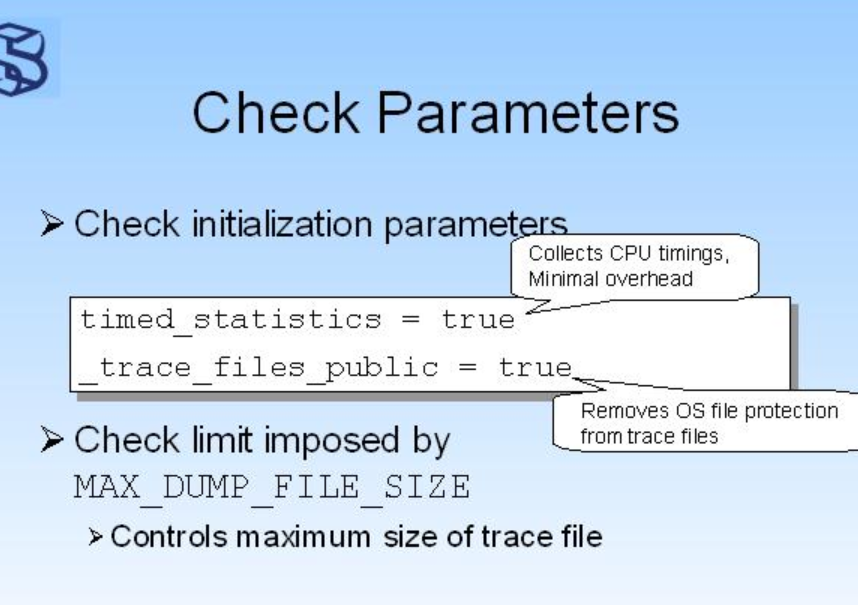
## [Lesson 4 SQL Trace and 10053 Event](JavaScript:parent.NavigateAbs(0))

学习目的： sql性能优化追逐数据库性能消耗大的SQL

会话级跟踪。SQL\_TRACE的通常使用方式是仅跟踪一个会话



第一步检查；

timed\_statistics boolean TRUE



第二部：

alter session set tracefile\_identifier='dave'; /\*XE\_ora\_2151\_dave.trc

主要命名 trc文件 以便好找\*/

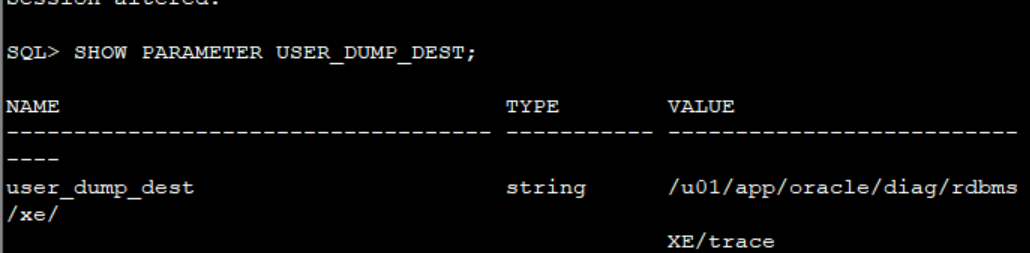
alter session set sql\_trace=true;/\*开启 会话及的trace\*/

----这里写需要追踪的SQL

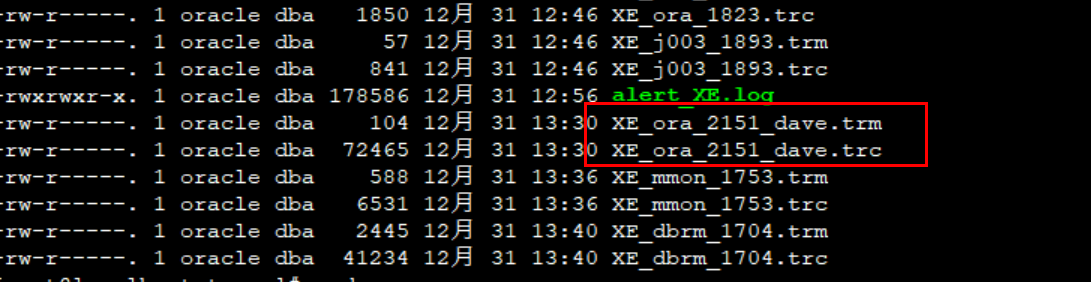
ALTER SESSION SET SQL\_TRACE=FALSE;/\*关闭会话及trace\*/

