Maximum Availability Architecture trifft auf Oracle Multitenant

Ludovico Caldara, Trivadis AG

Dieser Artikel zeigt die Funktionsweise der Pluggable Databases in einer Maximum Availability Architecture sowie deren Aufbau und Services.

Oracle hat mit der Datenbank 12*c* die Multitenant-Option herausgebracht. Sie galt von Anfang an als „die" Lösung für die Konsolidierung und den Aufbau privater Cloud-Lösungen. Die Konsolidierung vieler Pluggable Databases in eigenständige Instanzen ist jedoch nur beschränkt möglich. Dies betrifft vor allem die Skalierbarkeit des Servers: Sobald die Oracle-CDB-Instanz die gesamten Host-Ressourcen in Anspruch nimmt, muss die Konsolidierung neu erstellter Container Databases (CDB) auf verschiedenen Servern erfolgen *(siehe Abbildung 1)*.

Diese Art der Konsolidierung kann bald zu einem Überladen der CDB führen, sodass Kunden ihre Pluggable Databases (PDB) in unterschiedliche CDBs verschieben müssen, was zu einem steigenden Ressourcen-Verbrauch führt *(siehe Abbildung 2)*.

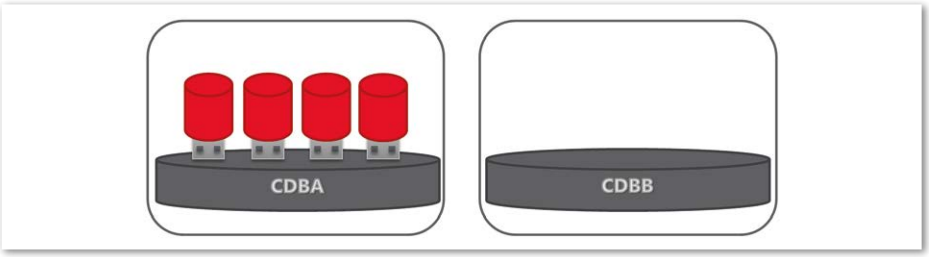

Abbildung 1: Konsolidierung neu erstellter CDBs auf verschiedenen Servern

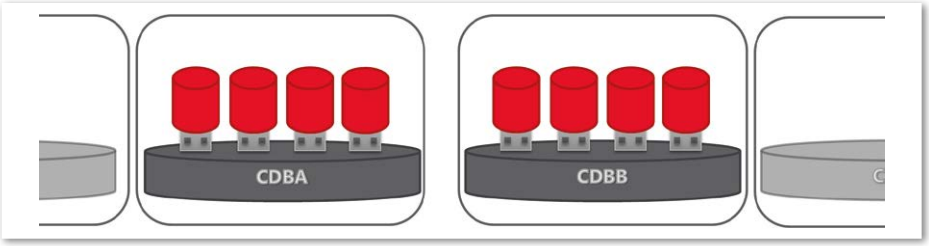

Abbildung 2: Steigender Ressourcen-Verbrauch

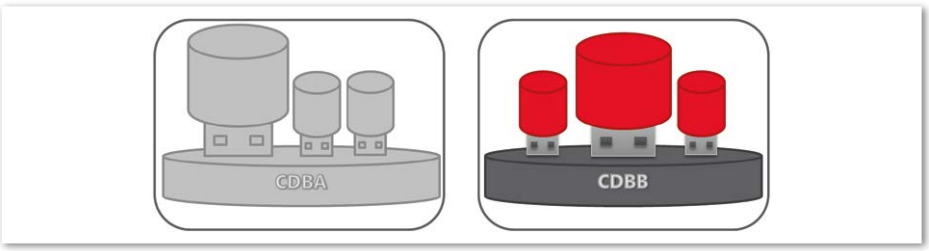

Abbildung 3: Unterschiedliche Wartungsfenster

Zudem entstehen Probleme bei der Verfügbarkeit: Sobald Kunden statische Parameter ändern oder die Soft- beziehungsweise Hardware warten oder patchen möchten, können die vielen auf einer CDB konsolidierten PDBs die Planung einer Ausfallzeit erschweren, weil die verschiedenen Anwendungen unterschiedliche Wartungsfenster benötigen *(siehe Abbildung 3)*.

