Borland
A MICRO FOCUS COMPANY

Silk Central 15.0

Datamart de reporting

Micro Focus
575 Anton Blvd., Suite 510
Costa Mesa, CA 92626

2013-12-02

Table des matières

Vue d'ensemble

Le datamart de reporting Silk Central permet d'accéder facilement aux données à des fins de reporting. Il déplace les données des tables de production vers les vues dédiées qui doivent être utilisées pour la création de rapports avancés. Les avantages sont, notamment :

- Dénomination claire des tables et des vues, ce qui vous permet de localiser rapidement les données recherchées.
- Données prétraitées, ce qui vous permet d'accéder aux données agrégées sans avoir à les calculer par vous-même.
- Performances optimisées, car les rapports peuvent utiliser des requêtes SQL beaucoup plus simples et rapides.
- Dépendance réduite sur la charge de la base de données de production, ce qui améliore également les performances et supprime les pics de charge.

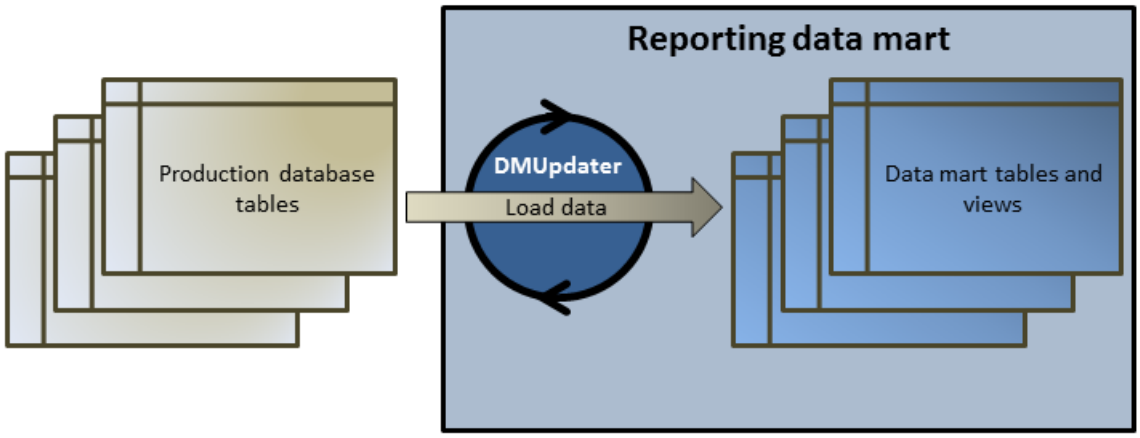
La version actuelle du datamart couvre l'espace de résultats. D'autres espaces destinés au reporting seront ajoutés au datamart dans les prochaines releases. Les tables et vues suivantes sont actuellement disponibles :

- La table `DM_TestStatus` constitue la table de base des vues relatives à l'état.
- La vue `RV_TestStatusExtended` fournit des informations détaillées pour une exécution de test précise.
- La vue `RV_LatestTestStatus` fournit l'état et des informations complémentaires sur la dernière exécution d'un test dans le cadre d'un plan d'exécution et d'un build précis.
- La vue `RV_MaxTestRunID` permet de récupérer le dernier ID d'exécution de test de chaque combinaison de test, plan d'exécution et build.
- La vue `RV_TestingCycleStatus` fournit des informations sur l'état des cycles de test.
- La vue `RV_ExecutionPlanStatusPerBuild` récupère les totaux des derniers états de test de chaque plan d'exécution dans le cadre de builds.
- La vue `RV_ExecutionPlanStatusRollup` récupère les totaux des tests réussis, en échec et non exécutés par plan d'exécution ou dossier dans le cadre d'un build.
- La vue `RV_ConfigurationSuiteStatus` répertorie les totaux par état pour toutes les suites de configurations et les configurations par build.

Architecture

Les données sont régulièrement extraites en arrière-plan des tables de la base de données de production, puis chargées dans les tables et les vues du datamart, permettant ainsi des recherches faciles et rapides. Si la base de données n'est pas trop chargée, ces données sont généralement disponibles en moins d'une minute après la validation des modifications effectuées. Si vous vous connectez en tant qu'administrateur système, vous pouvez vérifier l'état actuel du processus de chargement de données. Pour cela, accédez à `http://<server>:<port>/sctm/check/db` et consultez la **Table DM_TestStatus**.

