Borland®

Silk Central 16.5

Der Berichts-Data-Mart

Borland Software Corporation
700 King Farm Blvd, Suite 400
Rockville, MD 20850

2015-09-23

Inhalt

Übersicht

Der Silk Central Berichts-Data-Mart ermöglicht den mühelosen Zugriff auf Daten für die Erstellung von Berichten. Er verschiebt die Daten aus den Produktivtabellen in die Ansichten, die für die Erstellung von erweiterten Berichten verwendet werden sollen. Die Vorteile sind:

- Die klare Benennung von Tabellen und Ansichten, was Ihnen ermöglicht, die gesuchten Daten schnell zu finden.
- Vorverarbeitete Daten, was Ihnen die Möglichkeit gibt auf aggregierte Daten zuzugreifen, ohne dass Sie selbst die Berechnungen machen müssen.
- Leistungsverbesserung, da die Berichte viel einfachere und schnellere SQL-Abfragen verwenden können.
- Eine geringere Abhängigkeit von der Last der Produktivdatenbank, was ebenfalls die Leistung verbessert und Belastungsspitzen entfernt.

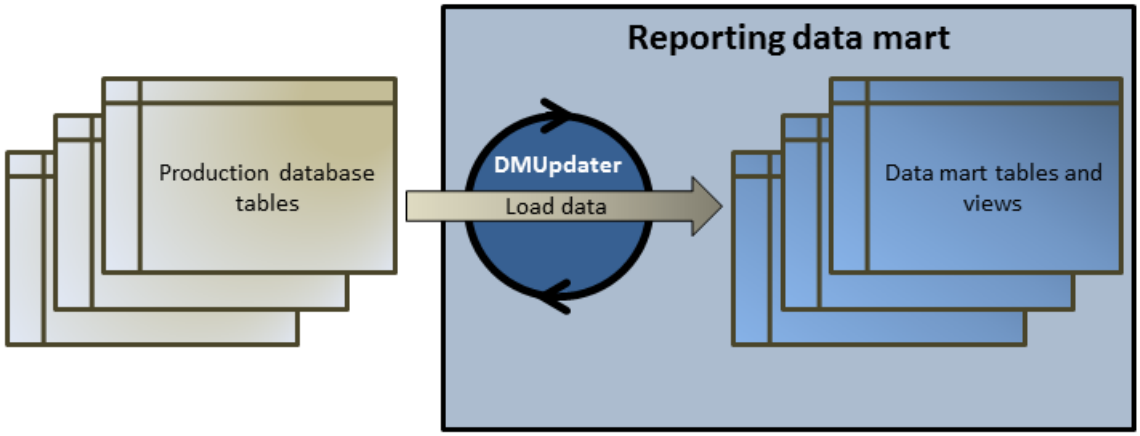
Die aktuelle Version des Data-Marts deckt den Ergebnisbereich ab. Weitere Bereiche für die Berichtserstellung werden in zukünftigen Versionen hinzugefügt. Die folgenden Tabellen und Ansichten sind derzeit verfügbar:

- Die `DM_TestStatus` Tabelle ist der Grundlage für statusbezogene Ansichten.
- Die `RV_TestStatusExtended` Ansicht liefert detaillierte Informationen zu einer bestimmten Testausführung.
- Die `RV_LatestTestStatus` Ansicht liefert Statusinformationen und erweiterte Informationen zu dem aktuellsten Testlauf eines Tests im Rahmen einer Testsuite und eines bestimmten Builds.
- Die `RV_MaxTestRunID` Ansicht ist ein Helfer um die aktuellste Testlauf-ID für jeden Test, jede Testsuite und jede Build-Kombination abzurufen.
- Die `RV_TestingCycleStatus` Ansicht liefert Status-Informationen zu Testzyklen.
- Die `RV_ExecutionPlanStatusPerBuild` Ansicht ruft die aktuellsten Summen der Teststatus für jede Testsuite im Rahmen eines Builds ab.
- Die `RV_ExecutionPlanStatusRollup` Ansicht ruft die Summen von bestehenden, fehlgeschlagenen und nicht ausgeführten Tests pro Testsuite oder im Rahmen eines Builds ab.
- Die `RV_ConfigurationSuiteStatus` Ansicht listet die Status-Anzahlen aller Konfigurationssuiten und Konfigurationen pro Build auf.

Architektur

Daten werden im Hintergrund aus den Produktivdatenbanktabellen extrahiert und in die Data-Mart Tabellen und Ansichten zur einfachen und schnellen Abfrage geladen. Wenn die Belastung der Datenbank nicht zu hoch ist, werden die Daten normalerweise in weniger als einer Minute verfügbar sein, nachdem Veränderungen durchgeführt wurden. Wenn Sie als Systemadministrator eingeloggt sind, können Sie den aktuellsten Stand der Datenladeprozesse prüfen, indem Sie zu `http://<server>:<port>/sctm/check/db` navigieren und das Kontrollkästchen **DM_TestStatus Table** anhaken.

Hinweis: Wenn Sie von einer Silk Central-Version aktualisieren, die den Data-Mart noch nicht enthielt (vor Version 13,0), werden die Data-Mart Tabellen und Ansichten mit den Daten vom Produktivssystem zunächst gefüllt. Abhängig von der Größe der Datenbank kann dieser Prozess einige Zeit dauern. Nachdem der Prozess abgeschlossen ist, können Sie auf die Daten zugreifen.



Wie Sie Berichte mit dem Data-Mart erstellen

Die folgenden Beispiele zeigen, wie Sie nützliche Berichte mit den Data-Mart Ansichten erstellen können.

