

Database Architecture: Federated vs. Clustered

An Oracle White Paper
March 2002

Database Architecture: Federated vs. Clustered

Executive Overview	3
Introduction	4
What is a Federated Database?	4
What is the Shared-Disk Cluster Database Architecture?	5
Clarifying the Terminology.....	6
Application Development	7
Porting Complex OLTP Applications.....	7
Developing New OLTP Applications.....	8
Scalability.....	9
Availability.....	10
Manageability.....	11
What Happens When You Add a Node?	12
TPC-C Benchmarks on Federated Databases	12
TPC-C Schema Is Inherently Partitionable	13
Most TPC-C SQL Accesses are Local	13
Conclusion	14

Database Architecture: Federated vs. Clustered

EXECUTIVE OVERVIEW

The parallel database market is rife with competing marketing claims, with each vendor touting the benefits of its own architecture. The buyer has to make the choice of a mission-critical software platform while sifting through a mass of rapidly evolving benchmark results, conflicting analyst reviews and uniformly positive customer testimonials.

This paper is a technical evaluation of two different database architectures: the federated architecture, as represented by Microsoft SQL Server, and the shared-disk cluster architecture of Oracle9i Real Application Clusters. We also explain how to interpret and use benchmark results from hardware and database vendors, and how various database features might be relevant in real customer situations.

This paper does not touch upon shared-nothing database clusters. Federated databases are a step towards shared-nothing clusters, but are in fact only distributed databases. However, a shared-nothing clustered database, such as IBM DB2 7.2 EEE, is a single database, with a single data dictionary, and not a collection of loosely coupled databases. Shared-nothing clusters are compared with shared-disk clusters in another Oracle white paper: *Technical Comparison of Oracle Real Application Clusters vs. IBM DB2 UDB EEE*, available from http://otn.oracle.com/deploy/performance/pdf/ibm_db2.pdf

We also include a brief section on terminology, contrasting the Microsoft use of the terms “database cluster” and “partition”, with the generally accepted definitions.

INTRODUCTION

What is a Federated Database?

A federated database is a logical unification of distinct databases running on independent servers, sharing no resources (including disk), and connected by a LAN.

Data is distributed across each participating server. For both the DBA as well as the Application Developer, there is a clear distinction between “local” data, which is on the disk attached to a particular server, and “remote” data, which is owned by another server in the federated database. Applications see a logical single view of the data through UNION ALL views and Distributed SQL – Microsoft calls this technology Distributed Partitioned Views (DPVs). The DPV is constructed differently at each node - it must explicitly consider which partitions are local and which are remote.

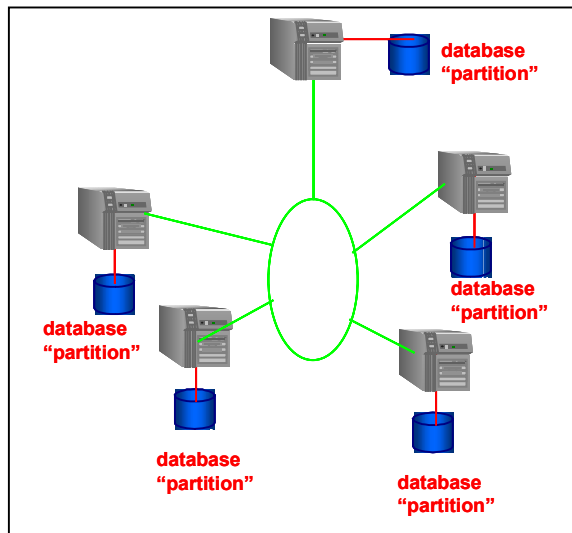


Figure 1: Architecture of a federated database

For example, here's a summary of how you would partition your customers across multiple servers in Microsoft SQL Server (code from the Microsoft web site at

http://msdn.microsoft.com/library/en-us/createdb/cm_8_des_06_17zr.asp).

