White-Paper: Datensicherheit mit SQL Anywhere®

Änderungsstand: 2014-07-28
Autor: Markus KARG (karg@quipsy.de)

Zusammenfassung

Dieses White-Paper erläutert Maßnahmen zur Erhöhung der Datensicherung bei Nutzung des relationalen Datenbank-Management-Systems (RDBMS) "Sybase SQL Anywhere®" der SAP AG.

Gilt für

- QUIPSY® CAQ, CPL, APQP, PMV, EMP, FMEA, RB 4.35
- QUIPSY® DMS und Audit bei Nutzung des SQL Anywhere®-Backends
- Sybase SQL Anywhere® 11.0.1

Einführung

Mit zunehmendem Umfang und Relevanz der durch die QUIPSY®-Software verarbeiteten Informationen steigt der Bedarf, administrative Maßnahmen zur Erhöhung der Datensicherheit zu ergreifen.

Im Folgenden sind beispielhafte Sicherungsmaßnahmen beschrieben, welche geeignet und von der QUIPSY QUALITY GmbH & Co. KG empfohlen sind, wenn SQL Anywhere® 11.0.1 zum Einsatz kommt.

Über dieses Whitepaper hinausgehend sind im Handbuch dieses RDBMS weitere Informationen zur Erhöhung der Datensicherheit beschrieben (siehe http://dcx.sybase.com/index.html#1100/en/dbadmin_en11/dbadmin_en11.html).

Dieses Whitepaper verbindet fachspezifisches Allgemeinwissen über Datenbankadministration mit den speziellen Belangen der Produkte SQL Anywhere® und QUIPSY® CAQ. Es ist grundsätzlich jedoch auch geeignet, um mit anderen Produkten angewendet zu werden.

Hintergrund

Die QUIPSY®-Software nutzt eine Datenbank, welche auf dem Sybase SQL Anywhere® Server in typischerweise zwei, gelegentlich auch mehr, Dateien vorliegt. Diese Dateien werden nicht direkt von der QUIPSY®-Software angesprochen, sondern stehen unter alleiniger Kontrolle des Datenbank-Serverdienstes (`dbsrv11.exe`), mit welcher die QUIPSY®-Software kommuniziert.

Theorie

Das Anhängen an bestehende Dateien geht sehr schnell, das Ändern in der Mitte von Dateien benötigt dagegen relativ viel Zeit. Umgekehrt ist das Auffinden einer bestimmten Information in einer strukturierten Datei ("Archiv") sehr schnell, während das Suchen der gleichen Information in einer chronologisch geordneten Datei ("Journal") sehr langsam ist. SQL Anywhere® kombiniert, wie die meisten relationalen Datenbank-Management-Systeme, daher beide Verfahren (sog. "Write Ahead Logging" bzw. "WAL") und gewährleistet darüber hinaus durch diese doppelte Buchführung in sich (d. h. ohne weitere technische Hilfsmittel) bereits eine einfache Redundanz der gespeicherten Informationen.

Der Serverdienst hängt gewünschte Änderungen daher zunächst an ein Journal an. Wenn das

Betriebssystem bestätigt, dass dieses Journal auf der Festplatte festgeschrieben ist, meldet der Serverdienst an die QUIPSY®-Software, dass die Änderungen erfolgreich waren - zunächst ohne das Archiv zu aktualisieren.

Hierdurch wird sichergestellt, dass nach einem Absturz und Neuanlaufen anhand des Journals das Archiv wieder den exakt gleichen Zustand hat wie vor dem Absturz und somit die Änderungen per Vorwärts-Wiederherstellung ("Redo") eingespielt werden können , das Archiv jedoch sicherheitshalber zunächst "unberührt" bleibt, falls der Absturz das Journal zerstören sollte.

Üblicherweise wird ein Journal zur Reduktion von Wartzeiten auf einem für hohe Schreibgeschwindigkeit optimierten Volume abgelegt.