Wie Sie Data-Mart Abfragen schreiben

1. Klicken Sie im Menü auf **Berichte** > **Detailansicht**.
2. Wählen Sie in der **Berichtshierarchie** den Ordner für den neuen Bericht aus.
   Dadurch legen Sie fest, wo der Bericht in der Ordnerhierarchie gespeichert wird.
3. Klicken Sie auf  in der Symbolleiste.Der Dialog **Neuen Bericht erstellen** wird angezeigt.
4. Geben Sie einen Namen für den neuen Bericht ein.
   Mit diesem Namen wird der Bericht in der **Berichtshierarchie** angezeigt.
5. Aktivieren Sie das Kontrollkästchen **Diesen Bericht anderen Benutzern zur Verfügung stellen**, wenn der Bericht für andere Benutzer freigegeben werden soll.
6. Geben Sie im Feld **Beschreibung** eine Beschreibung des Berichts ein.
7. Klicken Sie auf **Erweiterte Abfrage**, um das Feld **Berichtsdatenabfrage** zu öffnen. Fügen Sie vorhandenen Code ein, oder schreiben Sie neuen Code direkt in das Feld.
   Die Liste **Platzhalter einfügen** unterstützt Sie bei der Bearbeitung der SQL-Abfragen mit vordefinierten Funktionsplatzhaltern. Weitere Informationen finden Sie unter *SQL-Funktionen für benutzerdefinierte Berichte*.

   Hinweis: Nach der manuellen Bearbeitung von SQL-Anweisungen für die Abfrage sollte deren Gültigkeit mit **SQL überprüfen** geprüft werden.
8. Klicken Sie auf **Fertig stellen**, um die Änderungen zu speichern.

Die Zuverlässigkeit von Tests in einer Testsuite

Das Problem

In einer kontinuierlichen Integrationsumgebung werden Tests idealerweise mindestens einmal pro Tag ausgeführt, um den täglichen Build zu testen und die Qualität der zu testenden Anwendung zu sichern. Um zu verstehen, wie zuverlässig Ihr Test-Set für die Messung der Qualität Ihrer AUT ist, ist es notwendig gelegentlich zu überprüfen, wie sich die Testergebnisse im Laufe der Zeit verändern. Vielleicht haben Sie beispielsweise Tests in Ihrem Test-Set, die ständig ihren Status verändern, was bedeutet, dass sie kein echtes Maß für Qualität sind.

Die Lösung

Verwenden Sie die Data-Mart Ansicht *RV_TestStatusExtended* um einen Bericht zu erstellen, der die Ergebnisse für einen bestimmten Test im Rahmen von einer bestimmten Testsuite auflistet. Dadurch können Sie sehen, wie sich die Testergebnisse im Laufe der Zeit verändert haben. Der Einfachheit halber beschränken wir die Ergebnisliste auf Tests von markierten Builds, damit wir nur bestimmte Meilenstein-Builds der zu testenden Anwendung in Betracht ziehen. Dieser Bericht sammelt Testergebnisdaten für Tests im Rahmen von Testsuiten und Builds. In der folgenden Abfrage

- Wählen wir die Spalten aus, die wir von dieser Ansicht anzeigen möchten.
- Beschränken wir das Ergebnis auf die ID des Tests, den wir untersuchen möchten, und die ID der Testsuite, zu dem der Test gehört.
- Fügen wir eine Einschränkung hinzu, damit nur markierte Builds in Betracht gezogen werden.

```
SELECT TestName, ExecutionPlanName, VersionName, BuildName, TestRunID,
  PassedCount, FailedCount, NotExecutedCount
FROM RV_TestStatusExtended
WHERE TestID = ${TESTID|1|Test ID} AND ExecutionPlanID = ${EXECUTIONPLANID|1|
Execution Plan ID} AND BuildIsTagged = 1
ORDER BY BuildOrderNumber
```

Das Ergebnis der SQL-Abfrage liefert alle Testläufe für den ausgewählten Test in der ausgewählten Testsuite. Im folgenden Beispiel sehen Sie, dass der Test für den Build `579_Drop2` wiederholt wurde:

TestName	Execution PlanName	Version Name	BuildName	TestRunID	Passed Count	FailedCount	NotExecutedCount
UI Tests	EN| SQL2012| IE9|IIS	3.0	579_Drop02	7741797	59	5	0
UI Tests	EN| SQL2012| IE9|IIS	3.0	579_Drop02	7745078	63	1	0
UI Tests	EN| SQL2012| IE9|IIS	3.0	593_Drop03	7787437	63	1	0
UI Tests	EN| SQL2012| IE9|IIS	3.0	605_Drop04	7848720	63	1	0

Alle fehlgeschlagenen Tests in einem Ausführungsordner

Das Problem

Üblicherweise werden alle Testsuiten in einer Ordnerhierarchie strukturiert, die die verschiedenen Bereiche und Zwecke identifiziert, zu denen die Testsuiten und Tests in Verbindung stehen. Die Testsuiten werden regelmäßig in einer kontinuierlichen Integrationsumgebung oder bisweilen über einen gegebenen Zeitraum getriggert. Zur Folge haben Sie schöne Ausführungsstatistiken - leider nur für jede einzelne Testsuite.

Manchmal brauchen Sie jedoch eine allgemeine Übersicht über die Testergebnisse für einen bestimmten Bereich oder Zweck, um zu wissen und zu identifizieren, wo die Schwächen liegen.

Die Lösung

Verwenden Sie die Data-Mart-Ansicht *`RV_LatestTestStatus`* um einen Bericht zu erstellen, der eine Liste aller fehlgeschlagenen Tests für eine bestimmten Hierarchietiefe einer Testsuite zurückgibt.