Kurz gesagt, die Multitenant-Implementierung auf eigenständigen Instanzen löst das Silo-Paradigma nicht auf; sie verlagert lediglich das Problem von der Datenbank- auf CDB-Ebene.

Oracle Real Application Clusters kann als Backend-Lösung den Unterschied ausmachen, da sie die Verfügbarkeit und Skalierbarkeit bietet. Um die Verfügbarkeit sicherzustellen, läuft die CDB auf mehreren Instanzen; der Ausfall einer Instanz wirkt sich nicht auf die Anwendungen aus *(siehe Abbildung 4)*. Hinsichtlich der Skalierbarkeit ist es möglich, neue Instanzen hinzuzufügen, um weitere PDBs zu aktivieren, sobald neue Ressourcen erforderlich sind *(siehe Abbildung 5)*.

Die vielen PDBs konkurrieren nicht um die Ressourcen, weil jede PDB gezielt auf nur eine Teilmenge der Instanzen geöffnet werden kann, was den Ressourcen-Verbrauch auf alle Instanzen verteilt. Welche Instanzen auf welchem Server geöffnet sind, hängt von den Services ab, die auf Cluster-Ebene definiert sind.

Der Umgang mit PDBs und Services

Die PDBs werden nach dem Erstellen standardmäßig bereitgestellt. Im folgenden Beispiel hat eine PDB mit dem Namen „MAAZ" auf beiden Instanzen im Zwei-Knoten-RAC „CDB" den Status „MOUNT" *(siehe Listing 1)*. Wenn aber ein Service für

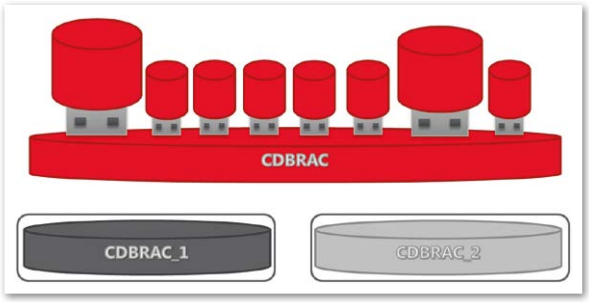

Abbildung 4: Der Ausfall einer Instanz wirkt sich nicht auf die Anwendungen aus

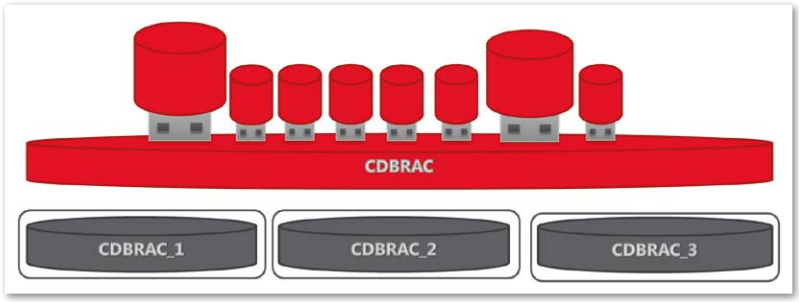

Abbildung 5: Neue Instanzen hinzufügen

die PDBs existiert, öffnet sich beim Start des Service die PDB *(siehe Listing 2)*.

Es ist zu sehen, dass die „Srvctl"-Befehle einen neuen Schalter „-Pdb" haben, der angibt, welcher PDB der neu erstellte Service zugewiesen wird. In diesem Fall wurde der Service als „Singleton" definiert: Er läuft nur in einer Instanz, deshalb wird die PDB nur für diese Instanz geöffnet *(siehe Listing 3)*.

Auf der anderen Seite schließt das Beenden des Service die PDB nicht: Alle Sitzungen bleiben verbunden, auch wenn der Dienst beendet wird. DBAs müssen die Pluggable Database manuell beenden, wenn sie die Instanz bereinigen möchten. Es ist möglich, PDBs selektiv für eine bestimmte Anzahl von Instanzen zu öffnen, indem man verschiedene Services mit unterschiedlichen Größen öffnet. Dies ermöglicht eine vollständige Trennung der Ressourcen sowie eine deutlich höhere Skalierbarkeit, gepaart mit Multitenant, da jede Instanz über einen eigenen Redo-Thread, Undo-Tablespace und vor allem einen eigenen Buffer-Cache verfügt: Der Buffer-Cache einer Instanz wird nur mit Blöcken gefüllt, die zu den PDBs dieser Instanz gehören *(siehe Abbildung 6)*.