查找trace 文件位置

SHOW PARAMETER USER\_DUMP\_DEST;



Trace文件名称



Tkprof 命令查看 trace 文件

命令查询

tkprof /u01/app/oracle/diag/rdbms/xe/XE/trace/XE\_ora\_2151\_dave.trc report.txt

生成 report.txt

通过 ftp 下载查看

CALL：每次SQL语句的处理都分成以下三个部分

Parse：这步将SQL语句转换成执行计划，包括检查是否有正确的授权和所需要用到的表、列以及其他引用到的对象是否存在。

Execute：这步是真正的由Oracle来执行语句。对于insert、update、delete操作，这步会修改数据，对于select操作，这步就只是确定选择的记录

Fetch：返回查询语句中所获得的记录，这步只有select语句会被执行。

COUNT：这个语句被parse、execute、fetch的次数。

CPU：这个语句对于所有的parse、execute、fetch所消耗的cpu的时间，以秒为单位。

ELAPSED：这个语句所有消耗在parse、execute、fetch的总的时间。

DISK：从磁盘上的数据文件中物理读取的块的数量。一般来说更想知道的是正在从缓存中读取的数据而不是从磁盘上读取的数据。

QUERY：在一致性读模式下，所有parse、execute、fetch所获得的buffer的数量。一致性模式的buffer是用于给一个长时间运行的事务提供一个一致性读的快照，缓存实际上在头部存储了状态。

CURRENT：在current模式下所获得的buffer的数量。一般在current模式下执行insert、update、delete操作都会获取 buffer。在current模式下如果在高速缓存区发现有新的缓存足够给当前的事务使用，则这些buffer都会被读入了缓存区中。

ROWS：所有SQL语句返回的记录数目，但是不包括子查询中返回的记录数目。对于select语句，返回记录是在fetch这步，对于insert、update、delete操作，返回记录则是在execute这步。

A、query+current/rows 平均每行所需的block数，太大的话（超过20）SQL语句效率太低

B、Parse count/Execute count parse count应尽量接近1，如果太高的话，SQL会进行不必要的reparse

C、rows Fetch/Fetch Fetch Array的大小，太小的话就没有充分利用批量Fetch的功能，增加了数据在客户端和服务器之间的往返次数。

D、disk/query+current 磁盘IO所占逻辑IO的比例，太大的话有可能是db\_buffer\_size过小(也跟SQL的具体特性有关)

E、elapsed/cpu 太大表示执行过程中花费了大量的时间等待某种资源

F、cpu　Or　elapsed 太大表示执行时间过长，或消耗了了大量的CPU时间，应该考虑优化

G、执行计划中的Rows 表示在该处理阶段所访问的行数，要尽量减少

几个重要参数的用法讲解

•sys参数，如果不指定默认值为yes.这个参数的含义是，输出文件中是否包含以SYS用户运行的sql语句。这个参数还是蛮有用的，我们执行sql语句的时候，后台经常会执行很多递归的语句，比如你输入了SELECT \* FROM TEST;如果这个语句是硬解析的话，那么会产生很多递归的SQL，递归的去查询表的统计信息，列的统计信息，索引的统计信息等，当然递归的不止是这些。这些递归的sql都是以SYS用户运行的，如果你不希望看到这些递归SQL，那么就加上这个参数sys=no.

•record参数，它指定的是一个路径下的文件，这个文件用来生成在跟踪文件中找到的所有的非递归SQL。比如你在SQLPLUS里执行了三条语句,select \* from a;select \* from b;select \* from c;，那么如果你指定了这个参数如：record=c:\test.log，那么你用tkprof格式化跟踪文件后，这个test.log里就会记录这三个SQL。这个特性在有些时候还是满有用的，因为跟踪文件往往都会比较大，找起来会比较费劲，我们可以通过指定这个参数先大体了解下，跟踪文件里都有哪些非递归SQL。而且这个功能还有助于我们重演SQL语句（绑定变量的不可以）。

