IBM FileNet Case Analyzer Cubes' Deep Customizations

Part 2: Organization-user drill up or drill down

Utilize OLAP parent-child relationship to enhance CA user dimension

3/19/2012
IBM
Er Xing Zhao & Yiwei Song

This is part 2 of the series, continues to fulfill report requirements in part 1. Case Analyzer cubes contain a User dimension which includes all users participated in processes. Various reports can be created base on this. However if organizational information is needed in the reports, CA User dimension is not sufficient. This part leverages OLAP parent-child relationship and LDAP, shows how to impghs Aement an organization-user dimension that can drill up or drill down.

ErXing Zhao, a member of IBM CDL ECM CoE (Center of Excellence) in China. Has rich experience in Enterprise Content Management and Business Process Management since 2007 as a developer and architect. Reach out to him at zhaoerx@cn.ibm.com

Yiwei Song, a member of CDL ECM CoE (Center of Excellence) in China, certified FileNet (4.0) developer. Has 5 years FileNet engagement experience, especially familiar with FileNet BPM. Acted as product expert and served several customer cases across insurance, banking and E&U industries. You may reach him at http://linkedin.com/in/yiweisong.

Refer to the first part of the article:
https://www.ibm.com/developerworks/mydeveloperworks/groups/service/html/communityview?communityUuid=e8206aad-10e2-4c49-b00c-fee572815374#fullpageWidgetId=Wf2c4e43b120c_4ac7_80ae_2695b8e6d46d&file=cd0373b5-30b2-482c-a9d9-dd0c949f15f6
This part of the series produces the report: Monthly ongoing/completed tasks by branch/sub-branch.

Requirement analysis

This report implies 3 factors: task amount, time period and branches list. The task amount measure and the time dimension is already in CA cubes. CA lacks of a branch dimension.

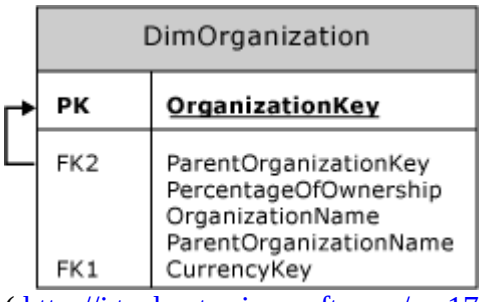
According to the organization chart, XYZ Insurance's organizational units and employees form a hierarchical tree. A hierarchical tree best fits an OLAP dimension with drill up or drill down capabilities.

It's not necessary to create a brand-new dimension, which would mean creating a full set of dimension table and foreign key relationships with fact tables. CA provides the User dimension as well as its underlying dimension table D_DMUser. Multiple fact tables like F_DMQueueLoad contain foreign key to it. It's possible to append organization hierarchies to D_DMUser and then to derive a new dimension.

Parent-child relationship in OLAP

An OLAP dimension may contain parent-child hierarchies. According to Microsoft TechNet document (http://technet.microsoft.com/en-us/library/ms174846(SQL.90).aspx), a parent-child hierarchy is a hierarchy in a standard dimension that contains a parent attribute. A parent attribute describes a self-referencing relationship, or self-join, within a dimension main table.

Figure 1. A sample dimension table presenting parent-child hierarchy



(http://i.technet.microsoft.com/ms174846.13317b8b-3540-48fd-b194-e287f0bddc12(en-US,SQL.90).gif)

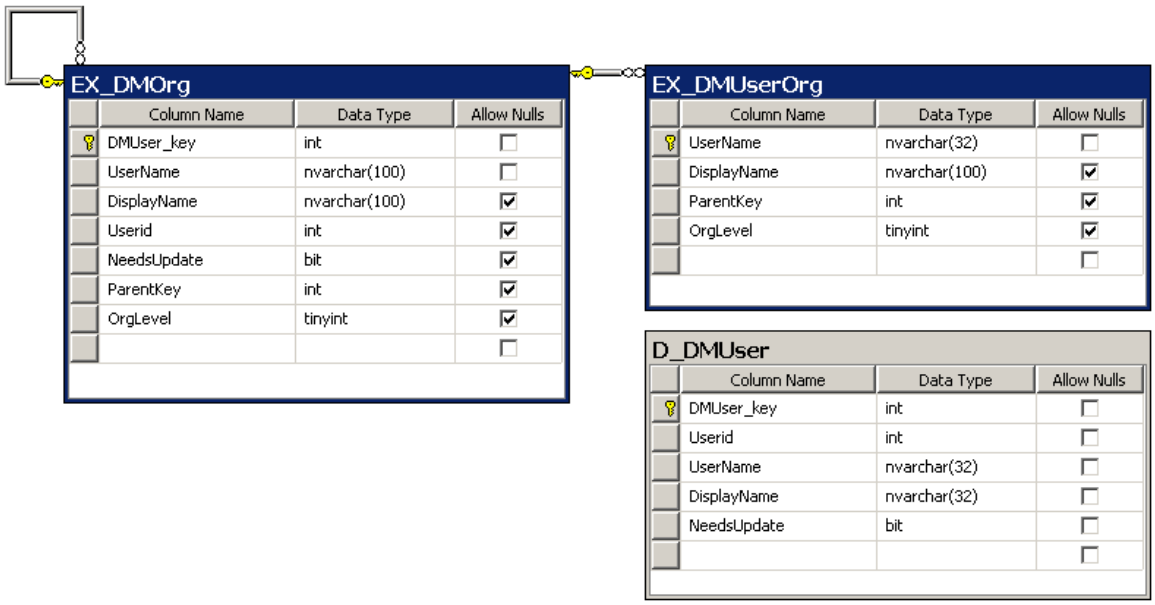
Extending CA user dimension table

The User dimension in CA OLAP only provides a flat view of users who participate in the processes. The dimension has to be extended by applying the parent-child hierarchy.

When customizing the OLAP model, it is best to modify the CA built-in tables/cubes as little as possible. So when needed, it's better to create new tables or dimensions rather than modifying existing ones.

In this article, CA data are stored in CADB database in SQL Server 2005. Below are two tables to be created in the same database. (D_DMUser is a CA built-in table, it's only listed here as reference.) EX_DMOrg will be used for storing organizational unit entries, the ParentKey column records parent-child relationship within the table; while EX_DMUserOrg stores the one-to-many relationship between organizational units and users. Note that DMUser_key is an auto-increase primary key in EX_DMOrg, and it has an increase step of -1.

Figure *2*. New tables supporting parent-child hierarchy



Then a database view EX_DMUser is created consuming newly created tables EX_DMOrg, EX_DMUserOrg and the CA table D_DMUser. The SQL of the view is as follows:

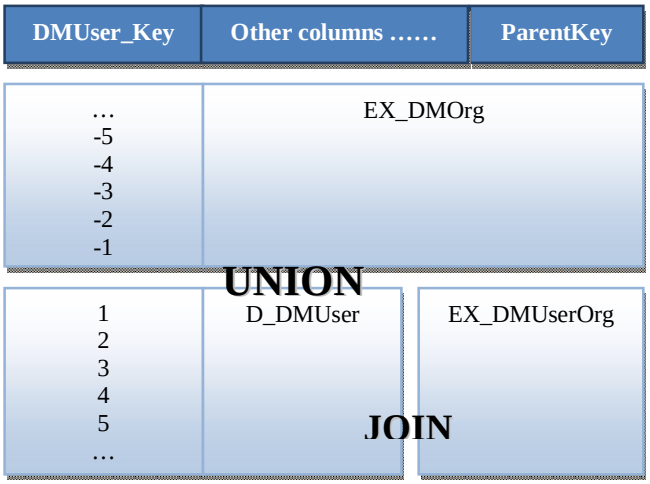
Listing 1. SQL of database view EX_DMUser

```
SELECT DMUser_key, DisplayName, Userid, UserName, NeedsUpdate, ParentKey, OrgLevel
    FROM (
        SELECT DMUser.DMUser_key, UserOrg.DisplayName, DMUser.Userid, DMUser.UserName,
DMUser.NeedsUpdate, UserOrg.ParentKey, UserOrg.OrgLevel
            FROM dbo.D_DMUser AS DMUser LEFT OUTER JOIN dbo.EX_DMUserOrg AS UserOrg ON
DMUser.UserName = UserOrg.UserName
```

```
        UNION ALL SELECT DMUser_key, DisplayName, Userid, UserName, NeedsUpdate,
ParentKey, OrgLevel
            FROM dbo.EX_DMOrg AS Org
    )
WHERE (OrgLevel > 0)
```

Looking into above SQL, it joins D_DMUser with EX_DMUserOrg on the UserName column. Then all EX_DMOrg and expended D_DMUser data are put together via a union. In this union operation, EX_DMOrg and D_DMUser both have the primary key DMUser_Key; however the values won't have any overlap, since the former has negative integer value, while the later has positive integer value. By such design, organizational unit entries and user entries are all mixed up. The foreign key ParentKey of the view points to the primary key DMUser_Key of itself. As a result, the corresponding parent-child relationship is applicable to both organizational units and users. The diagram below shows relationships among the three tables inside the view.

Figure *3*. Logical structure of the EX_DMUser view



This view EX_DMUser is the base of the new dimension.

Importing user-organization info from LDAP using UserOrgSync application

By now, newly created dimension tables are still empty.