Diese Trennung der Ressourcen zeigt, dass RAC die wichtigen Technologie-Voraussetzungen schafft, die nicht nur eine höhere Verfügbarkeit der Multitenant-Architektur bieten, sondern auch die Skalierbarkeit steigern: Bei großen Ressourcen-Anforderungen ist es möglich, dem Cluster einen oder mehrere Knoten hinzuzufügen und die Dienste/PDBs auf die neu erstellten Instanzen auszugleichen.

Die wichtigste Einschränkung ist die maximale Anzahl von Diensten: Es können nicht mehr als 512 Services auf einem einzigen CDB erstellt werden; danach ist eine neue CDB erforderlich. Die Anzahl der Services ist ein wichtiger Faktor für eine erfolgreiche Konsolidierungsstrategie in großen Umgebungen.

Ein weiterer wichtiger Punkt ist, dass, wenn ein Knoten abstürzt, bevor die Sessions umschalten können, die PDB zu öffnen ist: Wenn der Service „Singleton" ist und keine weiteren Instanzen die PDB geöffnet haben, müssen die Sessions warten, bis eine neue Instanz bereitsteht. Das Öffnen kritischer PDBs auf mehr als einer Instanz kann dieses Problem lösen.

RAC, Data Guard und Multitenant

Oracle Multitenant bringt in einer Oracle-Data-Guard-Umgebung einen wesentlichen Vorteil: Da die Redo-Threads über alle PDBs verteilt sind, ist nur ein Standby-CDB erforderlich, in der alle PDBs repliziert werden. Damit ist auch nur eine Data-Guard-Konfiguration notwendig, was zu einer einfachen Wartung und zu einem enorm reduzierten administrativen Aufwand führt *(siehe Abbildung 7)*.

Der Nebeneffekt dieser Architektur ist, dass die Granularität der Switchover- und Failover-Operationen auf CDB-Ebene erfolgt, was bedeutet, dass alle PDBs in einem CDB die gleiche Rolle wie die CDB innehaben; sie stehen entweder alle auf „primary" oder alle auf „standby". Aus Sicht der Konsolidierung ist es wichtig auszuwählen, welche PDBs auf welche CDBs konsolidiert werden müssen, sodass später eine gewisse Flexibilität für die kritischsten PDBs erreicht wird.

```
SYS@CDBATL_2> select INST_ID, CON_ID, name, OPEN_MODE
2  from gv$pdbs where con_id!=2 order by name, inst_id;

INST_ID          CON_ID                    NAME       OPEN_MODE
----------    ---------- ------------------------      ----------
1                     3                    MAAZ       MOUNTED
2                     3                    MAAZ       MOUNTED
```

Listing 1

```
$ srvctl add service -db CDBATL -service maazapp -serverpool CDBPOOL
-cardinality
singleton -role primary -failovertype select  -failovermethod basic
-policy
automatic -failoverdelay 2 -failoverretry 180 -pdb maaz
$ srvctl start service -db CDBATL -service maazapp -instance CDBATL_1
```

Listing 2

```
INST_ID          CON_ID                    NAME       OPEN_MODE
----------    ---------- ------------------------      ----------
1                     3                    MAAZ       READ WRITE
2                     3                    MAAZ       MOUNTED
```

Listing 3

Eine PDB erstellen und in einer MAA-Umgebung klonen

Bei der Erstellung einer neuen PDB aus „PDB$SEED“ („Standard“-Einstellung) werden die Datendateien der PDB$SEED-Datenbank kopiert und der neu erstellten PDB zugewiesen. Auf der Stand-by-Seite kopiert die Data-Guard-Technologie die PDB$SEED-Datendateien ebenfalls von der lokalen (Stand-by-)Seed-Datenbank, um den Erstellungsprozess transparent zu machen und dem Recovery-Prozess weiterhin das Wiederherstellen neuer Datendateien zu ermöglichen *(siehe Abbildung 8)*.