Die folgende Abfrage wählt fehlgeschlagene Tests in einem Testsuite-Ordner aus, mit Zusatzinformationen wie dem Namen der Testsuite und des Builds.

```
SELECT TestID, TestName, ExecutionPlanID, ExecutionPlanName, BuildName
FROM RV_LatestTestStatus lts
INNER JOIN TM_ExecTreePaths ON lts.ExecutionPlanID =
TM_ExecTreePaths.NodeID_pk_fk
WHERE TM_ExecTreePaths.ParentNodeID_pk_fk = ${executionFolderID|2179|
Execution Folder ID}
```

```
  AND StatusID = 2
ORDER BY TestName
```

Die Abfrage macht Folgendes:

- Verwendet die Ansicht `RV_LatestTestStatus`, um das aktuellste Testlaufergebnis abzurufen.
- Enthält die Testsuitehierarchie (`TM_ExecTreePaths`), um alle Tests von allen Testsuiten in der Hierarchie abzufragen.
- Verwendet die ID von dem Ordner der höchsten Ebene, wo die Analyse als `ParentNodeID_pk_fk` gestartet werden soll.
- Enthält nur fehlgeschlagene Tests (`StatusID = 2`).

Die `StatusID` finden Sie in der tabelle `TM_TestDefStatusNames`.

Das Ergebnis der SQL-Query ist alle Tests in dem ausgewählten Testsuite-Ordner, für den der letzte Testlauf fehlgeschlagen ist.

TestID	TestName	ExecutionPlanID	ExecutionPlanName	BuildName
14073	JUnitTestPackage	2184	CI Testing	352
14107	Volatile Tests	2191	Volatile Test	352

Status des Testzyklus

Das Problem

Ein Testzyklus kann komplex sein, da er Informationen über manuelle Tester, Tests, verschiedene Builds und Versionen von Produkten und möglicherweise auch Konfigurationen enthält. Um die Übersicht nicht zu verlieren, ist es wichtig die Antworten auf folgende Fragen zu finden:

- Wie viele Tests wurden beendet?
- Wie viele Tests pro Build haben bestanden oder sind fehlgeschlagen?
- Sind die manuellen Tester noch beschäftigt oder können sie zusätzliche Arbeit anfangen?

Die Lösung

Verwenden Sie die Data-Mart Ansicht *`RV_TestingCycleStatus`*, um einen Bericht zu erstellen, der den Status eines Testzyklus pro Tester und pro Build zeigt. Damit haben Sie eine Übersicht über die Anzahl von bestandenen, fehlgeschlagenen oder nicht ausgeführten Tests, die nach manuellem Tester, Konfiguration und Build gruppiert sind.

```
SELECT BuildName, TesterLogin, TesterExecutionName,
  PassedCount, FailedCount, NotExecutedCount
FROM RV_TestingCycleStatus
WHERE TestingCycleID = ${testingCycleID|3|Testing Cycle ID}
ORDER BY BuildOrderNumber, TesterLogin
```

Die Abfrage macht Folgendes:

- Sie verwendet die Ansicht `RV_TestingCycleStatus` als Datenquelle, da sie `BuildName`, `TesterLogin` und `TesterExecutionName` enthält, welche den generierten Namen von Tester, Konfiguration und Test darstellen.
- Begrenzt die Daten auf die relevante Testzyklus-ID.

Das Ergebnis der SQL-Abfrage zeigt den Status pro Build und pro Tester.

BuildName	TesterLogin	TesterExecution Name	PassedCount	FailedCount	NotExecuted Count
352		No specific tester (Test Assets)	0	0	1
351	admin	admin (English| SQL2008|FF| Tomcat - Test Assets)	0	1	0
352	admin	admin (English| SQL2008|FF| Tomcat - Test Assets)	0	0	1
352	gmazzuchelli	gmazzuchelli (English| Oracle10g|IE8| Tomcat - Test Assets)	0	1	1
352	jallen	jallen (German| Oracle11g|FF| Tomcat - Test Assets)	1	1	0
352	smiller	smiller (German| SQL2008|IE8|IIS - Test Assets)	1	1	0

Für nicht zugeordnete Tests wird eine Gruppe "kein zugewiesener Tester" erstellt, mit leeren Werten für `TesterLogin`,`TesterFirstName`, und `TesterLastName`.

Falls Sie den Fortschritt des Testzyklus bezogen auf die Leistung von den manuellen Testern wissen möchten, hilft eine leichte Abwandlung der Abfrage:

```
SELECT TesterLogin, TesterExecutionName, SUM(PassedCount) PassedCount,
  SUM(FailedCount) FailedCount, SUM(NotExecutedCount) NotExecutedCount
FROM RV_TestingCycleStatus
WHERE TestingCycleID = ${testingCycleID|3|Testing Cycle ID}
GROUP BY TesterLogin, TesterExecutionName
ORDER BY TesterLogin
```

Die Abfrage wurde um Folgendes erweitert:

- `GROUP BY TesterLogin, TesterExecutionName` um die übrigen Spalten zu bezeichnen.
- `SUM( )` zu den Zählern, um die Zahlen zusammenzufassen.

TesterLogin	TesterExecution Name	PassedCount	FailedCount	NotExecutedCount
	No specific tester (Test Assets)	0	0	1
admin	admin (English| SQL2008|FF|Tomcat - Test Assets)	0	1	1
gmazzuchelli	gmazzuchelli (English|Oracle10g|	0	1	1

TesterLogin	TesterExecution Name	PassedCount	FailedCount	NotExecutedCount
	IE8|Tomcat - Test Assets)			
jallen	jallen (German| Oracle11g|FF|Tomcat - Test Assets)	1	1	0
smiller	smiller (German| SQL2008|IE8|IIS - Test Assets)	1	1	0

Status von Testsuitehierarchien

Das Problem

Es ist üblich, Testsuiten in einer hierarchischen Struktur unterzubringen, welche die verschiedene Testbereiche und Testzwecke abbildet. In einigen Fällen werden Sie zum Beispiel die gesamte Anzahl von bestandenen, fehlgeschlagenen und nicht ausgeführten Tests wissen möchten, um den Teststatus und somit die Qualität eines Testbereichs oder Testzwecks zu wissen.

