## read-qry

### where-condition

#### exist pk in

in是把外表和内表作hash 连接；

exists 是对外表作loop 循环，每次loop 循环再对内表进行查询；

##### key

p, n/m

H(in) = p.id \* n + n / m ≈ p.id \* m + m.id ;

H(exist) = p.id \* m.id

###### in适合于外表大而内表小的情况；exists适合于外表小而内表大的情况

###### in不走索引

###### insert exist

在插入记录前，需要检查这条记录是否已经存在，只有当记录不存在时才执行插入操作，可以通过使用 EXISTS 条件句防止插入重复记录。

##### in

在select \*  from A where id in(select id from B); 中，in()中的子查询只执行一次，它查询出B表中的所有ID值并缓存起来；之后，在内存中检查A表的id是否与B表中的id值相等，如果相等则则将A表的记录加入到结果集中，直到遍历完A表中的所有记录。

它查询的过程类似于一下过程

Array A={select \* from A};

Array B={select id from B};

for (int i=0;i<A.length;i++){

for (int j=0;j<B.length;j++){

if (A[i].id == B[j].id){

resultSet.add(A[i]);

break;

}

}

}

如：A表有10000条记录，B表有1000000条记录，那么使用in最多可能有10000\*1000000次遍历，效率很差；

H(in) ,(p=1000,m=1w,result = 1000\*1w +1w)

H(exist), (p=1000,m=1w,result = 1000\*1000\*1w)

A表有10000条记录，B表有100条记录，那么使用in最多可能有10000\*100次遍历，遍历次数大大减少，效率大大提升；(p=1000,m=1w,result = 1000\*1w +1w)

 结论：in()适合B表比A表数据小的情况

##### exists

在select \* from A where exists (select 1 from B where A.id=B.id);之中  
exists()会执行A.length次，它并不缓存exists()结果集，因为exists()结果集的内容并不重要，重要的是其内查询语句的结果集空或者非空，空则返回false，非空则返回true。  
它的查询过程类似于以下过程

Array A=(select \* from A);

for(int i=0;i<A.length;i++) {

if(exists(A[i].id) { //执行select 1 from B where B.id=A.id是否有记录返回

resultSet.add(A[i]);

}

}

return resultSet;

当B表比A表数据大时适合使用exists()，因为它没有那么多遍历操作，只需要再执行一次查询就行。  
如：A表有10000条记录，B表有1000000条记录，那么exists()会执行10000次去判断A表中的id是否与B表中的id相等。  
如：A表有10000条记录，B表有100000000条记录，那么exists()还是执行10000次，因为它只执行A.length次，可见B表数据越多，越适合exists()发挥效果。

再如：A表有10000条记录，B表有100条记录，那么exists()还是执行10000次，还不如使用in()遍历10000\*100次，

因为in()是在内存里遍历比较，而exists()需要查询数据库，我们都知道查询数据库所消耗的性能更高，而内存比较很快

**结论：exists()适合B表比A表数据大的情况**

关于EXISTS：  
EXISTS用于检查子查询是否至少会返回一行数据，该子查询实际上并不返回任何数据，而是返回值TRUE 或FLASE   
EXISTS指定一个子查询，检测行的存在。  
语法：EXISTS subquery  
参数：subquery 是一个受限的 SELECT 语句 (不允许有 COMPUTE 子句和 INTO 关键字)。  
结果类型：Boolean 如果子查询包含行，则返回 TRUE ，否则返回 FLASE 。  
结论：select \* from A where exists (select 1 from B where A.id=B.id)   
EXISTS(包括 NOT EXISTS )子句的返回值是一个boolean值。 EXISTS内部有一个子查询语句(SELECT ... FROM...)， 我将其称为EXIST的内查询语句。其内查询语句返回一个结果集。 EXISTS子句根据其内查询语句的结果集空或者非空，返回一个布尔值。  
一种通俗的可以理解为：将外查询表的每一行，代入内查询作为检验，如果内查询返回的结果取非空值，则EXISTS子句返回TRUE，这一行行可作为外查询的结果行，否则不能作为结果。  
分析器会先看语句的第一个词，当它发现第一个词是SELECT关键字的时候，它会跳到FROM关键字，然后通过FROM关键字找到表名并把表装入内存。接着是找WHERE关键字，如果找不到则返回到SELECT找字段解析，如果找到WHERE，则分析其中的条件，完成后再回到SELECT分析字段。最后形成一张我们要的虚表。  
WHERE关键字后面的是条件表达式。条件表达式计算完成后，会有一个返回值，即非0或0，非0即为真(true)，0即为假(false)。同理WHERE后面的条件也有一个返回值，真或假，来确定接下来执不执行SELECT。

<https://www.cnblogs.com/niudaxianren/p/10018634.html>

## delete

### truncate

<https://www.cnblogs.com/zhoufangcheng04050227/p/7991759.html>

##### des

系统资源和事务日志资源更少；

truncate和delete只是删除表的数据(定义),drop语句将删除表的结构、被依赖的约束(constrain)、触发器 (trigger)、索引(index);依赖于该表的存储过程/函数将保留,但是变为invalid状态。

delete语句不影响表所占用的extent、高水线(high watermark)保持原位置不动；

drop语句将表所占用的空间全部释放；

truncate语句缺省情况下将空间释放到minextents的 extent,除非使用reuse storage。truncate会将高水线复位(回到最初)。

##### grammar

[ { database\_name.[ schema\_name ]. | schema\_name . } ]  
    table\_name  
[ ; ]

**参数**

database\_name  
数据库的名称。

schema\_name  
表所属架构的名称。

table\_name  
要截断的表的名称，或要删除其全部行的表的名称。

#### characteristic

##### #数据页

释放存储表数据所用的数据页来删除数据，并且只在事务日志中记录页的释放

##### #单行不记录-no trigger

删除表中的所有行，而不记录单个行删除操作；

TRUNCATE TABLE 不记录在日志中，所以它不能激活触发器，由FOREIGN KEY 约束引用的表，不能使用 TRUNCATE TABLE；

(delete每删除一行,在事务日志中为所删除的每一行记录)

##### #系统资源、事务日志资源更少

系统资源和事务日志资源

##### 约束、索引不变

表结构及其列、约束、索引等保持不变；新行标识所用的计数值重置为该列的；

想保留标识计数值，请改用 DELETE；

##### 非索引视图表

不能用于参与了索引视图的表

##### update statistics

对用TRUNCATE TABLE删除数据的表上增加数据时，要使用UPDATE STATISTICS来维护索引信息

##### no rollback

DELETE操作将被撤销，但TRUNCATE不会撤销；

delete是DML语句,不会自动提交；

drop/truncate都是DDL语句,执行后会自动提交。

##### reuse storage

如果是整理表内部的碎片，可以用truncate跟上reuse stroage，再重新导入、插入数据。