或者甚至用另一种分区类型（哈希而非范围）。ASE中的语义分区不支持多层级分区。

这些是额外的表分区优势，诸如隔离维护每个分区的行动。但是，它存在一个大问题每当一个时间段趋近结束时，不仅需要创建一个新的分区，而且涵盖了分区的视图必须更改，加入新表并移除旧表。虽然修改视图定义并不需要花一秒钟时间，但必须在无人使用该视图时完成，所以必须在行动时强制暂停。另外，真正完整的改变需要以下内容：

（1）创建新的“当前”分区表、索引和用户权限。

（2）更新视图定义，删除并重建的同时修改替代触发器和用户权限。

（3）去除最近的“当前”分区的权限。

（4）从最近的“当前”分区中支持OLTP的索引。

（5）在最近的“当前”分区中增加支持DSS查询的索引。

为了改变视图定义而造成的维护和缩短服务时间要求是本方法的劣势之一，还可能由于改变了对象ID而对预编译对象（例如存储过程）造成不确定的影响。另一个明确的考虑就是因为优化器并不知道每个分区与什么查询语义相关，结果就可能常常出现对所有数据的访问，频繁创建包含任意谓词结果的工作表，并将工作表用于查询处理中。若是语义分区，则优化器了解分区键，能避免中间的工作表并能执行分区消除。虽然表的分区是像上述那样离散的集合，而同样的键将触发优化器去了解没有可用的结果，并且可缩短union all查询的周期。

6 可配置的DSS查询优化

关于ASE 15，经常被问到的问题之一是对开发人员都有哪些特性？赞成开源系统的人们倾向那些产品所吹捧的“开发人员”特性，并抱怨主流的RDBMS系统长时间关注DBA，已经忽略了开发人员。那些产品中的一个严峻缺陷就是查询优化，查询优化器100%的是完全关注开发人员的特性。毕竟，日常运营的DBA通常不会关心查询优化器运行得如何，只要系统性能在可接受的资源消耗范围内。正是开发人员才去考虑该产品是否支持星型连接等。

如果考虑它，企业级DBMS产品的大部分精髓都被赋予了查询优化，一定程度地重写查询优化器将会耗费博士研发团队多年的辛勤工作。毫无疑问，在ASE 15.0中最大的改变正是它一完全重写的查询优化器。

重写优化器需要满足不断增长的混合负载、OLAP和甚至是DSS应用程序的需求，由于要满足实时报表或普通应用程序扩展以利用数据保持力的改变来满足业务需求，诸如市场、趋势分析或其他源于历史数据的信息，如客户波动。考虑以下OLTP与DSS应用程序的趋势变迁描绘图：

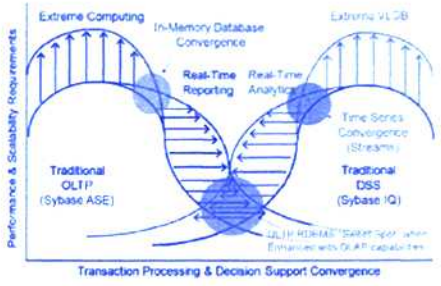

图3 OLTP和DSS应用程序趋势

注意OLTP与DSS系统的变迁，为了填补它们之间的空缺而产生了两类新的需求一实时报表和实时分析。为了支持实时报表复杂查询处理的变化，ASE完全重写了查询优化器，增加了新的典型DSS/报表优化特性，例如：DSS连接技术，诸如合并连接（merge join）、哈希连接（hash join）、N阵嵌套循环连接（n-ary nested loop join）和对星型模型的支持；支持聚合、联合和去重的更快算法；改进了对并行查询的支持，包括导向/矢量连接以减少资源使用。

非常有用的也是比较复杂的特性之一是优化目标和标准。对优化器工程师来说，关键需要考虑开发出的优化器能在最快的时间内能找到优化的执行计划，因为优化时间很容易超过执行时间。例如，一个12向的连接涉及12张相对大的表（每个表在10万至1 000万行），需要10 min优化但仅需30 s去执行。因此，如果优化器了解应用程序的本质，它就能避免在优化阶段的前期找到一个查询计划后，又竭尽全力去寻找可能不会比该计划更快的查询计划。ASE 15.0实施了3个特性以达此目的一优化目标、超时和标准，见表10。

表10 不同优化目标和一些关键查询技术之间的高层级关系

优化目标	嵌套循环连接	合并连接	哈希连接	廉洁N阵嵌套循环	并行查询	浓密连接	星型模式	积极聚合
ASE 12.5	✓				✓			
allrows_oltp	✓	✓						
allrows_mix	✓	✓		✓	✓			
allrows_dss	✓	✓	✓	✓	✓	✓	✓	✓

