(1) 判断语句:

语法：IF condition1 THEN …ELSIF condition2 THEN…ELSIF condition3 THEN…ELSE … END IF;

(2) 需要注意判断语句中对NULL的判断处理，(大概的判断级别就是 TRUE>NULL>FALSE)

AND：特殊情况 TRUE AND NULL =NULL; FALSE AND NULL =FALSE;(交换亦然)

OR:特殊情况 TRUE OR NULL =TRUE; FASLE OR NULL =NULL;

NOT:NOT NULL 还是NULL;

(3)结束时，END IF要分开，中间有空格.

(4)循环语句：

1）基本循环（先进入判断，类型do while()语句，再判断）

语法：

LOOP

语句……

EXIT WHEN 条件；

END LOOP;

2)FOR循环（大多数是利用后面讲的游标来配合操作）

语法：

FOR counter IN lower\_bound..upper\_bound LOOP

Statement1;

Statement2;

..

END LOOP

(3);WHILE循环（先判断，再进行操作）

语法：

declare

v\_number number:=10;

begin

while v\_number>6 loop

v\_number:=v\_number-1;

dbms\_output.put\_line(v\_number);

end loop;

end;

(4)嵌套循环

多个循环嵌套进行，注意的是每个循环利用普通的循环语句，虽然有EXIT WHEN…

用来退出循环，但是在end loop；之前 还是可以写输出语句。依然会合理输出。

## 5.PLSQL中复杂自定义数据类型

（1）自定义数据类型

定义记录类型：

TYPE emp\_record\_type IS RECORD(

last\_name varchar2(25),

job\_id varchar2(10),

salary number(8,2));

emp\_record emp\_record\_type; （名称 数据类型）；

（2）%ROWTPYE ---记录型数据类型一定是一条记录

emp\_rec emp%ROWTYPE;

（3）PLSQL内存表：index by table ,这种结构类似于数组,

这种类型必须包含两部分1.使用BINARY\_INTEGER 类型构成的索引主键；

2.另外一个简单类型或者用户自定义类型的字段作为具体的数组元素。这种类型可以自动增长，所以也类似于可变长数组。

TYPE ename\_table\_type IS TABLE OF employees.last\_name%TYPE INDEX BY BINARY\_INTEGER;

Ename\_table ename\_table\_type;

例：

*---内存表 和数组相似*

declare

type emp\_table\_type is table of emp%rowtype index by binary\_integer;

(type hiredate\_table\_type is table of date index by binary\_integer—数据类型-日期)

my\_emp\_table emp\_table\_type;

begin

FOR i IN 7369..7369 LOOP

SELECT \* INTO my\_emp\_table(i) from emp where empno=i;

END LOOP;

FOR i IN my\_emp\_table.first..my\_emp\_table.last loop

dbms\_output.put\_line(my\_emp\_table(i).ename);

end loop;

end;

## 6.PLSQL中的游标

（1）.Oracle中游标分为：隐式游标和显式游标

（2.）隐式游标的几个有用 的属性：

SQL%ROWCOUNT 受最近sql语句影响的行数

SQL%FOUND 最近的sql语句是否影响了一行以上的数据

SQL%NOTFOUND 最近的SQL语句是否未影响任何数据

SQL%ISOPEN 对于隐式游标而言永远为FALSE

（3.）显式游标 –普通游标

一般过程：

Declare ->open ->fetch -.>empty –yes->close;

声明->打开游标->提取当前行->测试行是否存在——>关闭游标

声明：cursor emp\_cursor IS

select empno,ename

from emp;

打开: open emp\_cursor;

提取当前行-判断

loop

FETCH emp\_cursor into v\_empno,v\_ename;

exit when emp\_cursor %rowcount>10 or emp\_cursor%notFOUND;

dbms\_output.put\_line(to\_char(v\_empno)||' '||v\_ename);

end loop;

关闭：close emp\_cursor;\

（4.）显式游标—for循环

1）For式游标不需要手动打开和关闭，其自动打开。

2）一种是在IN后面直接加select语句—类似于匿名函数只能被使用一次

3）另外一种

先声明好游标，然后直接利用,这样的游标可以重复被使用。

（5）.游标能带参数（在游标名后带参数）带的参数在打开的时候可以使用

（6）.FOR UPDATE NOWAIT语句：

在我们打开一个游标是为了更新或者删除一些数据，而往往我们需要锁定这些数据，让他更新完或者删除完，我们应该使用该语句，以免锁定失败后程序继续，造成死锁,\

用法：在select 语句后放置for update nowait;(多个用户操作相同的数据)

（7）.在利用游标update表我们需要向相应的地方，添加相应的值current of emp\_cursor

Begin

For emp\_date in emp\_cursor loop

Update emp e

Set e.sal=299999

Where current of emp\_cursor;(相似 e.empno=emp\_date.empno)

End loop;

End;

（8）带参数的游标，就是跟了两个参数，而不是将游标中的参数获取给该参数，for update

CURSOR emp\_cursor(p\_empno NUMBER) IS

select empno from emp where empno = p\_empno;

begin

open emp\_cursor(7369);

## 7.PLSQL中例外处理（异常处理）

（1）PLSQL中的一般有两种:

1.Oracle内部错误抛出的例外：1）预定义例外(有错误号+常量定义)和非预定义例外(

仅有错误号，无常量定义)

（2）.程序员显式抛出的例外

（3）.PLSQL中的例外捕获和传递：与其他语言类似，如果例外在当前中被处理，则到此为止，否则会被传递到外层（外层BLOCK或外层调用者函数）

（4）．PLSQL中处理例外的一般语法:

EXCEPTION

WHEN exception1 …THEN

Statement1;

,,,

When exception2 then

…;

WHEN OTHERS THEN

Statement1;

（5）.Oracle预定义的例外（异常）

-NO\_DATA\_FOUND 数据未找到

-TO\_MANY\_ROWS 数据过多

-INVALID\_CURSOR 无效游标

-ZERO\_DIVIDE 0作为除数

-DUP\_VAL\_ON\_INDEX 唯一索引上有重复值

（6）.OTHERS的处理:others表明程序员未能预计到这种错误,我们可以通过Oracle提供了两个内置函数SQLCODE和SQLERRM分别来返回Oracle错误号和错误描述

（7）．处理非预定义的Oracle错误：此类错误属于Oracle错误，有编号,但无错误名称定义，使用时需要先声明，并进行错误初始化。

（8）.自定义例外，先声明再使用

声明->抛出例外->处理例外

命名->使用RAISE语句显式的抛出例外->处理抛出的例外。

*--自定义异常*

declare

e\_invalid\_department exception; --声明异常

begin

update dept set dname=&p\_depart\_desc where deptno=&p\_dept\_number;

if sql%notfound then

raise e\_invalid\_department; --抛出异常

end if;

commit;

exception when e\_invalid\_department --处理异常

then dbms\_output.put\_line('No Such department\_id.');

end;

（9）.raise\_application\_error()函数,例外需要先定义非常麻烦,所以可以使用该函数简化处理..

……

If sql%notfound then

Raise\_application\_error(-20202,’this is not a valid manager’);

End if;

（10）.例外的传递

declare

e\_no\_rows exception;

e\_integrity exception;

pragma exception\_init(e\_integrity, -2292);

cursor emp\_cursor IS

select empno, ename from emp;

begin

for c\_record in emp\_cursor loop

begin

update emp a

set a.sal = a.sal + 1000

where a.empno = c\_record.empno + 10000;

if sql%notfound then

raise e\_no\_rows; --内部抛出异常

end if;

end;

end loop;

exception

when e\_integrity then

dbms\_output.put\_line('integrity');

when e\_no\_rows then -–外部处理异常c

dbms\_output.put\_line('not found');

end;

## 8.PLSQL中的存储过程(procedure)

（1）.PLSQLDEVELOPER 开发存储过程

File->new->program Windows->Procedure;

（2）.参数模式

in -默认模式 -用以把值传给过程 -参数可以是常数，变量，表达式

out -必须显式指定 -将值返回出来，-必须是个变量 –不能赋予默认值

in out –必须显示指定 –用以把变量传递给过程,并返回给外界 –必须是个变量 –不能赋予默认值

（3）.语法:

Create or procedure procedure pro\_name (

…in …,… out …)

IS

BEGIN

SELECT …,…,… into …,…,…from table\_name where …=…;

End pro\_name;

（4）.调用存储过程

EXECUTE proc\_name 或 CALL proc\_name(); --这是在SQL plus中

在PLSQL development 中：

Declare

…

Begin

Proc\_name(…,…);

End;

（5）.参数传递的方式：顺序传递 或使用=>符号传递

add\_proc(‘TRAINING’,2500);

add\_proc(p\_loc=>2400,p\_name=>’education’);

p\_loc一定要和存储过程开始的name一样，后面的名字随意。

（6）.过程调用的例外处理:

例外也可以被传递到外层之中，被处理

（7）删除存储过程：DROP PROCEDURE proc\_name;

## 9.PLSQL中的函数(function)

（1）函数语法：

Create OR REPLACE FUNCTION get\_sal

(p\_id IN emp.empno%type )

RETURN NUMBER ----无分号

IS

v\_sal emp.sal%type:=0;

BEGIN

SELECT sal

Into v\_sal

From emp

Where empno=p\_id;

Return v\_sal;

End get\_sal;

（2）用户自定义函数可以在sql大部分语句中被调用（存储过程不行）

（3）自定义函数在SQL中被调用限制如下：

1)只能有IN模式的参数（out,in out 不行）

2）只能接受sql数据类型的参数

3)函数返回的数据类型也必须是有效的数据类型，

4）在SQL中使用的函数，不能有DML语句

5)在UPDATE或DELETE 语句中调用的函数，其函数内部不能有针对同一张表的查询语句

6）不能有事务结束语句（COMMIT,ROLLBACK）;

（4）删除函数：DROP FUNCTION function\_name;

（5）赋予函数权限

Grant execute ONS function\_name TO green;

（6）定义者权限:函数执行时,对表的访问默认使用定义权限.

AUTHID CURRENT\_USER

Is… 只能调用函数，不能访问表

10. PLSQL中的包PACKAGE

(1)package 概念;按照业务逻辑/把相关的FUNC,PROC组织在一起,形成一个函数或过程集合.

(2)package组成：package由包说明和包体两部分构成；

(3)包说明：声明 异常，变量，函数，存储过程

包体：实现上述东西，

(4)包说明中定义的变量，过程都是公共的，在包体中定义其实现 是私有的。

(5) 好处:1)模块化

2）易设计

3）信息隐藏

4）加载性能高

5）重载

(6)包中函数先定义的才能被后定义的调用。（函数公有化 也是可以的）；

(7)包说明语法：

Create or replace package pack\_name is

…

Procedure proc\_name();

Function proc\_name() return field\_num;…

匿名块]

End pack\_name;

(8)包体；

create or replace package body pack\_name is

Procedure proc\_name()

Is

Begin

Null;

End;

Function proc\_name()

End pack\_name;

(9)存储过程可以被其他存储过程调用

(10)存储过程不能被函数调用.??