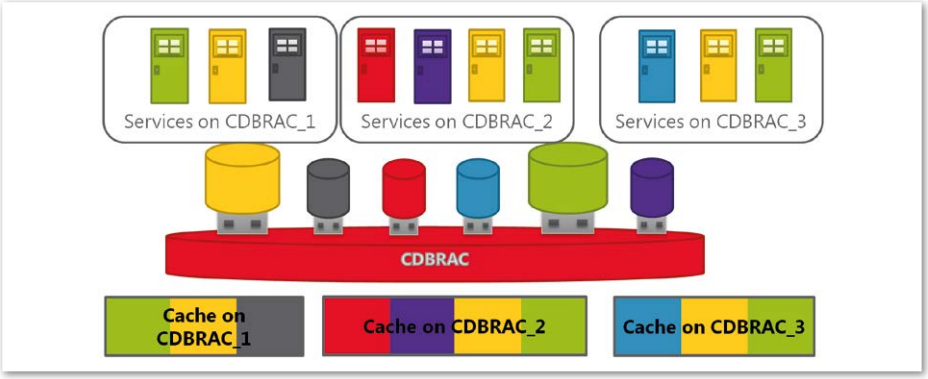

Abbildung 6: Der Buffer-Cache

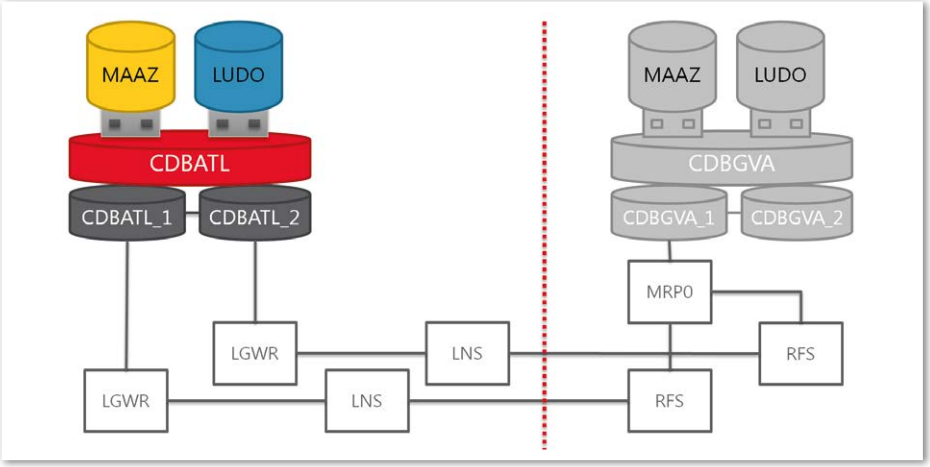

Abbildung 7: Nur ein Stand-by-CDB

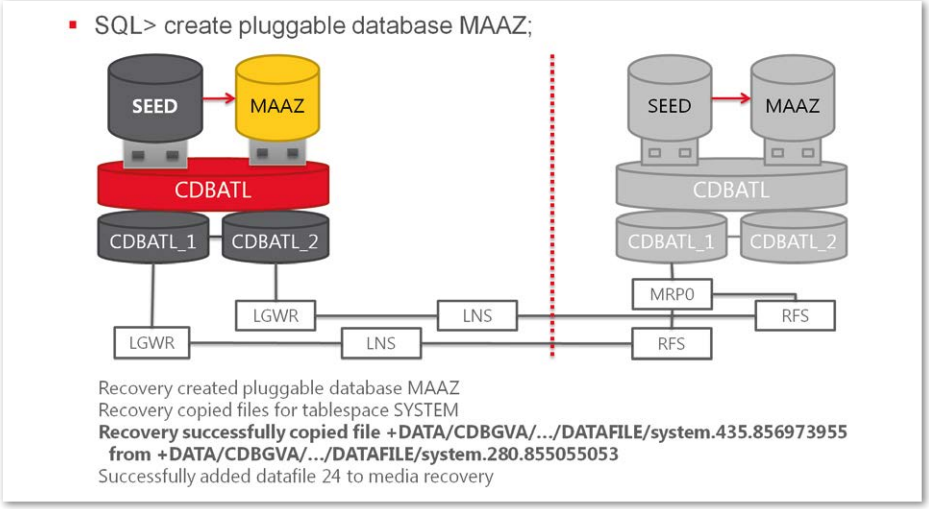

Abbildung 8: Eine neue PDB erstellen

Sobald die PDB durch das Klonen einer bestehenden PDB („create pluggable database A from B;“) erstellt ist, wird die Quelldatenbank nur dann auf beiden Seiten kopiert („primary“ und „stand-by“), wenn die Stand-by-Datenbank schreibgeschützt geöffnet ist, was eine Oracle-Active-Data-Guard-Lizenz erfordert *(siehe Abbildung 9)*.