1. First, create independent partitions on each node:

```
-- On Server1:  
CREATE TABLE Customers_33  
  (CustomerID  INTEGER PRIMARY KEY  
   CHECK (CustomerID BETWEEN 1 AND 32999),  
  ... -- Additional column definitions)  
-- On Server2:  
CREATE TABLE Customers_66  
  (CustomerID  INTEGER PRIMARY KEY  
   CHECK (CustomerID BETWEEN 33000 AND 65999),  
  ... -- Additional column definitions)  
-- On Server3:  
CREATE TABLE Customers_99  
  (CustomerID  INTEGER PRIMARY KEY  
   CHECK (CustomerID BETWEEN 66000 AND 99999),  
  ... -- Additional column definitions)
```

- Then, set up connectivity information (“linked server definitions”) and query optimization options on each participating server.
- Finally, create a DPV at each node. Here is the code for Server 1:

```
CREATE VIEW Customers AS
  SELECT * FROM
  CompanyDatabase.TableOwner.Customers_33
UNION ALL
  SELECT * FROM
  Server2.CompanyDatabase.TableOwner.Customers_66
UNION ALL
  SELECT * FROM
  Server3.CompanyDatabase.TableOwner.Customers_99
```

Similar - but **distinct** - views must be created at each node.

Federated databases are slight extensions of distributed databases with UNION ALL views logically unifying physically distributed data. They are not true database clusters.

When SELECT statements referencing the Customers view specify a search condition on the partition column (CustomerID), the query optimizer uses the CHECK constraint definitions to determine which member table contains the rows. With certain constraints, DPVs can be updated as well as queried from, and the query optimizer uses the same methodology to determine the target member table for an UPDATE. Federated databases have no global data dictionary, so the optimizer must access all nodes to determine the execution plan for a query.

Thus, a **federated database is a distributed database** overlaid by DPV technology. DPVs extend UNION ALL views and Distributed SQL by redirecting SQL statements accessing a UNION ALL view to distributed servers based on predicates on a partitioning key.

What is the Shared-Disk Cluster Database Architecture?

A cluster consists of servers (usually SMPs), a cluster interconnect and a shared disk subsystem. Shared disk database architectures run on hardware clusters that give every participating server equal access to all disks. However, servers do not share memory. Most major hardware vendors provide shared-disk cluster implementations today.

Shared-disk cluster databases implement sophisticated cache-sharing algorithms to mask the complexities of clustered hardware.

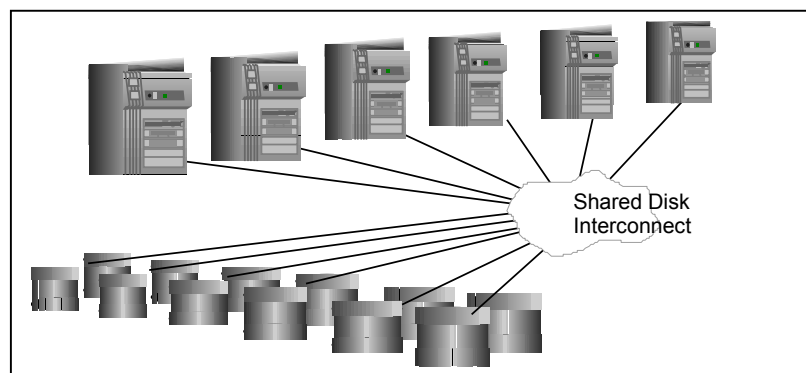


Figure 2: Architecture of a shared-disk clustered database

A database instance runs on every node of the cluster. Transactions running on any instance can read or update any part of the database – there is no notion of data ownership by a node. System performance is based on the database effectively utilizing a fast interconnect, such as the Virtual Interface Architecture (VIA), between cluster nodes. Oracle9i Real Application Clusters (RAC) is the first successful shared-disk cluster architecture and utilizes sophisticated Cache Fusion™ shared-cache algorithms to allow high performance and scalability without data or application partitioning.