Bring up the UserOrgSync project mentioned in part 1 of the series. Make sure the project is compiled successfully and configurations are adjusted accordingly. Now run the JUnit test case:

- com.ibm.cn.ecm.ca.ProcessAnalyzerServiceTest.testImportOrgAndUser()

Once it is successfully executed, the EX_DMUser view is filled with the following data.

Figure *4*. UserOrgSync application imports data into EX_DMUser view

View - dbo.EX_DMUser | Summary

DMUser_key	DisplayName	Userid	UserName	NeedsUpdate	ParentKey	OrgLevel
2	Mike Chen	151	mike	False	-8	99
3	Lei Li	150	lilei	False	-9	99
4	Lucy Liu	152	lucy	False	-10	99
6	Jun Lin	155	linjun	False	-12	99
5	Handson Zhou	153	handson	False	-11	99
7	Qiang Zhang	154	qiang	False	-12	99
-12	NingboSubbranch	NULL	ou=NingboSubbranch,ou...	False	-10	2
-11	XiAnBranch	NULL	ou=XiAnBranch,ou=XYZI...	False	-7	1
-10	ZhejiangBranch	NULL	ou=ZhejiangBranch,ou=...	False	-7	1
-9	BeijingHQ	NULL	ou=BeijingHQ,ou=XYZIn...	False	-7	1
-8	ShanghaiBranch	NULL	ou=ShanghaiBranch,ou=...	False	-7	1
-7	XYZInsurance	NULL	ou=XYZInsurance,dc=cn...	False	NULL	0

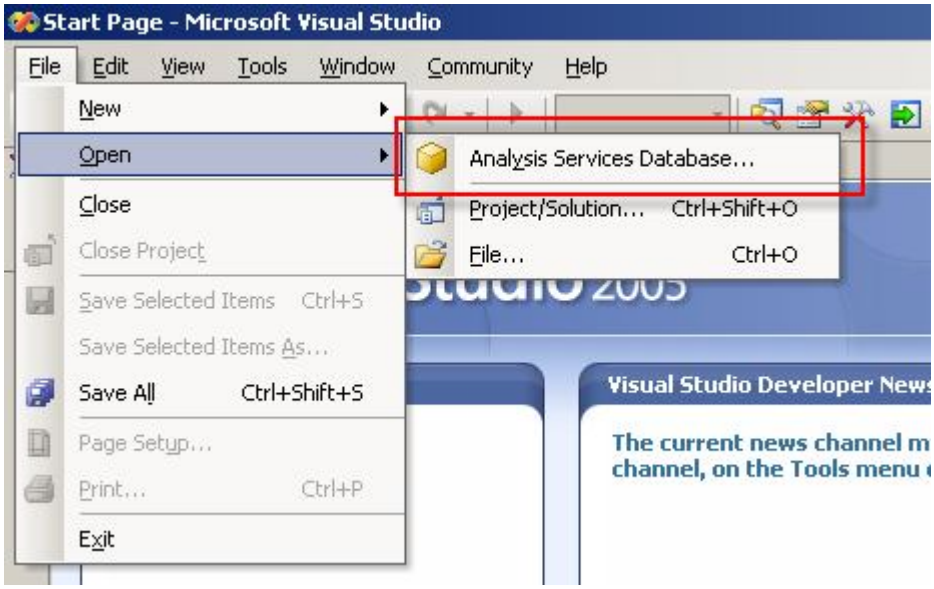
The import is not a one-shot action; the importer will be run regularly in order to reflect to latest organizational structure of the company. It is recommended to wrap the importer by some system service, so that it can be run manually or periodically by some scheduler.

Updating CA cubes with the new user-org dimension

This section will create a new dimension based on database view above and apply it to the CA cubes.

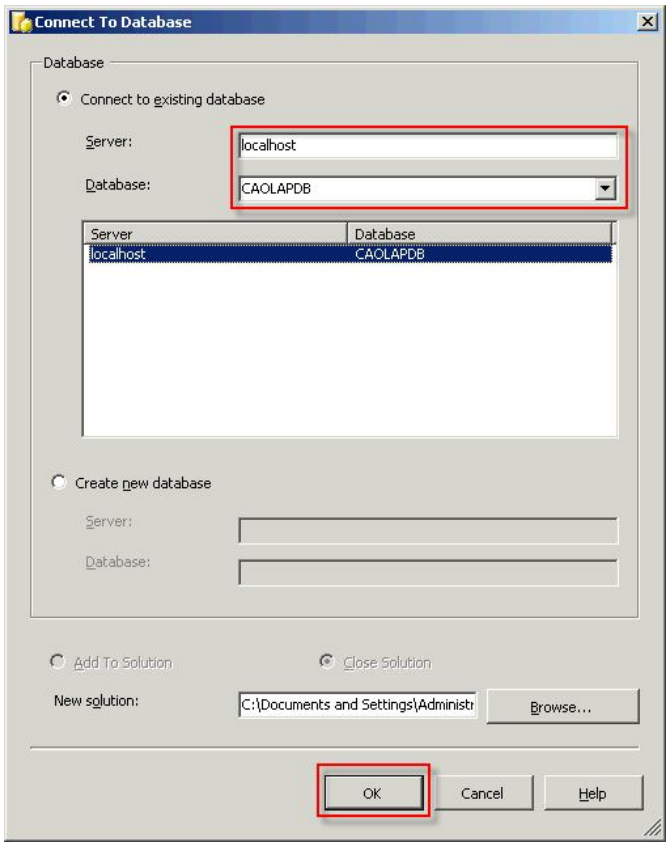
Bring up the Microsoft SQL Server Business Intelligence Development Studio (BI Studio for short), open the existing Analysis Services Database.

Figure *5*. Open existing Analysis Service database in BI Studio



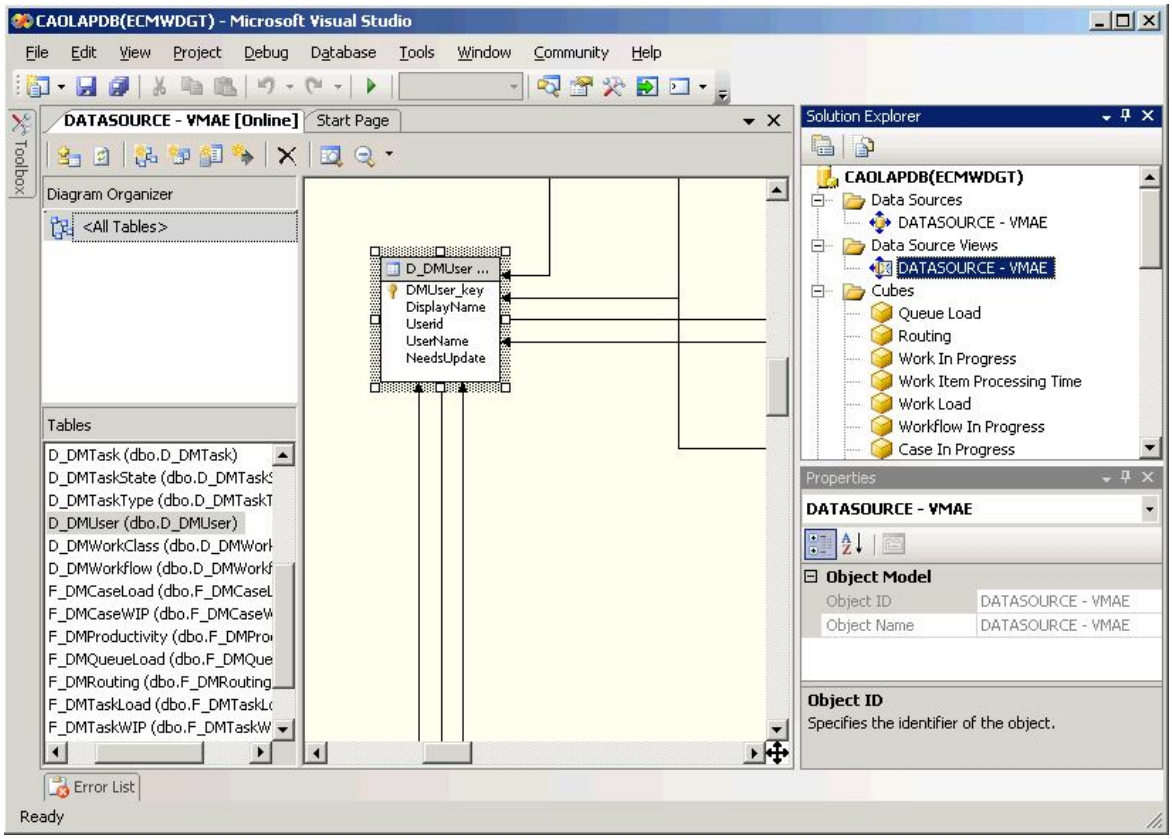
Choose the only database at localhost server.

Figure *6*. Choose a Analysis Service database to open



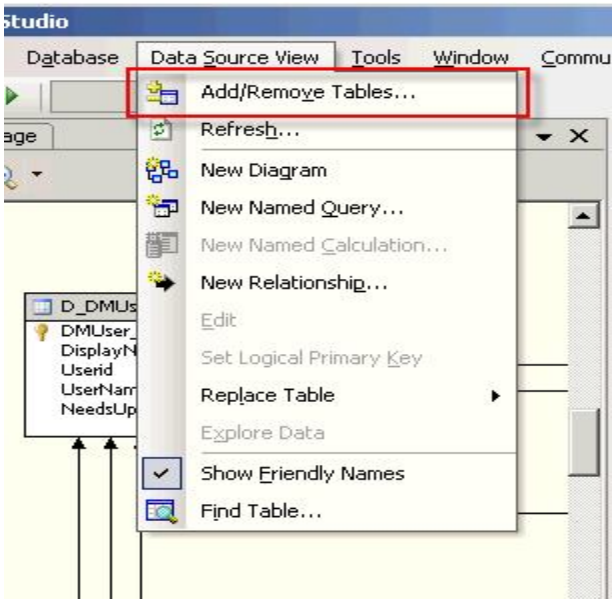
Open the Data Source View: VMAE.