Die Lösung

Verwenden Sie die Data-Mart Ansicht *RV_ExecutionPlanStatusRollup* um einen Bericht zu erstellen, der die Anzahl bestandener, fehlgeschlagener und nicht ausgeführter Tests für einen bestimmten Testsuite-Ordner auflistet, gruppiert nach Build.

```
SELECT BuildName, PassedCount, FailedCount, NotExecutedCount
FROM RV_ExecutionPlanStatusRollup
WHERE ExecutionFolderID = ${executionPlanID|43|Execution Plan ID}
```

Die Abfrage macht Folgendes:

- Wählt `BuildName` und die Status-Anzahlen von der Ansicht `RV_ExecutionPlanStatusRollup` aus.
- Bestimmt den Order der höchsten Ebene, von dem Sie den Status (`ExecutionFolderID`) haben wollen.

Das Ergebnis der SQL-Abfrage zeigt den Status der Testläufe in allen Testsuiten des ausgewählten Ordners an, aggregiert pro Build.

BuildName	PassedCount	FailedCount	NotExecutedCount
351	0	0	2
352	15	7	1

Sollten Sie an detaillierteren Informationen interessiert sein, beispielsweise an den Status-Anzahlen für jede Testsuite innerhalb einer ausgewählten Hierarchie, können Sie die Data-Mart Ansicht *RV_ExecutionPlanStatusPerBuild* verwenden:

```
SELECT eps.BuildName, eps.ExecutionPlanID, SUM(eps.PassedCount) PassedCount,
  SUM(eps.FailedCount) FailedCount, SUM(eps.NotExecutedCount) NotExecutedCount
FROM RV_ExecutionPlanStatusPerBuild eps
INNER JOIN TM_ExecTreePaths etp ON eps.ExecutionPlanID = etp.NodeID_pk_fk
WHERE etp.ParentNodeID_pk_fk = ${execFolderID|44|Execution Folder ID}
GROUP BY eps.ExecutionPlanID, eps.BuildOrderNumber, eps.BuildName
ORDER BY eps.BuildOrderNumber, eps.ExecutionPlanID
```

Die Abfrage macht Folgendes:

- Verwendet die Ansicht `RV_ExecutionPlanStatusPerBuild`, um auf Testsuite-bezogenen Daten zuzugreifen (`ExecutionPlanID` und `ExecutionPlanName`). Die zuvor verwendete Ansicht `RV_ExecutionPlanStatusRollup` enthält voraggregierte Daten (zusammengezählte Daten), die zu diesem Zweck nicht geeignet sind, da Sie nicht nur die Ergebnisse für Testsuiten liefert, sondern auch für Ordner-Knoten.
- Wählt alle Knoten eines bestimmten Ordners mittels `JOIN` der `TM_ExecTreePath` Tabelle aus, um die Hierarchieinformationen zu sammeln.
- Bestimmt den Ordner der höchsten Ebene mit `ExecutionFolderID`. Da die Tabelle `TM_ExecutionTreePaths` auch eine Selbstreferenz für jede Testsuite enthält, können Sie diese Abfrage auch für `ParentNodeID_pk_fk` mit einer Testsuite-ID ausführen, was die Reihen einer bestimmten Testsuite zurückgeben würde.
- Fügt `ORDER BY BuildOrderNumber` und `ExecutionPlanID` hinzu, um ein sauber geordnetes Ergebnis zu erhalten, beginnend mit dem ältesten Build und dessen Testsuiten.

Das Ergebnis der SQL-Abfrage liefert den Status der Testläufe in allen Testsuiten des ausgewählten Ordners.

BuildName	ExecutionPlanID	PassedCount	FailedCount	NotExecutedCount
351	2307	0	0	2
352	2184	11	2	0
352	2185	0	3	0
352	2186	2	1	0
352	2187	1	0	0
352	2191	0	1	0
352	2307	1	0	1

Status einer Konfigurationssuite

Das Problem

Konfigurationssuiten ermöglichen Ihnen die gleichen Tests für mehrere Konfigurationen, zum Beispiel mehrere Browser oder Betriebssysteme, auszuführen. Um eine vernünftige Aussage über die Qualität und Zuverlässigkeit der zu testenden Anwendung machen zu können, werden Sie die Ergebnisse für jede individuelle Konfiguration verfolgen wollen.

Die Lösung

Verwenden Sie die Data-Mart Ansicht *`RV_ConfigurationSuiteStatus`* um einen Bericht zu erstellen, der die Anzahl bestandener, fehlgeschlagener und nicht ausgeführter Tests für jede Konfiguration pro Build zurückgibt.

```
SELECT BuildName, ConfigurationName, PassedCount, FailedCount,
NotExecutedCount
FROM RV_ConfigurationSuiteStatus
WHERE ConfigurationSuiteID = ${configSuiteID|97|Configuration Suite ID}
ORDER BY BuildOrderNumber, ConfigurationName
```

Die Abfrage macht Folgendes:

- Ruft die Status-Anzahlen pro Build von Testläufen von der Ansicht `RV_ConfigurationSuiteStatus` ab.
- Beschränkt die Ergebnisse auf die Konfigurationssuite (`ConfigurationSuiteID`).