Cache Fusion™ works by:

- Utilizing the collective (fast) database caches of all the nodes in the system to satisfy application data requests at any node, effectively “fusing” the caches of all the nodes in the cluster into one cache.
- Removing (slow) disk operations from the critical path for inter-node data synchronization. If two nodes both operate on the same data block, they synchronize it in their caches, not by writing it to disk.
- Greatly reducing the required number of messages for inter-node synchronization
- Exploiting low-latency cluster interconnect protocols for both database messages and data shipping between caches

Applications see the same programming interface on SMP as well as RAC versions of Oracle’s Cache Fusion™ masks the underlying cluster architecture from the application.

In the subsequent sections, we compare and contrast Federated and Shared-Disk database architectures from the following perspectives:

- Application Development
- Scalability
- Availability
- Manageability
- Benchmarking

Clarifying the Terminology

Federated databases are not database clusters – even Microsoft admits this fact. According to Michael Otey, Senior Technical Editor, SQL Server Magazine: *“Technically, this setup isn’t a true clustered implementation. According to Microsoft, SQL Server won’t support true clustering*

until the post-SQL Server 2000 release, code named Yukon.”¹Yukon is not expected to be shipped until well into 2003.

Clustered databases have a single data dictionary, and a single system image. While Microsoft might market their federated solution as a cluster, the fact remains that federations are far from meeting the definition of a cluster.

Microsoft also calls nodes in a federation “partitions”. This is a misleading use of the term. In Oracle9i, and in general industry parlance, a partition is a smaller, more manageable piece of a schema object (table or index). “Partitions and subpartitions of a table or index all share the same logical attributes. For example, all partitions (or subpartitions) in a table share the same column and constraint definitions, and all partitions (or subpartitions) of an index share the same index options. Partitioning is transparent to existing applications and standard DML statements run against partitioned tables.”²

In contrast, tables in federated nodes are defined independently, and in a subsequent step they may be logically aggregated into a DPV. However, DPVs do not share the same logical attributes (each table in a DPV has different constraint definitions) and the DPV is not transparent to applications.

APPLICATION DEVELOPMENT

Oracle9i Real Application Clusters (RAC) appears just like a regular Oracle9i database – there are no additional constraints on the application developer. An application, even a complex OLTP application like SAP or the Oracle e-Business Suite, written using Oracle9i for an SMP platform runs with no modification on a shared-disk cluster running RAC. The single database image, with no data partitioning, carries over from the SMP to the cluster.

In contrast, federated databases such as Microsoft SQL Server do not have a single database image – they are multiple independent databases. Data must either be distributed across participating databases (for large, frequently updated transaction tables, e.g., Order Lines) or replicated (for reference data tables that can be accessed equally from every node, e.g., Item). Dividing data entities across databases requires creating Distributed Partition Views, which are distinct on every node. And replicated tables that are updated need to be kept in sync across databases using custom INSTEAD OF triggers

Porting Complex OLTP Applications

Popular OLTP applications such as those from SAP, PeopleSoft or Oracle have thousands of tables and unique global indexes. An index is considered global

¹ SQL Server Magazine, June 2000

² Oracle9i Server Documentation, http://download-west.oracle.com/otndoc/oracle9i/901_doc/server.901/a90117/partiti.htm#2165

when it indexes all records for a single logical table irrespective of how many physical tables it is divided across.

	Tables	Primary Key Indexes	Alternate Key Indexes
PeopleSoft	7493	6438	900
Oracle eBusiness Suite (ERP only)	8155	800	5100
SAP	16500	16239	2887

Table 1: Tables and Indexes in Popular OLTP Applications

These applications require global unique indexes on non-primary key columns both for speedy data access as well as for ensuring data integrity. An example of this type of index would be the unique index on *Customer_Number* in the *RA_Customers* table in the Oracle eBusiness Suite, which ensures that there is only one customer with a particular value of the unique business key – a key which is not the primary key for the table. Without these indexes, mission-critical application data can be corrupted, duplicated or lost.