•aggregate参数，它指定tkprof是否将同样文本内容的sql聚合处理，比如，你执行了十次select \* from a,如果你指定这个参数为no（默认情况），那么产生的输出文件会有十个这样语句的执行信息，如果你指定的是yes，那么tkprof会把这十次的执行信息汇总显示。这个参数怎么指定就看你的需要了，个人觉得还是满有用的一个参数。

•sort参数，这个参数是经常使用到的一个参数，它用来指定tkprof输出文件里sql语句按照什么排序，默认是按照执行的先后顺序排序的，我们可以指定它按照其他方式排序，比如磁盘读取数，CPU时间等。这个参数最经常用的方式是：sort=prsela,exeela,fchela,其实这三个值加起来就是响应时间，即按照响应时间排序。这里别产生误解，tkprof会根据prsela,exeela,fchela三个值的和进行排序，而不是像SQL语句似的一个个的排序。

•print参数，它经常搭配sort参数一起使用，用来指定tkprof输出sql语句的数量。这两个参数搭配使用起来就比较妙，比如你想知道一个跟踪文件里响应时间排前十的SQL，那么你就可以sort=prsela,exeela,fchela print=10来搭配使用。

•explain参数，这个参数的含义是为每一个SQL提供一个执行计划。使用的方法是explain=用户名/密码，其实原理很简单，就是通过你指定的用户名，密码登陆数据库，然后为每一个sql执行以下explain plan for sql，输出到plan\_table里，最后添加到输出文件里。注意，由于explain plan for 命令要求执行操作的用户要对sql语句里包含的对象都有执行权限，如果包含视图，也要对视图基于的基础表有执行权限，否则产生不了执行计划。注意增加了这个参数后，执行tkprof会比较慢。

•wait参数，指定输出文件中包含不包含等待事件，默认是包含的。一般都取默认值。

基本最常用到的就这些，其他的我就不介绍了，平时这些也就基本够用了。

## INVENTORY(库存模块)—ER图

HR.HR\_ALL\_ORGANIZATION\_UNITS ----业务组织和单元组织定义表

仓库定义表，

MTL\_PARAMETERS 物料参数表，

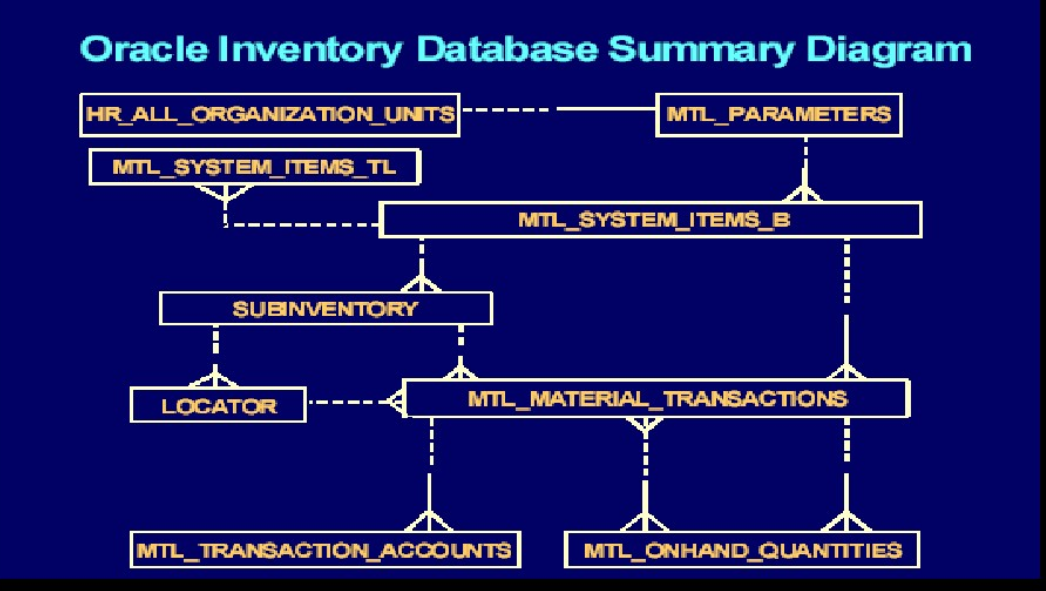
mtl\_system\_items\_tl ---物料表的多语言视图 lang(‘zhs’)多个倍的数据