Das Ergebnis der SQL-Abfrage liefert den Status der Testläufe für jede Konfiguration.

BuildName	ConfigurationName	PassedCount	FailedCount	NotExecutedCount
350	Chrome	0	1	0
350	Firefox	0	1	0
350	Internet Explorer	0	1	0
351	Chrome	1	0	0
351	Firefox	1	0	0
351	Internet Explorer	0	1	0
352	Chrome	1	0	0
352	Firefox	1	0	0
352	Internet Explorer	1	0	0

In diesem Beispiel verwenden wir die ID der Konfigurationssuite, um alle Konfigurationen zu sammeln. Es ist auch möglich das Ergebnis auf bestimmte Builds zu beschränken. In diesem Fall müssten Sie `BuildID`,`BuildName`, oder`BuildOrderNumber` in der WHERE-Klausel einschließen.

Hinweis: Die Ansicht *`RV_ConfigurationSuiteStatus`* enthält nur aggregierte Status-Anzahlen ohne testspezifische Daten. Um zusätzliche testspezifische Daten abzurufen, können Sie die Ansicht *`RV_LatestTestStatus`* verwenden.

Fehlerdiagnose

Falsche oder Fehlende Daten

Das Problem

Beim Abfragen von Daten aus einer Data-Mart Tabelle oder Ansicht sind die angegebenen Ergebnisse nicht aktuell oder sie fehlen.

Die Lösung

Die Data-Mart Tabellen und Ansichten werden regelmäßig im Hintergrund aber nicht in Echtzeit aktualisiert. Demzufolge kann es einige Sekunden bis Minuten dauern, bevor die Daten in die Data-Mart Tabellen geladen werden. Wenn das System unter schwerer Last läuft, wird die Leistung des Hintergrundprozess, der die Daten lädt, dadurch beeinflusst. Der Grund dafür ist, dass die anderen Prozesse eine höhere Priorität haben und den DataMartUpdater-Hintergrundjob temporär blockieren können. Führen Sie die Abfrage noch einmal aus, um die aktualisierten Daten abzufragen.

Wenn Sie als Systemadministrator eingeloggt sind, können Sie den aktuellsten Stand der Datenladeprozesse prüfen, indem Sie zu `http://<server>:<port>/sctm/check/db` navigieren und das Kontrollkästchen **DM_TestStatus Table** anhaken.

Hinweis: Tests und die abhängigen Testläufe werden vom Data-Mart entfernt, wenn der Test gelöscht wird. Dies gilt auch für Tests, die durch die Bereinigung eines Testpakets gelöscht wurden.

Der Data-Mart verlangsamt das System

Das Problem

Seit der Data-Mart ausgeführt wird, scheint die Gesamtleistung des Systems schlechter zu sein oder sich inkonsistent zu verhalten.

Die Lösung

Obwohl es nicht passieren sollte, ist es möglich den Data-Mart auszuschalten um zu prüfen, ob das Problem dadurch gelöst wird.

1. Stoppen Sie den Anwendungsserverdienst.
2. Öffnen Sie die Datei `TmAppServerHomeConf.xml` mit einem Texteditor. Die Datei befindet sich auf dem Anwendungsserver im Ordner `/conf/appserver` des Silk Central-Verzeichnisses.
3. Finden Sie den XML-Tag `Config/DataMart/Enabled` und stellen Sie den Wert auf `false` ein.
4. Starten Sie den Anwendungsserverdienst neu.

Referenz: Data-Mart Tabellen und Ansichten

Die folgenden Data-Mart Tabellen und Ansichten sind für die einfache und schnelle Berichterstellung verfügbar.

DM_TestStatus

Die `DM_TestStatus` Tabelle ist der Grundlage für statusbezogene Ansichten.

Die anderen Data-Mart Ansichten ermöglichen normalerweise einen einfacheren Zugriff auf detaillierten Daten, da diese Tabelle keinen direkten Zugriff auf Informationen wie den Namen eines Tests ermöglicht. Der Schlüssel dieser Tabelle ist die Kombination der Spalten `TestID`, `ExecutionPlanID`, `BuildID`, und `TestRunID`.

Reihe	Beschreibung
`TestID`	Die Kennung des Tests.
`ExecutionPlanID`	Die Kennung der Testsuite.
`BuildID`	Die Kennung des Builds.
`TestRunID`	Die Kennung des Testlaufs.
`ExecutionRunID`	Identifiziert den Ausführungslauf, in dem dieses Ergebnis generiert wurde.
`StatusID`	Der Status dieses Testlaufs (siehe `TM_TestDefStatusNames`).
`ReasonID`	Die Ursache für den Status dieses Testlaufs (siehe `TM_ResultStatusReasons`). Kann "null" sein.
`PassedCount`	Die Summe aller bestandenen Tests, die 0 oder 1 für allgemeine Tests ist und für Stammknoten von Testpaketen höher sein kann.
`FailedCount`	Die Summe aller fehlgeschlagenen Tests, die 0 oder 1 für allgemeine Tests ist und für Stammknoten von Testpaketen höher sein kann.
`NotExecutedCount`	Die Summe aller nicht ausgeführten Tests, die 0 oder 1 für allgemeine Tests ist und für Stammknoten von Testpaketen höher sein kann.
`ProjectID`	Die ID des Projekts, zu dem diese Reihe gehört.
`TestStartTime`	Der Zeitpunkt, zu dem der Testlauf gestartet wurde (UTC).
`ExecutionStartTime`	Der Zeitpunkt, zu dem der Ausführungslauf gestartet wurde (UTC).
`TestDurationInMilliseconds`	Die Dauer des Testlaufs in Millisekunden.