Figure 7. The VMAE Data Source View



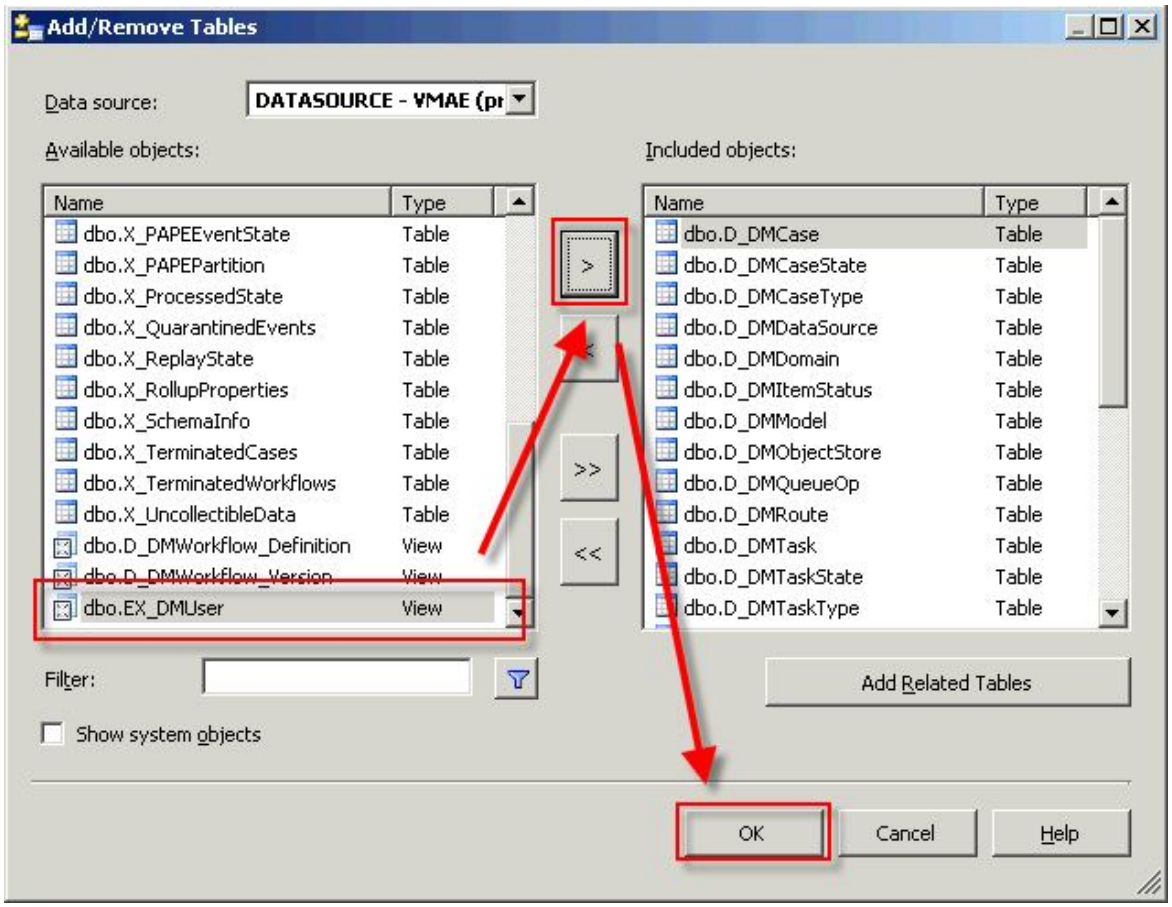
Add the EX_DMUser view from menu.

Figure 8. Add tables to data source view



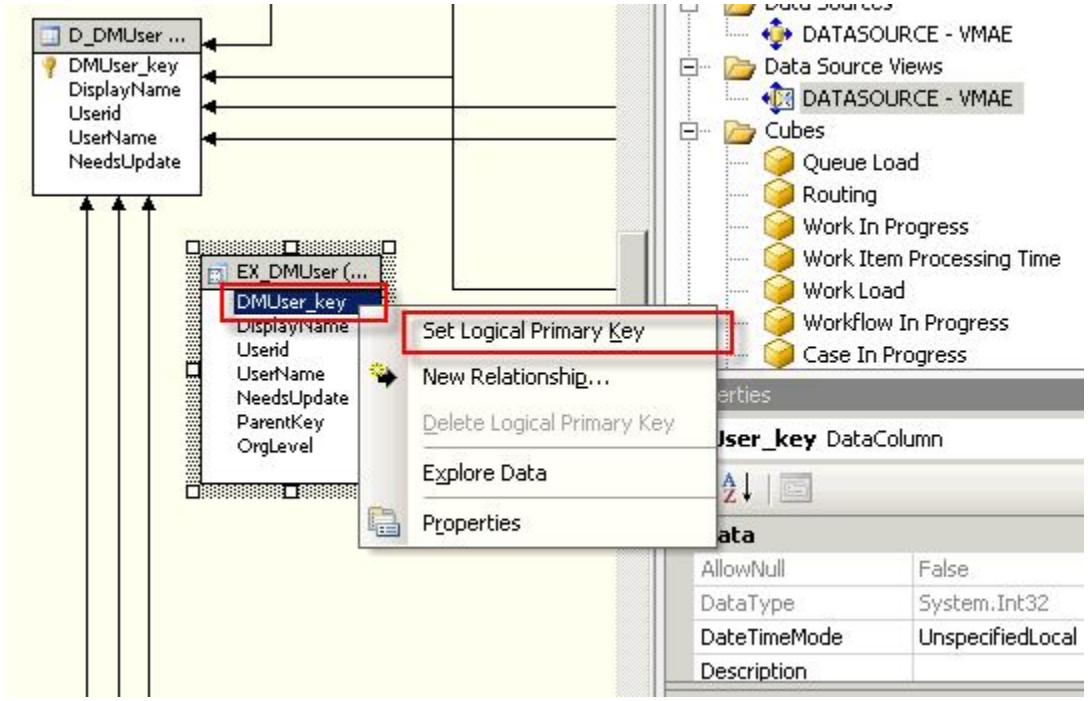
Include the newly created EX_DMUser view and click OK.

Figure *9*. Add newly created EX_DMUser to data source view



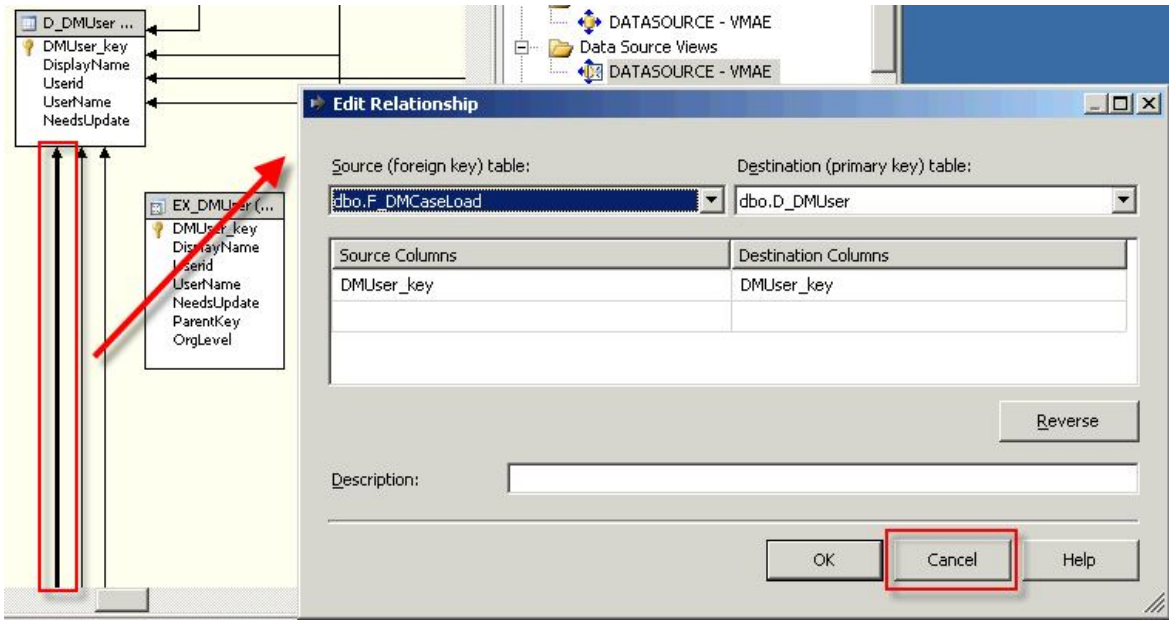
Find the EX_DMUser in the view, set the DMUser_key as Logical Primary Key.

Figure *10*. Set Logical Primary Key of the view



Check the relationships connected to original D_DMUser, double-click the line or right-click the line then choose Edit Relationship.