Nach einer statistisch ermittelten Anzahl an Änderungen nimmt der Serverdienst zu einem wesentlich späteren Zeitpunkt "im Hintergrund", d. h. ohne dass die QUIPSY®-Software darauf warten muss, einen sogenannten "Checkpoint" vor (sog. "Checkpointing"). Hierbei werden alle seit dem letzten Checkpoint im Journal angesammelten Informationen in einem Datenbank-Archiv auf den neuesten Stand gebracht. Sobald das Betriebssystem bestätigt, dass das Archiv auf der Festplatte festgeschrieben ist, vermerkt es diesen Zeitpunkt im Journal als letzten durchgeführten Checkpoint, womit das Journal logisch gesehen von vorne beginnt. Physisch wird das Journal im Normalfall jedoch nie von vorne angefangen.

Hierdurch wird sichergestellt, dass die "eigentliche" Datenbank (also das Archiv) regelmäßig einen konsistenten Stand hat, somit also alle Daten aus Sicht des Datenbank-Serverdienstes "sicher" sind.

Üblicherweise wird ein Archiv zur Reduktion von Wartezeiten auf einem für hohe Lesegeschwindigkeit optimieren Volume abgelegt.

Aus Sicht der optimalen Geschwindigkeit ist es sinnvoll, Journal und Archiv auf getrennten Volumes abzulegen, welche verschiedenartige Performance-Optimierungen aufweisen. Die hierdurch zwangsläufig entstehende Redundanz (Verteilung auf getrennte Medien) stellt eine grundlegende Sicherheit der Daten her, welche mit RAID1 vergleichbar ist, jedoch eine geringere Verfügbarkeit (sog. "Availability") aufweist, da diese im Fehlerfall einen Neustart des Dienstes erfordert.

Dateien

QUIPSY.log

Die Datei `QUIPSY.log` enthält das Journal der Datenbank und sollte sich auf einem für hohe Schreibgeschwindigkeit optimierten Volume befinden. Die logische(*) Position der Datei ist in binärer Form in der Datei `QUIPSY.db` gespeichert und kann über das Werkzeug `dblog` verändert werden.

QUIPSY.db

Die Datei `QUIPSY.db` enthält das Archiv der Datenbank und sollte sich auf einem für hohe Lesegeschwindigkeit optimierten Medium befinden. Die logische(*) Position der Datei wird dem Serverdienst über einen Kommandozeilen-Parameter mitgeteilt und kann über das Werkzeug `dbsvc` verändert werden.

QUIPSY.mlg (opt.)

Die Datei `QUIPSY.mlg`, sofern genutzt, enthält eine Kopie des Journals, welche zeitgleich mit dem Original-Journal geschrieben wird. Die logische(*) Position der Datei ist in binärer Form in der Datei `QUIPSY.db` gespeichert und kann über das Werkzeug `dblog` verändert werden.

Temporäre Dateien

Temporäre Dateien werden vom Datenbankserver ausschließlich während der Laufzeit einer Transaktion verwendet und sind daher aus Sicht der Datensicherheit irrelevant. Sie sollten auf einem Medium liegen,

welches höchstmögliche Schreibgeschwindigkeit aufweist. Die Position von temporären Dateien wird über die Umgebungsvariable `SATMP` festgelegt. Sofern diese nicht vorhanden ist, wird das allgemeine Temporär-Verzeichnis des Betriebssystems genutzt (Umgebungsvariable `TEMP`).

(*)Tip: Nicht die logische Position der Dateien ändern (`dblog` / `dbsvc`), sondern statt dessen die physische. Hierzu die Dateien zunächst physisch verschieben (`MOVE`) und danach über `MKLINK` auf Windows® bzw. `link` auf Linux® einen neuen Eintrag im ursprünglichen Directory erstellen. Dies erleichtert die Übersichtlichkeit über die logische Zusammengehörigkeit der Dateien erheblich, da sie logisch gesehen weiterhin im gleichen Ordner liegen, physisch jedoch auf getrennten Volumens).

Maßnahmen

Zur Erzielung möglichst optimaler Datensicherheit sollten alle folgenden Maßnahmen angewendet werden. Aus Gründen der Wirtschaftlichkeit kann es jedoch angebracht sein, einige Maßnahmen nicht anzuwenden, oder statt dessen gleichwertige Ersatzmaßnahmen anzuwenden.

Einige der genannten Maßnahmen können nur von QUIPSY®-Servicepersonal durchgeführt werden.