(11)Package中的初始化代码：可以统一在包中初始化

BEGIN

Select rate\_value into tax

From tax\_rates

Where name=’TAX’;

End pack\_name;

(12)Package 中的变量的持久状态：

在同一session中一个函数改变一个变量的值后，后面就是该变量变化后的值。另外session值是不改变的。

10.PLSQL工具包

1.动态SQL:运行时，临时拼接的SQL语句。

2.动态SQL可以使用Oracle内置包DBMS\_SQL来执行,也可以使用,EXECUTE IMMEDIATE 语句来执行。

3.普通的执行动态SQL--- EXECUTE IMMEDIATE 语句来执行。

execute immediate ' update '||p\_table\_name||' set sal = '||p\_sal\*2||' where dname = '||p\_dname;】

## 10.触发器\*

1.数据库触发器：对数据库对象的操作可引发很多其他的事件，比如before insert,before update等

2.注意：除非迫不得已,尽量避免使用Trigger，因为这会导致维护困难。

3.创建Triggle:Triggle的定义语句中涉及的关键因素：

时机：Before或者After或 instead of

事件：update或insert 或delete

对象：表名（视图名）

条件：满足特定where条件才执行；

内容：通常是PL/SQL代码块

4.重点注意：

Instead of:用Trigger的内容替换事件本身的动作。

ROW级触发器：SQL语句影响到的每一行都会引发Trigger

Statement级:一句SQL语句引发一次，不过他影响了多少行（甚至0行）

5.Triggle的触发顺序”

1,)Before statement trigger

2)before row trigger

3)after row trigger

4)after statement trigger

6.语句级触发器

CREATE OR REPLACE TRIGGER secure\_emp

BEFORE INSERT ON employees

BEGIN

IF (TO\_CHAR(SYSDATE, 'DY') IN ('SAT', 'SUN')) OR

(TO\_CHAR(SYSDATE, 'HH24:MI') NOT BETWEEN '08:00' AND '18:00') THEN

RAISE\_APPLICATION\_ERROR(-20500,

'You may insert into EMPLOYEES table only during business hours.');

END IF;

END;

7.多事件触发器

在不同事件的时候触发器做不同的事情。

8.行级触发器

--行级触发器

CREATE OR REPLACE TRIGGER restrict\_salary

BEFORE INSERT OR UPDATE OF salary ON employees

FOR EACH ROW —-不同于语句级触发器的部分

BEGIN

IF NOT (:NEW.job\_id IN ('AD\_PRES', 'AD\_VP')) AND :NEW.salary > 15000 THEN

RAISE\_APPLICATION\_ERROR(-20202, 'Employee cannot earn this amount');

END IF;

END;

9.old和new

Old是插入前的记录表（无），new是插入后的记录表。

Old,new update 都有

Old new(无) delete

例：

/\*--触发器1. 在表HAND\_STUDENT上创建一个触发器，当表数据新增、更

新或删除时，都在表HAND\_STUDENT\_HIS新增一条记录，记录LAST\_UPDATE\_DATE及STATUS\*/。

create or replace trigger hand\_student\_trigger

after insert or update or delete on hand\_student

for each row

declare

-- local variables here

last\_update\_date hand\_student\_his.last\_update\_date%type;

status hand\_student\_his.status%type;

begin

IF INSERTING THEN

last\_update\_date:=sysdate;

status:='N';

insert into hand\_student\_his values

(:new.student\_no,:new.student\_name,:new.student\_age,:new.student\_gender,last\_update\_date,status);

ELSIF UPDATING THEN

last\_update\_date:=sysdate;

status:='U';

insert into hand\_student\_his values

(:new.student\_no,:new.student\_name,:new.student\_age,:new.student\_gender,last\_update\_date,status);

ELSE

last\_update\_date:=sysdate;

status:='D';

insert into hand\_student\_his values

(:old.student\_no,:old.student\_name,:old.student\_age,:old.student\_gender,last\_update\_date,status);

end if;

end hand\_student\_trigger;

使用 :old.ename;

10.instead of

会将原来的操作失效，并且执行其他语句代替它。

11.管理Trigger

失效/生效

ALTER TRIGGER trigger\_name DISABLE | ENABLE （ALL TRIGGERS）;

重新编译

ALTER TRIGGER trigger\_name COMPILE;

删除

DROP TRIGGER trigger\_name；

12.触发器和约束。

前台应该收收到错误提示，与某某约束冲突。  
但不会说是那个触发器干的。  
只说当前操作某个表的某个约束冲突。

所以我们应该少用触发器。

## 11.数据库对象的依赖关系

1.注意view直接依赖于表，存储过程等也是直接或间接依赖于表，在修改表结构的时候，我们需要注意view，等是否会发生改变，是否能够在改变表结构后，在进行运行，若成功该view等继续变为有效，若不行则无效。

## 12.易错地方

1．For游标中循环变量无法直接输出，只能利用 . 操作

2.在写IF语句的时候，条件部分不要加括号，并且结尾要加END IF ，循环也类似。

3. *--聚集函数 sum(),avg()单独使用 又不会出现no\_data\_found异常*

*--其他聚集函数 sum(),avg()+group by 会出现no\_data\_found异常*

4.字符串可以使用‘&name’来输入

5.

execute immediate ' update '||p\_table\_name||' set sal = '||p\_sal\*2||' where dname = '''||tab1 ||'''';

前三个引号：前两个 转义，后一个圈起来

后四个引号，前2个转义为引号，后两个在将其圈起来。

6.注意PL/SQL中赋值是 ：= 比较是 = 类似于 PASCAL;

7.内存表： 先要自己定义，然后使用的时候，将其当成一个数据类型。

8.user1/user1 -平常plsql

User2/user2 –sql

User3/user3 –pl sql

8.入参 p\_,,, 出参 x\_,,,

9.linux改文件权限： chmod 777 文件名

10.函数在异常中也要返回值，不然会报错误

11.declare后面是参数的声明，而在函数和存储过程中IS后面是参数的声明。

## 13.例题

1.存储过程

CREATE OR REPLACE PROCEDURE upd\_sal

(p\_jobid IN jobs.job\_id%type,

p\_minsal IN jobs.min\_sal%type,

p\_maxsal IN jobs.max\_sal%type)

IS *---is*

v\_dummy VARCHAR2(1);

e\_resource\_busy EXCEPTION;

sal\_error EXCEPTION;

PRAGMA EXCEPTION\_INIT (e\_resource\_busy , -54); *--初始化异常*

BEGIN

IF (p\_maxsal < p\_minsal) THEN

DBMS\_OUTPUT.PUT\_LINE ('ERROR ... MAX SAL SHOULD BE > MIN SAL');

RAISE sal\_error;

END IF;

SELECT ''*---处理资源正忙，在update前面增加select语句*

INTO v\_dummy

FROM jobs

WHERE job\_id = p\_jobid

FOR UPDATE OF min\_sal NOWAIT;

UPDATE jobs

SET min\_salary = p\_minsal,

max\_salary = p\_maxsal

WHERE job\_id = p\_jobid;

EXCEPTION

WHEN e\_resource\_busy THEN

RAISE\_APPLICATION\_ERROR (-20001, 'Job information is

currently locked, try later.');

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR

(-20001, 'This job ID does not exist');

WHEN sal\_error THEN

RAISE\_APPLICATION\_ERROR(-20001,'Data error..Max salary should be more than min salary');

END upd\_sal;

2/DECLARE

v\_job USER\_JOBS.job%TYPE;

BEGIN

DBMS\_JOB.SUBMIT ( v\_job,

'BEGIN DBMS\_OUTPUT.PUT\_LINE (get\_job\_count(110)); END; ',

SYSDATE,

'ADD\_MONTHS(SYSDATE, 6)');

DBMS\_JOB.RUN(v\_job);

DBMS\_OUTPUT.PUT\_LINE('JOB: '|| v\_job ||

' COMPLETED AT - ' || SYSDATE);

END;

p76 ppt

3.写文件

create or replace directory MY\_DIR

as '/home/oracle/plog';--创建目录

grant create any directory to user3;

grant read,write on directory MY\_DIR to user3;---赋予权限

declare

fhandle utl\_file.file\_type; *--声明一个文件操作句柄*

begin

fhandle :=utl\_file.fopen('MY\_DIR','student','w');*--实例化一个句柄，并且打开*

utl\_file.put\_line(fhandle , 'eygle test write one'); *--写入信息*

utl\_file.fclose(fhandle); *--关闭句柄*

end;

4.读文件,并且插入

DECLARE

filehandle utl\_file.file\_type;

filebuffer varchar(200);

BEGIN

filehandle := utl\_file.fopen('MY\_DIR','student','r');

loop

begin

utl\_file.get\_line(filehandle,filebuffer);

dbms\_output.put\_line(filebuffer);

INSERT INTO TEST(a) VALUES(filebuffer);

EXCEPTION

WHEN no\_data\_found THEN

exit ;

End;

END LOOP;

utl\_file.fclose(filehandle);

COMMIT;

END;

# 3.蚂蚁书 PL/SQL 摘记

## 1．概念部分

1.PL/SQL 不是作为一种独立的语言设计的，必须在一个特定的宿主环境内才能执行。（SQL PLUS PLSQL DEVELOPER）

2.Pl/sql的特点：1）高度化，易懂，易理解

2）标准的，可移植性的Oracle的开发语言

3）嵌入式语言

4）高性能，高集成度

3.Oracle用通过提供AUTHID子句为PL/SQL的执行授权模型 （‘？？？？？？）、

4.ODBC –开发数据库连接 JDBC—Java数据库连接

5.实现java输入数据库账号和密码，cmd中成功连接数据库

6.SQL \*plus中两种独立的变量类型：1）DEFINEs 和绑定变量

DEFINE:

DEFINE x=’qeqwe’;

查询 ： DEFINE x;

引用 '&x';

绑定变量：

先声明：variable x varchar(10)

赋值：可以在存储过程中赋值

区别于其他变量 :x=’hello’;

输出（打印）：PRINT :X;

7.匿名块 可以多级嵌套。

Declare

…

Begin

Declare

Begin

...

End;

End;

8.存储过程中可以使用匿名块.

create or replace procedure test

is

begin

declare

begin

dbms\_output.put\_line('test');

end;

end;

9.PL/SQL中特殊符号的意义：

1）\*\* 双星号：幂运算

2）<> != 或者^= ~=(波浪线)都代表不相等

3）@ At符号：远程位置指示符

10. ！=之间不得有空格或者tab符号，不然会报出错误

11.to\_date 和to\_timestamp的区别

select to\_date('2012-7-28 03:12:00','yyyy-mm-dd hh24:mi:ss')-sysdate from dual；

两个日期相减得到的是 天数（后续的小时，分钟 全转化为天数）

 select to\_timestamp('2018-7-28 03:12:00','yyyy-mm-dd hh24:mi:ss')-systimestamp from dual

得到的是

+000000360 17:10:23.391632000

360天 17小时：10分钟：23秒… 具体的时间差值

12．在Oracle pl/sql中，一个空字符串和零个字符串的直接量是没有区别，判定不出来（未实现区别）

CHAR(N)赋值一个长度0的字符串，数据库会自动帮他填满空格，因此它不会是NULL

13.字符串拼接问题

select '''hello''' from dual; --输出 'hello'

select '''''' from dual; --输出 ''（q’{…}’）

因为外部两个'相当于标志，然后里面两两一对’‘''代表一个'

14.PRAGMA关键字（契约—动作）

语法：

PRAGMA instruction\_to\_compiler;

提示编译器，如何编译。

## 2.pl/sql程序结构

1.三值逻辑：布尔类型的值： TRUE FALSE NULL（UNKNOWN 只是PLSQL中没有区别，但是在其他地方可能会有区别，所以要自己有所区别）;

2.如果IF中没有ELSE的话，NULL值是不会被执行的，可能和想要的结果就有差距，需要特别注意。

3．IF 和CASE语句都利用了短路求值（不符合则跳过该判断，进入下一个判断）

4. CASE语句的语法

CASE expression

WHEN result1 THEN

Statements1

When … then

,…

Else …

End case;

5.在case …when 后面加表达式 case when … then…

Case true(true 可以忽略)

When expression then …

Else

End case;

6.goto语句

begin

goto second\_output;

dbms\_output.put\_line('11`1111111111111111111');

<<second\_output>>

dbms\_output.put\_line('222222222222222222');

end;

测试的时候感觉可以使用，其他地方少量使用。

7.空语句

NULL;

8.终止简单循环：

EXIT和EXIT WHEN 语句

9.varchar 和varchar2的区别：varchar2会计算字符串前后的长度，（：？）

10.数据库管道通信（会产生死锁 未执行成功）

代码如下：

---管道通信

declare

pipename Constant varchar2(12) := 'signaler';

result integer;

pipebuf varchar2(64);

begin

/\*创建一个指定名字的私有管道\*/

result := DBMS\_PIPE.create\_pipe(pipename);

LOOP

--data\_gathering\_procedure;

DBMS\_LOCK.sleep(10);

/\*检查管道里是否有消息\*/

IF DBMS\_PIPE.receive\_message(pipename, 0) = 0 THEN

/\* 解释消息表采取动作\*/

DBMS\_PIPE.unpack\_message(pipebuf);

EXIT WHEN pipebuf = 'stop';

END IF;

END LOOP;

end;

--

--同一管道中 伙伴进行终止上述程序

DECLARE

pipename varchar2(12):='signler';

result integer :=DBMS\_PIPE.CREATE\_PIPE(pipename);

begin

dbms\_pipe.pack\_message('stop');

end;

11 for循环：数值型FOR循环和游标FOR循环。

12.REVERSE 关键字可以迫使循环从上界到下界递减式进行。

For loop\_counter IN REVERSE 1..10 (这里还是显示正序)

LOOP

…可执行语句…

END LOOP;

13. continue 在循环中可以使用。（continue可以直接跳过此次循环，表达逻辑更加直接）

语法如下：(还是略有问题)

CONTINUE WHEN MOD(l\_index,2)=0;

14.异常处理 6.4.1

# 4.Aurora框架

1.配置虚拟机tomcat和本地tomcat ，及其工程路径



2.