Figure *11*. Check incoming relationships of D_DMUser



There’s no necessary to modify the existing relationships, just remember it.

You may find there are 7 incoming relationships targeting D_DMUser in CA OLAP (fewer in PA):

- F_DMCaseLoad.DMUser_key -> D_DMUser.DMUser_key
- F_DMCaseWIP.DMUser_key -> D_DMUser.DMUser_key
- F_DMTaskWIP.DMUser_key -> D_DMUser.DMUser_key
- F_DMRouting.DMUser_key -> D_DMUser.DMUser_key
- F_DMWIP.DMUser_key -> D_DMUser.DMUser_key
- F_DMQueueLoad.DMUser_key -> D_DMUser.DMUser_key
- F_DMProductivity.DMUser_key -> D_DMUser.DMUser_key

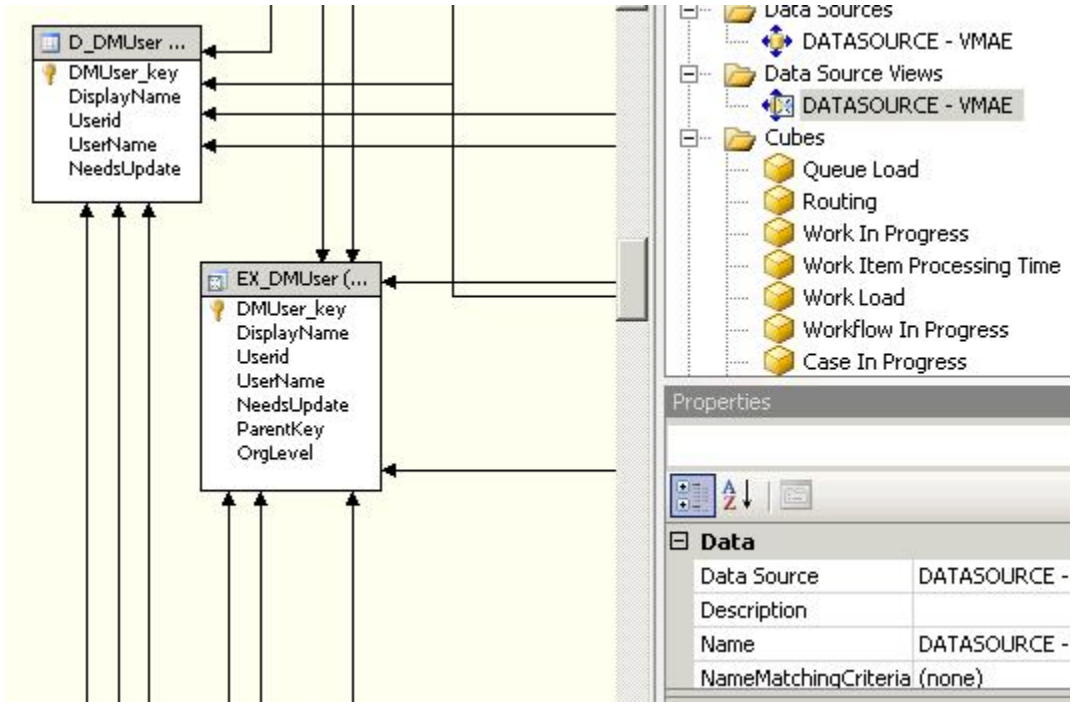
Comparing this, create new relationships for EX_DMUser:

- F_DMCaseLoad.DMUser_key -> EX_DMUser.DMUser_key
- F_DMCaseWIP.DMUser_key -> EX_DMUser.DMUser_key
- F_DMTaskWIP.DMUser_key -> EX_DMUser.DMUser_key
- F_DMRouting.DMUser_key -> EX_DMUser.DMUser_key
- F_DMWIP.DMUser_key -> EX_DMUser.DMUser_key
- F_DMQueueLoad.DMUser_key -> EX_DMUser.DMUser_key
- F_DMProductivity.DMUser_key -> EX_DMUser.DMUser_key

Figure *12*. Create new relationship

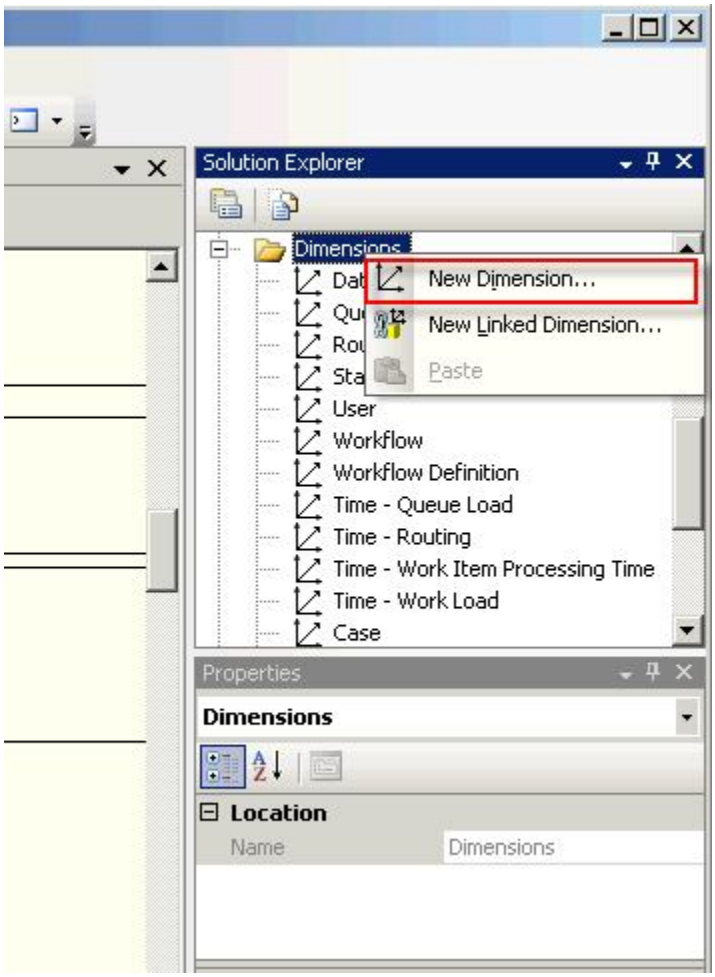
As a result:

Figure *13*. Create similar relationships for EX_DMUser



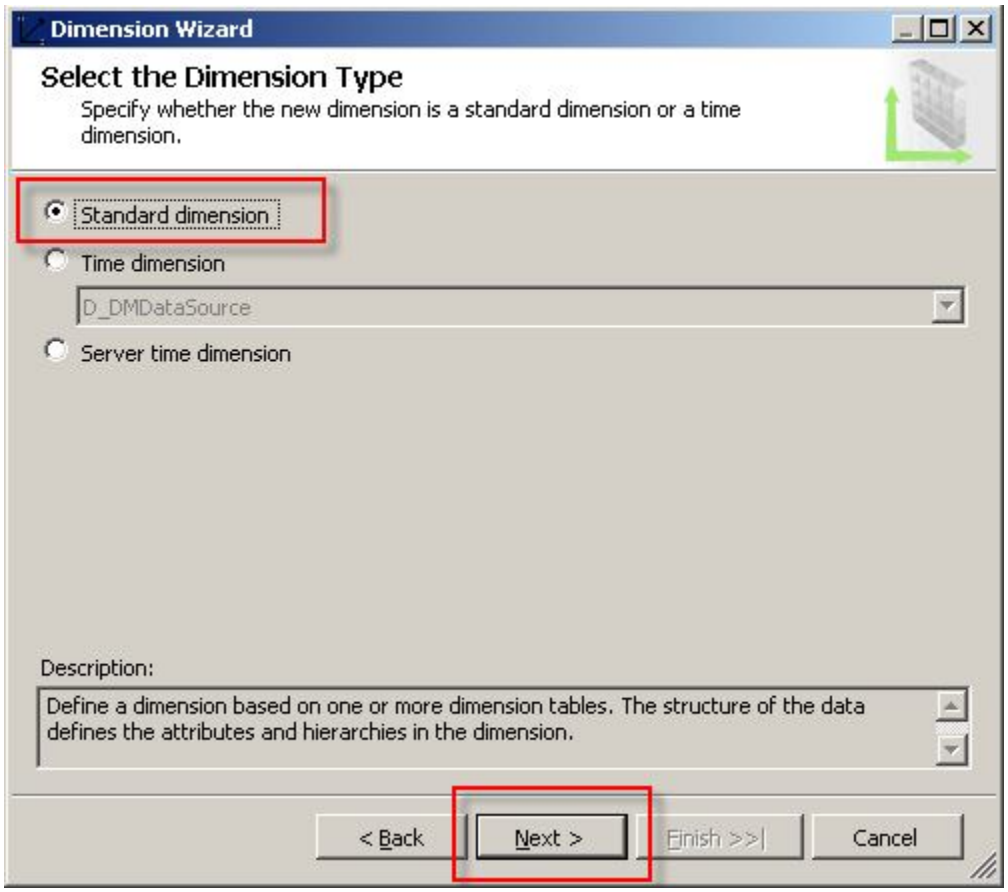
Save the changes. Now create new dimension:

Figure *14*. Create new dimension



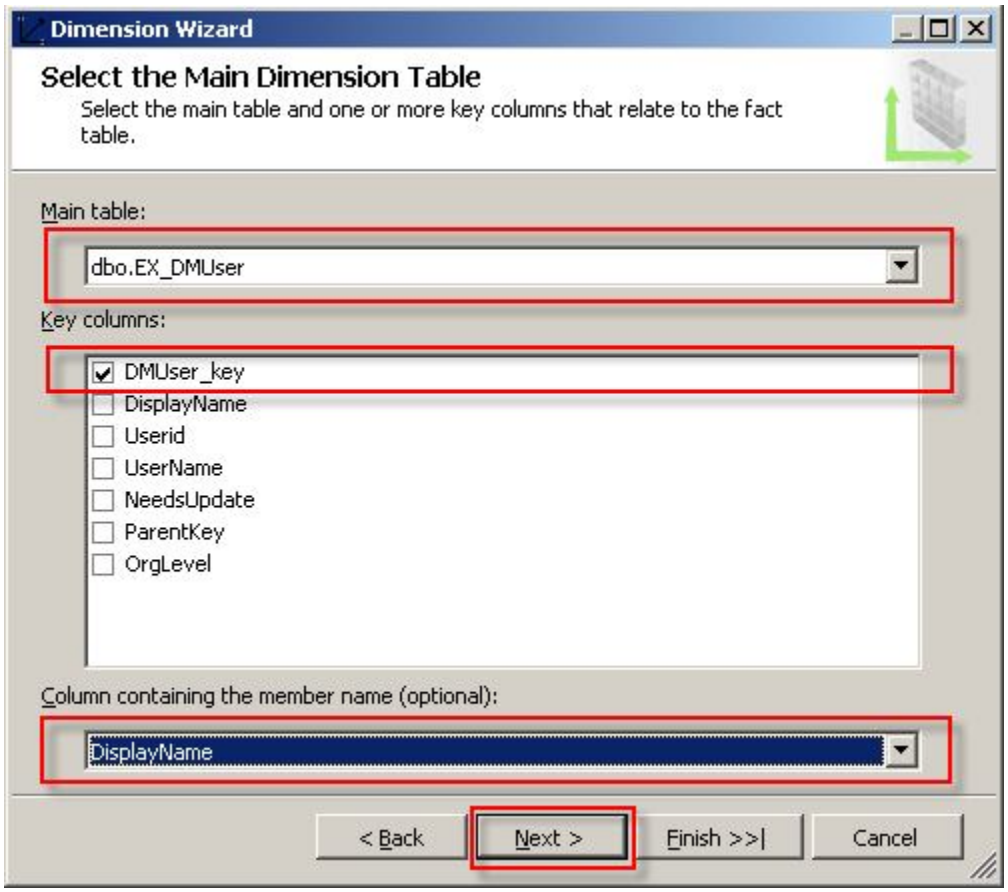
Keep the default values and click Next buttons in the pop-up Dimension Wizard dialog.

Figure *15*. Choose standard dimension type



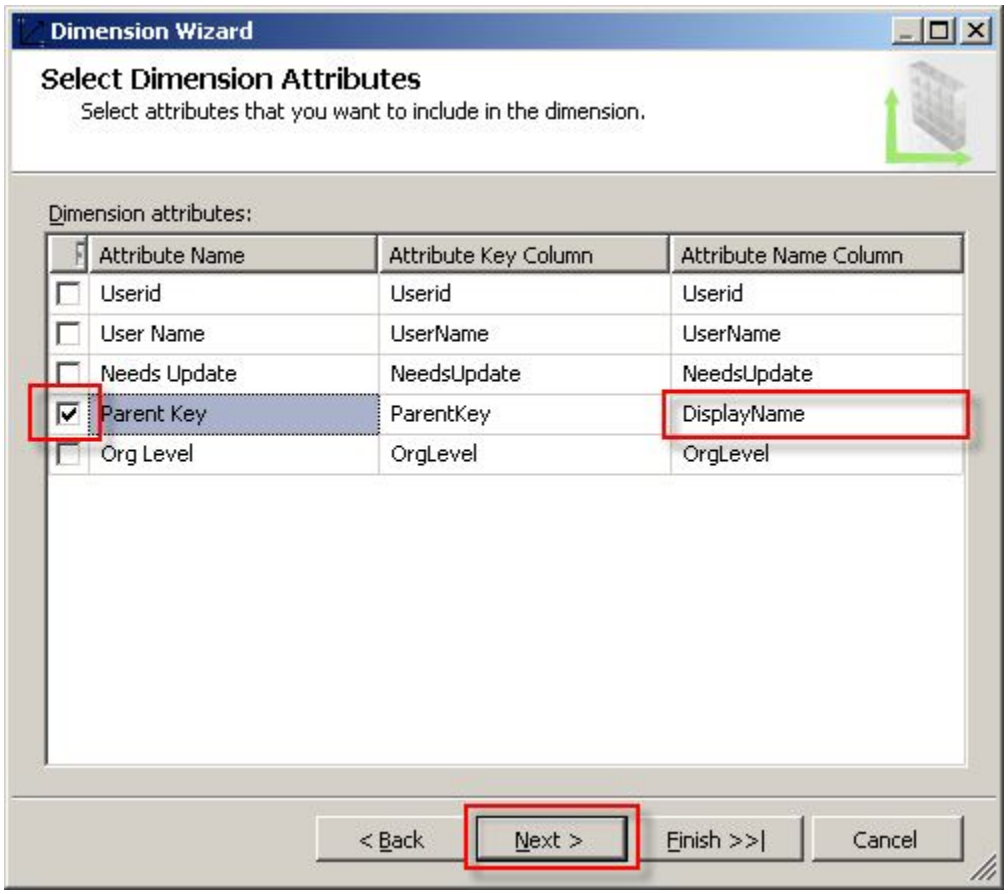
Until the step Select the Main Dimension Table, specify the view EX_DMUser, select DMUser_key as Key column, use DisplayName as member name column. Click Next.

Figure *16*. Select the main dimension table



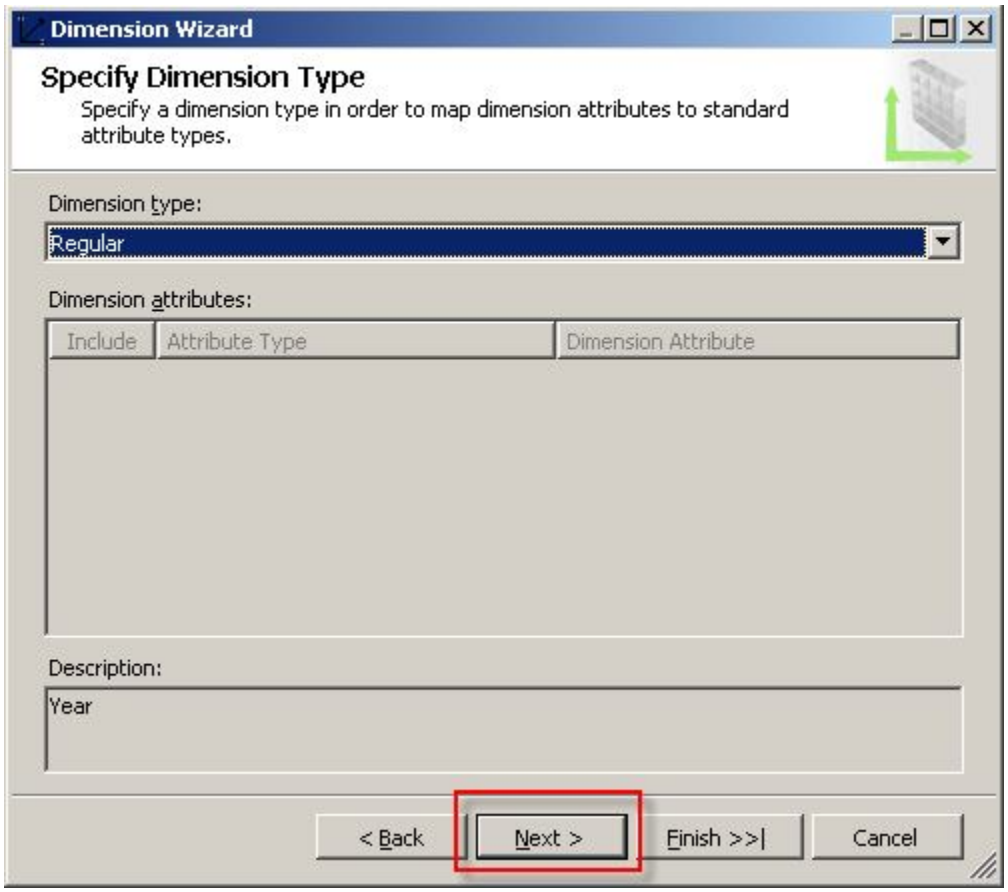
Include the Parent Key attribute in step Select Dimension Attributes.