Maßnahmen zur Datenrettung (Backup und Recovery)

Die folgenden Maßnahmen *zur Datenrettung* stellen das absolute Minimum dar, ohne welches ein produktiver Betrieb auf keinen Fall stattfinden darf, da auch die Anwendung *aller* genannten Maßnahmen zur Vermeidung von Datenverlust keine *absolute* Sicherheit gewährleistet!

Backup

Hinweis: Zur Anfertigung einer Sicherungskopie dient das Werkzeug `dbbackup`. Zur Validierung eines Archivs (`QUIPSY.db`) dient das Werkzeug `dbvalid`. Diese Werkzeuge können grundsätzlich von der Kommandozeile (CLI), als auch über die grafische Benutzeroberfläche (GUI; `dbisql` oder *Sybase Central*) aufgerufen werden. Ebenso ist es möglich, Sicherung und Validierung über die Befehle `BACKUP DATABASE` und `VALIDATE DATABASE` durch den Serverdienst selbst auszuführen. **Letzteres ist die Standardeinstellung durch den QUIPSY®-Support als *Minimal*-Lösung**).

- **Es MUSS eine tägliche Sicherung des Archivs (`QUIPSY.db`) auf einem portablen Medium (z. B. Band oder DVD-RAM) abgelegt und an einem sicheren und für alle Beteiligten bekannten und im Notfall zugänglichen Ort verwahrt werden. Die Sicherungskopie DARF NICHT auf dem gleichen Server aufbewahrt werden.**

  Hierdurch wird sichergestellt, dass im Notfall lediglich die Änderungen eines einzigen Tages verloren sind und dass ein defekter Server neben der Produktivdatenbank auch die Sicherungskopie zerstört.

- Es SOLLTE, sofern irgend möglich, eine tägliche Sicherung des Journals (`QUIPSY.log`) und (sofern vorhanden) des Journal-Spiegels (`QUIPSY.mlg`) auf dem gleichen portablen Medium wie die Sicherung des Archivs (`QUIPSY.db`) durchgeführt werden. Diese Sicherung MUSS zeitgleich oder Später mit der Sicherung des Archivs erfolgen. Es ist daher sinnvoll, beide Dateien in einem einzigen Durchgang zu sichern.

  Hierdurch wird sichergestellt, dass trotz defekter Sicherungskopie des Archivs ein aktuelles Archiv anhand des Journals erstellt werden kann. Zudem ist der Datenverlust geringer als bei reiner Sicherung des Archivs, da die Differenz zum letzten Checkpoint nicht verloren geht. Ebenso beschleunigt dies im Notfall das Wiederanlaufen, da der Datenbankserver standardmäßig ein zum Archiv passendes Journal erfordert, welches ansonsten manuell erzeugt werden muss, indem der letzte Checkpoint ignoriert wird, wodurch dessen Differenzinformation verloren geht (Option `-f` beim Start des Serverdienstes).

- Eine Kürzung (Beschneidung) des Journals SOLLTE, sofern irgend möglich, unterbunden werden.

- Es SOLLTE eine Validierung der Sicherungskopie des Archivs erfolgen.

  Hierdurch wird sichergestellt, dass die Sicherungskopie im Notfall tatsächlich lesbar und fehlerfrei ist. Ein nicht lesbares Backup ist etwa so nützlich wie ein leerer Feuerlöscher. Es ist technisch durchaus möglich, dass ein Datenbankserver sehr lange mit defekten Dateien arbeitet, ohne einen Fehler zu melden.

  Hinweis: Die Validierung der Sicherheitskopie SOLLTE nicht auf dem Original-Datenbankserver erfolgen.

  Hierdurch wird verhindert, dass ein Fehler im Controller des Servers dazu führt, dass eine defekte Kopie als nicht-defekt gemeldet wird.

- Es KANN (alternativ oder zusätzlich) eine Validierung der produktiven Datenbank erfolgen.

  Hierdurch wird sichergestellt, dass sich versteckte Fehler nicht verschleppen und möglichst sofort festgestellt wird, wenn das Archiv beschädigt ist. Ebenso wird hierdurch erkannt, ob `dbbackup` selbst fehlerhaft ist (Differenz des Validierungsergebnisses).