mtl\_system\_items\_b ---物料信息表 物料ID,物料编号，库存组织。

mtl\_material\_transactions –--事务交易记录信息

mtl\_transaction\_accounts---会计分录 每个物料的总账信息。

Mtl\_onhand\_quantities --物料表更新记录

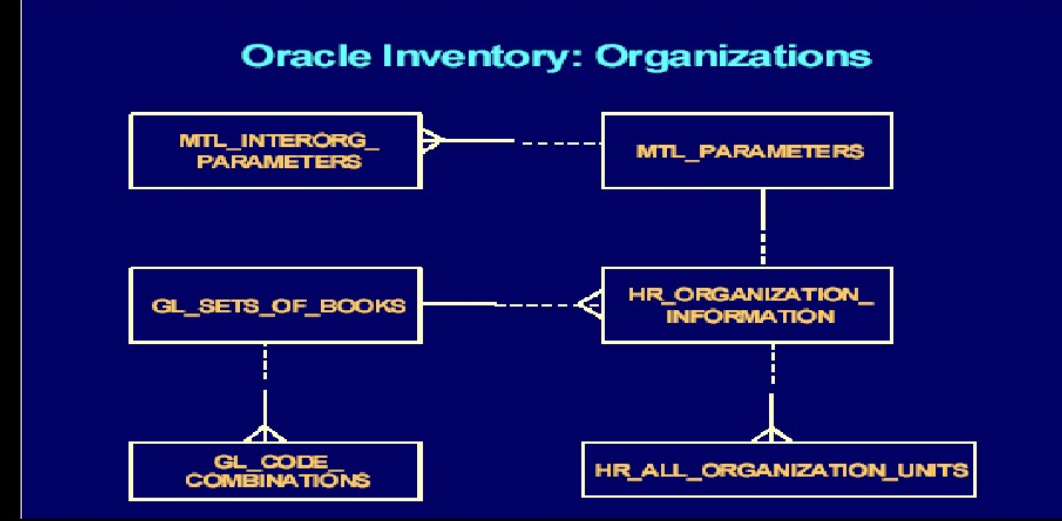


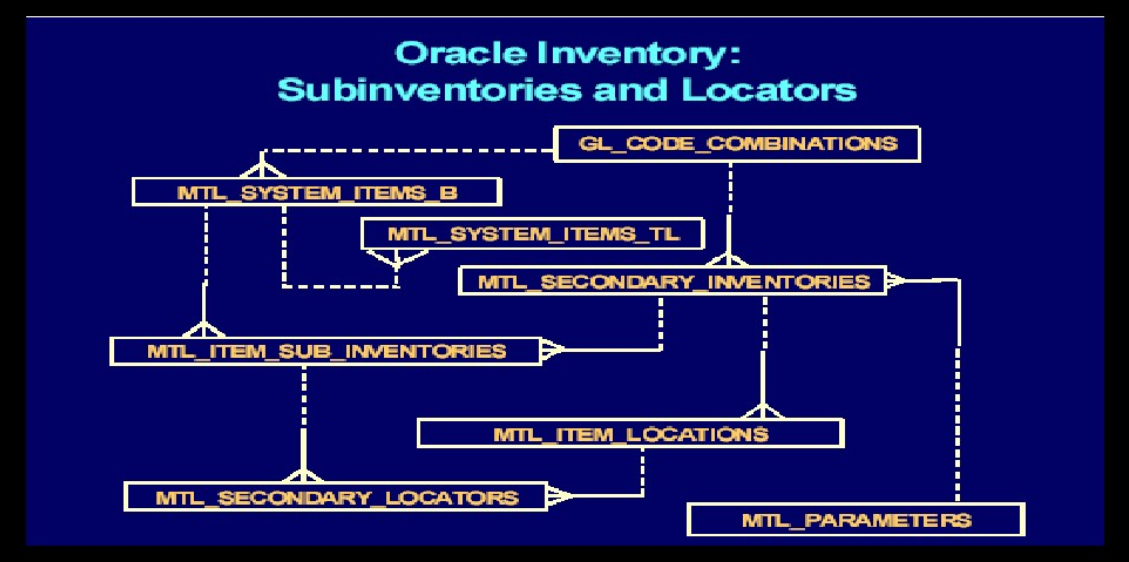
MTL\_INTERORG\_PARAMETERS—记录两个库存组织之间 FROM-ORGANIZATION,TO\_ORGANAZITION