# 5．GIT

## 1.创建版本库

现在总结一下今天学的两点内容：初始化一个Git仓库，使用git init命令。

添加文件到Git仓库，分两步：

第一步，使用命令git add <file>，注意，可反复多次使用，添加多个文件；

第二步，使用命令git commit，完成。

## 2. 查看工作区状态及修改数据

要随时掌握工作区的状态，使用git status命令。

如果git status告诉你有文件被修改过，用git diff可以查看修改内容。

## 3.版本回退以及前进

现在总结一下：

HEAD指向的版本就是当前版本，因此，Git允许我们在版本的历史之间穿梭，使用命令git reset --hard commit\_id。（回退上个版本 git reset –hard head^）

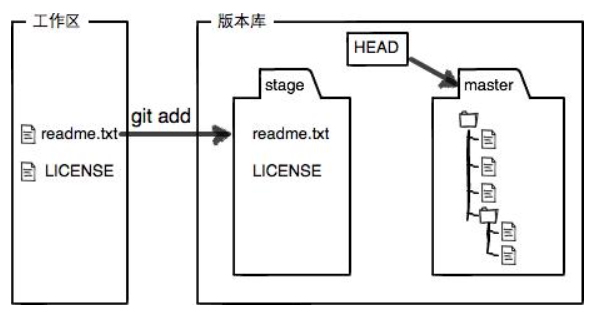
穿梭前，用git log可以查看提交历史，以便确定要回退到哪个版本。

要重返未来，用git reflog查看命令历史，以便确定要回到未来的哪个版本。

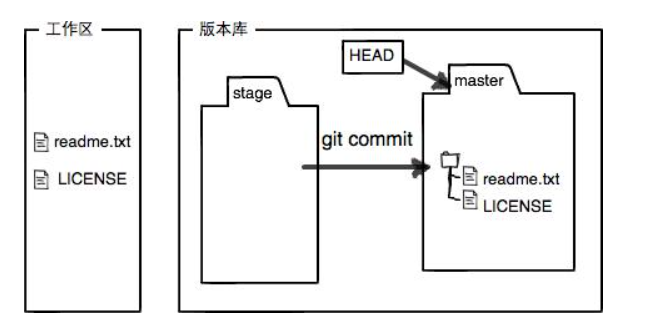
## 4.工作区和暂存区

工作区：就是在你电脑上能看到的目录

版本库：工作区中隐藏目录 .git 是git的版本库



当未提交的时候,stage中有两个文件，而master分支上挂着这两个文件，如果，commit之后，



## 5.管理修改

现在，你又理解了Git是如何跟踪修改的，每次修改，如果不add到暂存区，那就不会加入到commit中。

## 6.撤销修改

又到了小结时间。

场景1：当你改乱了工作区某个文件的内容，想直接丢弃工作区的修改时，用命令git checkout -- file。

场景2：当你不但改乱了工作区某个文件的内容，还添加到了暂存区时，想丢弃修改，分两步，第一步用命令git reset HEAD file，就回到了场景1，第二步按场景1操作。

场景3：已经提交了不合适的修改到版本库时，想要撤销本次提交，参考[版本回退](https://www.liaoxuefeng.com/wiki/0013739516305929606dd18361248578c67b8067c8c017b000/0013744142037508cf42e51debf49668810645e02887691000)一节，不过前提是没有推送到远程库

## 7.删除文件

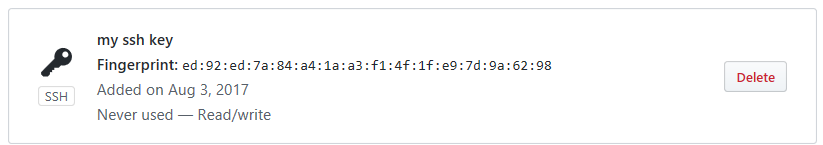
1．现在你有两个选择，一是确实要从版本库中删除该文件，那就用命令git rm删掉，并且git commit

2. git checkout其实是用版本库里的版本替换工作区的版本，无论工作区是修改还是删除，都可以“一键还原”。

3. 命令git rm用于删除一个文件。如果一个文件已经被提交到版本库，那么你永远不用担心误删，但是要小心，你只能恢复文件到最新版本，你会丢失**最近一次提交后你修改的内容**。

8.远程仓库

先利用 $ ssh-keygen -t rsa -C [youremail@example.com](mailto:youremail@example.com) 来获取公钥和私钥，并且在GITHUB打开SETTING 设置SSH KEYS ，



## 9.添加远程库&

1.要关联一个远程库，使用命令$ git remote add origin [git@github.com:793404933/learngit.git](mailto:git@github.com:793404933/learngit.git)关联后，使用命令git push -u origin master第一次推送master分支的所有内容；此后，每次本地提交后，只要有必要，就可以使用命令git push origin master推送最新修改；

分布式版本系统的最大好处之一是在本地工作完全不需要考虑远程库的存在，也就是有没有联网都可以正常工作，而SVN在没有联网的时候是拒绝干活的！当有网络的时候，再把本地提交推送一下就完成了同步，真是太方便了

推送：git push -u origin master

## 10.从远程库克隆 &

1.克隆代码：git clone [git@github.com:793404933/gitskills.git](mailto:git@github.com:793404933/gitskills.git)（仓库地址）

添加关联：$ git remote add origin [git@github.com:793404933/learngit.git](mailto:git@github.com:793404933/learngit.git)

2．要克隆一个仓库，首先必须知道仓库的地址，然后使用git clone命令克隆。

3.Git支持多种协议，包括https，但通过ssh支持的原生git协议速度最快。

## 11.创建与合并分支 &

Git鼓励大量使用分支：

查看分支：git branch

创建分支：git branch <name>

切换分支：git checkout <name>

创建+切换分支：git checkout -b <name>

合并某分支到当前分支：git merge <name>

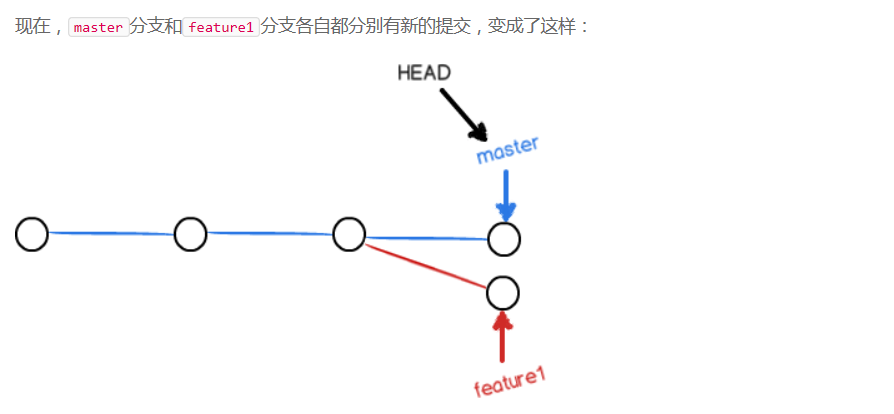
删除分支：git branch -d <name>

Cat readme.txt 查看文件内容。

## 12.解决冲突 &

当Git无法自动合并分支时，就必须首先解决冲突。解决冲突后，再提交，合并完成。

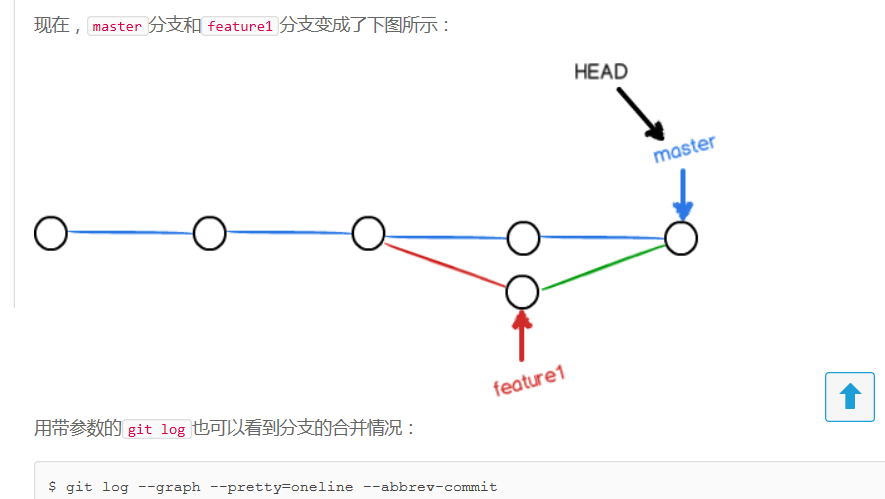
用git log --graph命令可以看到分支合并图



此时两个分支都有新的提交，Git无法执行快速合并，只能先解决冲突后进行合并。

git merge feature1 –快速合并的语法

我们修改冲突部分就能提交了。



合并的时候，先创建新的分支（git checkout –b feature1），修改东西，然后回到Master分支上(git checkout master)，然后看一下是否能快速合并(git merge feature1)，如果不能则先解决冲突 然后进行操作。解决完了以后，删除后续分支(git branch -d feature1)。

## 13分支管理策略

合并分支时，加上--no-ff参数就可以用普通模式合并，合并后的历史有分支，能看出来曾经做过合并，而fast forward合并就看不出来曾经做过合并。

3.BUG分支

4.feature分支，丢弃

如果要丢弃一个没有被合并过的分支，可以通过git branch -D <name>强行删除。

推送分支，就是把该分支上的所有本地提交推送到远程库。推送时，要指定本地分支，这样，Git就会把该分支推送到远程库对应的远程分支上：

$ git push origin master

## 14.多人协作



## 15.标签及其操作

命令git tag <name>用于新建一个标签，默认为HEAD，也可以指定一个commit id；

git tag -a <tagname> -m "blablabla..."可以指定标签信息；

git tag -s <tagname> -m "blablabla..."可以用PGP签名标签；

命令git tag可以查看所有标签。

----操作

命令git push origin <tagname>可以推送一个本地标签；

命令git push origin --tags可以推送全部未推送过的本地标签；

命令git tag -d <tagname>可以删除一个本地标签；

命令git push origin :refs/tags/<tagname>可以删除一个远程标签。

16.后续部分忽略（参考网站：https://www.liaoxuefeng.com/）

## 16.将本地项目git到远程仓库

<http://blog.csdn.net/u011043843/article/details/33336625>

步骤如下

1：创建一个文件夹并且初始化为git仓库

Mkdir testMapper

Cd testMapper

Git init

2.关联远程仓库

Git remote add origin [git@github.com:793404933/testMapper](mailto:git@github.com:793404933/testMapper)

3.将所需上传文件放入git本地仓库中

4.添加并且在本地仓库中提交

Git add MybatisMapper

Git commit –m”first commit”

5.push放入远程仓库

Git push –u origin master --第一次需要这样

后面： git push ;

# 6.Maven

## 1.概念

Maven是基于项目对象模型（POM）,可以通过一小段描述信息来管理项目的构建,报告和文档的软件项目管理工具。

mvn clean install -Dmaven.test.skip=true

## 2.mvn –v 查看maven版本

Compile 编译

Test 测试

Package 打包项目

Clean 删除target

Install 安装jar包到我们本地仓库中

## 3.archetype插件

–用于创建符合maven规定的目录骨架。

Mvn archetype:generate

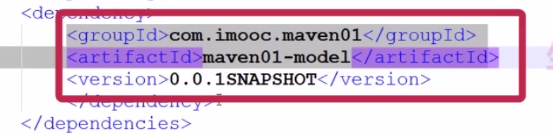
创建目录的两种方式：

1）archetype:generate 按照提示进行选择

2）archetype:generate –DgroupID=…配置自己所需要的。

## 4.maven中的坐标和仓库

1其中

. 