Remarque: Si vous effectuez une mise à jour depuis une version de Silk Central qui ne comprenait pas de datamart (versions antérieures à la 13.0), les tables et les vues de datamart sont initialement alimentées avec des données issues du système de production. Selon la taille de votre base de données, ce processus peut prendre du temps. Une fois le processus terminé, vous pouvez accéder aux données.



Création de rapports à l'aide du datamart

Les exemples suivants illustrent comment créer des rapports utiles à l'aide des vues du datamart.

Écriture de requêtes de datamart

1. Dans le menu, cliquez sur **Rapports** > **Vue Détaillée**.
2. Dans l'arborescence **Rapports**, sélectionnez le dossier dans lequel vous voulez afficher le nouveau rapport.

   Ceci détermine l'emplacement de stockage du rapport dans l'arborescence de répertoires.
3. Cliquez sur  dans la barre d'outils.La boîte de dialogue **Créer un Nouveau Rapport** s'ouvre.
4. Indiquez le nom du nouveau rapport.

   Il s'agit du nom affiché dans l'arborescence **Rapports**.
5. Cochez la case **Partager ce rapport avec d'autres utilisateurs** si vous voulez que ce rapport soit accessible à d'autres utilisateurs.
6. Saisissez une description du rapport dans le champ **Description**.
7. Cliquez sur **Requête Avancée** pour ouvrir le champ **Requête de données de rapport**. Insérez le code précédemment écrit ou écrivez le nouveau code directement dans le champ.

   La liste **Insérer un espace réservé** vous aide à modifier les requêtes SQL à l'aide d'espaces réservés de fonctions prédéfinies. Pour plus de détails, consultez la section *Fonctions SQL pour les rapports personnalisés*.

   Remarque: Si vous éditez manuellement du code SQL pour la requête, cliquez sur **Vérifier le SQL** afin de confirmer votre saisie.
8. Cliquez sur **Terminer** pour enregistrer vos paramètres.

Fiabilité des tests d'un plan d'exécution

Problème

Dans un environnement d'intégration continue, les tests sont idéalement exécutés au moins une fois par jour afin de tester le build quotidien et de garantir la qualité de l'application testée. Pour connaître la fiabilité de votre série de tests et savoir si la qualité de l'application testée est correctement mesurée, il est indispensable d'examiner de temps en temps les variations des résultats dans le temps. Votre série de tests peut, par exemple, contenir des tests dont l'état change fréquemment, et qui, par conséquent, ne mesurent pas réellement la qualité.

Solution

Utilisez la vue datamart *`RV_TestStatusExtended`* pour créer un rapport qui répertorie les résultats d'un test donné dans le cadre d'un plan d'exécution spécifique. Vous pourrez ainsi constater comment les résultats de ce test ont évolué au fil du temps. Pour des raisons pratiques, nous allons restreindre la liste des résultats aux résultats associés aux builds marqués, et examiner ainsi uniquement des builds intermédiaires spécifiques de l'application testée. Ce rapport collecte les données de résultats de test pour des tests effectués dans le cadre de plans d'exécution et de builds. Dans la requête suivante, nous avons :

- Sélectionné les colonnes que nous souhaitons afficher à partir de cette vue.

- Limité les résultats à l'ID du test que nous souhaitons examiner et à l'ID du plan d'exécution auquel le test appartient.
- Ajouté une contrainte pour ne prendre en compte que les builds marqués.

```
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
```

La requête SQL renvoie toutes les exécutions de test pour le test sélectionné dans le plan d'exécution sélectionné. Dans l'exemple suivant, le test a été réexécuté avec le build `579_Drop2` :

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

Tous les tests en échec d’un dossier d’exécution

Problème

Généralement, tous les plans d'exécution sont structurés dans une hiérarchie de dossiers qui identifie les différents domaines ou champs d'application auxquels les plans d'exécution et leurs tests sont associés. Les plans d'exécution sont déclenchés régulièrement dans un environnement d'intégration continue, ou occasionnellement dans l'intervalle de temps couvert par la release, générant dès lors de bonnes statistiques d'exécution, mais uniquement pour chaque plan d'exécution malheureusement.