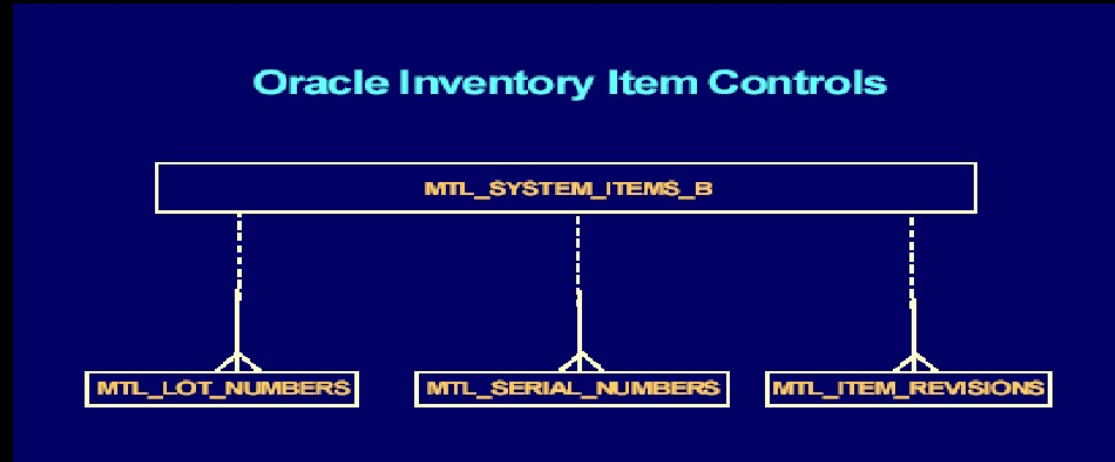
HR\_ORGANIZATION\_INFORMATION----记录组织信息的上下问配置信息。