Wenn Active Data Guard nicht lizenziert ist, sagt die Dokumentation, dass es notwendig ist, die Datendateien vor dem Klonen in die Stand-by-Datenbank zu kopieren *(siehe „http://docs.oracle.com/database/121/SBY-DB/create_ps.htm#SBYDB5260“)*: „If you plan to create a PDB as a clone from a different PDB, then copy the data files that belong to the source PDB over to the standby database. This step is not necessary in an Active Data Guard environment because the data files are copied automatically when the PDB is created on the standby database.“

In einer RAC-Umgebung werden die Datendateien jedoch von ASM verwaltet und es gibt keine Möglichkeit, diese im gleichen Pfad zu kopieren, der in den Controlfiles angegeben ist. Deshalb ist nach dem Klonen eine manuelle Wiederherstellung der neuen PDB auf der Stand-by-CDB erforderlich *(siehe Abbildung 10)*.

Oracle hat dazu eine MOS-Note veröffentlicht: „Making Use of the STANDBYS=NONE Feature with Oracle Multitenant“ (Doc ID 1916648.1). Sie beschreibt den besten Weg für eine PBD-Erstellung in einer Data-Guard-Umgebung ab Release 12.1.0.2.

Der Umgang mit PDBs und Services

In einer Data-Guard-Umgebung hat sich die Möglichkeit, neue Services zu erstellen, nicht geändert. Wenn man einen Read-only-Service auf der Stand-by-Datenbank (Active Data Guard) erstellt, ist nur die entsprechende Rolle („physical_standby“) anzugeben. Der einzige Unterschied besteht darin, die richtige PDB mit dem „-pdb“-Schalter anzugeben *(siehe Listing 4)*. Man sollte nochmal im Auge behalten, dass es ein Gesamtlimit von 512 Services in einem CDB gibt. In einer Active-Data-Guard- und Multitenant-Umgebung gibt es für jede PDB:

- Einen Standard-Service mit dem Namen der PDB
- Einen Read-Write-Service (mehr als empfohlen)
- Einen Read-only-Service für die Stand-by-Datenbank

Die maximale Anzahl von PDBs pro CDB sind dann 512/3 =170 PDBs, also weit unter der Grenze von 252 PDBs pro CDB. Je mehr Services, die man pro PDB erstellt, desto weniger PDBs kann man in der gleichen CDB konsolidieren. Die „connection description" hat sich gegenüber dem herkömmlichen MAA- Connection-String nicht geändert, außer dass der „SERVICE_NAME" auf den in der ausgewählten PDB zugewiesenen Service zeigen muss *(siehe Listing 5)*. Aus Sicht der Konsolidierung ist ein hochverfügbares Oracle Internet Directory (OID) zur Auflösung der Namen dringend angeraten.