Applications also usually do not partition their data accesses cleanly. It is generally not feasible to find partitioning keys for application tables that yield a high proportion of “local” data accesses. Local accesses are those in which the requirements of a query can be satisfied exclusively by the contents of a single partition of data. Most significant queries in SAP, PeopleSoft or the Oracle eBusiness Suite join multiple tables, and different queries use different alternate keys in the join predicates. And non-local data accesses incur the unacceptable performance overhead of frequent distributed transactions.

Even if a suitable partitioning key could be found, thousands of application tables would have to be partitioned. Thus, to port PeopleSoft or SAP to a federated database SQL Server configuration would require the creation and management of thousands of DPVs (one per partitioned table) – a Herculean task. And, since DPVs cannot support global unique indexes on alternate keys, this effort would guarantee serious violations of the integrity of critical business information. Hence, existing OLTP applications cannot be ported to run on federated databases.

As noted earlier, applications developed for Oracle9i on an SMP require no porting effort to run on Oracle9i Real Application Clusters.

Developing New OLTP Applications

Federated database configurations lack several other features that are essential for developing any OLTP application, including brand-new applications that

Real-world OLTP applications cannot reasonably be ported to run on federated databases.

might try to work around the lack of global unique alternate-keys indexes. Single-node SQL Server provides the following features, which are not supported when using DPVs across multiple-node federated databases:

- No unique columns across the federation, other than the primary key
- Triggers
- Check constraints
- DEFAULT, TIMESTAMP, IDENTITY attributes
- Referential integrity constraints

These features are essential to OLTP applications, and all major database vendors provide them in their SMP offerings. Oracle9i Real Application Clusters, however, also supports these features on shared-disk architectures.

Thus, federated databases are unsuitable for developing OLTP applications, while shared-disk architectures are ideal for OLTP development.

SCALABILITY

Scalability is the primary driver for a customer to move from an SMP platform to a multi-node environment, such as a federated or shared-disk architecture. However, there are inherent barriers to the scalability of a federated database. For instance, queries on DPVs in federated SQL Server will scale up only if the partitioning key is included in the query predicate. Queries will actually provide slower performance in a federated database if the partitioning key is not part of the query predicate.

Consider a 10-node federated database with a DPV on the *Employee* table with columns (Employee Number Primary Key, Email, ...). A basic OLTP query on an alternate key, such as *SELECT * FROM Employee WHERE EMAIL = 'joe.smith@acme.com'* will be broadcast to all 10 nodes, although it will return only one record. As you add nodes (go from 10 nodes to 12, say) this query will perform worse, since it will now require responses from 12 nodes instead of 10.

This behavior of federated databases is not a function of the quality of implementation; it is inherent in the architecture.

In contrast, even early 90's versions of shared-disk clusters (such as IBM DB2-Sysplex for the Mainframe and Oracle Parallel Server) scaled up very well on alternate-key access for read-mostly workloads. And Oracle9i Real Application Clusters scales up very well on alternate-key access for the read-write workloads typical of OLTP. Clearly, shared-disk cluster scaling is already very good, and is constantly improving as vendors incrementally tune their implementation of shared-cache technology. There are no architectural roadblocks to scaling up shared-disk cluster databases.