都可以相应组成坐标。

ArtifactID –对应 项目名-模块名

2．仓库管理项目的依赖 。

仓库分为两种：本地仓库和远程仓库。

镜像仓库

访问国内的镜像仓库。

修改镜像仓库 –打开maven –打开settings.xml –146行

可以修改为阿里云（<http://maven.aliyun.com/nexus/content/groups/public/>）

配置代码：

<mirrors>  
<mirror>  
<id>alimaven</id>  
<name>aliyun maven</name>  
<url>http://maven.aliyun.com/nexus/content/groups/public/</url>  
<mirrorOf>central</mirrorOf>  
</mirror>  
</mirrors>

修改仓库默认位置 setting.xml –55行

<localRepository>D:\work\_dir\package</localRepository>

## 5.正反斜杠的区别：

1）浏览器地址栏网址使用 斜杆/ ;

（2）windows文件浏览器上使用 反斜杠\ ;

（3）出现在html url() 属性中的路径，指定的路径是网络路径，所以必须用 斜杆/ ;

<div style="background-image:url(/Image/Control/title.jpg); background-repeat:repeat-x; padding:10px 10px 10px 10px"></div>

// 如果url后面用反斜杠，就不会显示任何背景

（4）出现在普通字符串中的路径，如果代表的是windows文件路径，则使用 斜杆/ 和 反斜杠\ 是一样的；如果代表的是网络文件路径，则必须使用 斜杆/ ;

<img src=".\Image/Control/ding.jpg" /> // 本地文件路径，/ 和 \ 是等效的

<img src="./Image\Control\cai.jpg" />

<img src="http://hiphotos.baidu.com/yuhua522/pic/item/01a949c67e1023549c163df2.jpg" /> // 网络文件路径，一定要使用 斜杆/

斜杆/ 和 反斜杠\ 的区别基本上就是这些了，下面再讨论一下相对路径和绝对路径。

./SRC/  这样写表示，当前目录中的SRC文件夹；

 ../SRC/  这样写表示，当前目录的上一层目录中SRC文件夹；

/SRC/   这样写表示，项目根目录（可以只磁盘根目录，也可以指项目根目

总体来说：正斜杠/ 他是在指定的一般是网络中的路径，而本地磁盘上的花，正反/是等效的，但是windows默认 ‘\’ 路径分隔符：/ 因此，我们可以统一使用 /

## 6.在Eclipse安装maven插件以及创建maven项目

1.配置环境（[http://www.imooc.com/video/8644 3-4](http://www.imooc.com/video/8644%203-4)）

2.完整的项目构建过程包括：

清理，编译，测试，打包，集成测试，验证，部署。

3.maven的生命周期:

Clean 清理项目 default 构建项目 site 生成项目站点。

Clean 又分为3个阶段： pre-clean 执行清理前的工作，

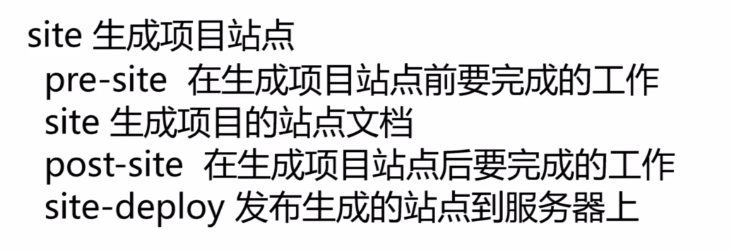
Clean 清理上一次构建生成的所有文件。‘

Post-clean 执行清理后的文件，

4.default为构建项目之中最核心的步骤。

Compile test package install 都属于default之中的。

5.site是用来生成项目站点。Pom中的信息，自动生成



--Mybatis 增删改查

## 7.pom.xml常用元素介绍

前面是pom的约束信息。

<modelVersion>固定版本 pom版本

坐标信息：

<groupId> --主项目标识,和我们实际创建的包是类似的。反写的公司网址+项目名

<artifactId>--项目名+模块名

<version> --当前项目版本号 0.0.1

--第一个0大版本号，第二个0分支版本号，第三个0表示小版本

<packaging>--默认是jar war zip,pom

<name>项目描述名

<url>项目地址

<dependencies>依赖列表,

<scope >依赖范围，test

<optional>--设置依赖是否可选。

<exclusions>排除依赖列表。--Ajar 依赖于B B依赖C A不想依赖C就能使用该标签。

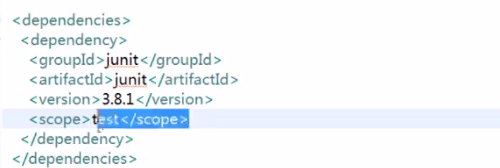
<build><plugin>…<groupId><artifactId><version>插件列表

<parent >pom继承

<modules >指定多模块

## 8.依赖

1.maven中依赖的范围



说明该依赖的范围为test中。 Runtime 编译运行都有效。

2.maven中依赖是传递的

3.排除依赖 可以将传递依赖解除 （192行来解决jre显示过低版本）

4.依赖冲突

原则：1.短路优先， A-B-C-X（jar） A-D-X优先解析第二个版本

2.先声明先优先。（路径相同的时候）

5.’利用module 聚合

<modules>

<module>../hongxing-bge</module>

<module>../hongxing-nange</module>

<module>../hongxing-shanji</module>

</modules>

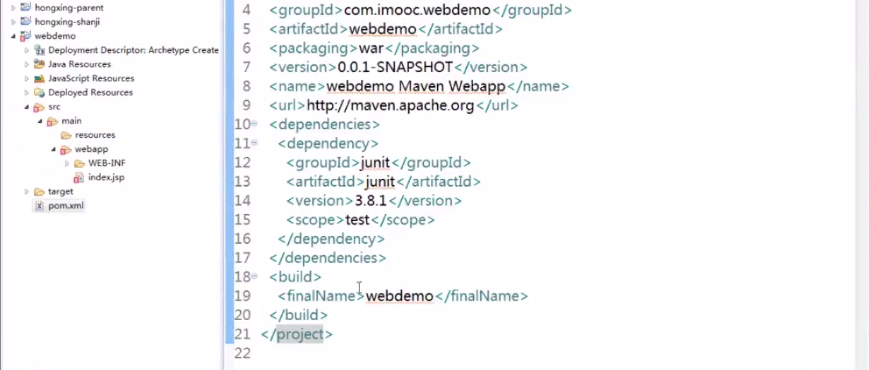
继承：



6.使用maven构建web项目

1.选择：Webapp

2.出错：

3.没有servlet的坐标配置

<scope>provided</scope>

4.属性—》project Facets 动态的web medules

5. 

发布移除test

6.设置服务器 tomcat的坐标

7.cmd 构建工程

一、基于mvn创建java工程：DbScriptGenerator  
  
  
1. 创建默认的JAVA工程:  
mvn archetype:generate  -DgroupId=DbScriptGenerate  -DartifactId=DbScriptGenerate   
  
2. 指定原型来创建WEB工程：  
mvn archetype:generate -DgroupId=packageName   -DartifactId=webappName   -DarchetypeArtifactId=maven-archetype-webapp   
-DinteractiveMode=false  
  
3. 交互式创建：   
mvn  archetype:generate 后续的命令，按提示输入  
  
4. 生成 eclipse项目：进入生成的工程家目录下执行,否则找不到pom文件  
mvn eclipse:eclipse  
  
5. 在eclipse中导入对应的工程  
上面是通过命令行，生成基本maven工程骨架

# 7.Mybatis

## 1.基本知识

1. 官网：<http://www.mybatis.org/>

2. maven 构建项目

加入 dependency 在pom.xml中即可

<dependency>

<groupId>org.mybatis</groupId>

<artifactId>mybatis</artifactId>

<version>3.4.3</version>

</dependency>

3.Mybatis取数据的大概思路：

SqlSessionFactoryBuilder (XML文件) --获得-->sqlSessionFactory—打开—>sqlSession

--通过xml中构建的sql语句取得信息—>sqlSession.selectOne(…,..);

或者

Xml的namespace 为Mapper名字 –>通过sqlSession.getMapper(name);获取Mapper

--使用其接口的方法名称—>name.findById(2);

UserMapper userMapper = sqlSession.getMapper(UserMapper.**class**);

List<Users> list = userMapper.findAll();

4.XML文件配置

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!DOCTYPE configuration

PUBLIC "-//mybatis.org//DTD Config 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-config.dtd">

<configuration>

<!-- 和spring整合后environments配置将废除 -->

<environments default=*"development"*>

<environment id=*"development"*>

<!-- 使用JDBC事务管理 -->

<transactionManager type=*"JDBC"*/>

<!-- 数据库连接池 -->

<dataSource type=*"POOLED"*>

<property name=*"driver"* value=*"oracle.jdbc.driver.OracleDriver"*/>

<property name=*"url"* value=*"jdbc:oracle:thin:@192.168.159.130:1521:mas"*/>

<property name=*"username"* value=*"hec3dev"*/>

<property name=*"password"* value=*"hec3dev"*/>

</dataSource>

</environment>

</environments>

<mappers>

<!-- 注册userMapper.xml文件，

userMapper.xml位于com.iflytek.mapper这个包下，所以resource写成

sqlmapper/user.xml -->

<mapper resource=*"sqlmapper/Users.xml"*/>

</mappers>

</configuration>

5.构建sqlSessionFactory以及构建sqlSession (利用XML的构建)

String resource = “SqlMapConfig.xml”;

InputStream inputStream = Resources.getResourceAsStream(resource);

SqlSessionFactory sqlSessionFactory = new SqlSessionFactoryBuilder().build(inputStream);

SqlSession sqlSession = sqlSessionFactory.openSession();

6.映射的SQL语句

XML 文件：

<?xml version="1.0" encoding="UTF-8" ?>

<!DOCTYPE mapper

PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"

"http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="org.mybatis.example.BlogMapper">

<select id="selectBlog" resultType="Blog">

select \* from Blog where id = #{id}

</select>

</mapper>

调用XML文件中的sql语句

Blog blog = (Blog) session.selectOne("org.mybatis.example.BlogMapper.selectBlog", 101);

Java命名空间式映射SQL

package org.mybatis.example;

public interface BlogMapper {

@Select("SELECT \* FROM blog WHERE id = #{id}")

Blog selectBlog(int id);

}

调用JAVA文件中的SQL语句

BlogMapper mapper = session.getMapper(BlogMapper.class);