Figure *17*. Select dimension attributes



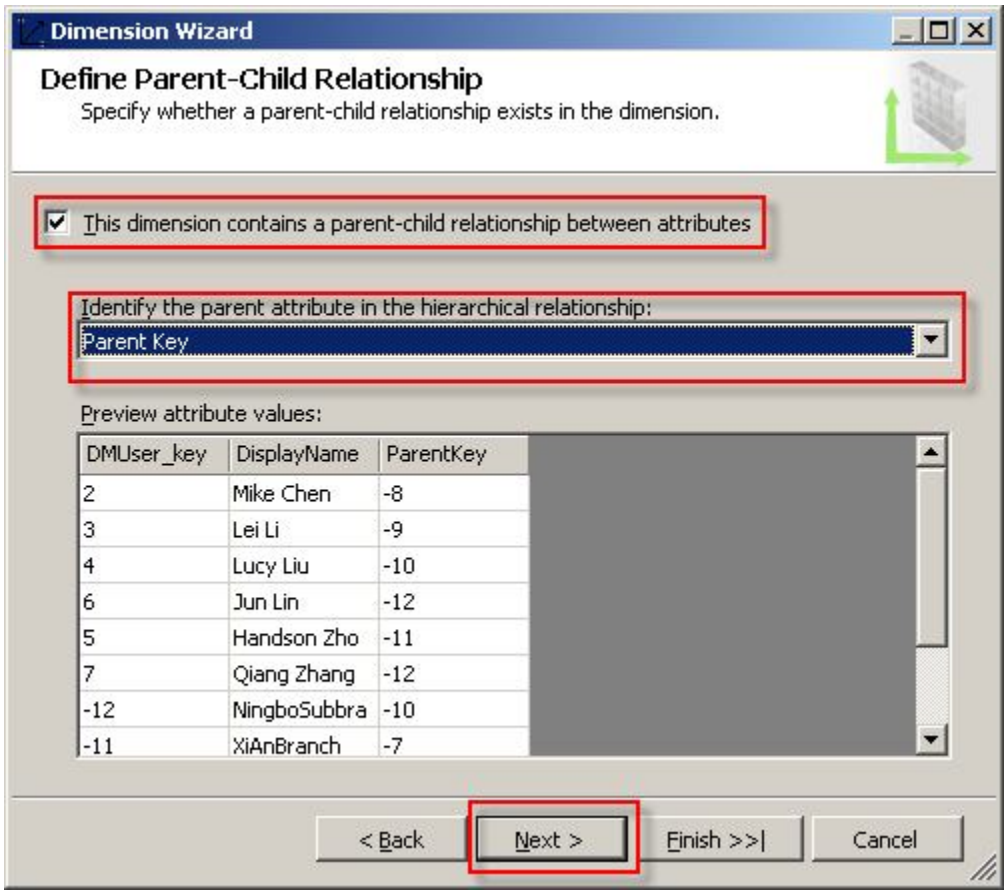
It’s a dimension of regular type.

Figure *18*. Specify dimension type to regular



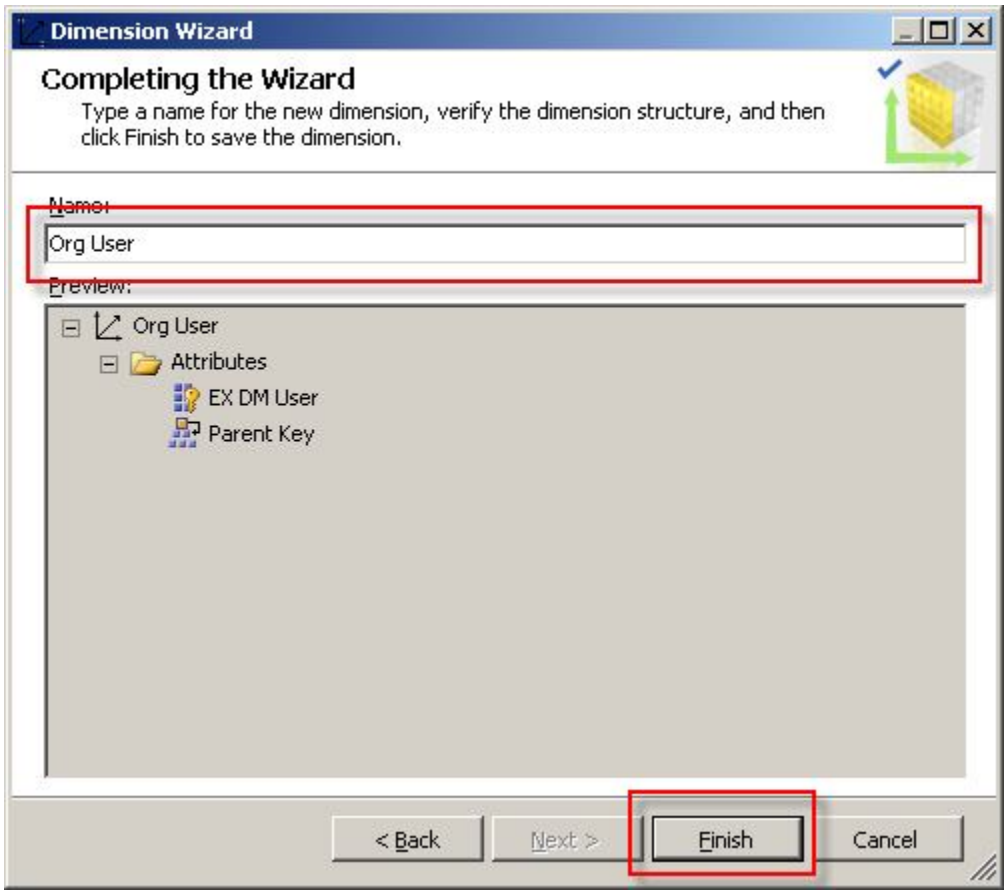
The wizard will find there may be Parent-Child Relationship in the dimension, select Parent Key here.

Figure *19*. Define parent-child relationship



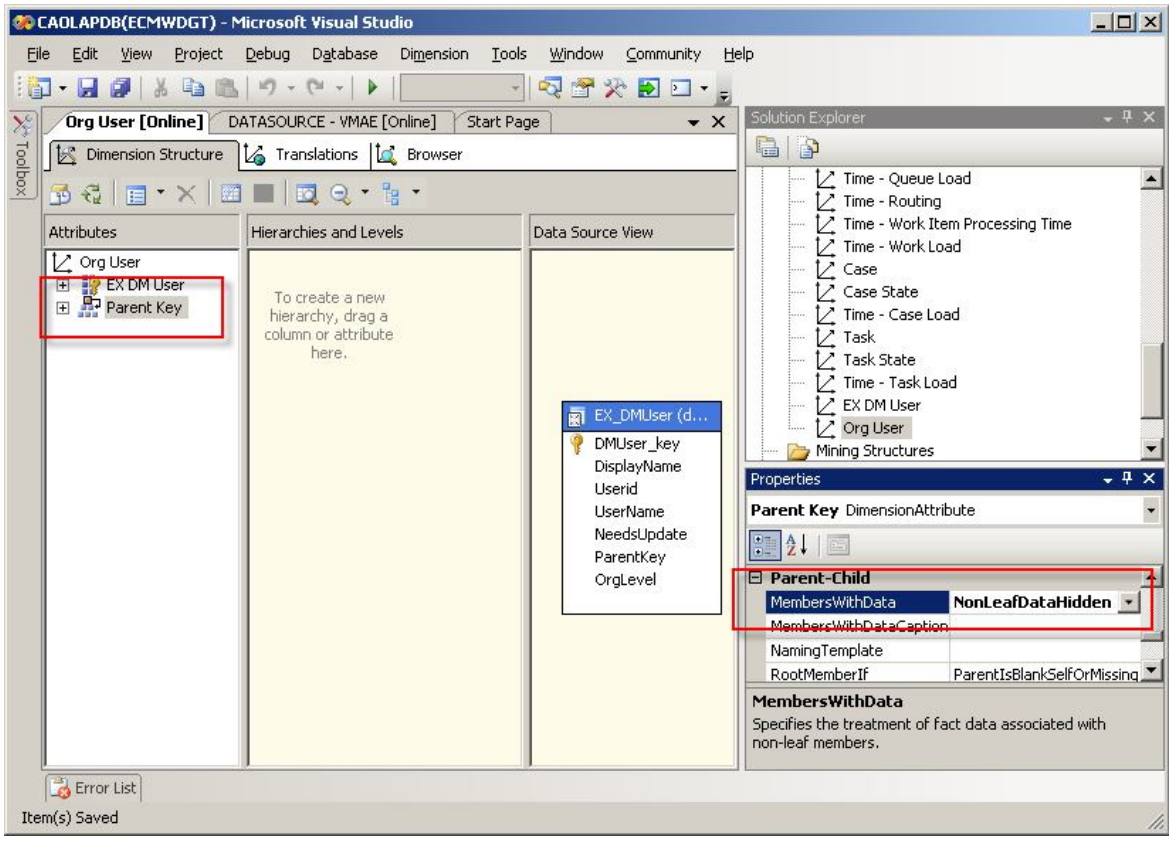
Name as Org User.