Reihe	Beschreibung
IsBlocked	Kennzeichnet den Testlauf als blockiert oder nicht blockiert
DbChangedAt	Der Zeitpunkt, zu dem diese Reihe zuletzt durch den Berichts-Data-Mart aktualisiert wurde.

Die Ansicht RV_TestStatusExtended

Die RV_TestStatusExtended Ansicht liefert detaillierte Informationen zu einer bestimmten Testausführung.

Diese Ansicht liefert Informationen zu allen Testläufen, im Gegensatz zur Ansicht *RV_LatestTestStatus*, welche Statusinformationen und erweiterte Informationen zu dem aktuellsten Testlauf eines Tests im Rahmen einer Testsuite und eines bestimmten Builds liefert. Sie können diese Ansicht verwenden, um zum Beispiel *einen Bericht, der jeden Testlauf von markierten Builds auflistet*, zu erstellen. Der Schlüssel dieser Tabelle ist die Kombination der Spalten TestID, ExecutionPlanID, BuildID und TestRunID.

Hinweis: Tests und die abhängigen Testläufe werden vom Data-Mart entfernt, wenn der Test gelöscht wird. Dies gilt auch für Tests, die durch die Bereinigung eines Testpakets gelöscht wurden.

Reihe	Beschreibung
TestID	Die Kennung des Tests.
ExecutionPlanID	Die Kennung der Testsuite.
BuildID	Die Kennung des Builds.
TestRunID	Die Kennung des Testlaufs.
ExecutionRunID	Identifiziert den Ausführungslauf, in dem dieses Ergebnis generiert wurde.
StatusID	Der Status dieses Testlaufs (siehe TM_TestDefStatusNames).
ReasonID	Die Ursache für den Status dieses Testlaufs (siehe TM_ResultStatusReasons). Kann "null" sein.
PassedCount	Die Summe aller bestandenen Tests, die 0 oder 1 für allgemeine Tests ist und für Stammknoten von Testpaketen höher sein kann.
FailedCount	Die Summe aller fehlgeschlagenen Tests, die 0 oder 1 für allgemeine Tests ist und für Stammknoten von Testpaketen höher sein kann.
NotExecutedCount	Die Summe aller nicht ausgeführten Tests, die 0 oder 1 für allgemeine Tests ist und für Stammknoten von Testpaketen höher sein kann.
ProjectID	Die ID des Projekts, zu dem diese Reihe gehört.
TestStartTime	Der Zeitpunkt, zu dem der Testlauf gestartet wurde (UTC).
ExecutionStartTime	Der Zeitpunkt, zu dem der Ausführungslauf gestartet wurde (UTC).

Reihe	Beschreibung
TestDurationInMilliseconds	Die Dauer des Testlaufs in Millisekunden.
IsBlocked	Kennzeichnet den Testlauf als blockiert oder nicht blockiert
DbChangedAt	Der Zeitpunkt, zu dem diese Reihe zuletzt durch den Berichts-Data-Mart aktualisiert wurde.
TestName	Der Name des Tests.
TestDescription	Die Beschreibung des Tests.
TestParentID	Die ID des übergeordneten Tests.
PlannedTimeInMinutes	Geplante Dauer dieses Tests in Minuten.
Reason	Name der Ursache. Kann Ursachen beinhalten, welche in der Zwischenzeit gelöscht wurden.
ExecutionPlanName	Der Name der Testsuite.
ExecutionPlanDescription	Die Beschreibung der Testsuite.
ExecutionParentFolderID	Die ID der übergeordneten Testsuite.
Priority	Die Priorität der Testsuite: 0 = Niedrig, 1 = Mittel, 2 = Hoch.
BuildName	Der Name des Builds, der für diesen Testlauf verwendet wurde.
BuildDescription	Die Beschreibung des Builds.
BuildOrderNumber	Die Reihenfolge des Builds.
BuildIsTagged	1 wenn der Build markiert ist, 0 wenn nicht.
VersionID	Die ID der Version, zu der der Build gehört.
VersionName	Der Name der Version.
VersionDescription	Die Beschreibung der Version.
VersionOrderNumber	Die Reihenfolge der Version.
ProductID	Die ID des Produkts, zu dem der Build gehört.
ProductCode	Der Name des Produkts.
ProductDescription	Die Beschreibung des Produkts.
ProductOrderNumber	Die Reihenfolge des Produkts.

Die Ansicht RV_LatestTestStatus

Die RV_LatestTestStatus Ansicht liefert Statusinformationen und erweiterte Informationen zu dem aktuellsten Testlauf eines Tests im Rahmen einer Testsuite und eines bestimmten Builds.

Benutzen Sie die Ansicht *RV_TestStatusExtended*, um Informationen über alle Testläufe abzufragen. Sie können diese Ansicht verwenden, um *einen Bericht, der alle fehlgeschlagenen Tests in einem Ausführungsordner auflistet* zu erstellen. Der Schlüssel dieser Tabelle ist die Kombination der Spalten TestID, ExecutionPlanID, BuildID, und TestRunID.