Blog blog = mapper.selectBlog(101);

7.存取数据的模板

SqlSession session = sqlSessionFactory.openSession();

try {

// do work

} finally {

session.close();

}

8.resultMapper

<resultMap id="userResultMap" type="User">

<id property="id" column="user\_id" />

<result property="username" column="user\_name"/>

<result property="password" column="hashed\_password"/>

</resultMap>

## 2.XML中配置SQL语句

1.select --id 唯一标识

parameterType 接受一个 int（或 Integer）类型的参数

resultType 返回HashMap 类型的对象

<select id="selectPerson" parameterType="int" resultType="hashmap">

SELECT \* FROM PERSON WHERE ID = #{id}

</select>

Select 参数列表：



2.insert ,update 和delete(<http://www.mybatis.org/mybatis-3/zh/sqlmap-xml.html>)

如果你的数据库支持自动生成主键的字段（比如 MySQL 和 SQL Server），那么你可以设置 useGeneratedKeys=”true”，然后再把 keyProperty 设置到目标属性上就OK了。例如，如果上面的 Author 表已经对 id 使用了自动生成的列类型，那么语句可以修改为:

<insert id="insertAuthor" useGeneratedKeys="true"

keyProperty="id">

insert into Author (username,password,email,bio)

values (#{username},#{password},#{email},#{bio})

</insert>

2)三种SQL映射的代码

<insert id="insertAuthor">

insert into Author (id,username,password,email,bio)

values (#{id},#{username},#{password},#{email},#{bio})

</insert>

<update id="updateAuthor">

update Author set

username = #{username},

password = #{password},

email = #{email},

bio = #{bio}

where id = #{id}

</update>

<delete id="deleteAuthor">

delete from Author where id = #{id}

</delete>

3.动态 sql

1) if

<select id="findActiveBlogWithTitleLike"

resultType="Blog">

SELECT \* FROM BLOG

WHERE state = ‘ACTIVE’

<if test="title != null">

AND title like #{title}

</if>

</select>

2)choose when otherwise

<select id="findActiveBlogLike"

resultType="Blog">

SELECT \* FROM BLOG WHERE state = ‘ACTIVE’

<choose>

<when test="title != null">

AND title like #{title}

</when>

<when test="author != null and author.name != null">

AND author\_name like #{author.name}

</when>

<otherwise>

AND featured = 1

</otherwise>

</choose>

</select>

3)Trim where set

<trim prefix="WHERE" prefixOverrides="AND |OR ">

...

</trim>

解决where and condition where ;直接忽略 and

Set 类似 解决 ，

4）foreach

动态 SQL 的另外一个常用的必要操作是需要对一个集合进行遍历，通常是在构建 IN 条件语句的时候。比如：

<select id="selectPostIn" resultType="domain.blog.Post">

SELECT \*

FROM POST P

WHERE ID in

<foreach item="item" index="index" collection="list"

open="(" separator="," close=")">

#{item}

</foreach>

</select>

此元素可用于定义可以包含在其他语句中的SQL代码的可重用片段。它可以是静态的（在负载阶段）参数化。不同的属性值可以在包含实例中变化。例如：

相当于可以重复利用。<include refid=”userColumns”>…

<sql id = “userColumns” > $ {alias} .id，$ {alias} .username，$ {alias} .password </ sql>

后续可以运用：

<select id="selectUsers" resultType="map">

select

<include refid="userColumns"><property name="alias" value="t1"/></include>,

<include refid="userColumns"><property name="alias" value="t2"/></include>

from some\_table t1

cross join some\_table t2

</select>

# 8.spring

## 1.spring 前戏

1.spring官网：<http://projects.spring.io/spring-framework/>

2.maven配置（部分）

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>4.3.10.RELEASE</version>

</dependency>

3.maven依赖包

，(配置spring+mybatis+maven 将下方文件改名为pom就可以了 )



特别注明：该pom.xml中ojdbc需要下载到本地，并且自己将其导入maven的库中。

## 2.事务管理

事务就是对一系列的数据库操作（比如插入多条数据）进行统一的提交或回滚操作，如果插入成功，那么一起成功，如果中间有一条出现异常，那么回滚之前的所有操作。这样可以防止出现脏数据，防止数据库数据出现问题。

所以我们通过spring事务管理可以减少事务之间异常的发生，并且防止数据库数据出现问题。在spring中不需要自动提交，该事务管理会帮你自动提交。

<bean id="transactionManager"

class="org.springframework.jdbc.datasource.DataSourceTransactionManager">

<property name="dataSource" ref="dataSource"/>

</bean>

</beans>

## 3.两种配置方式

1.XML配置

<!-- 采用注解的方式配置bean 注解开启 -->

<context:annotation-config />

<!-- 配置要扫描的包 -->

<!-- <context:component-scan base-package="com.lgh.\*"></context:component-scan> -->

配置扫描包。

Test的时候

引入使用：

@RunWith(SpringJUnit4ClassRunner.**class**)

@ContextConfiguration(locations = {"classpath:SpringConfig.xml"})

这两个注解

2．JavaConfig配置

@Configuration

@ImportResource("classpath:SpringConfig.xml")

@ComponentScan(basePackageClasses={DaoConfig.**class**,PojoConfig.**class**})

**public** **class** ConfigAll {

}

@Configuration 声明该类是配置作用的Bean

由于我使用了两种混合配置，所以需要引入XML文件

@ComponentScan 是启动组件扫描，来包含其base包。

大概的意思就是：你在本包中，添加一个Config接口，然后引入为base包，则在该包下的所有类都能被扫描。

在Test的时候

引入使用：

@RunWith(SpringJUnit4ClassRunner.**class**)

@ContextConfiguration(classes=ConfigAll.**class**)

这2个注解

为什么需要引入这两个注解呢？

因为，在原本基础上，test类并不属于Spring容器管理，因此spring容器根本不了解该类的任何信息，这种写法就是为了让测试在Spring的容器环境下执行。

第二个注解则是，配置上下文为该ConfigAll类。

## 4.应用上下文

包含：java配置类应用上下文和xml配置文件应用上下文。（其他不考虑）

## 5.混合配置

1）在java配置类中引入XML配置

@ImportResource(“classpath:cd-config.xml”)

2）在XML配置中引入java配置类

<bean class=”soundsystem.CDconfig”>

## 6. springmvc控制登录用户session失效处理

1.准备解决的方案：

1）拦截器

2）Spring AOP (既然学习Spring ,那就使用AOP)

2.AOP切面大部分的作用：是为了横向的获取系统运行时的日志文件，但是在这种登录权限控制时，在跳转页面的时候，也是可以使用AOP处理的。在每次跳转的时候，就判断该session是否存在，如果存在，让其不做任何操作，如果不存在，则可以说明该用户Session失效或者用户未登录，因此我们可以让其跳转到相应的登陆界面。

3.切面有五种类型的通知：

前置通知 Before

后置通知：After

返回通知：After-returning

异常通知：After-throwing

环绕通知：Around

既然我们需要每时每刻的需要判断跳转前后Session是否有值，所以我们使用环绕通知。

4.切点和连接点的区别和联系

连接点是各个能够插入切面的点（而不是已经插入切面），由于连接点可能是函数，异常等等，因此连接点是说的一个大范围

而切点，则是根据功能而使部分所需连接点插入切面，则这部分连接点也被称作切点，是一个小的范围。因此 连接点包含切点

5.spring只支持方法级别的连接点，

这个非常容易理解，就是连接点最小是一个方法。

6.重要的2个AspectJ指示器

1)execution()

用于匹配是连接点的执行方法：大概的意思就是符合其括号的，就会变成切点

2）within()

限制连接点匹配制定的类型：限制连接点所在的包，类等

7.几个excecution表达式和within表达式：

execution(public \* \* (..))//匹配所有public方法

execution(\* set\*(..))//匹配以set开始的方法

execution(\* com.abc.service.AdviceManager.\* (..))//匹配AdviceManager中任意方法

execution(\* com.abc.service.\*.\* (..))//匹配com.abc.servcie包中任意类的任意方法

within：限定匹配特定类型的连接点，当使用Spring AOP的时候，只能匹配方法执行的连接点。下面是几个例子：

within(com.abc.service.\*)//匹配com.abc.service包中的任意连接点

within(com.abc.service..\*)//匹配com.abc.service包或子包中任意的连接点

8.注意事项：

Spring MVC 和 Spring 整合的时候，SpringMVC的springmvc.xml文件中 配置扫描包，不要包含 service的注解，Spring的applicationContext.xml文件中 配置扫描包时，不要包含controller的注解，如下所示：  
SpringMVC的xml配置：

<context:component-scan base-package="com.insigma">

  <context:exclude-filter type="annotation" expression="org.springframework.stereotype.Service"/>

</context:component-scan>

Spring MVC启动时的配置文件，包含组件扫描、url映射以及设置freemarker参数，让spring不扫描带有@Service注解的类。为什么要这样设置？因为springmvc.xml与applicationContext.xml不是同时加载，如果不进行这样的设置，那么，spring就会将所有带@Service注解的类都扫描到容器中，等到加载applicationContext.xml的时候，会因为容器已经存在Service类，使得cglib将不对Service进行代理，直接导致的结果就是在applicationContext 中的事务配置不起作用，发生异常时，无法对数据进行回滚。以上就是原因所在。  
同样的在Spring的xml配置如下：

<context:component-scan base-package="com.insigma">

 <context:exclude-filter type="annotation" expression="org.springframework.stereotype.Controller"/>

</context:component-scan>

扫描包路径，不扫描带有@Controller注解的类。因为这些类已经随容器启动时，在springmvc.xml中扫描过一遍了。  
完成以上工作

7.spring 注解

在我们自动装配某个对象的时候，要考虑到，构造这个对象的时候，需不需要其他对象，如果需要的话，我们也应该将其他对象放入Spring容器之中，这样spring就能管理对象，自动将对象注入到我们所需要的对象之中。（？？）

# 9.rocket MQ

## 1.概述

<http://www.jianshu.com/p/453c6e7ff81c> 该网站的讲述很不错。

1.分布式消息系统作为实现分布式系统可扩展/可伸缩性的关键组件,需要具有高吞吐量，高可用性等特点，而谈到消息系统的设计。

2.可是谈到消息系统的设计会存在两个非常难以避免并且非常重要的问题：

1）消息的顺序问题

2）消息的重复问题

## 2.关键特性以及其实现原理

（1）.第一大问题的解决：（消息的顺序问题）

1. RocketMQ通过轮询所有队列的方式来确定消息被发送到哪一个队列（负载均衡策略）。

2.为了要实现严格的顺序消息，最简单且可行的方法就是

保证生产者 - MQServer - 消费者是一对一对一的关系

3.虽然上述设计简单可行，但是存在的问题也很大，吞吐量会不够，异常处理变多

阿里的解决方式：通过合理设计和问题分解 –1，不关注乱序 2.队列乱序不代表消息乱序。

3.RocketMQ通过MessageQueueSelector中实现的算法来确定消息在哪个队列之中

而两种默认的MessageQueueSelector实现为: 随机和Hash

可以根据业务实现自己的MessageQueueSelector来决定消息按照何种策略发送到消息队列中

SendResult sendResult =producer.send(msg,new MessageQueueSelector(){

@Override

Public MessageQueue select (List<MessageQueue> mqs,Message msg,Object arg){

Integer id=(Integer)arg;

Int index =id % mqs.size();

Return mqs.get(index);

}

} ,orderId);

//匿名类并且重写select方法获取消息队列中的所取得Id

3.在获取到路由信息以后，通过MessageQueueSeletor 实现的算法来选择一个队列（同一OrderId获取的一定在同一个队列）

Private SendResult send(){

//获取路由信息

TopicPublishInfo topicPublishInfo=this.tryToFindTopicPublishInfo(msg

.getTopic());

If(topicPublishInfo != null && topicPublishInfo.ok()){

MessageQueue mq=null;