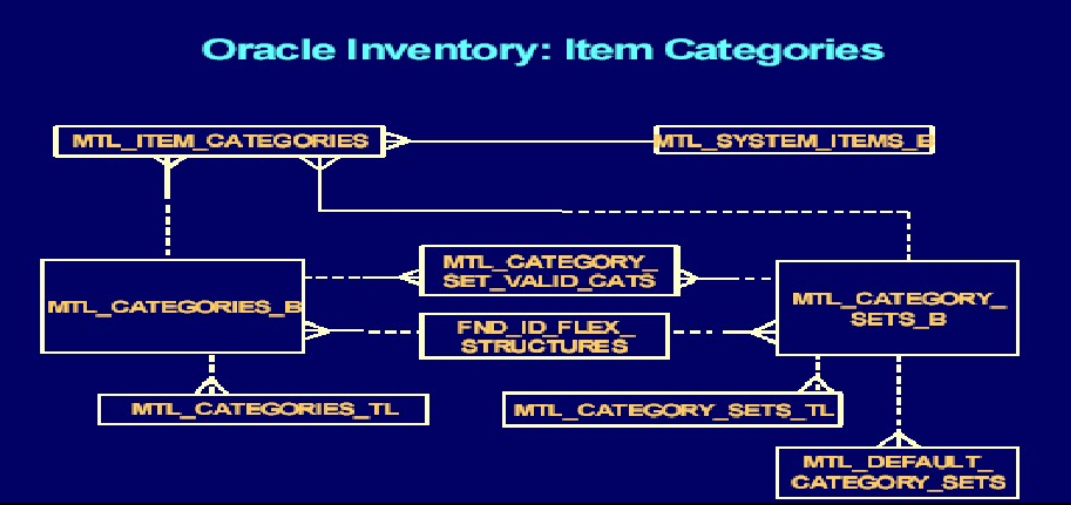
GL\_SETS\_OF\_BOOKS---总账账套信息

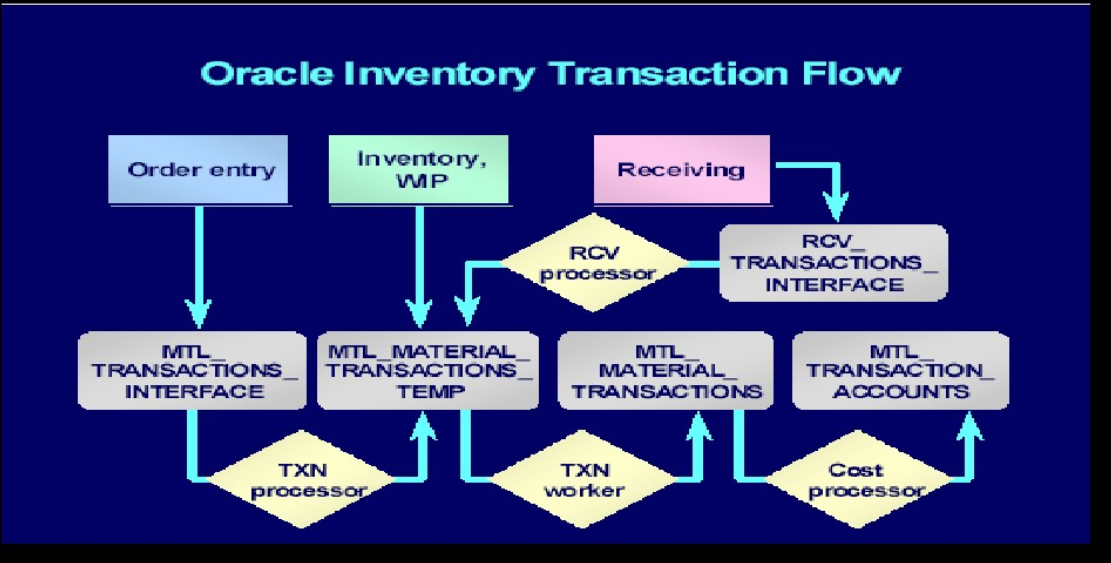
GL\_CODE\_COMBINATIONS ---科目信息

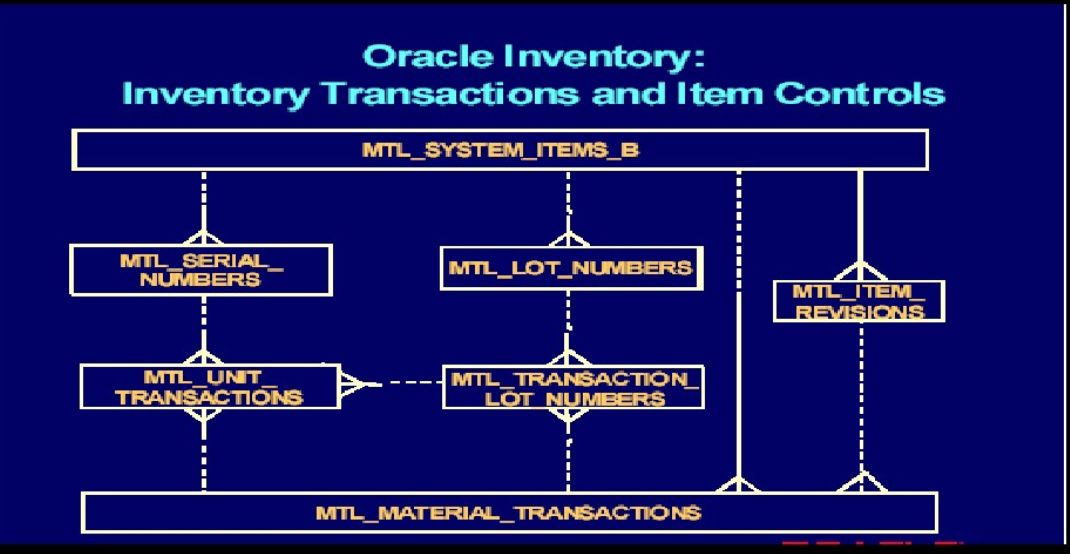