Toutefois, vous avez parfois besoin d'informations globales pour connaître le fonctionnement de tous vos tests dans un domaine ou un champ d'application spécifique, afin de localiser les faiblesses.

Solution

Utilisez la vue datamart *`RV_LatestTestStatus`* pour créer un rapport contenant la liste de tous les tests en échec pour un niveau donné de la hiérarchie d'organisation des exécutions.

La requête suivante sélectionne des tests en échec figurant dans un dossier d'organisation des exécutions, ainsi que des informations contextuelles telles que le nom du plan d'exécution et le nom du build :

```
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 IN (2, 9)
ORDER BY TestName
```

La requête fonctionne comme suit :

- Elle utilise la vue `RV_LatestTestStatus` pour récupérer les résultats de la dernière exécution de test.
- Elle inclut la hiérarchie de l'arborescence des exécutions (`TM_ExecTreePaths`) pour pouvoir rechercher tous les tests dans tous les plans d'exécution de la hiérarchie.
- Elle utilise l'ID de dossier de niveau supérieur à partir duquel l'analyse doit débuter, comme `ParentNodeID_pk_fk`.
- Elle inclut uniquement les états de test en échec et non résolu (`StatusID IN (2,9)`).

Les numéros 2 et 9 associés à `StatusID` figurent dans la table `TM_TestDefStatusNames`. Silk Central mappe les états comme suit :

- réussi et non supporté -> réussi.
- en échec et non résolu -> en échec.
- non exécuté et N/A -> non exécuté.

La requête SQL renvoie tous les tests du dossier d'exécution sélectionné, dont la dernière exécution a échoué.

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

Etat d’un cycle de test

Problème

Les cycles de test peuvent être des objets complexes car ils contiennent des informations sur les testeurs manuels, les tests, les différents builds et versions des applications, voire même des configurations. Pour ne pas perdre le fil, il est important d'obtenir les réponses aux questions suivantes :

- Combien de tests ont été réalisés ?
- Combien d'entre eux ont réussi ou ont échoué, par build ?
- Mes testeurs manuels sont-ils toujours occupés ou peuvent-ils effectuer d'autres tâches ?

Solution

Utilisez la vue de datamart *`RV_TestingCycleStatus`* pour créer un rapport qui affiche l'état d'un cycle de test par testeur et par build, vous offrant ainsi un aperçu du nombre de tests réussis, en échec, non exécutés, groupés par testeur manuel, par configuration et par build.

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

La requête fonctionne comme suit :

- Elle utilise la vue `RV_TestingCycleStatus` comme source de données, car elle contient les éléments `BuildName`, `TesterLogin` et `TesterExecutionName` représentant les noms générés qui reflètent le testeur, la configuration et le test.
- Limite les données à l'ID du cycle de test qui vous intéresse.

Le résultat de la requête SQL affiche l'état par build et par testeur.

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

Pour les tests non assignés, un groupe « Aucun testeur spécifique » est créé avec des valeurs vides pour `TesterLogin`,`TesterFirstName` et `TesterLastName`.

Si vous souhaitez simplement connaître l'évolution de votre cycle de test à partir des performances des testeurs manuels, il vous faudra modifier légèrement la requête comme suit :

```
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
```

La requête est élargie par :

- `GROUP BY TesterLogin, TesterExecutionName` pour indiquer les colonnes restantes.
- `SUM( )` qui, appliqué aux totaux, permet d'additionner les différents états de test.

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

État de l'arborescence des exécutions

Problème

Il est courant d'organiser les plans d'exécution en structure hiérarchique représentant différents objectifs ou domaines de tests. Dans certains cas par exemple, pour connaître l'état du test et par conséquent la qualité atteinte pour un objectif ou un domaine, vous devrez connaître le nombre total de tests réussis, en échec et non exécutés.

Solution

Utilisez la vue de datamart *RV_ExecutionPlanStatusRollup* pour créer un rapport qui indique le nombre de tests réussis, en échec et non exécutés groupés par build, pour un dossier d'organisation des exécutions spécifique.

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

La requête fonctionne comme suit :

- Elle sélectionne `BuildName` et les totaux par état depuis la vue `RV_ExecutionPlanStatusRollup`.
- Elle spécifie le dossier de niveau supérieur à partir duquel vous souhaitez connaître les états (`ExecutionFolderID`).