//根据算法，选择一个队列

//这里arg=ordered

mq = selector.select(topicPublishInfo.getMessageQueueList(), msg, arg);

if(mq!=null){

return this.sendKernellImpl(msg,mq,communicationMode,sendCallback,timeout);

}

}

}

（2）第二大问题的解决：（消息重复）

4.造成消息重复的根本原因是：网络不发达。只要通过网络交换数据，就无法避免这个问题。

所以解决这个问题的办法就是绕过这个问题。那么问题就变成了：如果消费端收到两条或者多条一样的信息，应该怎么处理。

1.消息重复的处理

1）消息端处理消息的业务逻辑保持等幂性:（这个的理解是，如果两条相同的数据，务必对它们操作是相同的 不属于消息系统的范围）

2）保证每条消息都有唯一编号且保证消息处理成功与去重表的日志同时出现。

（由于正常情况下重复消息的概率本来就低，可以在消息系统和业务层面实现。如果直接在消息系统中来实现的话，会对消息系统的吞吐量和高可用性有大小不一的影响，在后续的工作中可以在业务中实现去重，这样能使得消息系统不受影响并且业务增加的只是轻量级的东西。）

5. **RocketMQ不保证消息不重复，如果你的业务需要保证严格的不重复消息，需要你自己在业务端去重。**

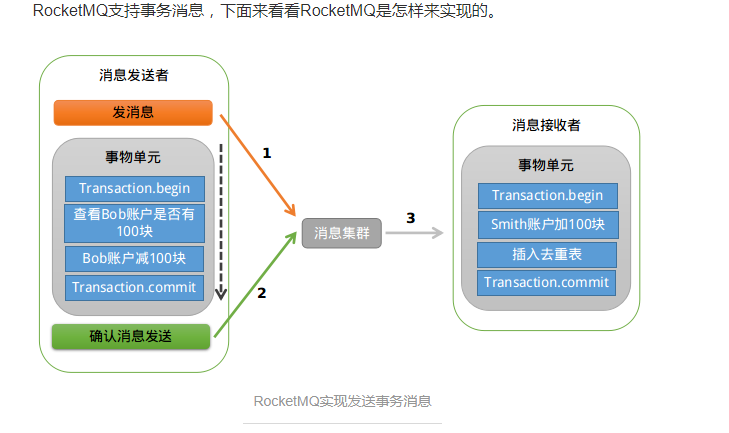
（3）事务消息

1. RocketMQ除了支持普通消息，顺序消息，另外还支持事务消息。

2.当用户增长到一定程度，业务双方信息不在同一台服务器中，查找需要少许时间。而如果看整个业务过程的话一台服务器整个业务过程的2倍时间，显然接受不了，所以来规避这个问题是利用了:

大事务 = 小事务+ 异步(在2个服务器上分别需要做的事情)

但是同时在解决这个问题的时候,产生另外一个问题，数据不一致。因此MQ利用了下述原理解决：



解释：

RocketMQ第一阶段发送Prepared消息时，会拿到消息的地址，第二阶段执行本地事物，第三阶段通过第一阶段拿到的地址去访问消息，并修改消息的状态。

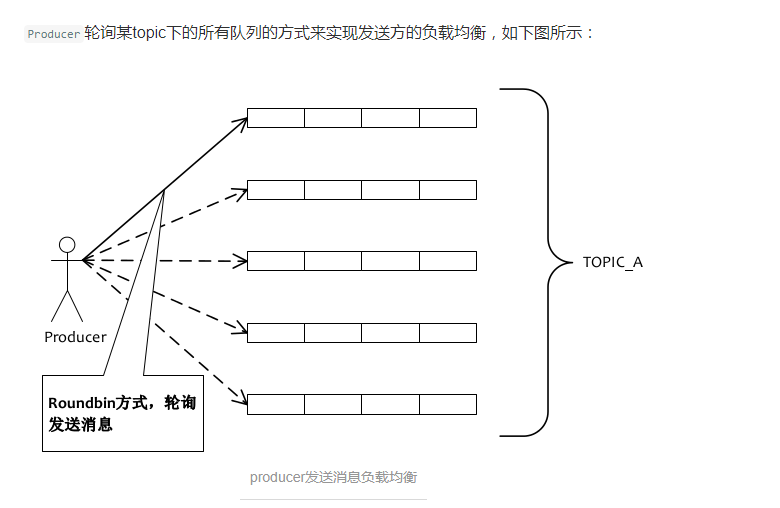
细心的你可能又发现问题了，如果确认消息发送失败了怎么办？RocketMQ会定期扫描消息集群中的事物消息，如果发现了Prepared消息，它会向消息发送端(生产者)确认，Bob的钱到底是减了还是没减呢？如果减了是回滚还是继续发送确认消息呢？RocketMQ会根据发送端设置的策略来决定是回滚还是继续发送确认消息。这样就保证了消息发送与本地事务同时成功或同时失败。

3.事务消息的处理过程



（4）Producer如何发送消息

1.发送消息图



2.在生命周期中，生产者调用一次初始化方法，（1.如果没有指定namesrv地址，将会自动寻址 2.启动定时任务：更新namesrv地址、从namsrv更新topic路由信息、清理已经挂掉的broker、向所有broker发送心跳...3.启动负载均衡的服务），初始化后开始发送消息

private SendResult sendDefaultImpl(Message msg,......) {

// 检查Producer的状态是否是RUNNING

this.makeSureStateOK();

// 检查msg是否合法：是否为null、topic,body是否为空、body是否超长

Validators.checkMessage(msg, this.defaultMQProducer);

// 获取topic路由信息

TopicPublishInfo topicPublishInfo = this.tryToFindTopicPublishInfo(msg.getTopic());

// 从路由信息中选择一个消息队列

MessageQueue mq = topicPublishInfo.selectOneMessageQueue(lastBrokerName);

// 将消息发送到该队列上去

sendResult = this.sendKernelImpl(msg, mq, communicationMode, sendCallback, timeout);

}

代码中需要关注的两个方法tryToFindTopicPublishInfo和selectOneMessageQueue。前面说过在producer初始化时，会启动定时任务获取路由信息并更新到本地缓存，所以tryToFindTopicPublishInfo会首先从缓存中获取topic路由信息，如果没有获取到，则会自己去namesrv获取路由信息。selectOneMessageQueue方法通过轮询的方式，返回一个队列，以达到负载均衡的目的。

3.如果Producer发送消息失败，会自动重试，重试的策略：

1.重试次数 < retryTimesWhenSendFailed（可配置）

2.总的耗时（包含重试n次的耗时） < sendMsgTimeout（发送消息时传入的参数）

3.同时满足上面两个条件后，Producer会选择另外一个队列发送消息

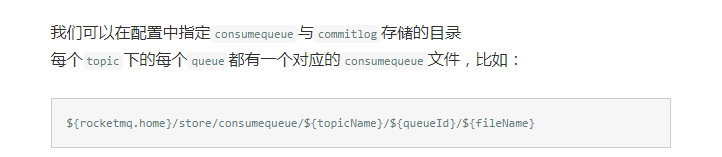
代码：略；

（5）消息存储 \*

1.RocketMQ的消息存储是由 consume queue 和 commit log 配合完成的

2. Consume Queue

consume queue是消息的逻辑队列，相当于字典的目录，用来指定消息在物理文件commit log上的位置。

3．

4.Commit Log

CommitLog：消息存放的物理文件，每台broker上的commitlog被本机所有的queue共享，不做任何区分。

(6)消息订阅

1.RocketMQ消息订阅有两种模式，一种是Push模式，即MQServer主动向消费端推送；另外一种是Pull模式，即消费端在需要时，主动到MQServer拉取。但在具体实现时，Push和Pull模式都是采用消费端主动拉取的方式。

2.后续太过抽象，后面再工程中遇到再解决。

# 10.Dubbo

## 1.Dubbo概念

1. Dubbo |ˈdʌbəʊ| 是一个分布式服务框架，致力于提供高性能和透明化的RPC远程服务调用方案，以及SOA服务治理方案。

2,由于Dubbo是一种分布式服务框架，所以说，解决什么是分布式服务框架等问题，很有必要。

1. 什么是java反射？(下述是标准解释及用法)

[**反射基本原理**](http://www.cnblogs.com/chalk/p/4709001.html)

**反射的概念：**  
   **在Java中的反射机制是指在运行状态中，对于任意一个类，都能够知道这个类的所有属性和方法；**  
**对于任意一个类，都能调用它的任意一个方法；**  
**这种动态获取信息以及动态调用对象的功能成为Java语言的反射机制。**

**（在类运行的时候，我们是可以知道（不是获取？）到这个类的所有属性和方法）**  
  
  
**我们写的类在载入内存的时候jvm虚拟机是怎么识别到底哪个是变量，哪个是方法，哪个是修饰符的呢？**  
          因为java中的Class也是一个对象，那么有对象就有对应的类,当在硬盘中的时候是一个文件，当载入内存的时候就可以看成一个对象。这个Class对象的成员变量就是        Field、Constructor、Method、Modifier,值就是变量名、修饰符、方法名。类里面的field、constructor、method、modifier,在java.lang.reflect中都有这些方法、变量等对应的类，其中有类Field、类Constructor、类Method、类Modifier，有对象就有对应的类。例如，我们写了一个动物类，成员变量有毛色、体重，毛色=红色、体重=70，毛色的值是红色，体重的值是70，这个很好理解。那么，在jvm虚拟机中，java.lang.Class就相当于这个动物类，成员变量就是Field、Constructor、Method、Modifier，值就是变量名、修饰符、方法名。因此我们就可以通过Class对象去获得成员变量的值，也就是我们想要得到的变量名、修饰符列表、方法名等等，这就是反射的基本原理。  
（由于java中Class也是对象，他有成员变量为Field Constructor Method Modifier ,因此我们可以通过获取Class对象属性的值来获取我们所得到的变量，方法等 ---反射）  
  
**Java反射机制能够实现的功能:**  
    **1 在运行时判断任意一个对象所属的类**  
**2 在运行时构造任意一个类的对象**  
**3 在运行时判断任意一个类所具有的方法和属性**  
**4 在运行时调用任意一个对象的方法**  
**5 生成动态代理**  
在动态代理中应用的反射最多，可也说有动态代理必有反射。  
  
  
**获取Class类对象有三种方法，这个对象是java.lang.Class类的，**  
    **1 使用Class类的forName()静态方法**  
**Class c1 = Class.forName("Employee");//"类的全路径"**  
  
  
**2 调用某个类的class属性来获取该类对应的Class对象**  
**每个类都默认具备Class属性，这个属性保存的是它对应那个Class对象的内存地址**  
**Class c2 = Employee.class;**  
  
**3 调用某个对象的getClass()方法**  
**java中任何一个对象都有getClass()方法，e是运行时类**  
**Employee e = new Employee();**  
**Class c3 = e.getClass();**

**（java反射只能对java类进行获取其属性，并不能对其实例化后的对象进行获取，所以我们没必要考虑安全性方面问题）**

2）什么是分布式服务框架，它与传统服务框架有什么区别。

简单来说，就是同一个服务，把数据库的不同部分分开建立到不同的服务器上，以缓解数据库大量数据访问的压力。可以提高处理响应速度。传统的服务框架是一台服务器处理所有数据。

3）分布式服务和微服务联系和区别，微服务的缺点和优点

分布式只是把东西分散部署，是一个比较笼统的概念。微服务一般都是分布式的，而微服务更强调敏捷和健壮，他的边界应该更加清晰，包括你部署的程序和数据都应该独立，这里可能借助docker。

优点：

业务解耦，方便扩容，方便系统按模块升级，模块重用，开发新业务简单，开发人员可以专注某一业务，方便数据库优化

微服务的**坏处**:(分布式服务框架要解决的问题)  
每个系统之间的关系变得非常复杂  
随着调用的业务增多,底层的模块需要高可用性和并发  
需要分布式Session框架支持  
分层后增加测试复杂度