Figure *20*. Name the dimension



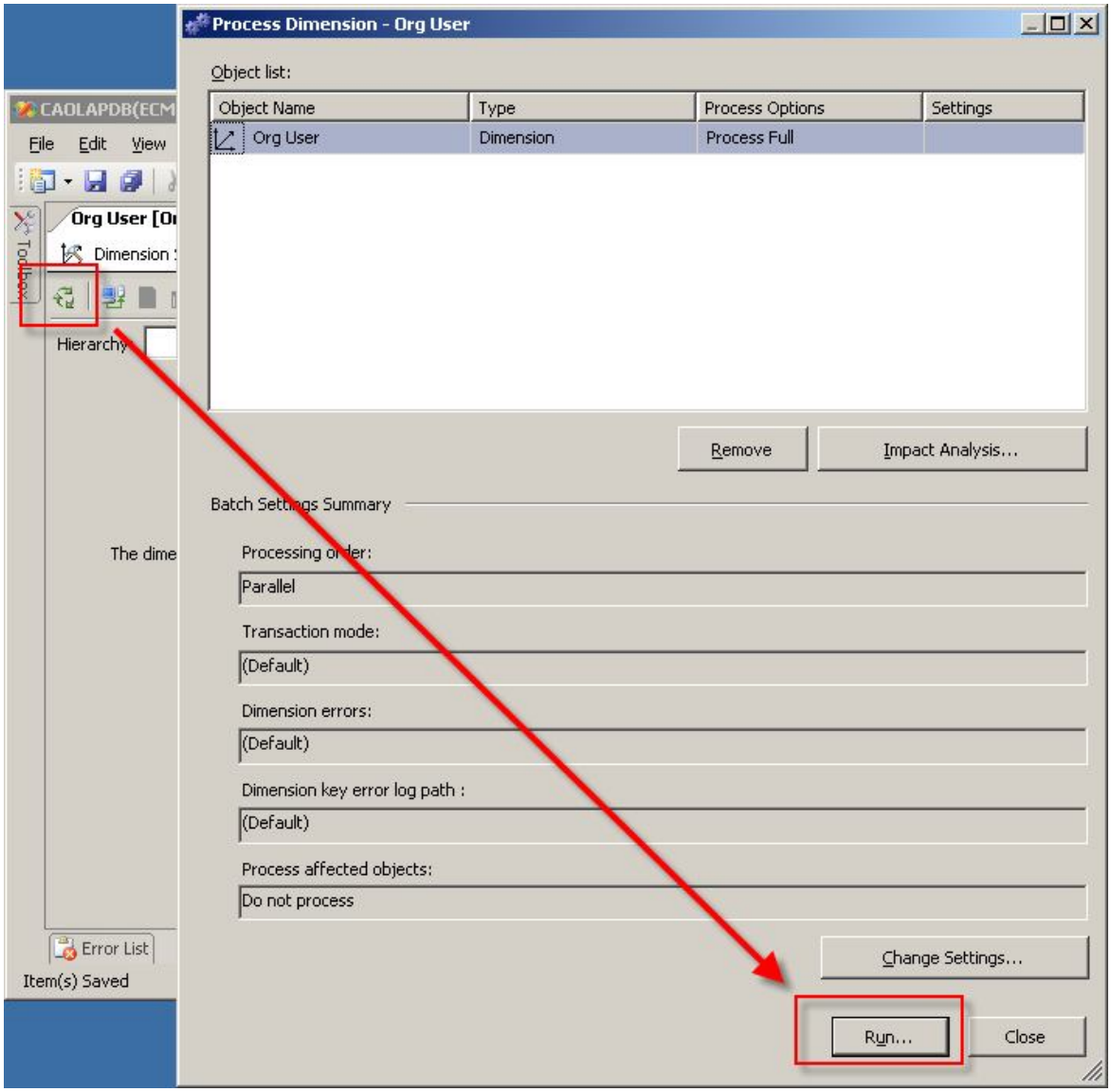
Open the just created Org User dimension, select Parent Key, and then change MembersWithData property to NonLeafDataHidden. Save the change.

Figure *21*. Adjust dimension properties



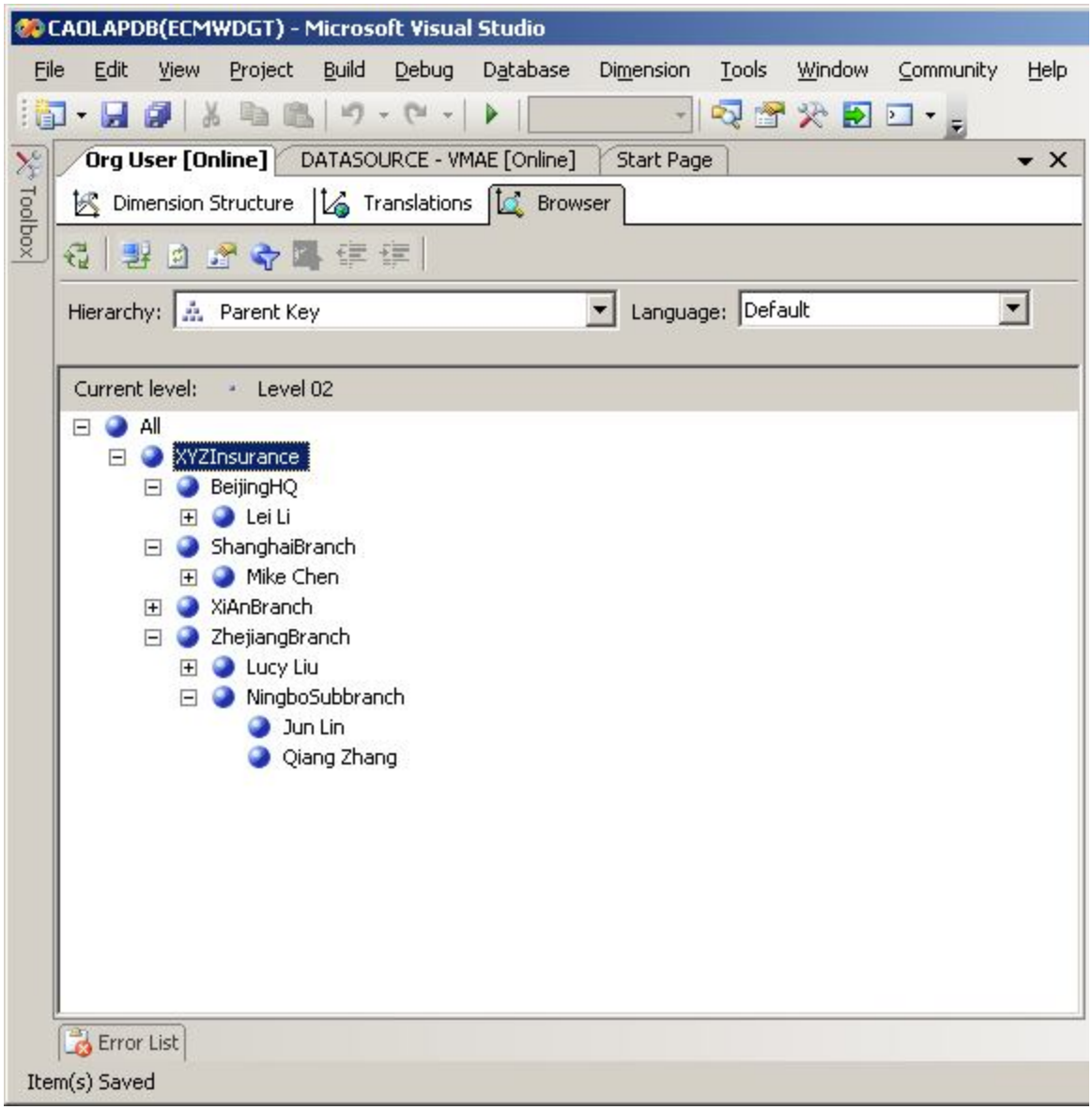
Switch to Browser tab, click Process button.

Figure *22*. Process the dimension



When processing finished, click the Reconnect or Refresh to preview the new dimension.

Figure *23*. Preview the dimension



Now add the new dimension Org User into CA cube “Queue Load”.