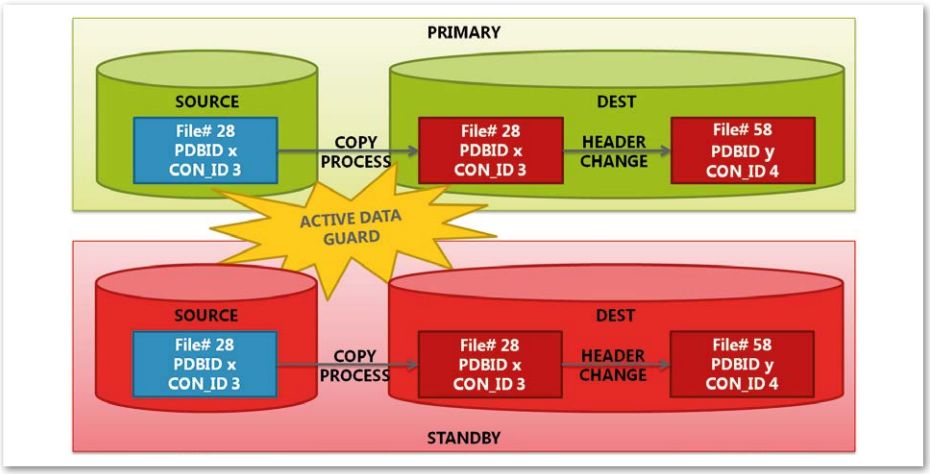

Abbildung 9: Oracle-Active-Data-Guard-Lizenz erforderlich

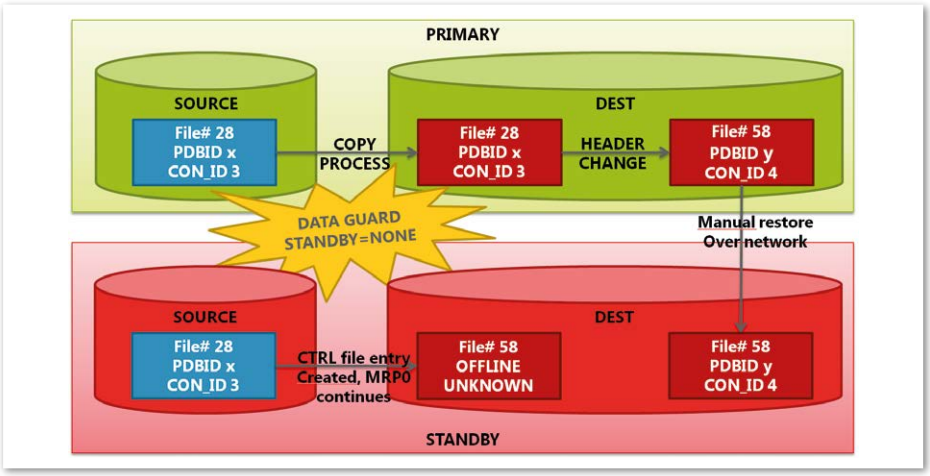

Abbildung 10: Manuelle Wiederherstellung der neuen PDB

```
$ srvctl add service -db db_unique_name-service ro_service_name \
-serverpool server_pool -cardinality uniform -role physical_standby \
-failovertype select -failovermethod basic -policy automatic \
-failoverdelay 2 -failoverretry 180 -pdb pluggable_database
```

Listing 4

```
LUDOAPP = (DESCRIPTION_LIST= (LOAD_BALANCE=off) (FAILOVER=on) (DESCRIP-
TION = (CONNECT_TIMEOUT=5)
(TRANSPORT_CONNECT_TIMEOUT=3) (RETRY_COUNT=3) (ADDRESS_LIST=(LOAD_
BALANCE=on)
(ADDRESS = (PROTOCOL = TCP)(HOST = raca-scan)(PORT = 1521))) (CONNECT_
DATA = (SERVICE_NAME = LUDOAPP))    )
(DESCRIPTION =    (CONNECT_TIMEOUT=5) (TRANSPORT_CONNECT_TIMEOUT=3)
                  (RETRY_COUNT=3) (ADDRESS_LIST=
                  (LOAD_BALANCE=on)
(ADDRESS = (PROTOCOL = TCP)(HOST = racb-scan)(PORT = 1521))) (CONNECT_
DATA = (SERVICE_NAME = LUDOAPP)) )  )
```

Listing 5

Fazit

Active Data Guard ist eine zunehmend attraktive Option geworden und trotz der Beschränkungen beim PDB-Klonen auf nicht aktiven Stand-bys kein Nice-to-have-Produkt. Active Data Guard ist die beste Lösung für eine Data-Guard- und Multitenant-Beschränkung. Die Konsolidierung mit Multitenant erfordert im Idealfall alle drei Optionen (Multitenant, Real Application Cluster, Active Data Guard), die für Kunden trotz der vielfältigen Vorteile nicht leicht erschwinglich sind.

Aus Sicht der Konsolidierung ist die MAA- und Multitenant-Architektur die beste Wahl in Richtung einer Cloud-Infrastruktur, denn sie erfüllt alle Anforderungen in Bezug auf Konsolidierungsdichte, Skalierbarkeit, Verfügbarkeit und einfache Administration; deshalb würde der Autor nicht zögern, sie zu empfehlen. Zudem hat Oracle kürzlich die Freigabe der Nicht-CDB-Architektur angekündigt: Ein Weg, der Kunden die neue Architektur zur sicheren Wahl macht.

Hinweis: Ins Deutsche übersetzt von global syntax Language Management Services. Die englische Fassung des Artikels kann unter „*www.doag.org/go/News/201503/Caldara*" heruntergeladen werden.

Ludovico Caldara
Ludovico.Caldara@trivadis.com