2.Dubbo分布式服务框架结构

4）RPC远程服务是什么？

RPC（Remote Procedure Call Protocol，远程过程调用协议）

5）SOA是什么？

面向服务的架构（SOA）是一个组件模型，它将应用程序的不同功能单元（称为服务）通过这些服务之间定义良好的接口和契约联系起来。接口是采用中立的方式进行定义的，它应该独立于实现服务的硬件平台、操作系统和编程语言。这使得构建在各种各样的系统中的服务可以以一种统一和通用的方式进行交互。

SOA的过程中牢记以下特征：

可从企业外部访问

随时可用

粗粒度的服务接口分级

松散耦合

可重用的服务

服务接口设计管理

标准化的服务接口

支持各种消息模式

精确定义的服务契约

6）Dubbo实现其高性能RPC远程调用方案和SOA服务治理方案，其核心部分为？

（1）远程通讯: 提供对多种基于长连接的NIO框架抽象封装，包括多种线程模型，序列化，以及“请求-响应”模式的信息交换方式。

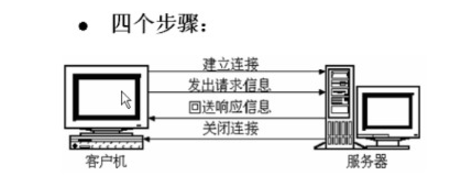
（2）集群容错: 提供基于接口方法的透明远程过程调用，包括多协议支持，以及软负载均衡，失败容错，地址路由，动态配置等集群支持

（3）自动发现: 基于注册中心目录服务，使服务消费方能动态的查找服务提供方，使地址透明，使服务提供方可以平滑增加或减少机器。

7）NIO,请求-响应模式，负载均衡，地址路由名词意思

NIO：java.nio全称java non-blocking IO，是指jdk1.4 及以上版本里提供的新api（New IO） ，为所有的原始类型（boolean类型除外）提供[缓存](https://baike.baidu.com/item/%E7%BC%93%E5%AD%98)支持的数据容器，使用它可以提供非阻塞式的高伸缩网络性。

请求-响应模式：

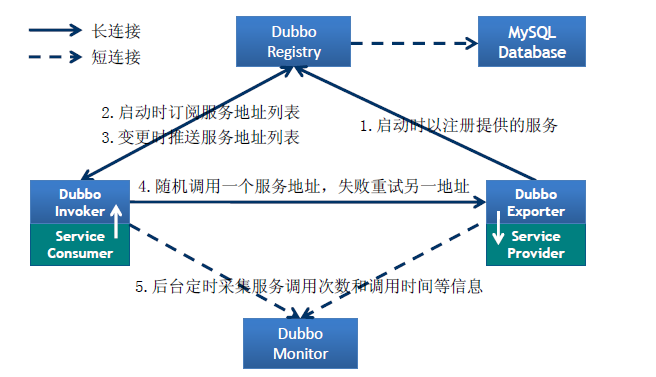


它描述的是HTTP1.0中的请求/响应过程。我们分析一下，整个过程中花费的时间包括：建立TCP三次握手的时间、客户端发送请求的时间、服务器返回响应的时间。

负载均衡：大概的意思就是分摊到多个操作单元上进行执行，例如Web[服务器](https://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8)、[FTP服务器](https://baike.baidu.com/item/FTP%E6%9C%8D%E5%8A%A1%E5%99%A8)、[企业](https://baike.baidu.com/item/%E4%BC%81%E4%B8%9A)关键应用服务器和其它关键任务服务器等，从而共同完成工作任务。分担操作。就像百度有多台服务器一样。

地址路由：暂无。我的理解：地址路由是指将地址分组从源到目的地时，决定端到端路径的网络范围的进程。

## 2.Dubbo框架图



1. 系统角色

Privoder:暴露服务的服务提供方

Consumer:调用远程服务的服务消费方

Registry:服务注册与发现的注册中心

Monitor: 统计服务的调用次数和调用时间的监控中心

Container: 服务运行容器

1. 调用关系：

服务容器负责启动，加载，运行服务提供者。

服务提供者 在启动时，向 注册中心 注册自己提供的服务。

服务提供者 在启动时，向 注册中心 订阅自己所需的服务。

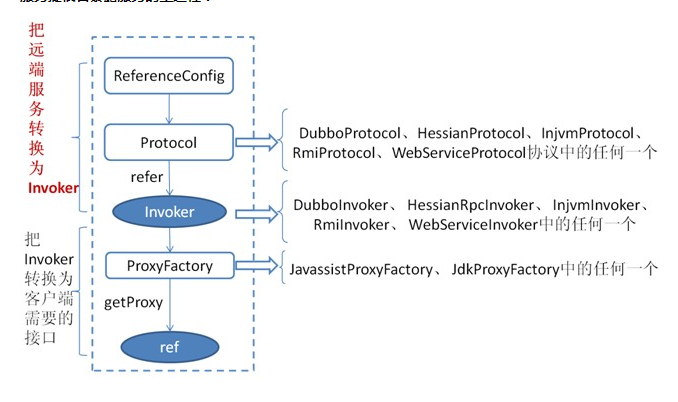
注册中心 返回 服务提供者 地址列表给 消费者 ，如果有变更，注册中心 将基于长连接推送变更数据给 消费者。

服务消费者 ，从提供的地址列表中，基于 软负载均衡 算法 ，选择一台 提供者 进行调用，如果调用失败 ，再选另一台调用

服务者消费者和提供者，在内存中累计调用的次数和调用的时间，定时每分钟发送一次统计数据到监控中心。

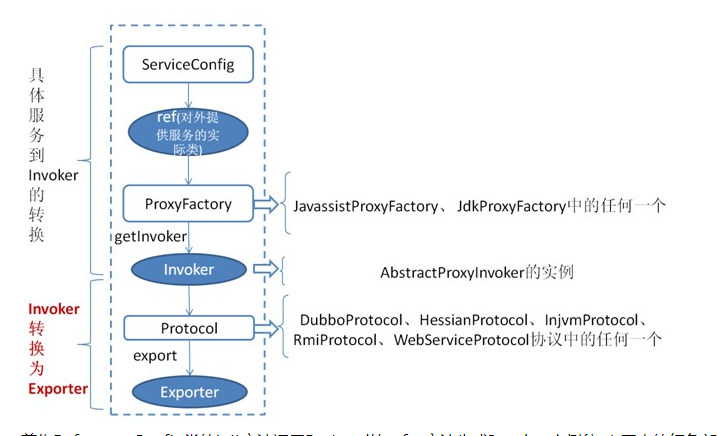
注：软负载均衡算法：软件件负载均衡的算法，平均分配（轮询），加权轮询；ip hash；fair（最小值负载均衡）。

## 3. 服务暴露和消费的详细过程



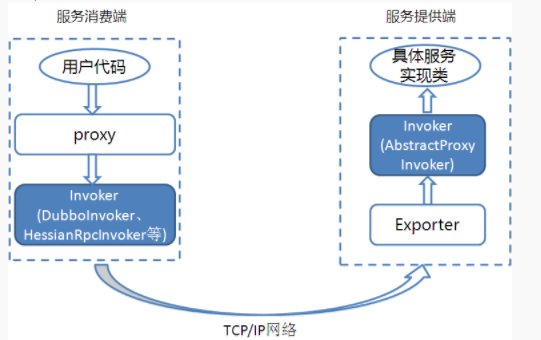


## 4.消费者消费详细过程



## 5.详解Invoker

1)由于Invoker是Dubbo领域模型中非常重要的一个概念,很多思路都是向它靠拢的。



在服务提供端：具体的服务实现类会被封装成为一个AbstractProxyInvoker实例，并新生成一个Exporter 实例。

这样当我们调用服务消费的时候，用户代码会将proxy（代理服务器）调用其对应的DubboInvoker、 HessianRpcInvoker、 InjvmInvoker、 RmiInvoker、 WebServiceInvoker中的任何一个),而该Invoker通过网络通讯层的传输，会找到对应的Exporter实例，并调用它对应的AbstractProxyInvoker 实例。从而真正的调用了服务提供者的代码，实现了远程服务调用。

# 11.Redis

http://www.runoob.com/redis/redis-hashes.html

## 1.Redis 简介

1.Redis是完全开源免费的，遵守BSD协议（可修改源代码），是一个高性能的key-value数据库。

2.Redis与其它key-value缓存产品有以下三个特点：

* Redis支持数据的持久化，可以将内存中的数据保存在磁盘中，重启的时候可以再次加载进行使用。

（可以利用这个来重新改造下载器）

* Redis不仅仅支持简单的key-value类型的数据，同时还提供list，set，zset，hash等数据结构的存储。
* Redis支持数据的备份，即master-slave模式的数据备份

3.Redis优势：

1）性能高 Redis能读的速度是110000次/s,写的速度是81000次/s 。

2）丰富的数据类型

3）原子

4）丰富的特性

## 2.Redis数据类型

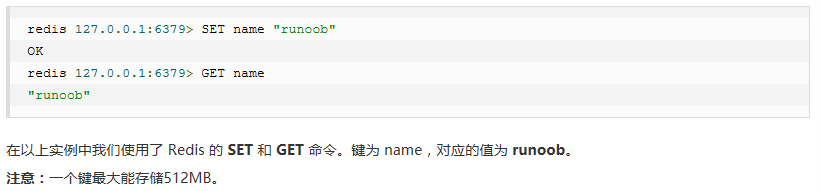
1.Redis支持五种数据类型：String (字符串) hash(哈希) list(列表) set(集合)及 zset(sorted set: 有序集合)。

2.String （set name “ ”）

string是redis最基本的类型，你可以理解成与Memcached一模一样的类型，一个key对应一个value。

string类型是二进制安全的。意思是redis的string可以包含任何数据。比如jpg图片或者序列化的对象 。

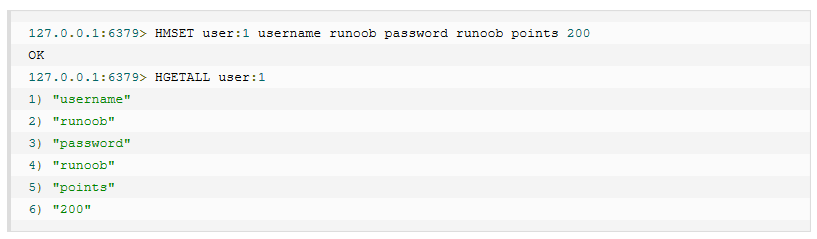
string类型是Redis最基本的数据类型，一个键最大能存储512MB。



3.hash(‘HMSET”) 他是能相同的

Redis hash 是一个键名对集合。

Redis hash是一个string类型的field和value的映射表，hash特别适合用于存储对象。



4list（lpush key member ）



5,set集合(sad key member)



## 3.Redis命令

1. Redis 命令用于在 redis 服务上执行操作。要在 redis 服务上执行命令需要一个 redis 客户端。Redis 客户端在我们之前下载的的 redis 的安装包中。

2.Redis客户端的基本语法：

Redis-cli

例子：

$redis-cli （该命令会连接本地的redis服务）



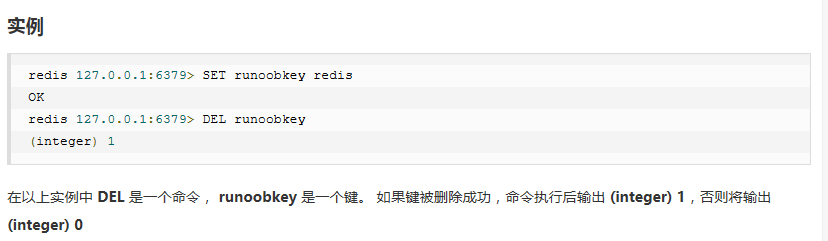
3.在远程服务上执行命令：

语法：$redis-cli –h host –p –port –a password

4.Redis键（key）

Redis键命令是管理redis键的。

语法如下：



5.键命令表



## 4.数据类型命令

1．字符串（String）：

命令（部分）：SET key value 设置制定key的值

GET key 获取制定key的值

GETRANGE key start end 返回key中的字符串值的子字符

STRLEN key 返回key所存储的字符串值得长度。

…

2.哈希（hash）

命令（部分）：HDEL key field1[fiedld2] 删除一个或多个哈希表字段。





3.列表（List）





4.set(集合)