Figure *24*. Add the new dimension to a cube

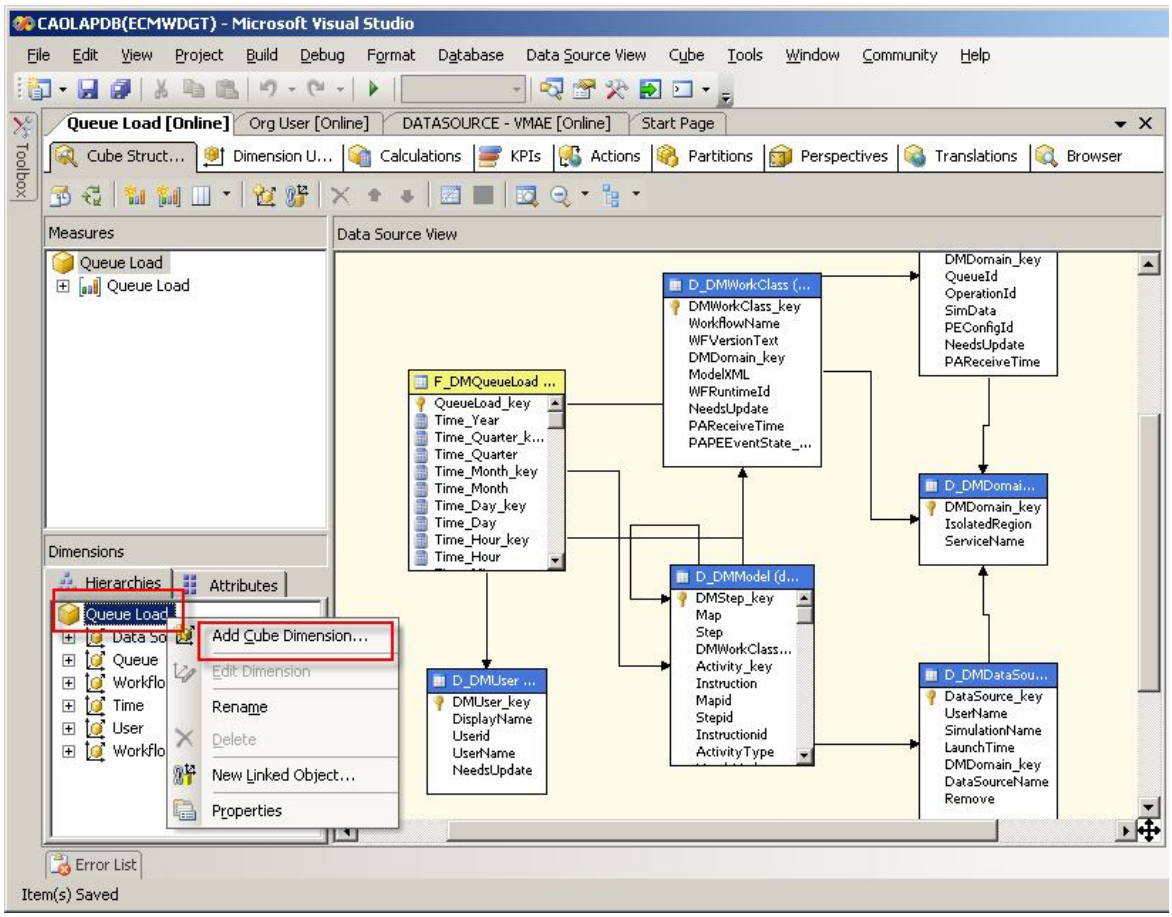
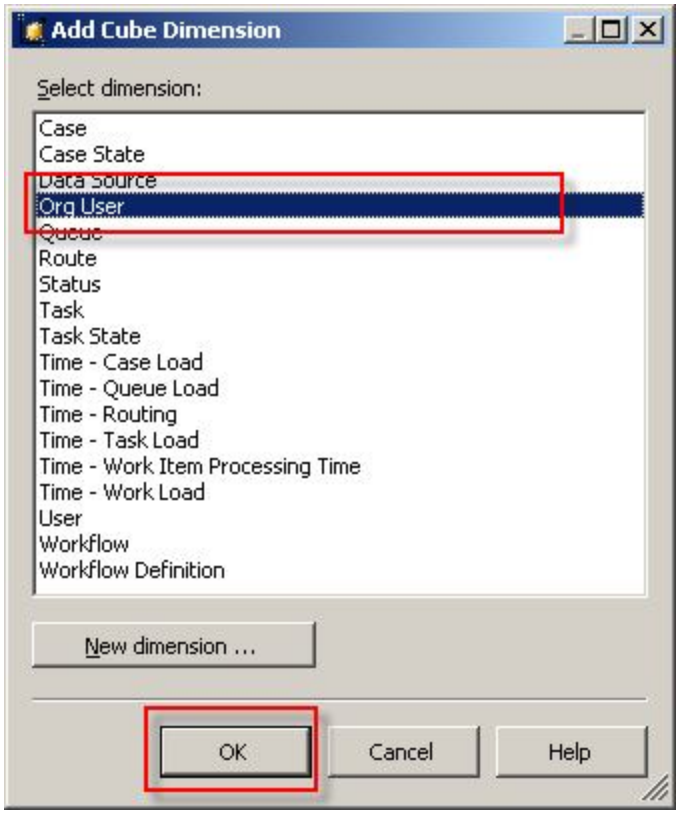
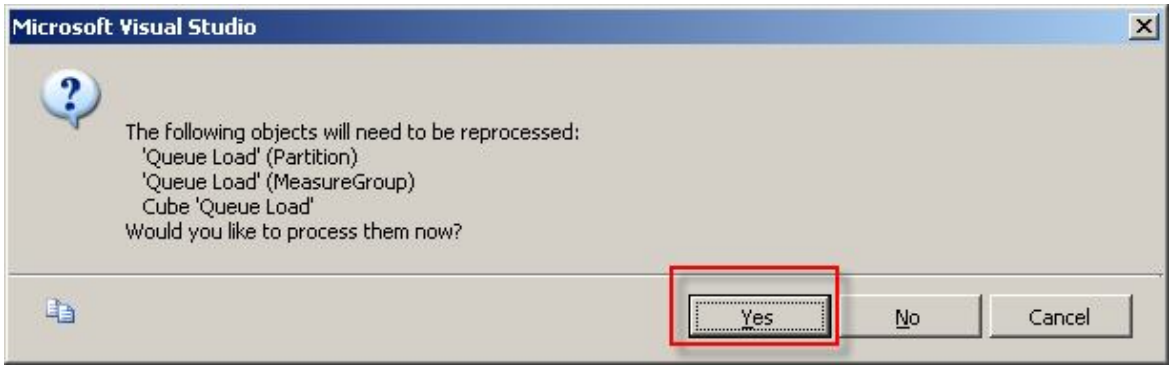


Figure *25*. Choose the Org User dimension to add



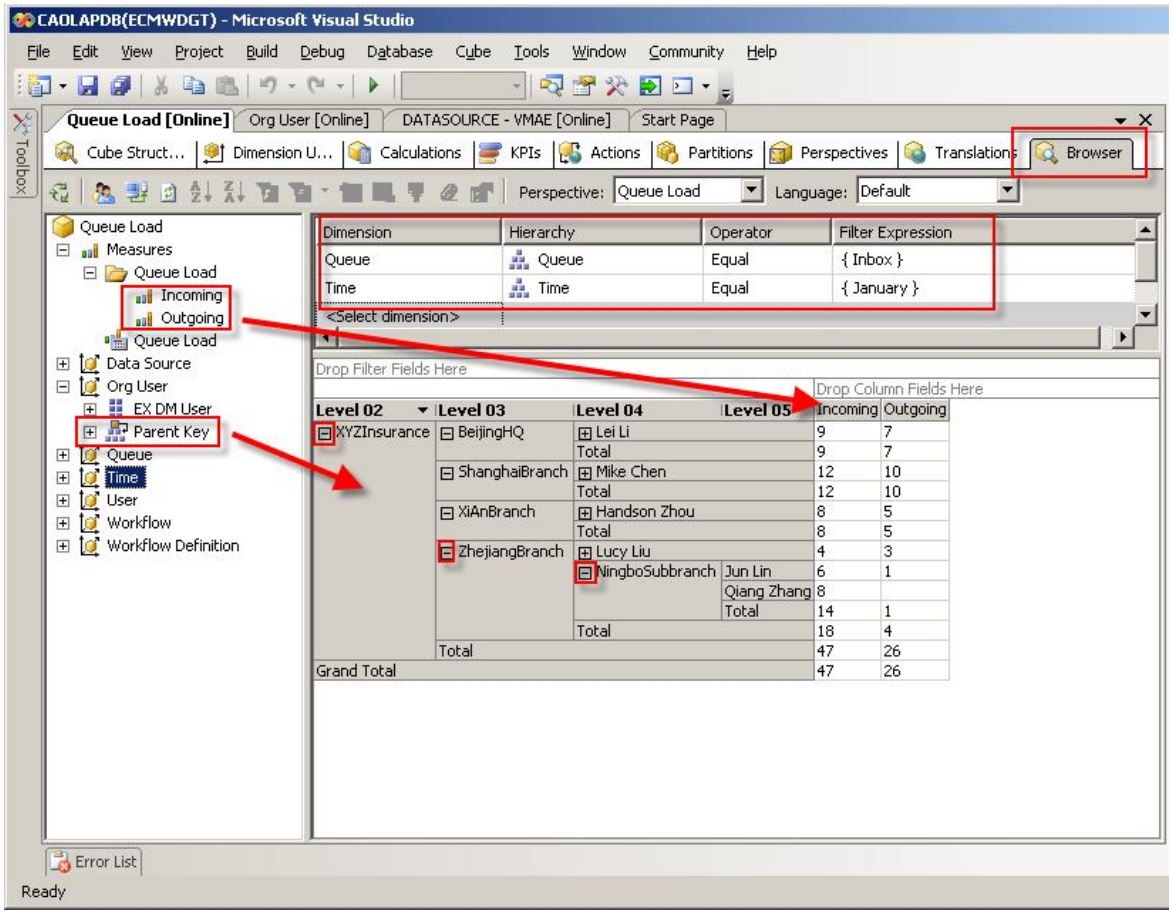
Save and let it process.

Figure *26*. Save and process the cube



Finally check the updated cube in Browser tab, drag Parent Key of Org User to Row Field, add Incoming/Outgoing measures, and apply Queue/Time filters. Here is the result. You can see you may drill up or drill down freely in Org User dimension, and for higher level members, like branches/sub-branches, the measures are automatically summed.

Figure *27*. Preview the cube with new dimension added



This updated cube can be either presented by Excel Pivot Table or Cognos BI.

As CA periodically extracts data and processes cubes, the cube data can be refreshed automatically. Above update won't break the mechanism of CA itself.

Note that, in real use case, if an employee of company is transferred from one branch to another branch (or department), the LDAP data usually changes as well. The Java application that imports user/organization data should be re-invoked. After that, if you try to bring up old reports, you'll find some branches' Total changed. That's because the Total value is calculated in OLAP's runtime. Archive old reports to avoid this.

Summary

Above content not only gives a step by step description of how to implement an organization-user dimension that can drill up or drill down, but also explains the underlying principle, parent-child relationship in OLAP.

The next part 3 continues to introduce how to implement last month amount and month-on-month growth rate.