White Paper:
How QlikTech uses QlikView - SalesForce.com

Best Practices for Reloading Applications relying on SalesForce.com Data

v1.0

MLW

QlikTech

February 2009

www.qlikview.com

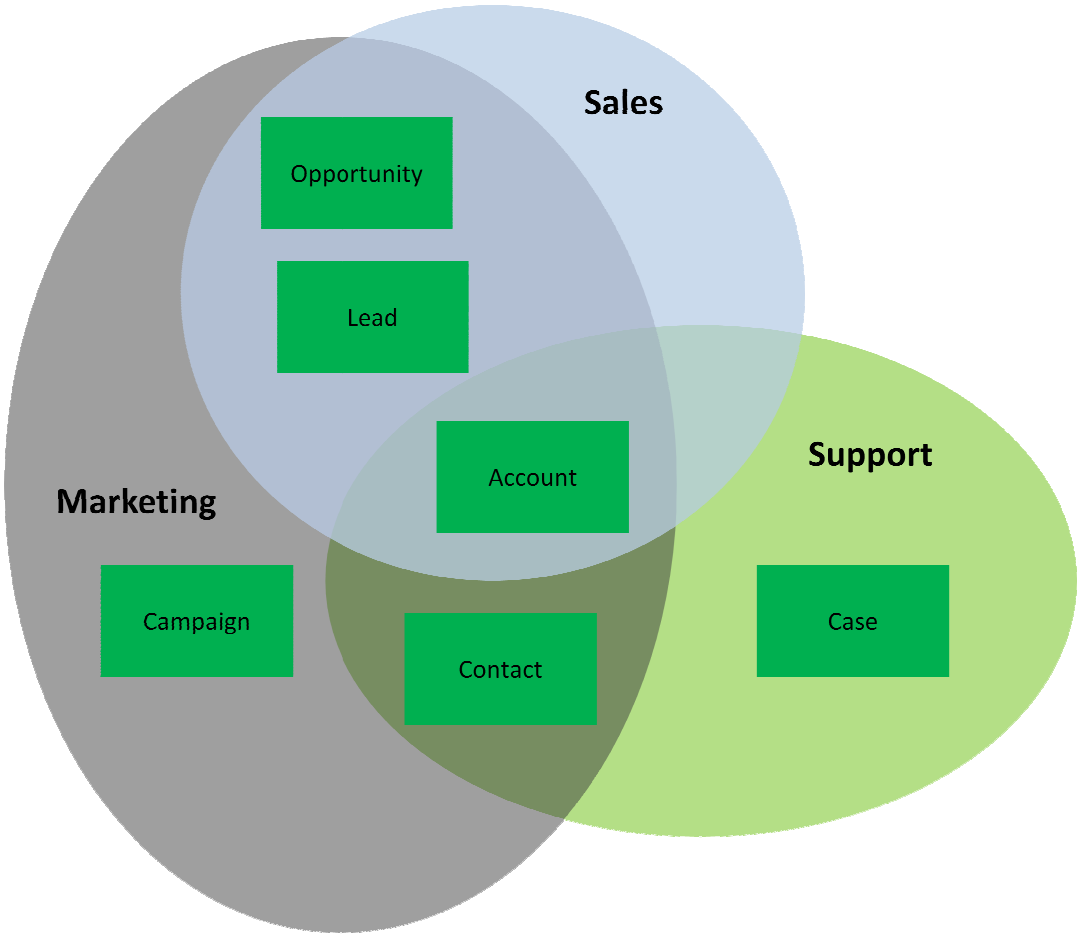
QlikView

White Paper: How QlikTech uses QlikView - SalesForce.com

Table of Contents:

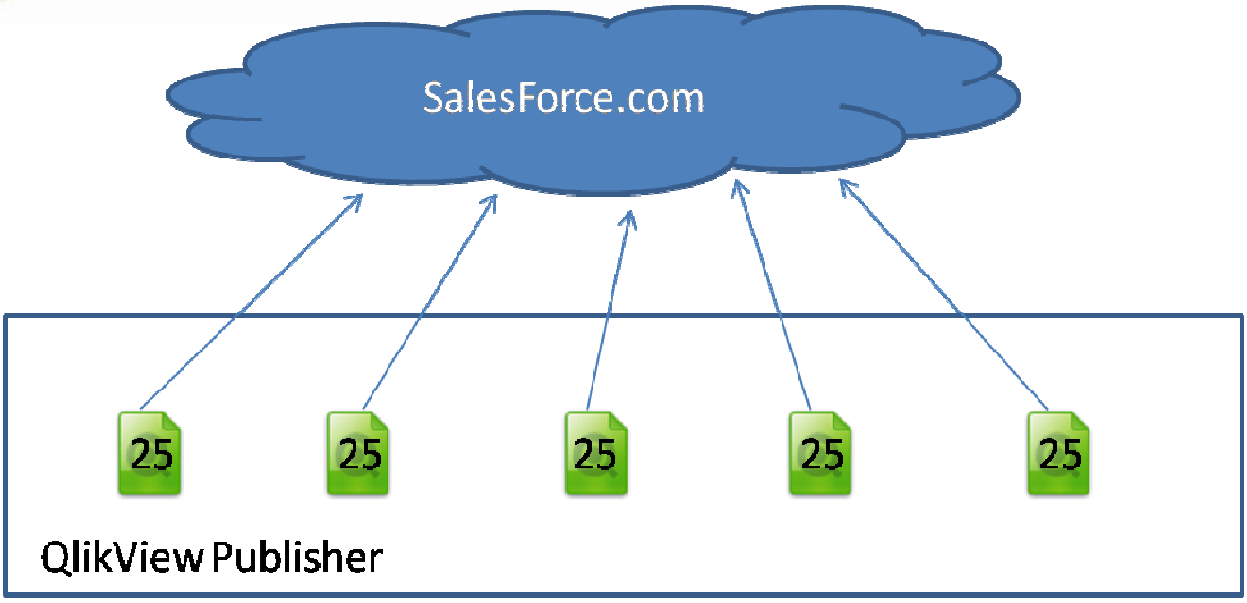
Overview of Loading Data from SalesForce.com

SalesForce.com is an "easy-to-use Web-based CRM solution for sales, service, marketing, and call center operations that streamlines customer relationship management and boosts customer satisfaction." Many organizations rely on SalesForce.com to manage a large part of their business and, as a result, most will have several QlikView applications that will rely on SalesForce.com data for analysis. However, not all applications will require the same data out of SalesForce.com, but a large subset of data will be common among most applications and the organizations that will rely on them.



In order to keep these applications updated, a process would need to be implemented to ensure data is updated as needed. A standard methodology to do this would be to configure each application to connect to SalesForce.com directly, as in the image below.

Scenario I:

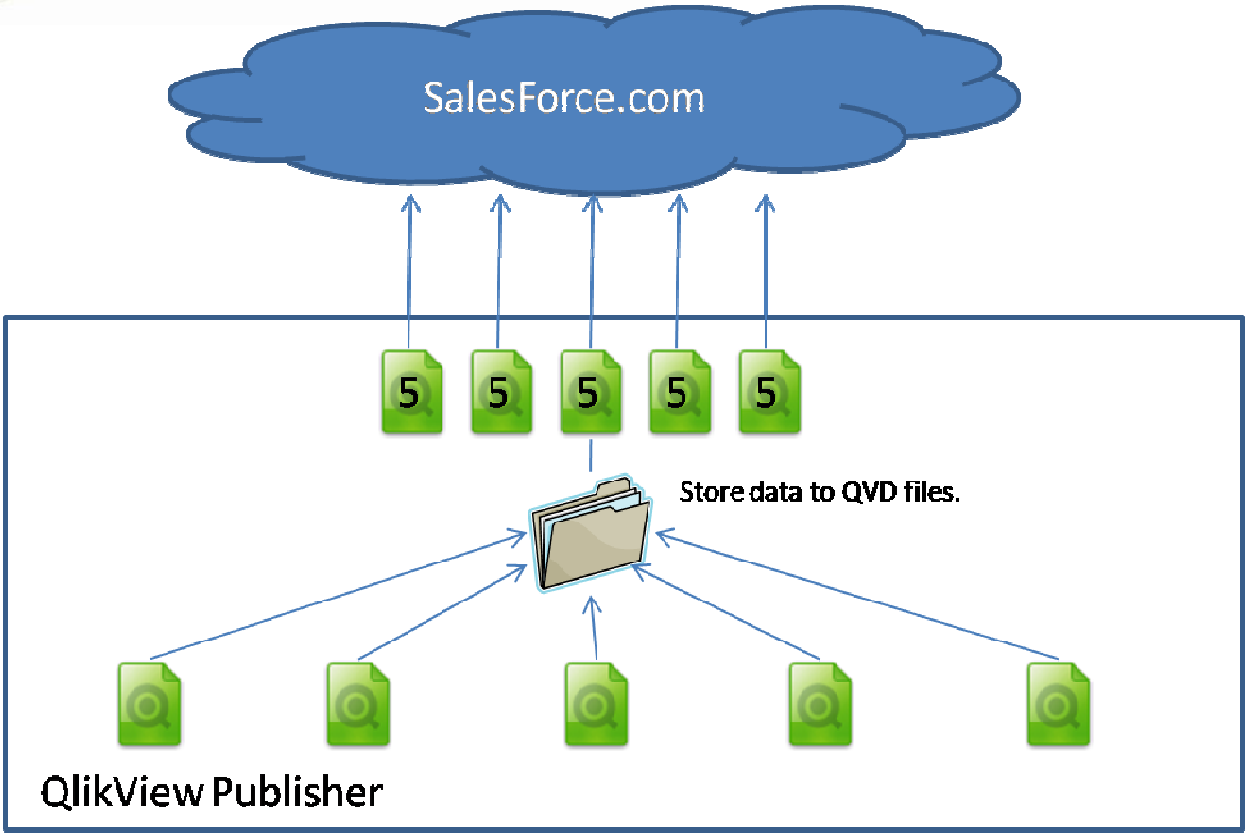


Each application is then setup within QlikView Publisher to reload on schedule. In this example, this would require Publisher to make five separate connections to SalesForce.com, one for each application that is reloading. Assuming each application needs to download 25 tables from SalesForce.com, each single reload process will make 125 connections to SalesForce.com (5 apps * 25 tables). If the intent is to reload these applications every hour, than the number of connections to SalesForce.com per day increases to 3120 daily calls (5 apps * 25 tables * 24 hours). This method has several drawbacks:

- SalesForce.com has a limited number of allowable API calls per day, depending on the organizations license agreement. The more connections made through Publisher, the more risk of running out of API calls needed to reload the applications.
- There is a potential risk of multiple applications attempting to access the same table(s) within SalesForce.com at the same time and therefore one, or all, not getting all the data downloaded properly.

Although each application can be scripted to connect directly to SalesForce.com, this is not a recommended solution. In order to reload the QlikView applications that utilize SalesForce.com, it is suggested to make a single connection to SalesForce.com for each table required, download the necessary data to QVD files, and then reload applications using these QVD files. This is beneficial because it minimizes the number of connections made to the SalesForce.com data source over the internet. It also makes the reloads of the applications much faster as they now use the optimized QVD load as opposed to each individual application connecting to SalesForce.com independently and downloading the same data.

Scenario II:



In this manner, five separate QlikView applications can be created to break out the set of the 25 needed tables. These applications can be scheduled to run simultaneously and download all 25 tables and save them to QVD files. Once the QVD files are created, the applications that require this data can then be scripted to reload off these files and scheduled through Publisher as needed. Using this methodology, the number of calls to SalesForce.com would be reduced to 624 per day (5 apps * 5 tables * 24 hours).

Additionally, it is not recommended to do a full data load of SalesForce.com to QVD multiple times a day. To do this can take a large amount of time and download a very large amount of data over the course of a day. Since all the data is downloaded initially, and all that is necessary afterwards are the changes made to the data over the course of time, a mechanism to handle incremental loads of SalesForce.com data is a more efficient process. This handles loading just a subset of SalesForce.com data that has changed and combining it with data that has not changed, making the reload of the QVD faster as well.

To improve reload time even further, an organization may also want to stagger the timeframes to when certain tables are reloaded out of SalesForce.com. There may be some tables that are rarely updated and, hence, may only require downloading every 3 or 6 hours, as opposed to every hour. Other tables that are updated more frequently can be scheduled to be loaded more frequently such as every hour.

In order to ensure proper data integrity, it is also recommended to run a full SalesForce.com data refresh on a set interval, or as needed. This can be scheduled based on an organizations needs and timeframes. A good practice would be to schedule this run once a week during off business hours.

Incremental Load Process

In order to decrease the reload time of data extracted from SalesForce.com, an incremental process for downloading data should be created. This process uses the dates within SalesForce.com as well as the runtime of the reload process to generate a delta set of data to merge with existing data already downloaded from SalesForce.com. Since existing data within SalesForce.com can be modified at any time, and the changed data is not new data generated after a certain point in time, the script needs to take modified data into account as well as all new data.

The incremental load is a five step process. The process is repeated for every table that needs to be downloaded from SalesForce.com. Here the script is broken down into a step-by-step description. The full script can be found at the end of this section.