Le résultat de la requête SQL affiche les états des exécutions de test de tous les plans d'exécution du dossier sélectionné, regroupés par build.

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

Si vous souhaitez obtenir plus de détails comme les totaux par état de tous les plans d'exécution de la hiérarchie sélectionnée, vous pouvez utiliser la vue du datamart *RV_ExecutionPlanStatusPerBuild* :

```
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
```

La requête fonctionne comme suit :

- Elle utilise la vue `RV_ExecutionPlanStatusPerBuild` pour accéder aux données propres au plan d'exécution (`ExecutionPlanID` et `ExecutionPlanName`). La vue `RV_ExecutionPlanStatusRollup` utilisée précédemment contient des données préagrégées (synthétisées), ce qui n'est pas adapté à l'objectif visé ici car les résultats obtenus ne couvrent pas uniquement les plans d'exécution, mais aussi les nœuds de dossier.
- Elle sélectionne tous les nœuds contenus dans un dossier spécifique avec un `JOIN` de la table `TM_ExecTreePath` afin de récupérer les informations relatives à la hiérarchie.
- Elle spécifie le dossier de niveau supérieur avec `ExecutionFolderID`. Étant donné que la table `TM_ExecutionTreePaths` contient également une autoréférence pour chaque plan d'exécution, vous pouvez également exécuter cette requête avec un ID de plan d'exécution pour `ParentNodeID_pk_fk`. Vous obtiendrez alors les lignes correspondant au plan d'exécution en question.
- Elle ajoute `ORDER BY BuildOrderNumber` et `ExecutionPlanID` pour obtenir des résultats bien triés, avec les plus anciens builds et leurs plans d'exécution en premier.

Le résultat de la requête SQL affiche les états des exécutions de test de tous les plans d'exécution du dossier sélectionné.

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

État d'une suite de configurations

Problème

Les suites de configurations vous permettent d'exécuter le même jeu de tests sur plusieurs configurations, par exemple plusieurs navigateurs ou systèmes d'exploitation. Pour être en mesure de faire des déclarations raisonnables sur la qualité et la fiabilité d'une application testée, vous devez conserver un suivi des résultats pour chacune des configurations.

Solution

Utilisez la vue datamart *`RV_ConfigurationSuiteStatus`* pour créer un rapport qui indique le nombre de tests réussis, en échec et non exécutés, par build et pour chaque configuration.

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

La requête fonctionne comme suit :

- Récupère les totaux par état des exécutions de tests par build, à partir de la vue `RV_ConfigurationSuiteStatus`.
- Restreint les résultats à la suite de configurations (`ConfigurationSuiteID`).

Le résultat de la requête SQL indique l'état des exécutions de tests pour chaque configuration.

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

Dans cet exemple, nous utilisons l'ID de la suite de configurations pour obtenir toutes les configurations. Il est également possible de restreindre les résultats à des builds spécifiques. Dans ce cas, vous devez inclure `BuildID`,`BuildName` ou `BuildOrderNumber` dans la clause WHERE.

Remarque: La vue *`RV_ConfigurationSuiteStatus`* contient uniquement les totaux agrégés par état, sans aucune donnée spécifique du test. Pour récupérer des données spécifiques du test, vous pouvez par exemple utiliser la vue *`RV_LatestTestStatus`*.

Dépannage

Données erronées ou manquantes

Problème

Lorsque des données sont interrogées dans une vue ou une table du datamart, les résultats répertoriés ne sont pas à jour ou sont manquants.

Résolution

Les vues et les tables du datamart sont régulièrement mises à jour en arrière-plan, mais pas en temps réel. Le chargement des données dans les tables du datamart peut ainsi prendre quelques secondes, voire quelques minutes. Si votre système exécute une charge importante, cela influence les performances du processus de chargement de données qui s'exécute en arrière-plan. En effet, les autres processus, dotés de priorités plus élevées, peuvent bloquer temporairement la tâche DataMartUpdater exécutée en arrière-plan. Réexécutez votre requête plus tard afin de récupérer les données mises à jour.

Si vous vous connectez en tant qu'administrateur système, vous pouvez vérifier l'état actuel du processus de chargement de données. Pour cela, accédez à `http://<server>:<port>/sctm/check/db` et consultez la **Table DM_TestStatus**.