  ***Hinweis:** Eine Validierung im laufenden Betrieb kann sogenannte "False Postives" zur Folge haben, d. h. Fehler melden, die effektiv nicht vorhanden sind. Daher SOLLTE eine Meldung, dass die laufende Datenbank korrupt ist, unbedingt im Stillstand verifiziert werden.*

- Jeder Beteiligte SOLLTE sich mit der Bedienung und den Optionen der Werkzeuge `dbbackup` und `dbvalid` vertraut machen und regelmäßig (jährlich) seine Kenntnisse auffrischen. Die Auswirkungen der genutzten Optionen SOLLTEN jedem Beteiligten klar und verständlich sein.

Recovery

Hinweis: Bei ordnungsgemäß durchgeführtem Backup und Recovery ist es möglich, einen Störfall mit lediglich minimalem Datenverlust (noch nicht gespeicherte Daten) zu überstehen, und binnen weniger Minuten wieder produktiv zu sein.

- Jeder Beteiligte SOLLTE sich mit dem Ersetzen der defekten Datenbank durch eine Kopie von Archiv und Journal (`QUIPSY.db` und `QUIPSY.log`), dem Ersatz des Journals durch einen Journal-Spiegel (`QUIPSY.mlg`), den Notstart ohne Journal (Option `-f`) sowie dem Zusammenfügen von beschnittenen Journalen vertraut machen, sowie seine Kenntnis regelmäßig (jährlich) auffrischen und die Anwendung der notwendigen Prozeduren anhand von lokalen Kopien üben. Die Auswirkungen der genutzten Optionen SOLLTEN jedem Beteiligten klar und verständlich sein.

  Dies verkürzt erheblich die Zeit bis zur Wiederinbetriebnahme und verhindert versehentliches Zerstören von möglicherweise noch funktionsfähigen Daten.

- Recovery SOLLTE mit Kopien aller Dateien auf einer separaten Maschine erfolgen.

  Dies verhindert, dass versehentlich funktionsfähige Dateien durch unsachgemäße Behandlung ("Experimentieren") beschädigt und somit wertlos werden.

- Eine defekte Datenbank SOLLTE, sofern vorhanden, durch eine vollständige Kopie aller Datenbank-Dateien (Archiv, Journal, opt. Journal-Spiegel) ersetzt werden. Die defekten Dateien sind vorher sicherzustellen.

  Aus dem sichergestellten Journal bzw. Journal-Spiegel lässt sich, sofern nicht beschädigt, der Datenverlust des laufenden Tages vermeiden.

- Sofern vorhanden SOLLTE ein noch funktionsfähiges Journal (opt. Journal-Spiegel) über `dbeng11 -a` angewendet werden.

  Hierdurch wird der Datenverlust des laufendes Tages vermieden, indem die im Journal vorhandene Differenz zum letzten Backup in das Backup-Archiv eingespielt wird.

- Es KANN im Notfall ein Neuanfahren mit einem gesicherten Archiv (`QUIPSY.db`) ohne Journal erfolgen, sofern dies notwendig ist. Hierzu ist zunächst ein leeres Journal mittels `dbeng11 -f QUIPSY.db` anzulegen und gemeinsam mit dem dadurch veränderten Archiv auf den Produktivserver zu kopieren.

  Hinweis: *Die Differenz zur letzten Datensicherung ist somit verloren!*

- Sofern kein unbeschnittenes Journal vorhanden ist, kann ein solches durch Einspielen einer durchgehenden Kette von beschnittenen Journalen mittels `dbeng11 -ad` erstellt werden.

  Hinweis: *Die Kette muss absolut durchgehend sein vom Zeitpunkt des letzten Checkpoints vor dem Backup des Archivs bis zum Zeitpunkt des Störfalls. Beginnt die Kette erst nach dem letzten Checkpoint des gesicherten Archivs können keine Daten aus der Journalkette in das Archiv übernommen werden. Ist ein "Loch" in der Kette, können keine dem "Loch" folgenden Daten in das Archiv übernommen werden.*

Maßnahmen zur Vermeidung von Datenverlust

- Abschalten des Write-Caches

  Der Write Cache MUSS auf allen Ebenen (Betriebssystem, SAN, Controller, HDDs) abgeschaltet werden.