Step 1:
Set the current time of the reload process. This value will determine the upper level timestamp to download data. Since all data in SalesForce.com uses a UTC timestamp, a variable is set and then offset by 15 minutes. The offset is just a method to ensure we do not interfere with any data that is currently being modified by a user. This variable is then used throughout the reload process as the upper level timestamp. The variable is set as follows:

```
LET vQVDPath = 'Qvd\';  // Set QVD storage Directory

SET vOffset = '00:15:00';
LET vUTC = UTC();
LET vExecTime = replace(timestamp(vUTC-vOffset, 'YYYY-MM-DDXhh:mm:ssZ'), 'X',
'T');

// SalesForce.com uses a timestamp format of 'YYYY-MM-DDThh:mm:ssZ', hence the
need for the replace statement here.
```

***NOTE*:** This timestamp is only set once within each QVW application, and is not reset for each table downloaded.

Step 2:
Determine the last timestamp of data from the previous load. This is done by loading the current QVD file into memory and determining the maximum of the **LastModifiedDate** and setting this to a variable. In some cases, such as history tables, there is no **LastModifiedDate** so the **CreatedDate** is used instead.

```
SET vLastExecTime             = 0; // resetting vLastExecTime

// As long as a QVD already exists (the isnull() check), find the latest
timestamp for modified records.  This will be used to generate the delta set.
if not isnull(QVDCreateTime('$(vQVDPath)<tablename>.qvd')) then

        LoadTime:
        Load Max(LastModifiedDate) as LastModifiedDate  // Or max(CreatedDate)
        From $(vQVDPath)<tablename>.qvd (qvd);

        Let vLastExecTime =
        replace(timestamp(peek('LastModifiedDate',0,'LoadTime'),'YYYY-MM-
        DDXhh:mm:ssZ'), 'X', 'T');

        Drop Table LoadTime;

end if
```

Step 3:
Download the new and modified data from SalesForce.com. This is accomplished by making a proper connection to SalesForce.com and downloading data from each individual table where the **LastModifiedDate** (or **CreatedDate**) is between the established **vLastExecTime** and the **vExecTme** for the reload process.

```
SQL SELECT Id,
…
FROM <tablename>
WHERE LastModifiedDate >=$(vLastExecTime) and LastModifiedDate < $(vExecTime);
```

Step 4:
Once the delta set is downloaded, the QVD that was created from the previous load is then loaded into the application. A WHERE EXISTS clause is used to concatenate data from the current QVD to the delta set that was downloaded. If the Id of the delta set exists within the QVD, the assumption is that the data in the delta set is the relevant data and so any data within the QVD that matches the unique identifier in the delta set is not loaded. All additional data is concatenated to the delta set table.

```
// Check to see if this is the first reload.  If it is, skip this step
if Not isnull(QvdCreateTime('$(vQVDPath)<tablename>.qvd')) then
       Concatenate (<tablename>)
       LOAD *
       FROM $(vQVDPath)<tablename>.qvd (qvd)
       WHERE Not(Exists (Id));
end if
```

Step 5:
As long as the new table is not empty, the table is then stored to a QVD file to be used for reloading of all other applications that require the SalesForce.com data. This new QVD will also be the source of data in Step 4 for the next incremental reload scheduled task.

```
//If data exists within table, store to QVD.
if NoOfRows('<tablename>') > 0 then
       STORE <tablename> INTO $(vQVDPath)<tablename>.qvd;
       Drop Table <tablename>;
end if
```

The incremental load process can be broken into as many threads as necessary to balance timing with performance and need. Some tables may take longer to load based on data volumes. It is recommended to know all the tables that will be required for all internal QlikView applications and determine the best groupings of these tables based on an organizations data.

An overview of how QlikTech uses this methodology can be found in Section 4.