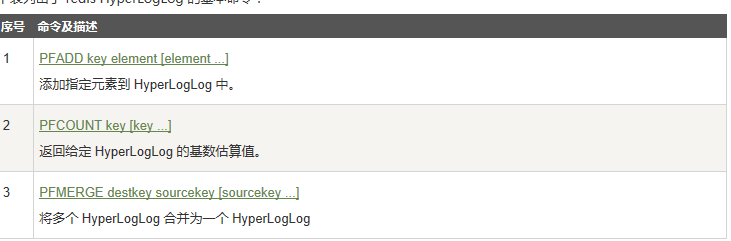


·5.Redis有序集合命令（略）

6.Redis HyperLogLog

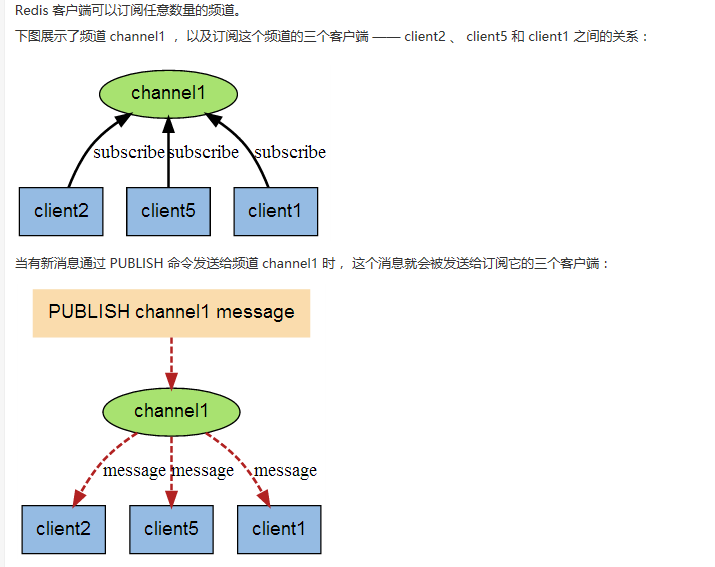
计算基数.。

比如数据集 {1, 3, 5, 7, 5, 7, 8}， 那么这个数据集的基数集为 {1, 3, 5 ,7, 8}, 基数(不重复元素)为5。 基数估计就是在误差可接受的范围内，快速计算基数。

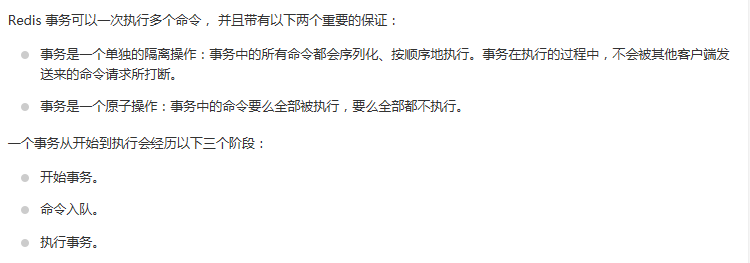


## 5.Redis发布订阅

Redis发布订阅（pub/sub）是一种消息通信模式：发送者(pub)发送消息，订阅者(sub)接收消息。



## 6.Redis事务



# 12.大概的开发流程

## 1.流程

1.通过设置两部分Mybatis generator 文件，让es表部分和表转化为 3部分 dto（实体类）,dao,mapper

由于还有业务表和ES业务表，还有接口表，所以说有9部分自动生成的。

2.执行generator文件

java -jar mybatis-generator-core-1.3.2.jar -configfile generatorConfig.xml -overwrite

然后根据PLSQL包中逻辑转化为java代码逻辑。

## 2.错误及其解决

1）业务表中主键和关联ES表中外键不相等。

出现原因：两边都加了

<selectKey keyProperty="purOrgEsId" resultType="Long" order="BEFORE">

select pur\_organizations\_es\_s.nextval from dual

</selectKey>

解决方式：在两个.xml文件中删除这句话，这句话大概的意思就是，在插入之前，设置主键+1,但是我们是不需要这部分的。因为我们在java代码中就已经取得了主键ID；

2）获取ESValue为空，而插入失败

（1）出现原因：cache缓存不是完整的，查不到ES数据，导致，明明是更新操作，直接进入了插入操作，使得和很多约束冲突，就会导致插入不进来

解决方式（步骤）：1.在cache.xml设置 true,重新加载一遍缓存 2.在cache-data.xml中entireLoad="false"，改成false意思是：不是完整加载，如果未找到就会去数据库中找。

3）出现cache中未找到该Es类，

出现原因：在cache-data中未配置该ES类。

解决方式：只需在cache-data中配置该ES类，并且重新加载就行了。

4）在测试的时候不能装配实现类。

出现原因：和错误显示的一样，就是没有装配实现类。

解决方式：在META-INF/spring目录下的application.xml文件下配置：

<bean class="com.goinglink.middleware.srm.itf.imp.org.PurOrganization"

scope="prototype">

<property name="codingRuleFactory" ref="codingRuleFactory"></property>

</bean>

5）序列化无法生成错误

出现原因：

解决方式：

6）xml中的常见错误：

（1）.es\_id错误

出现原因（解决）在对应的时候，是externalSystemId在insert和update中修改成externalSystemId就可以了。

（2）data\_version,last\_update\_version缺少，因为这两个字段是需要自己添加的。

解决：自己添加这两个字段。

1. 主键在批次号表中重复，

出现原因：主键重复

解决方式：由于是在测试的时候，我们可以在数据库表中删除该批次号记录，继续进行测试。

7）既不进行insert又不进行update而且不报错

出现原因：大概就是接口数据版本低于业务表ES中数据版本，不进行更新。

解决方式：将测试数据数据版本设高一些。

8）接口表字段和业务表字段相同，但是没有对应进入表中

出现原因：没有设置字段相同对应

解决方式： BeanConvertor.convert(itfFndUomClasses, uocValue);// 对应字段自动set

9）类型对应不上等错误就不列举了。

10）

# 13.大数据日知录—框架与算法

## 1.数据分片与路由

1.数据量衡量的单位，从小到大依次为：KB,MB,GB,TB,PB,EB和ZB，换算均为1024

2.NoSQL数据库

3.一致性哈希路由算法

1）先判断 c<j<=s，如果为真结束查找，说明key如果存在，则在Nc的后继节点Ns上，，所以Nc发送消息给Ns查找key的值value,Ns将查询结果返回给Ni(消息源)

2）否则，Nc查找其对应的路由表，找到小于j的最大编号节点Nh(如果所有路由表都大于j,则选择第m-项路由信息内的数据作为Nh)，Nc向Nh发送消息，请求它代表Ni查找key的值value,Nh此时成为当前节点Nc,继续按照步骤1和步骤2递归进行查找操作。

大概讲的就是第一从后面第一个节点查，如果不是，则查询该节点的数据路由表，找到最大的比key(j)小的节点进行与第一个节点一样的循环操作，直到找到为止。

4，添加节点

1. 通过路由算法查找到后继节点，Nnew的后继节点指向Ns,前继节点置为空值null.
2. 后续，稳定性检测

大概就是你都找到了后继节点，为什么前继不是直接插入呢，因为虽然在一般情况下不会出现后继节点的原前趋节点在后继节点和New节点之间。但是以防万一还是要检测该原前趋结点是不是小于new节点。

## 2.数据复制和一致性

5.CAP（Consistency/Availability/Partition Tolerance）

强一致性：即在分布式

可用性：

分区容忍性：

目前虽然三者不能兼得，但是在系统未产生网络分区的情形下，应该尽可能保证AC两者兼得，也即大多数情况下考虑CAP三者兼得，当发生网络分区时，系统应该能够识别这种情况并对其进行正确处理，具体：

首先能够识别网络分区发生，然后网络分区场景下进入明确的分区模式，此时可能会限制某些系统操作，最后在网络分区解决后能够进行善后处理，即回复数据的一致性或者弥补分区模式中产生的错误。

## 3.大数据常用的算法与数据结构

1.布隆过滤器（Bloom Filter）

基本原理：BF可以高效地表征集合数据。其使用长度为m的位数组来存储集合信息，同时使用k个相互独立的哈希函数将数据映射到位数组空间。

基本思想：首先，将长度为m的位数组元素全部置为0。对于集合S中的某个成员a，分别使用k个哈希函数对其计算，如果hi(a)=x(1<=i<=k,1<=x<=m),则将位数组的第x位置为1，对于成员a来说，经过k个哈希函数计算后，可能会将位数组中的w位（w<=k）设置为1.对于集合中的其他成员也如此处理，这样即可完成为数组空间的集合表示。

2）误判率及相关计算

由于BF使用的是位数组和哈希函数来表征集合，并不需要实际存储集合数据本身的内容。所以其空间利用率非常高，但是潜在一个问题就是，在查询某个成员是否属于这个集合的时候会发生误判，加入x1,x2成员已经通过上文算法表现到位数组集合中。而x3计算的位置正好和x1,x2的重叠了。就会产生误判的情况，函数本身就会认为其是属于集合的。

3）在误判时，产生误判被许多变量关联着。例如：哈希函数的个数，位数组的大小，集合大小，等

而最优的哈希函数的个数为：k=m/nln2;

4)改进：计数Bloom Filter

由于基本的BF无法删除集合成员，然后改进成计数BF，容易想到的是，在普通的BF上，删除成员的话，反向过程，利用hash函数找到其对应的位数组的位置，其所有的位置必定都为1，然后将其置为0即可，但是这样明显是错误的。因为其他数据也可能占用了该位数组中的这一位，而导致影响到了其他数据查找等。

计数BF就是基本信息比特位扩展为多个比特位，来进行计数，这样在删除的时候就不会影响其他数据的查找。（添加就对应位置加1，减少就减一，判断是否存在就，看其相应位置是否全为1）.

5）BF使用的领域非常广泛，尤其是数据量极大且容忍一定误判率的场合。

（1）Google Chrome浏览器使用它进行恶意URL的判断；

（2）网络爬虫使用它对已经爬过的URL进行判断，

（3）缓存使用BF来对海量数据进行查找，

（4）比特币使用BF对历史交易进行验证等

2.SkipList(http://blog.csdn.net/ict2014/article/details/17394259)

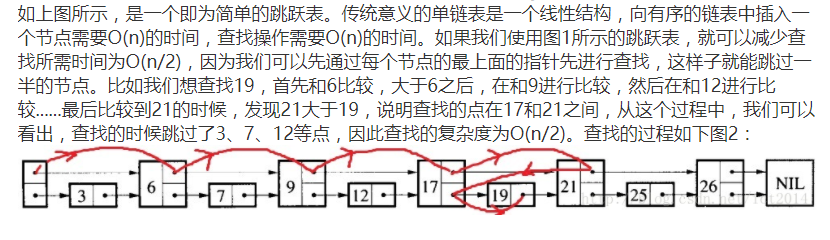
1)一种可代替平衡树的数据结构

2）SkipList依靠随机生成数以一定概率来保持数据平衡分布。

3）插入，删除，查找的时间复杂度都是O(log(N));

4)优点：高效，维护简单。

5）Redis在实现Sorted Set数据结构时，采用的就是SkipList



6）其实实质上，SkipList有上下左右指向。