虽然allrows_dss能运行所有内容看起来很诱

人，但是请记住前面关于优化时间超出执行时间的讨论。一个关键的原因是合并与哈希连接被认为是DSS类型的操作，因为它们假定连接不会被索引完全覆盖，通常可能并不是即席报表查询的情况。因此，这样的连接条件通常会因数据排序而受益，要么是物理地要么是内存哈希表，与连接排序的连接列。较严厉的OLTP系统应用索引来强制主外键。

虽然“容量规划”常常被包含在开发周期中，而“负载管理”却常被忽略。不幸的是，业务用户通常让硬件/软件来自动执行负载均衡，还需要考虑业务优先权，简直就是不可能的任务。事实是不同的应用程序常常会有不同的优化需求，使用错误的技术同时支持所有的需求将造成失败。请认真考虑，如何去满足冲突的需求：高速插入/最小化优化；复杂查询与更长时间的优化需求。在不断改变的现实中，日趋快速的CPU，大量廉价的内存。现在更倾向于用更多的CPU和内存来执行密集算法么？四核=低价的大型SMP。它不仅带来了第2阶段的应用程序整合，它增加了复杂度，还在更低的价格点上提供了并行策略的CPU资源。在从未如此扩展的领域中，更大的数据量和更大的使用人群。

好的方法是通过会话级别和查询相关的优化控制，让开发人员来控制优化。因此，ASE 15不仅增加了DSS优化技术，它还令其可配置并通过优化目标提供了3个初始的优化情形。开发人员必须考虑应用程序组件的本质并与DBA一道工作，为不同的应用程序实施“优化情形”，然后这些可以根据应用程序名称或其它特质来通过登录触发器实现。通过这些特性，OLTP应用程序能避免不适合OLTP的长时间优化或复杂优化算法，而报表应用程序则可利用诸如索引联合、浓密连接等的新功能。使用支持表来存储优化情形的登录触发器例子如下：

```
-- created in master to allow prevent login failures in case the server needs to
-- be booted with traceflag to recover master database only
use master
go
create table optimization_profile (
appname    varchar(30)      not null,
opt_goal   varchar(20)      not null,
parallel_deg      tinyint   not null,
use_stmtcache     bit       not null,
use_mergejoin     bit       not null,
use_hashjoin      bit       not null,
use_idxunion      bit       not null,
delayed_commit    bit       not null,
primary key (appname)
)
go
exec sp_logintrigger 'drop' go
if exists (select 1 from sysobjects
where name='sp_optimization_profile' and type='P' and uid=user_id()) drop procedure sp_optimization_profile
go
print "...creating procedure: 'sp_optimization_profile'" go
create procedure sp_optimization_profile as begin
declare @appname   varchar(30),
@opt_goal          varchar(20),
@use_stmtcache     bit,
@use_mergejoin     bit,
@use_hashjoin      bit,
@use_idxunion      bit,
@delayed_commit    bit
if suser_name()='sa' return 0
if has_role('sa_role',0)=1 return 0
select @appname=(case
when clientapplname is not null then clientapplname else program_name
end)
from master..sysprocesses
where spid=@@spid
set export on
if exists (select appname
from master..optimization_profile where appname=@appname)
begin
select @opt_goal=opt_goal,
@use_stmtcache=use_stmtcache from master..optimization_profile
where appname=@appname
if @opt_goal="allrows_oltp" set plan optgoal allrows_oltp if @opt_goal="allrows_mix" set plan optgoal allrows_mix if @opt_goal="allrows_dss" set plan optgoal allrows_dss
if @use_stmtcache=1
begin
set statement_cache on set literal_autoparam on
end else begin
set statement_cache off
end
if @delayed_commit=1 set delayed_commit on ...
end
return 0
end
go
grant exec on sp_optimization_profile to public go
exec sp_logintrigger 'master..sp_optimization_profile' go
```

如果一个单独的服务器试图支持不同类型的应用程序(OLTP和DSS)，则强烈推荐使用该技术，因其支持应用程序开发人员控制每个应用程序必要的优化配置，而无需在应用程序中进行硬编码配置。除了优化情形外，开发人员也必须考虑临时表和应用程序相关tempdb的负载管理。当一个服务器与DSS用户共享时，这将改善需要支持混合负载的OLTP应用程序响应度。当大型的DSS查询充满tempdb时，通过避免tempdb挂起问题，它也能改善OLTP应用程序的可用性。

文/赛贝斯软件（中国）有限公司

（未完待续）