  Sofern dies nicht möglich ist, MUSS alternativ das Verfahren von "Write Back" auf "Write Through" umgeschaltet werden (auf allen Ebenen).

  Sofern dies nicht möglich ist, sind Hardware (SAN, Controller, HDDs) und ggf. Betriebssystem zu ersetzen.

  Die Nutzung des sogenannten "Write Back"-Verfahrens bei Write Caches meldet dem Datenbankserver, dass Daten auf dem Medium unveränderlich festgeschrieben sind, obwohl diese sich erst in einem flüchtigen Zwischenpuffer befinden. Tritt nach diesem Zeitpunkt ein Störfall ein, ist der tatsächliche Datenbestand auf dem Medium in einem inkonsistenten Zustand, und damit zerstört.

  *Liegen Journal und Archiv auf dem selben Volume, und fällt auf diesem Volume beim Schreiben des Archivs der Strom aus, ist das Archiv in der Folge zwingend defekt (teilweise überschriebener Block))! Der Grund ist, dass das Archiv generell partiell überschrieben werden muss (sog. "In-Place Modification"). Datenbankserver versuchen sich dagegen zu schützen, indem sie zuvor die geplante Änderung an das Journal anhängen. Ist jedoch ein "Write Back" Cache aktiv (z. B. als Teil des Betriebssystems, der Festplatte, usw.), ist nicht garantiert, dass das Journal zum Zeitpunkt der Archivänderung physisch auf der Festplatte festgeschrieben ist. In der Folge ist die automatische Fehlerkorrektur des Datenbankservers **unmöglich**, und Datenverlust die zwangsläufige Folge.*

  ### Geräte melden abgeschaltete Caches, obwohl sie weiterhin eingeschaltet sind

  *Einige Geräte verhalten sich irreführend (sei es absichtlich oder als Ergebnis von Fehlern und Defekten), indem sie melden, das "Write Back"-Verfahren nicht zu nutzen, obwohl sie es faktisch **trotzdem** tun. Dies betrifft leider auch teure HDDs und SSDs bekannter Markenhersteller.*

  ### Batterie-Puffer für Caches (sog. "Battery Backup Unit" bzw. "BBU") sind unwirksam

*Batteriepuffer mindern dieses Problem, können jedoch ebenfalls **nicht erkennbare** Defekte aufweisen und wirken generell nur auf **ein** Glied in der Kette (das Controller-RAM), nicht jedoch auf das RAM der Festplatten, des SAN-Heads, oder das Betriebssystem des Hosts -- womit sie faktisch **wirkungslos** sind, sofern an den nicht-gepufferten Stellen "Write Back" (absichtlich oder unabsichtlich) zum Einsatz kommt.*

Unabhängige Stromversorgung (sog. "UPS") sind unwirksam

Jede unabhängige Stromversorgung hat eine begrenzte Überbrückungszeit. Ist diese erreicht, schaltet sie angeschlossene Geräte entsprechend deren Priorität ab (d. h. unwichtige Geräte vor wichtigen Geräten). Hierbei kann es vorkommen, dass ein Plattensystem abgeschaltet wird, während es gerade einen Schreibvorgang durchführt (z. B. weil das Plattensystem die Vorwarnung der UPS nicht erhält oder die Geräteprioritär falsch konfiguriert ist). Hat der Controller bereits Vollzug gemeldet ("Write Back") sind diese Daten trotz UPS verloren.

Write Cache ist für SQL Anywhere®-Performance nicht notwendig

Write Caches mit "Write Back" sind bei SQL Anywhere® zur Steigerung der Schreib-Performance nicht notwendig, wenn das Journal auf Volumes abgelegt wird, die für hohe Schreibleistung optimiert sind.

Write Caches mit "Write Through" sind bei SQL Anywhere® zur Steigerung der Lese-Performance nicht notwendig, wenn das Archiv auf Volumes abgelegt wird, die für hohe Leseleistung optimiert sind und / oder der Datenbankserver über genügend RAM verfügt.