Reihe	Beschreibung
TestID	Die Kennung des Tests.
ExecutionPlanID	Die Kennung der Testsuite.
BuildID	Die Kennung des Builds.
TestRunID	Die Kennung des Testlaufs.
ExecutionRunID	Identifiziert den Ausführungslauf, in dem dieses Ergebnis generiert wurde.
StatusID	Der Status dieses Testlaufs (siehe TM_TestDefStatusNames).
ReasonID	Die Ursache für den Status dieses Testlaufs (siehe TM_ResultStatusReasons). Kann "null" sein.
PassedCount	Die Summe aller bestandenen Tests, die 0 oder 1 für allgemeine Tests ist und für Stammknoten von Testpaketen höher sein kann.
FailedCount	Die Summe aller fehlgeschlagenen Tests, die 0 oder 1 für allgemeine Tests ist und für Stammknoten von Testpaketen höher sein kann.
NotExecutedCount	Die Summe aller nicht ausgeführten Tests, die 0 oder 1 für allgemeine Tests ist und für Stammknoten von Testpaketen höher sein kann.
ProjectID	Die ID des Projekts, zu dem diese Reihe gehört.
TestStartTime	Der Zeitpunkt, zu dem der Testlauf gestartet wurde (UTC).
ExecutionStartTime	Der Zeitpunkt, zu dem der Ausführungslauf gestartet wurde (UTC).
TestDurationInMilliseconds	Die Dauer des Testlaufs in Millisekunden.
IsBlocked	Kennzeichnet den Testlauf als blockiert oder nicht blockiert
DbChangedAt	Der Zeitpunkt, zu dem diese Reihe zuletzt durch den Berichts-Data-Mart aktualisiert wurde.
TestName	Der Name des Tests.
TestDescription	Die Beschreibung des Tests.
TestParentID	Die ID des übergeordneten Tests.
PlannedTimeInMinutes	Geplante Dauer dieses Tests in Minuten.
Reason	Name der Ursache. Kann Ursachen beinhalten, welche in der Zwischenzeit gelöscht wurden.
ExecutionPlanName	Der Name der Testsuite.
ExecutionPlanDescription	Die Beschreibung der Testsuite.
ExecutionParentFolderID	Die ID der übergeordneten Testsuite.
Priority	Die Priorität der Testsuite: 0 = Niedrig, 1 = Mittel, 2 = Hoch.

Reihe	Beschreibung
BuildName	Der Name des Builds, der für diesen Testlauf verwendet wurde.
BuildDescription	Die Beschreibung des Builds.
BuildOrderNumber	Die Reihenfolge des Builds.
BuildIsTagged	1 wenn der Build markiert ist, 0 wenn nicht.
VersionID	Die ID der Version, zu der der Build gehört.
VersionName	Der Name der Version.
VersionDescription	Die Beschreibung der Version.
VersionOrderNumber	Die Reihenfolge der Version.
ProductID	Die ID des Produkts, zu dem der Build gehört.
ProductCode	Der Name des Produkts.
ProductDescription	Die Beschreibung des Produkts.
ProductOrderNumber	Die Reihenfolge des Produkts.

Die Ansicht RV_MaxTestRunID

Die RV_MaxTestRunID Ansicht ist ein Helfer um die aktuellste Testlauf-ID für jeden Test, jede Testsuite und jede Build-Kombination abzurufen.

Der Schlüssel dieser Tabelle ist die Kombination der Spalten TestID, ExecutionPlanID und BuildID.

Reihe	Beschreibung
TestID	Die Kennung des Tests.
ExecutionPlanID	Die Kennung der Testsuite.
BuildID	Die Kennung des Builds.
MaxTestRunID	Identifiziert den aktuellsten Testlauf des Tests im Kontext des Ausführungsplans und des Builds.

RV_TestingCycleStatus

Die RV_TestingCycleStatus Ansicht liefert Status-Informationen zu Testzyklen.

Sie können diese Ansicht verwenden um *einen Bericht, der den aktuellen Status eines Testzyklus zeigt* zu erstellen.

TestingCycleID bezeichnet den Testzyklus und TesterExecutionID (sowie TesterExecutionName, UserID, CapacityInCycle, TesterLogin, TesterFirstName, TesterLastName) und wird verwendet, den zugeordneten Tester im Testzyklus zu identifizieren. Für Tests die keinem Tester zugeordnet, sind UserID, CapacityInCycle, TesterLogin, TesterFirstName, und TesterLastName null. Der Schlüssel dieser Tabelle ist die Kombination der Spalten TesterExecutionID und BuildID.

Reihe	Beschreibung
TestingCycleID	Die Kennung des Testzyklus.
TesterExecutionID	Identifiziert die Gruppe von Tests, die einem Tester zugeordnet sind.
TesterExecutionName	Der generierte Name der Gruppe von Tests, die einem Tester zugeordnet sind.
UserID	Die User-ID des Testers.
CapacityInCycleInMinutes	Die Kapazität für diesen Benutzer in diesem Testzyklus in Minuten.
TesterLogin	Der Loginname des Testers.
TesterFirstName	Der Vorname des Testers
TesterLastName	Der Nachname des Testers
PassedCount	Die Summe aller bestandenen Tests.
FailedCount	Die Summe aller fehlgeschlagenen Tests.
NotExecutedCount	Die Summe aller nicht ausgeführten Tests.
ProjectID	Die Kennung des Projekts.
BuildID	Die Kennung des Builds.
BuildName	Der Name des Builds, der für diesen Testlauf verwendet wurde.
BuildDescription	Die Beschreibung des Builds.
BuildOrderNumber	Die Reihenfolge des Builds.
BuildIsTagged	1 wenn der Build markiert ist, 0 wenn nicht.
VersionID	Die ID der Version, zu der der Build gehört.
VersionName	Der Name der Version.
VersionDescription	Die Beschreibung der Version.
VersionOrderNumber	Die Reihenfolge der Version.
ProductID	Die ID des Produkts, zu dem der Build gehört.
ProductCode	Der Name des Produkts.
ProductDescription	Die Beschreibung des Produkts.
ProductOrderNumber	Die Reihenfolge des Produkts.