Federated databases cannot scale when queries access data by alternate keys that are not the partitioning key

Performance characterizations of real-world applications, like SAP, have revealed that the application scales nearly linearly (90%) as the number of nodes in the RAC cluster grows from 2 to 4 to 8.³

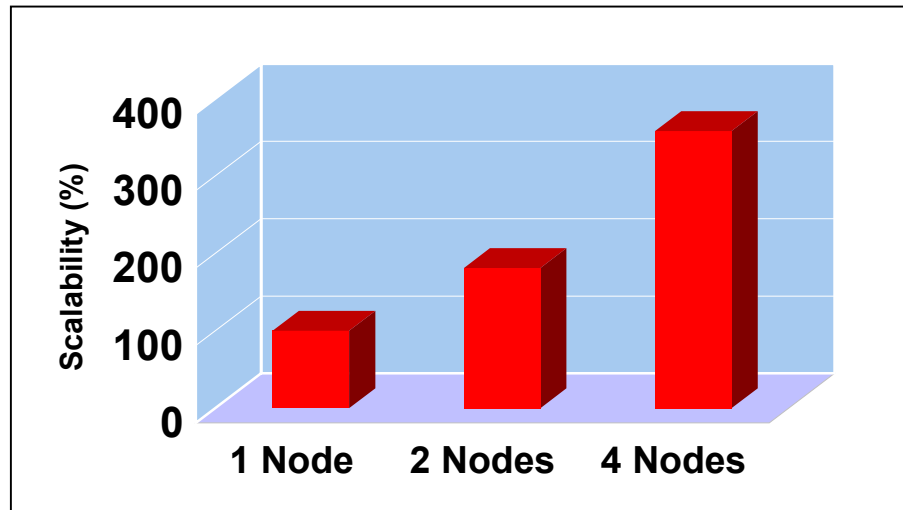


Figure 3: Near-linear Scalability of SAP on RAC using SAP SD 3-tier benchmark

AVAILABILITY

Data in federated databases is divided across databases – and each database is owned by a different node. The only way to access data owned by a node is to request it from the node and have it service the request. Thus, when the node fails, the data it owns becomes unavailable and the entire system becomes unavailable as well. And any in-flight distributed transactions controlled by that node might have locked data on other nodes, so recovering from node failures requires additional work in resolving those in-flight transactions. Recovery involves a series of manual steps that require the system to be offline. Since **each node is a single point of failure for the entire system**, Microsoft's recommendation is to have a failover node for each participating node, thus doubling the hardware requirements, as well as the complexity and cost of the configuration. Each node and its secondary failover node would have to run Microsoft Cluster Service (MSCS). While MSCS provides hardware and O/S clustering, this is not database clustering. In fact, MSCS in Windows 2000 Advanced Server can support only two nodes, and in Windows 2000 Datacenter Server, up to four nodes.⁴ For availability, each primary node might be clustered with a secondary, but the federation of databases, consisting of independent pairs of nodes, is not a cluster.

³ Performance test in Dec 2001 by Compaq and Oracle on Tru64 UNIX ES45 AlphaServer clusters, Oracle9i RAC, SAP® R/3® 4.6C. Each database server node had 4 Alpha CPUs, 1GHz each, with 8MB L2 cache

⁴ Gartner Group, Product Report DPRO-90486, February 7, 2002

On the other hand, when a node in a shared-disk cluster fails, all data remains accessible to the other nodes. In-flight transactions spanning nodes are rolled back, so no data remains locked as a result of the failure. In Oracle9i Real Application Clusters, recovery after node failure is automatic. After detecting node failure, the cluster is automatically reconfigured and the same roll-forward/roll-back recovery processes that work in the SMP environment are applied. For applications using Transparent Application Failover functionality provided by the Oracle Call Interface (OCI), users on the failed node are failed over to another node in the cluster and can continue to work without interruption.

After days or weeks of uptime, each instance in a high-end multi-node database will have a large database block buffer cache. For example, Oracle and HP published a TPC-C benchmark result in November, 2001 with over 100GB of database block buffer cache. With Cache Fusion™ technology in Oracle9i Real Application Clusters, many data blocks will be resident in the caches of more than one node in a cluster. This is because OLTP data cannot reasonably be partitioned by node and there will be a significant overlap in data block accesses. Hence, when a node recovers after failure, it doesn't have to refresh its cache from disk for every data access. Instead, it can get data blocks served from the caches of its peers in the cluster. This significantly reduces the I/O overhead when recovering from node failure.