Write Caches sind generell nicht notwendig, wenn Journal und Archiv auf getrennten Festplatten liegen, welche dem Datenbankserver exclusiv zur Verfügung stehen, d. h. nicht von anderen Anwendungen / Diensten oder dem Betriebssystem mitbenutzt werden.

Aufgrund der aktuellen Preise für SSDs und RAM liegt teilweise sogar der Einsatz einer professionellen BBU bei einem schlechteren Preis-/Leistungsverhältnis als der Einsatz eines SSD-RAID1 und genügend Server-RAM!

- Verteilung von Archiv und Journal auf mehrere Medien

  Archiv (`QUIPSY.db`), Journal (`QUIPSY.log`) und Journal-Spiegel (`QUIPSY.mlg`, sofern verwendet) SOLLTEN, sofern irgend möglich, auf getrennten Medien liegen.

  Hierdurch wird sichergestellt, dass bei Ausfall eines Mediums die darauf befindliche Datenbank-Datei aus der jeweils anderen Datenbank-Datei vom noch funktionierenden Medium erstellt werden kann (siehe Abschnitt über Recovery).

  #### RAID ist nicht unbedingt eine Lösung gegen Datenverlust

  Abgesehen von RAID1 (10, 50, 60) (sog. "Mirroring") stellen RAID-Sets keine getrennten Medien dar: Viele RAID-Level, allen voran RAID0, RAID5 und RAID6, verteilen Daten zur Erhöhung der Lesegeschwindigkeit über mehrere (teilweise alle) Festplatten hinweg (sog. "Striping") womit jeder Datensatz der Datenbank von der einwandfreien Funktion aller beteiligten Platten im RAID-Set abhängig ist.

  ***Warnung:** RAID-Systeme sind daher kein Ersatz für die genannte Maßnahme! Hätte das RAID-System einen Defekt, wären Archiv und Journal gleichzeitig betroffen. Zwei wiederholt bekannt gewordene Probleme mit RAID-Systemen sind, dass (1) die Firmware von Controllern Defekte aufgewiesen hat, sodass nach einem Rebuild (Ersatzplatte eingebaut) das Stripe Set (und damit die darauf liegenden Volumes) fehlerhaft waren und (2) die Platten alle exakt gleiche Laufleistung aufweisen und während eines Rebuilds (Ersatzplatte) in Kombination ausfallen, da hierdurch eine erheblich gesteigerte Belastung ansteht - wodurch alle Daten verloren sind. Wird RAID eingesetzt,*

sollte daher unbedingt mit getrennten RAID-Sets pro Datenbank-Datei gearbeitet werden.

Hintergrund: RAID dient primär nicht zum Schutz vor Datenverlust, sondern beschleunigt die Lesegeschwindigkeit (RAID0, RAID1, RAID5, RAID6), erhöht die Verfügbarkeit durch Reduktion der Ausfallzeit (RAID1, RAID5, RAID6) und bietet eine höhere Gesamtkapazität als einzelne Festplatten.

- Verteilung auf mehrere Controller

  Archiv (`QUIPSY.db`), Journal (`QUIPSY.log`) und Journal-Spiegel (`QUIPSY.mlg`, sofern verwendet) SOLLTEN von getrennten Controllern bedient werden.

  Dies vermindert die Wahrscheinlichkeit, dass eine defekte BBU bzw. ein anderer Controller-Defekt alle Datenbank-Dateien gleichzeitig betrifft.

  Optimaler Weise werden Controller von verschiedenen Herstellern verwendet, sofern wirtschaftlich tragbar.

  Hierdurch wird verhindert, dass ein Defekt in der Firmware des Controllers oder ein anderer Controller-spezifischer Konstruktionsfehler sich auf alle Datenbank-Dateien gleichzeitig auswirkt.

  ***Warnung**: Zwei Volumes auf einem SAN-Speicher werden üblicherweise weiterhin vom gleichen Controller bedient. Ein Ausweg sind getrennte SAN-Devices oder Devices mit mehr als einem Controller.*

- Nutzung eines Journal-Spiegels

  Es KANN ein Journal-Spiegel eingesetzt werden.

  Der Einsatz eines Journal-Spiegels verhindert Datenverlust, da im Falle eines defekten Journals der Journal-Spiegel in ein Journal umgewandelt werden kann.