Remarque: Les tests et les exécutions de test dépendantes sont supprimés du datamart si un test est supprimé. Cela s'applique également aux tests supprimés en raison de la réinitialisation des packages de tests.

Le datamart ralentit le système

Problème

Depuis l'exécution du datamart, les performances globales du système semblent être inférieures ou inégales.

Résolution

Même si cette situation ne devrait pas se produire, vous pouvez désactiver le datamart pour voir si cela résout votre problème :

1. Arrêtez le service du serveur d'application.
2. Ouvrez le fichier `TmAppServerHomeConf.xml` à l'aide d'un éditeur de texte. Ce fichier se trouve dans le dossier `/conf/appserver` du répertoire Silk Central sur le serveur d'application.
3. Localisez la balise XML `Config/DataMart/Enabled`, et positionnez la valeur à `false`.
4. Redémarrez le service du serveur d'application.

Référence : Tables et vues de datamart

Les tables et les vues de datamart suivantes permettent un reporting facile et rapide.

DM_TestStatus

La table `DM_TestStatus` constitue la table de base des vues relatives à l'état.

Les autres vues de datamart offrent généralement un accès plus facile aux données détaillées, car cette table ne fournit pas d'accès direct aux informations telles que le nom d'un test. La clé de cette table est la combinaison des colonnes `TestID`,`ExecutionPlanID`,`BuildID` et `TestRunID`.

Ligne	Description
`TestID`	Identifiant du test.
`ExecutionPlanID`	Identifiant du plan d'exécution.
`BuildID`	Identifiant du build.
`TestRunID`	Identifiant de l'exécution de test.
`ExecutionRunID`	Identifie l'exécution au cours de laquelle ce résultat a été généré.
`StatusID`	État de cette exécution de test (voir `TM_TestDefStatusNames`).
`ReasonID`	Motif de l'état de cette exécution de test (voir `TM_ResultStatusReasons`). Peut afficher une valeur nulle.
`PassedCount`	Somme de tous les tests réussis, qui est égale à 0 ou 1 pour les tests et qui peut être plus élevée pour les regroupements de tests.
`FailedCount`	Somme de tous les tests ayant échoué, qui est égale à 0 ou 1 pour les tests et qui peut être plus élevée pour les regroupements de tests.
`NotExecutedCount`	Somme de tous les tests non exécutés, qui est égale à 0 ou 1 pour les tests et qui peut être plus élevée pour les regroupements de tests.
`ProjectID`	ID du projet auquel cette ligne appartient.
`TestStartTime`	Heure à laquelle l'exécution de test a démarré (UTC).
`ExecutionStartTime`	Heure à laquelle l'exécution a démarré (UTC).
`TestDurationInMilliseconds`	Durée de l'exécution du test en millisecondes.
`IsBlocked`	Marque l'exécution du test comme étant bloquée/débloquée.
`DbChangedAt`	Heure de la dernière mise à jour de cette ligne par le datamart de reporting.

Vue RV_TestStatusExtended

La vue `RV_TestStatusExtended` fournit des informations détaillées pour une exécution de test précise.

Cette vue affiche toutes les exécutions de test, contrairement à la vue *`RV_LatestTestStatus`* qui affiche uniquement la dernière exécution d'un test dans le cadre d'un plan d'exécution et d'un build précis. Vous pouvez utiliser cette vue pour créer un rapport *qui répertorie toutes les exécutions de test des builds marqués*. La clé de cette table est la combinaison des colonnes `TestID`,`ExecutionPlanID`,`BuildID` et `TestRunID`.

Remarque: Les tests et les exécutions de test dépendantes sont supprimés du datamart si un test est supprimé. Cela s'applique également aux tests supprimés en raison de la réinitialisation des packages de tests.