In contrast, a node in a federated database starts out with an empty buffer cache when recovering from failure. Thus, the user sees slower response times due to increased reads from disk after recovery. This greatly increases recovery time when viewed from the perspective of a user – who only cares about when the application response time returns to normal, and not when some unknown process in the database kernel is deemed to have started up.

MANAGEABILITY

The manageability of an RDBMS is proportional to the number of entities to be defined and maintained. When moving from a single-instance (SMP) database to either a federated or shared-disk multi-instance architecture, you incur the management overhead of initializing and managing instance-specific configuration parameters. Thus, moving from single instance of Oracle9i to multi-node Real Application Clusters requires you to maintain:

- Instance-specific configuration and tuning parameters (*init* parameters). Most of these parameters are likely to be identical across nodes
- Recovery logs per instance

Similar instance-specific entities have to be maintained on each node of a federated database. However, Real Application Clusters do not require any other significant maintenance tasks that grow with the number of nodes in the cluster.

In addition, in a federated configuration every database will require separate backup and recovery, tuning, security, user management, space management, etc. The thousands of tables and indexes typical of real-world complex OLTP applications will also have to be split across the participating federated nodes – managing each object will require tremendous extra work **per node**.

What Happens When You Add a Node?

DPVs present a logical image of data distributed across each node of a federated SQL Server database. Adding a node to a federated database changes the partitioning of data, and results in the need to change the implementation of the DPVs. In order to add a node to a federated SQL Server database, a DBA or Sysadmin has to:

- Add hardware
- Configure new instance (set instance-specific parameters, etc.)
- Create new database
- Disconnect all users
- Unload data from existing tables
- Re-define partitioned tables and indexes
- Re-define triggers on partitioned or replicated tables
- Re-define DPVs
- Reload the data to spread it across a larger number of partitions
- Reconnect all users

This is a significant set of management tasks, and will require that all the databases be offline for the duration of the whole operation.

Consider, on the other hand, the management tasks needed when you add a node to Oracle9i Real Application Clusters:

- Add hardware
- Configure new instance (set instance-specific parameters, etc.)

And that's it! No data unload and reload is required, and no maintenance tasks that take the existing nodes offline- just a seamless scale-up.

Thus we see that shared-disk clusters like Oracle9i Real Application Clusters are much easier to manage than federated databases like Microsoft SQL Server.

TPC-C BENCHMARKS ON FEDERATED DATABASES

The TPC-C benchmark (currently, version 5.0) is widely used as an indicator of the scalability of OLTP databases and associated hardware. The benchmark models part of an order-entry application for a supplier of discrete products. The

Adding a node to a shared-disk cluster is seamless when using Oracle 9i Real Application Clusters.

benchmark measures New Order transactions/minute (tpmC) while enforcing a mix of specific percentages of Payment, Order Status, Delivery and Stock Level transactions. Results are reported on the Transaction Processing Council (TPC) web site (<http://www.tpc.org>) under two categories: Cluster and Non-Cluster results.

The TPC considers TPC-C benchmarks on federated databases to be clustered results. By amassing huge numbers of CPUs in loosely coupled configurations, various hardware vendors have obtained very large TPC-C numbers on federated Microsoft SQL Server databases. In prior sections of this paper, we have demonstrated that federated databases are unsuitable for mainstream OLTP applications. Why, then, are the highest TPC-C benchmark results on federated Microsoft SQL Server databases?

The answer is that TPC-C, being a synthetic benchmark (rather than a workload from a real customer) has certain peculiarities that have been exploited to obtain good results in a federated architecture. These attributes of TPC-C, which are not representative of real-world OLTP applications, are:

TPC-C Schema Is Inherently Partitionable

The TPC-C schema consists of only 9 tables, unlike real-world applications, and 7 out of these 9 tables have Warehouse_ID as part of their primary key.