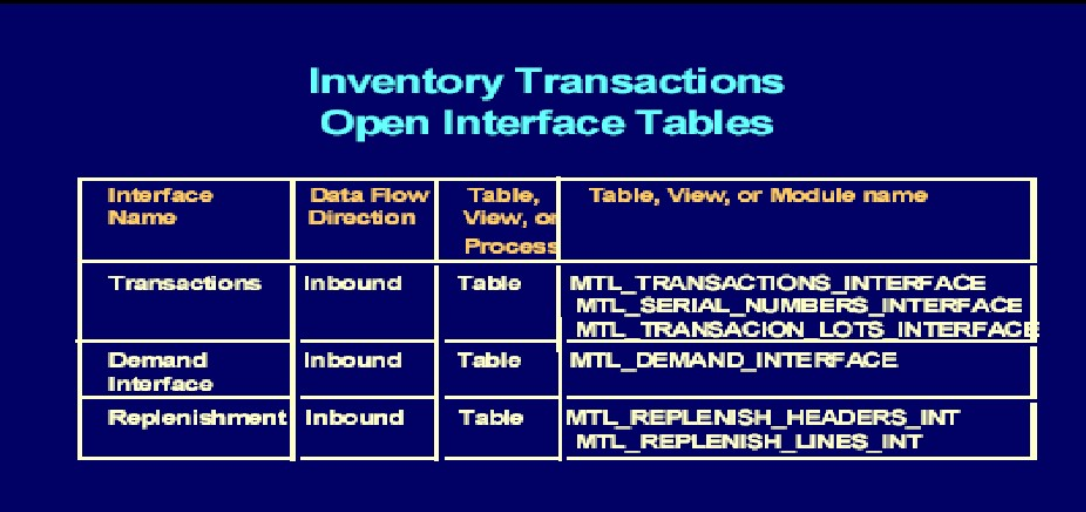


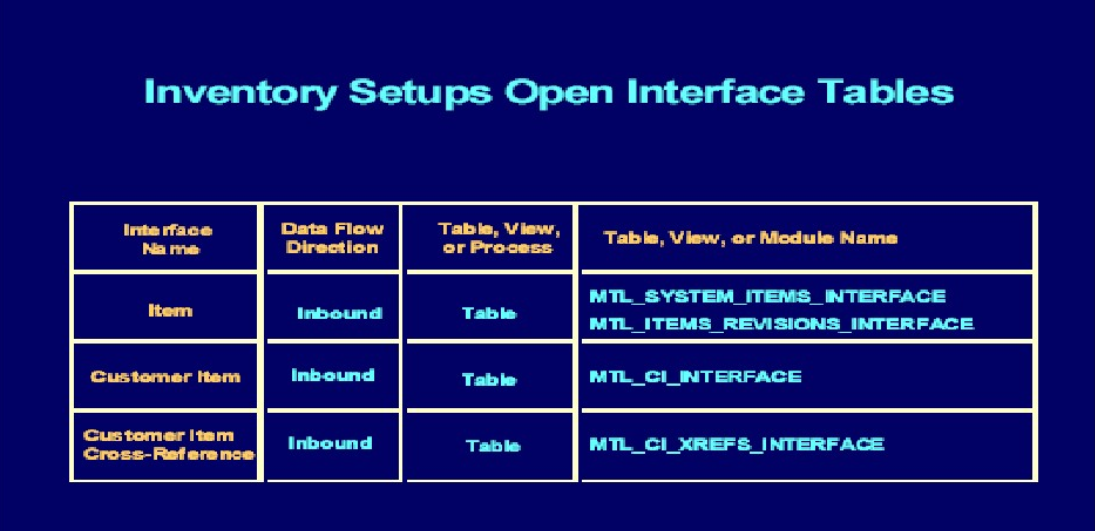




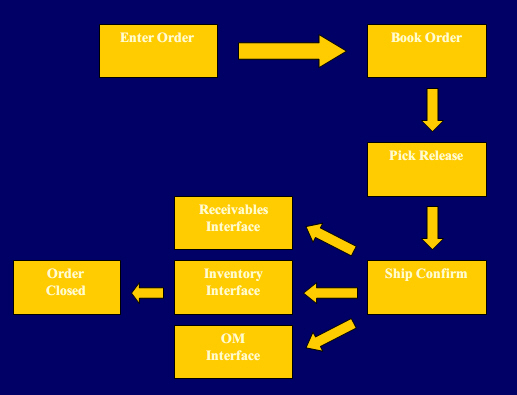








## 销售订单



订单每个状态变化：