Ligne	Description
`TestID`	Identifiant du test.
`ExecutionPlanID`	Identifiant du plan d'exécution.
`BuildID`	Identifiant du build.
`TestRunID`	Identifiant de l'exécution de test.
`ExecutionRunID`	Identifie l'exécution au cours de laquelle ce résultat a été généré.
`StatusID`	État de cette exécution de test (voir `TM_TestDefStatusNames`).
`ReasonID`	Motif de l'état de cette exécution de test (voir `TM_ResultStatusReasons`). Peut afficher une valeur nulle.
`PassedCount`	Somme de tous les tests réussis, qui est égale à 0 ou 1 pour les tests et qui peut être plus élevée pour les regroupements de tests.
`FailedCount`	Somme de tous les tests ayant échoué, qui est égale à 0 ou 1 pour les tests et qui peut être plus élevée pour regroupements de tests.
`NotExecutedCount`	Somme de tous les tests non exécutés, qui est égale à 0 ou 1 pour les tests et qui peut être plus élevée pour les regroupements de tests.
`ProjectID`	ID du projet auquel cette ligne appartient.
`TestStartTime`	Heure à laquelle l'exécution de test a démarré (UTC).
`ExecutionStartTime`	Heure à laquelle l'exécution a démarré (UTC).
`TestDurationInMilliseconds`	Durée de l'exécution du test en millisecondes.
`IsBlocked`	Marque l'exécution du test comme étant bloquée/ débloquée.
`DbChangedAt`	Heure de la dernière mise à jour de cette ligne par le datamart de reporting.
`TestName`	Nom du test.

Ligne	Description
TestDescription	Description du test.
TestParentID	ID du parent du test.
PlannedTimeInMinutes	Durée planifiée pour ce test en minutes.
Reason	Nom du motif. Peut contenir des motifs qui ont été supprimés entre-temps.
ExecutionPlanName	Nom du plan d'exécution.
ExecutionPlanDescription	Description du plan d'exécution.
ExecutionParentFolderID	ID du parent du plan d'exécution.
Priority	Priorité du plan d'exécution : 0 = faible, 1 = moyenne, 2 = élevée.
BuildName	Nom du build utilisé pour cette exécution de test.
BuildDescription	Description du build.
BuildOrderNumber	Numéro d'ordre du build.
BuildIsTagged	1 si le build est marqué, 0 s'il ne l'est pas.
VersionID	ID de la version à laquelle ce build appartient.
VersionName	Nom de la version.
VersionDescription	Description de la version.
VersionOrderNumber	Numéro d'ordre de la version.
ProductID	ID de l'application à laquelle ce build appartient.
ProductCode	Nom de l'application.
ProductDescription	Description de l'application.
ProductOrderNumber	Numéro d'ordre de l'application.

Vue RV_LatestTestStatus

La vue RV_LatestTestStatus fournit l'état et des informations complémentaires sur la dernière exécution d'un test dans le cadre d'un plan d'exécution et d'un build précis.

Utilisez la vue *RV_TestStatusExtended* pour récupérer des données sur toutes les exécutions de test. Vous pouvez l'utiliser pour créer un rapport *qui répertorie tous les tests en échec figurant dans un dossier d'exécution*. La clé de cette table est la combinaison des colonnes TestID,ExecutionPlanID,BuildID et TestRunID.

Ligne	Description
TestID	Identifiant du test.
ExecutionPlanID	Identifiant du plan d'exécution.
BuildID	Identifiant du build.
TestRunID	Identifiant de l'exécution de test.

Ligne	Description
ExecutionRunID	Identifie l'exécution au cours de laquelle ce résultat a été généré.
StatusID	État de cette exécution de test (voir TM_TestDefStatusNames).
ReasonID	Motif de l'état de cette exécution de test (voir TM_ResultStatusReasons). Peut afficher une valeur nulle.
PassedCount	Somme de tous les tests réussis, qui est égale à 0 ou 1 pour les tests et qui peut être plus élevée pour les regroupements de tests.
FailedCount	Somme de tous les tests ayant échoué, qui est égale à 0 ou 1 pour les tests et qui peut être plus élevée pour regroupements de tests.
NotExecutedCount	Somme de tous les tests non exécutés, qui est égale à 0 ou 1 pour les tests et qui peut être plus élevée pour les regroupements de tests.
ProjectID	ID du projet auquel cette ligne appartient.
TestStartTime	Heure à laquelle l'exécution de test a démarré (UTC).
ExecutionStartTime	Heure à laquelle l'exécution a démarré (UTC).
TestDurationInMilliseconds	Durée de l'exécution du test en millisecondes.
IsBlocked	Marque l'exécution du test comme étant bloquée/ débloquée.
DbChangedAt	Heure de la dernière mise à jour de cette ligne par le datamart de reporting.
TestName	Nom du test.
TestDescription	Description du test.
TestParentID	ID du parent du test.
PlannedTimeInMinutes	Durée planifiée pour ce test en minutes.
Reason	Nom du motif. Peut contenir des motifs qui ont été supprimés entre-temps.
ExecutionPlanName	Nom du plan d'exécution.
ExecutionPlanDescription	Description du plan d'exécution.
ExecutionParentFolderID	ID du parent du plan d'exécution.
Priority	Priorité du plan d'exécution : 0 = faible, 1 = moyenne, 2 = élevée.
BuildName	Nom du build utilisé pour cette exécution de test.
BuildDescription	Description du build.
BuildOrderNumber	Numéro d'ordre du build.
BuildIsTagged	1 si le build est marqué, 0 s'il ne l'est pas.
VersionID	ID de la version à laquelle ce build appartient.