- Serverdienst nicht anhalten

  Sofern nicht zwingend notwendig, SOLLTE der Serverdienst niemals angehalten werden.

  Solange der Serverdienst läuft, sind Archiv, Journal und (sofern eingesetzt) Journal-Spiegel gegen Zugriffe anderer Software geschützt, wodurch die Möglichkeit der absichtlichen oder unabsichtlichen Beschädigung wirkungsvoll verhindert wird. Diese Schutzwirkung entfällt, sobald der Dienst angehalten wird.

  ***Hinweis:** Zur Anfertigung einer Sicherungskopie oder zur Validierung der Datenbank ist das Anhalten des Dienstes nicht notwendig. Ggf. ist die Prozedur zur Sicherung und Validierung anzupassen, siehe Abschnitt über Backup. Ebenso ist es möglich, Backups mittels VSS (siehe SQL Anyhere®- Handbuch unter Stichwort `dbvss11`) im laufenden Betrieb anzufertigen.*

- SAN statt NAS

  Statt eines NAS sollte ein SAN (alternativ: direkt angeschlossene Festplatten) verwendet werden.

  Einige NAS verwenden zur Leistungssteigerung Write Caches ohne BBU, verfügen aber über keine Option, um diese abzuschalten oder auf Write Through umzuschalten. Zudem ist unklar, inwieweit die verschiedenen Server-Betriebssysteme Write Caches bei der Ansteuerung von NAS verwenden, und ob diese abschaltbar oder zumindest auf Write Through einstellbar sind.

- VMware- und SAN-Snapshots vermeiden

  Zur Anfertigung einer Sicherheitskopie SOLLTE kein Snapshot in VMware oder SAN verwendet werden, sondern ausschließlich `dbbackup`.

Der Datenbankserver wird im Normalfall nicht von VMware oder dem SAN-Gerät informiert, dass ein solcher Snapshot stattfindet. Da das Archiv im Normalfall dem Journal um Minuten bis Stunden nacheilt, und während des Snapshots möglicherweise Schreibzugriffe auf Journal und / oder Archiv erfolgen, ist die Gefahr gegeben, dass die Datenbank-Dateien in den Snapshots defekt sind. Sollen Snapshot-basierte Lösungen zum Einsatz kommen, ist unbedingt VSS zu verwenden (siehe SQL Anywhere®-Handbuch unter Stichwort `dbvss11` sowie Handbuch zu VMware und zum SAN-Gerät), oder es ist sicherzustellen, dass direkt vor dem Snapshot per `dbbackup` eine Kopie von Archiv und Journal erstellt werden, welche beim Wiederanlaufen des Snapshots die defekten Dateien ersetzen.

- VMware GSX durch VMware ESXi ersetzen

  VMware GSX DARF NICHT eingesetzt werden.

  VMware GSX enthält einen Write Cache, der nur mit großer Mühe abschaltbar ist und die Leistung erheblich beeinflusst sowie aufgrund möglicher Software-Defekte dieser nicht mehr vom Hersteller gewarteten Software Datenverlust ermöglicht. VMware ESXi hingegen enthält keinen Schreibcache und übertrifft die Leistung von VMware GSX um ein Vielfaches.

Sonstige Maßnahmen

- Einsatz von Warm Stand-by

  SQL Anywhere ® KANN einen Warm Stand-by auf einer separaten Maschine bereitstellen.

  Hierdurch wird das Wiederanlaufen auf einem Ersatzgerät automatisiert und erheblich beschleunigt gegenüber manuellem Recovery.

- Einsatz von Mirror-Servern

  SQL Anywhere® KANN vollständige Spiegel bereitstellen (kostenpflichtige Zusatzoption). Die Umschaltung erfolgt transparent, erfordert jedoch trotzdem einen Neustart des Clients.

  Hierdurch wird Datenausfall nahezu vollständig verhindert und die Ausfallzeit minimiert. Die QUIPSY®-Software ist jedoch nicht mit Mirror-Konfigurationen getestet und es erfolgt daher keine Unterstützung bei der Einrichtung und Wartung von Spiegel-Konfigurationen durch die QUIPSY QUALITY GmbH & Co. KG.

Rechtliche Hinweise