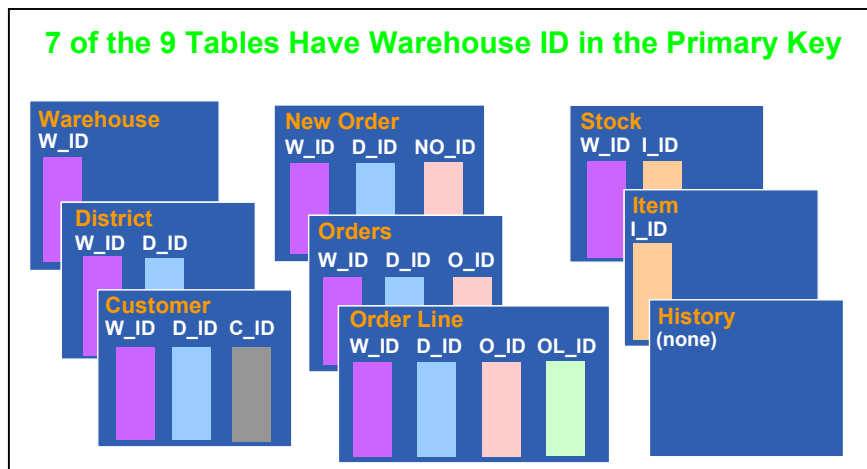


Figure 4: TPC-C Schema

Most TPC-C SQL Accesses are Local

We have already noted that the TPC-C schema can be easily distributed using Warehouse_ID as the partitioning key. With this partitioning, the vast majority (over 99%) of data accesses are local (node running a query or update only accesses data local to that node to service the request):

TPC-C benchmark results obtained on federated databases lack relevance to performance on real-world OLTP applications.

Transaction Type	% of Mix	% of "Local" SQL
New Order	45	99.5
Payment	43	95
Order Status	4	100
Delivery	4	100
Stock Level	4	100

Table 2: Local data access in partitioned TPC-C schema

Thus, a TPC-C benchmark implementation distributed across a federated database is nothing more than a number of independent databases running mostly local transactions. As explained in previous sections of this paper, this situation cannot be replicated in real-world applications; neither the data nor the transactions can be shoehorned to fit neatly into well-defined partitions.

The TPC-C benchmark also does not require any alternate key data access, and as noted earlier, (see section on OLTP Applications) that real-world applications will always require alternate-key access to data.

The consumer of benchmark data - the database customer- should be aware of these idiosyncrasies of the TPC-C benchmark before using "clustered" benchmark results. Results obtained on federated databases have little, if any, relevance to database performance for real-world OLTP applications.

CONCLUSION

This paper compared Federated Databases (as represented by Microsoft SQL Server) with Shared-Disk Cluster Databases (as represented by Oracle9i Real Application Clusters). This paper explained why federated databases are poor candidates for use in deploying real-world OLTP applications, while shared-disk clusters support OLTP applications very well. In addition, federated databases are deficient in the following areas:

- Application Development
- Scalability
- Availability
- Manageability

Finally, this paper explained why TPC-C benchmark results obtained using federated databases should not be a factor in evaluating database performance in the OLTP arena.



Database Architecture: Federated vs. Clustered

March 2002

Author: Vineet Buch

Contributing Authors: Sandra Cheevers

Oracle Corporation

World Headquarters

500 Oracle Parkway

Redwood Shores, CA 94065

U.S.A.

Worldwide Inquiries:

Phone: +1.650.506.7000

Fax: +1.650.506.7200

www.oracle.com

**Oracle Corporation provides the software
that powers the internet.**

**Oracle is a registered trademark of Oracle Corporation. Various
product and service names referenced herein may be trademarks
of Oracle Corporation. All other product and service names
mentioned may be trademarks of their respective owners.**

Copyright © 2002 Oracle Corporation

All rights reserved.