Ligne	Description
VersionName	Nom de la version.
VersionDescription	Description de la version.
VersionOrderNumber	Numéro d'ordre de la version.
ProductID	ID de l'application à laquelle ce build appartient.
ProductCode	Nom de l'application.
ProductDescription	Description de l'application.
ProductOrderNumber	Numéro d'ordre de l'application.

Vue RV_MaxTestRunID

La vue RV_MaxTestRunID permet de récupérer le dernier ID d'exécution de test de chaque combinaison de test, plan d'exécution et build.

La clé de cette table est la combinaison des colonnes TestID, ExecutionPlanID et BuildID.

Ligne	Description
TestID	Identifiant du test.
ExecutionPlanID	Identifiant du plan d'exécution.
BuildID	Identifiant du build.
MaxTestRunID	Identifie la dernière exécution du test dans le cadre du plan d'exécution et du build.

RV_TestingCycleStatus

La vue RV_TestingCycleStatus fournit des informations sur l'état des cycles de test.

Cette vue permet de créer un rapport *qui affiche l'état actuel d'un cycle de test.*.

TestingCycleID indique le cycle de test et TesterExecutionID (ainsi que TesterExecutionName,UserID,CapacityInCycle,TesterLogin,TesterFirstName,TesterLastName) sert à identifier le testeur assigné au cycle de test. Pour les tests qui ne sont pas assignés à un testeur spécifique, les lignes UserID,CapacityInCycle,TesterLogin,TesterFirstName et TesterLastName contiennent la valeur null. La clé de cette table est la combinaison des colonnes TesterExecutionID et BuildID.

Ligne	Description
TestingCycleID	Identifiant du cycle de test.
TesterExecutionID	Identifie le groupe de tests assigné à un testeur spécifique.
TesterExecutionName	Nom généré pour le groupe de tests assigné à un testeur spécifique.
UserID	ID utilisateur du testeur.
CapacityInCycleInMinutes	Capacité accordée à cet utilisateur dans ce cycle de test en minutes.

Ligne	Description
TesterLogin	Identifiant de connexion du testeur.
TesterFirstName	Prénom du testeur.
TesterLastName	Nom de famille du testeur.
PassedCount	Somme de tous les tests réussis.
FailedCount	Somme de tous les tests ayant échoué.
NotExecutedCount	Somme de tous les tests non exécutés.
ProjectID	Identificateur du projet.
BuildID	Identificateur du build.
BuildName	Nom du build utilisé pour cette exécution de test.
BuildDescription	Description du build.
BuildOrderNumber	Numéro d'ordre du build.
BuildIsTagged	1 si le build est marqué, 0 s'il ne l'est pas.
VersionID	ID de la version à laquelle ce build appartient.
VersionName	Nom de la version.
VersionDescription	Description de la version.
VersionOrderNumber	Numéro d'ordre de la version.
ProductID	ID de l'application à laquelle ce build appartient.
ProductCode	Nom de l'application.
ProductDescription	Description de l'application.
ProductOrderNumber	Numéro d'ordre de l'application.

RV_ExecutionPlanStatusPerBuild

La vue RV_ExecutionPlanStatusPerBuild récupère les totaux des derniers états de test de chaque plan d'exécution dans le cadre de builds.