Sample Incremental Load Script:

```
LET vQVDPath = 'Qvd\';  // Set QVD storage Directory

SET vOffset = '00:15:00';
LET vUTC = UTC();
LET vExecTime = replace(timestamp(vUTC-vOffset, 'YYYY-MM-DDXhh:mm:ssZ'), 'X',
'T');

// SalesForce.com uses a timestamp format of 'YYYY-MM-DDThh:mm:ssZ', hence the
need for the replace statement here.

SET vLastExecTime             = 0; // resetting vLastExecTime

// As long as a QVD already exists (the isnull() check), find the latest
timestamp for modified records.  This will be used to generate the delta set.
if not isnull(QVDCreateTime('$(vQVDPath)<tablename>.qvd')) then

       LoadTime:
       Load Max(LastModifiedDate) as LastModifiedDate  // Or max(CreatedDate)
       From $(vQVDPath)<tablename>.qvd (qvd);

       Let vLastExecTime =
       replace(timestamp(peek('LastModifiedDate',0,'LoadTime'),'YYYY-MM-
       DDXhh:mm:ssZ'), 'X', 'T');

       Drop Table LoadTime;

end if

SQL SELECT Id,
…
FROM <tablename>
WHERE LastModifiedDate >=$(vLastExecTime) and LastModifiedDate < $(vExecTime);

// Check to see if this is the first reload.  If it is, skip this step
if Not isnull(QvdCreateTime('$(vQVDPath)<tablename>.qvd')) then
       Concatenate (<tablename>)
       LOAD *
       FROM $(vQVDPath)<tablename>.qvd (qvd)
       WHERE Not(Exists (Id));
end if

//If data exists within table, store to QVD.
if NoOfRows('<tablename>') > 0 then
       STORE <tablename> INTO $(vQVDPath)<tablename>.qvd;
       Drop Table <tablename>;
end if
```

Full Data Reload

The full reload process is independent of any date filters. This process simply makes a connection to SalesForce.com and does a full SQL Select on the table(s) requested. It then downloads all the data within these tables and stores them to individual QVD files. These QVD files can then be used as a starting point for the incremental load process.

```
<tablename>:
SQL SELECT Id,
…
FROM <tablename>;;
Store <tablename> into $(vQVDPath)<tablename>.qvd;
Drop Table <tablename>;
```

The incremental load process will download a full data set if no QVD file already exists, but it is recommended to run a full reload process before beginning the incremental loads. Once the initial full reload completes, the incremental load will use the QVD files to bring down the updated/added data from SalesForce.com. Although the incremental data load process is much faster to run throughout the day, it is highly recommended to refresh the QVD files with a full data load from SalesForce.com. This should be planned and scheduled on a routine basis.

An overview of how QlikTech uses this methodology can be found in Section 4.

How QlikTech loads SalesForce.com Data

In this section, we will provide an overview of how QlikTech has setup the process of updating our internal applications that utilize SalesForce.com data. The methods and timings documented here are specific to the data required for our implementation and may not correlate to the data and timing needs of other organizations. This is provided as an overview of the process and for use as a guideline.

QlikTech has several applications that require data from SalesForce.com for analysis. After much work, it was determined that all these applications require many of the same tables. Rather than have each developer build their own script and process to load data from SalesForce.com, it was determined that a centralized process for the Organization was a better practice.

Initially, a single application was built to download the data from SalesForce.com. This application made a single connection to SalesForce.com and downloaded each table that was required for all applications. The application was configured to download the required 26 tables needed from SalesForce.com. Because this was a single threaded approach, and only one table could be downloaded at a time, this process took nearly three hours to complete. This is due to the fact that several tables are extremely large and took a very long time to download all the data. All told, the data required to download exceeded 310MB. Using this method, we could only comfortably reload our internal applications every four hours.

To get better performance of the reload process, we initially broke this out into four threads, each downloading a subset of the required tables but still downloading the full data set each time. By spreading the process out into four applications, we were able to reduce the timeframe of a reload to approximately 1hr. 30mins, allowing a reload of internal applications approximately every two hours. However, in both these cases, as the use of SalesForce.com increases, so will the time for the full reload of data since the amount of data that will be downloaded will continue to grow.

By implementing the documented incremental data load process, QlikTech was able to cut the reload time down to less than 5 minutes, allowing for a very frequent reloading of internal applications; anywhere from 30 minutes to an hour refresh of data, depending on needs and scheduling.

Additionally, because the incremental load process is only downloading data that has changed from the last process time (within the last hour in this case), the download process time should not increase as dramatically, even as the use of SalesForce.com increases, because the data being downloaded is only new and/or modified data within the timeframe.

As mentioned, a 15 minute offset variable was also implemented. This was done to minimize the risk of attempting to download data that is currently being updated within SalesForce.com. Although each application within Publisher runs independently, utilizing not only the current time of the reload process (minus the offset), but also using the dates within the current data set as a boundary, there is minimal risk of the updated QVD files being out of synch with each other.

However, to ensure a high quality of the data loaded into the internal applications, QlikTech currently runs a full reload of the QVD files from SalesForce.com on a weekly basis. This is run at Saturdays at 6:00pm EST to minimize the number of users accessing SalesForce.com at the time of reload. The incremental load process is not run Saturdays and restarts Sundays as 12:00am EST to allow for the full download to complete and provide a fresh set of QVD files to run incremental loads again.