RV_ExecutionPlanStatusPerBuild

Die RV_ExecutionPlanStatusPerBuild Ansicht ruft die aktuellsten Summen der Teststatus für jede Testsuite im Rahmen eines Builds ab.

Ordner und untergeordnete Knoten werden nicht berücksichtigt. Sie können diese Ansicht benutzen, um *einen Bericht, der den Status Ihrer Testläufe für jede Testsuite in einem Ordner zeigt* zu erstellen. Im Gegensatz zu *RV_ExecutionPlanStatusRollup*, hat diese Ansicht einen leichten Leistungsvorteil, da

keine Hierarchie berücksichtigt wird um die Daten abzurufen. Der Schlüssel dieser Tabelle ist die Kombination der Spalten `ExecutionPlanID` und `BuildID`.

Reihe	Beschreibung
`ExecutionPlanID`	Die Kennung der Testsuite.
`BuildID`	Die Kennung des Builds.
`ExecutionPlanName`	Der Name der Testsuite.
`ExecutionParentFolderID`	Die ID der übergeordneten Testsuite.
`PassedCount`	Die Summe aller bestandenen Tests.
`FailedCount`	Die Summe aller fehlgeschlagenen Tests.
`NotExecutedCount`	Die Summe aller nicht ausgeführten Tests.
`ProjectID`	Die ID des Projekts, zu dem die Testsuite gehört.
`BuildName`	Der Name des Builds, der für diesen Testlauf verwendet wurde.
`BuildDescription`	Die Beschreibung des Builds.
`BuildOrderNumber`	Die Reihenfolge des Builds.
`BuildIsTagged`	`1` wenn der Build markiert ist, `0` wenn nicht.
`VersionID`	Die ID der Version, zu der der Build gehört.
`VersionName`	Der Name der Version.
`VersionDescription`	Die Beschreibung der Version.
`VersionOrderNumber`	Die Reihenfolge der Version.
`ProductID`	Die ID des Produkts, zu dem der Build gehört.
`ProductCode`	Der Name des Produkts.
`ProductDescription`	Die Beschreibung des Produkts.
`ProductOrderNumber`	Die Reihenfolge des Produkts.

RV_ExecutionPlanStatusRollup

Die `RV_ExecutionPlanStatusRollup` Ansicht ruft die Summen von bestehenden, fehlgeschlagenen und nicht ausgeführten Tests pro Testsuite oder im Rahmen eines Builds ab.

Wenn es sich um Ordner handelt, enthalten die Zähler die Zahlen von allen Tests, auch in untergeordneten Ordnern. Sie können diese Ansicht benutzen, um *einen Bericht, der den Status aller Testläufe in einem Ordner zeigt* zu erstellen. Der Schlüssel dieser Tabelle ist die Kombination der Spalten `ExecutionFolderID` und `BuildID`.

Reihe	Beschreibung
`ExecutionFolderID`	Die Kennung der Testsuite.
`BuildID`	Die Kennung des Build.
`PassedCount`	Die Summe aller bestandenen Tests.
`FailedCount`	Die Summe aller fehlgeschlagenen Tests.

Reihe	Beschreibung
NotExecutedCount	Die Summe aller nicht ausgeführten Tests.
ProjectID	Die ID des Projekts, zu dem die Testsuite gehört.
BuildName	Der Name des Builds, der für diesen Testlauf verwendet wurde.
BuildDescription	Die Beschreibung des Builds.
BuildOrderNumber	Die Reihenfolge des Builds.
BuildIsTagged	1 wenn der Build markiert ist, 0 wenn nicht.
VersionID	Die ID der Version, zu der der Build gehört.
VersionName	Der Name der Version.
VersionDescription	Die Beschreibung der Version.
VersionOrderNumber	Die Reihenfolge der Version.
ProductID	Die ID des Produkts, zu dem der Build gehört.
ProductCode	Der Name des Produkts.
ProductDescription	Die Beschreibung des Produkts.
ProductOrderNumber	Die Reihenfolge des Produkts.

RV_ConfigurationSuiteStatus

Die RV_ConfigurationSuiteStatus Ansicht listet die Status-Anzahlen aller Konfigurationssuiten und Konfigurationen pro Build auf.

Sie können diese Ansicht benutzen, um *einen Bericht, der den Status aller Testläufe für jede Konfiguration in einer Konfigurationssuite zeigt* zu erstellen. Der Schlüssel dieser Tabelle ist die Kombination der Spalten ConfigurationID und BuildID.

Reihe	Beschreibung
ConfigurationSuiteID	Die Kennung der Konfigurationssuite.
ConfigurationID	Die Kennung der Konfiguration.
ConfigurationName	Der Name der Konfiguration.
BuildID	Die Kennung des Builds.
PassedCount	Die Summe aller bestandenen Tests.
FailedCount	Die Summe aller fehlgeschlagenen Tests.
NotExecutedCount	Die Summe aller nicht ausgeführten Tests.
ProjectID	Die ID des Projekts, zu dem diese Reihe gehört.
BuildName	Der Name des Builds, der für diesen Testlauf verwendet wurde.
BuildDescription	Die Beschreibung des Builds.
BuildOrderNumber	Die Reihenfolge des Builds.
BuildIsTagged	1 wenn der Build markiert ist, 0 wenn nicht.

Reihe	Beschreibung
VersionID	Die ID der Version, zu der der Build gehört.
VersionName	Der Name der Version.
VersionDescription	Die Beschreibung der Version.
VersionOrderNumber	Die Reihenfolge der Version.
ProductID	Die ID des Produkts, zu dem der Build gehört.
ProductCode	Der Name des Produkts.
ProductDescription	Die Beschreibung des Produkts.
ProductOrderNumber	Die Reihenfolge des Produkts.