Les dossiers et les nœuds enfants ne sont pas pris en compte. Cette vue permet de créer un rapport *qui affiche l'état des exécutions de test pour chacun des plans d'exécution contenus dans un dossier*. Contrairement à *RV_ExecutionPlanStatusRollup*, cette vue dispose d'un léger avantage en termes de performances car la récupération de données s'effectue indépendamment de la hiérarchie. La clé de cette table est la combinaison des colonnes ExecutionPlanID et BuildID.

Ligne	Description
ExecutionPlanID	Identifiant du plan d'exécution.
BuildID	Identifiant du build.
ExecutionPlanName	Nom du plan d'exécution.
ExecutionParentFolderID	ID du parent du plan d'exécution.
PassedCount	Somme de tous les tests réussis.

Ligne	Description
FailedCount	Somme de tous les tests ayant échoué.
NotExecutedCount	Somme de tous les tests non exécutés.
ProjectID	ID du projet auquel le plan d'exécution appartient.
BuildName	Nom du build utilisé pour cette exécution de test.
BuildDescription	Description du build.
BuildOrderNumber	Numéro d'ordre du build.
BuildIsTagged	1 si le build est marqué, 0 s'il ne l'est pas.
VersionID	ID de la version à laquelle ce build appartient.
VersionName	Nom de la version.
VersionDescription	Description de la version.
VersionOrderNumber	Numéro d'ordre de la version.
ProductID	ID de l'application à laquelle ce build appartient.
ProductCode	Nom de l'application.
ProductDescription	Description de l'application.
ProductOrderNumber	Numéro d'ordre de l'application.

RV_ExecutionPlanStatusRollup

La vue RV_ExecutionPlanStatusRollup récupère les totaux des tests réussis, en échec et non exécutés par plan d'exécution ou dossier dans le cadre d'un build.

Dans le cas des dossiers, les compteurs incluent les nombres issus de tous les enfants. Cette vue permet de créer un rapport *qui affiche l'état de toutes les exécutions de test figurant dans un dossier*. La clé de cette table est la combinaison des colonnes ExecutionFolderID et BuildID.

Ligne	Description
ExecutionFolderID	Identifiant du plan d'exécution.
BuildID	Identifiant du build.
PassedCount	Total des tests réussis.
FailedCount	Total des tests ayant échoué.
NotExecutedCount	Total des tests non exécutés.
ProjectID	ID du projet auquel le plan d'exécution appartient.
BuildName	Nom du build utilisé pour cette exécution de test.
BuildDescription	Description du build.
BuildOrderNumber	Numéro d'ordre du build.
BuildIsTagged	1 si le build est marqué, 0 s'il ne l'est pas.
VersionID	ID de la version à laquelle ce build appartient.
VersionName	Nom de la version.

Ligne	Description
VersionDescription	Description de la version.
VersionOrderNumber	Numéro d'ordre de la version.
ProductID	ID de l'application à laquelle ce build appartient.
ProductCode	Nom de l'application.
ProductDescription	Description de l'application.
ProductOrderNumber	Numéro d'ordre de l'application.

RV_ConfigurationSuiteStatus

La vue RV_ConfigurationSuiteStatus répertorie les totaux par état pour toutes les suites de configurations et les configurations par build.

Cette vue permet de créer un *rapport qui affiche l'état de toutes les exécutions de test pour chaque configuration d'une suite de configurations*. La clé de cette table est la combinaison des colonnes ConfigurationID et BuildID.

Ligne	Description
ConfigurationSuiteID	Identifiant de la suite de configurations.
ConfigurationID	Identifiant de la configuration.
ConfigurationName	Nom de la configuration.
BuildID	Identifiant du build.
PassedCount	Total des tests réussis.
FailedCount	Total des tests ayant échoué.
NotExecutedCount	Total des tests non exécutés.
ProjectID	ID du projet auquel cette ligne apparaît.
BuildName	Nom du build utilisé pour cette exécution de test.
BuildDescription	Description du build.
BuildOrderNumber	Numéro d'ordre du build.
BuildIsTagged	1 si le build est marqué, 0 s'il ne l'est pas.
VersionID	ID de la version à laquelle ce build appartient.
VersionName	Nom de la version.
VersionDescription	Description de la version.
VersionOrderNumber	Numéro d'ordre de la version.
ProductID	ID de l'application à laquelle ce build appartient.
ProductCode	Nom de l'application.
ProductDescription	Description de l'application.
ProductOrderNumber	Numéro d'ordre de l'application.