Configuring External Blob (Document) Storage

Introduction

This is the mechanism for storing blobs outside the database. Blob (Binary Large OBject) is a generic term for documents (CVs, letters, photos, videos, spreadsheets etc) stored in the database.

By default these blobs are stored inside the database. The advantage of this is that they are backup with the database and no separate arrangements are needed for looking after them. However there can be times when it is more convenient to store them separately. IQX has an alternative storage mechanism which will allow this and which is transparent to the end IQX user. All access to the blobs is still via the database engine - **not directly from IQX client machines** - so no additional folder rights or controls are needed at the client machine level.

Once documents are stored externally, they are **no longer included** in database backups. IQX Customers must be asked to acknowledge IN WRITING that they understand that they must make proper provision for backing up the file structure and contents.

Configuration

Steps to start using External Storage

1. Create a folder on the database machine under which the blob files will be stored. Ensure the server process has full rights to it including the ability to create subfolders.
2. The most important bit – ensure that the designated folder is backed up, preferably by some kind of live sync arrangement.
3. It is highly desirable that the designated folder is ONLY accessible by the Windows user account that the database engine is run under and the user account used for the sync / backup process. This will reduce the chance of any inadvertent editing / deleting of the blob files.
4. Set params.BlobExternalRootFolder to the path to the above folder - for example:

   ```
   update params set BlobExternalRootFolder = 'C:\\IQXDocs';
   ```

   note the doubled-up slashes in the file path Do NOT use spaces in the folder name.

5. Set params.BlobExternalStorage to 1 ie:

   ```
   update params set BlobExternalStorage = 1;
   ```

As users re-enter IQX, every blob that is created or edited will be saved externally. It is important therefore to get all users to exit and re login to IQX once the changes have been made. The BlobRelocate.xml job can be used to shift existing blobs out incrementally.

Blobstore records contain the full path to the file containing the blob. If the value of params.BlobExternalRootFolder is subsequently changed, these paths will need to be updated in the database if documents are to be found.

Accessing blobs in SQL

To access blobs in reports or other queries, use the BlobStoreFetch() database function. This function will work for all blobs whether stored internally or externally.

For example: to retrieve candidate images use

```
BlobStoreFetch(‘J’,person.personid)
```

Moving Blobs Back Into The Database

To reverse the process, change the setting of BlobExternalStorage - new and edited blobs will now be stored internally. Again BlobRelocate will move the existing blobs for you.

BlobRelocate Job

```
<?xml version="1.0"?>
<Job AutoClose="YES" title="Blob Relocation Tool" dateformat="yyyy-mm-dd">
    <Parameters>
        <Parameter name="xMins" type="N" value="60" required="YES">Minutes
to Run</Parameter>
    </Parameters>
    <IfSQL condition="BlobstoreExtFolder(null) is null">
        <SetVariable name="xDest" value="Internal Storage (blobstore
table)"/>
        <SetVariable name="xCond" value="blob is null"/>
    </IfSQL>
    <Else>
        <SetVariable name="xDest" value="External Storage (individual
files)"/>
        <SetVariable name="xCond" value="externalfilepath is null"/>
    </Else>
    <IfNoDialog text="This job will migrate blobs to {xDest} for {xMins}
minutes. Proceed?">
        <Cancel/>
    </IfNoDialog>
    <SQLExec ignoreerror="YES">
        create variable migjobend timestamp
    </SQLExec>
    <SQLExec>
        set migjobend = dateadd(minute, :xMins, current timestamp)
    </SQLExec>
    <SQLQuery>
        <SQLSelect>
            select class, id from blobstore where {xCond}
        </SQLSelect>
        <ForeachRow>
            <IfSQL condition="migjobend &lt;= current timestamp">
                <Finish/>
            </IfSQL>
            <Message text="{Row}"/>
            <SQLExec>call BlobstoreRelocate(:class,:id)</SQLExec>
        </ForeachRow>
    </SQLQuery>
</Job>
```

Validating the External Storage

Once the blobs are stored outside the database, the normal database validation cannot check that everything is present and correct. The stored procedure below (which will only work with SQL Anywhere v12 and later) checks that each external file referred to in the BlobStore table exists and can be accessed by the database server. It does not identify “orphan blobs” ie files that exist in the file structure but which are not listed in the BlobStore.

```
CREATE PROCEDURE "pears"."ValidateExternalBlobStoreFilesExist"(IN
"@AlternativeLocation" long varchar DEFAULT NULL)
  RESULT ("Class" char(1),"ID" char(20),"ExternalFilePath" long varchar,
"Issue" SmallInt, "Description" char(250))
BEGIN
  DECLARE "@Folder" long varchar;
  DECLARE "@StandardLocation" long varchar;
  -- create temp table to hold errors
  DECLARE LOCAL TEMPORARY TABLE BlobStoreCheckIssue("Class" char(1),"ID"
char(20),"ExternalFilePath" long varchar,"Issue" char(10)) NOT
TRANSACTIONAL;
  -- regularise Standard and Alternative locations
  SET "@StandardLocation" = (SELECT "BlobExternalRootFolder" from params);
  IF "@AlternativeLocation" is NULL THEN SET "@AlternativeLocation" =
"@StandardLocation" END IF;
  -- @AlternativeLocation now holds the actual location of the files we want
to check
  IF right("@StandardLocation",1) != '\' THEN SET "@StandardLocation" =
string("@StandardLocation",'\') END IF;
  IF right("@AlternativeLocation",1) != '\' THEN SET "@AlternativeLocation"
```

```
= string("@AlternativeLocation",'\') END IF;
  -- loop through external blobs
    FOR BlobLoop as BlobCursor NO SCROLL CURSOR
      FOR select "Class" as BClass, "ID" as BID,
replace("ExternalFilePath","@StandardLocation","@AlternativeLocation") as
BExtPath from BlobStore where "ExternalFilePath" is not null order by
"Class", "ID"
          FOR READ ONLY
      DO
        if (select byte_substr(xp_read_file(BExtPath,1),0,1)) is NULL // ie
error reading file
            then insert into
BlobStoreCheckIssue("Class","ID","ExternalFilePath","Issue") values (BClass,
BID, BExtPath, 1) end if;
    END FOR;
  -- recheck errors in case files were in use
    FOR BlobLoop2 as BlobCursor2 NO SCROLL CURSOR
      FOR select "Class" as BClass, "ID" as BID, "ExternalFilePath" as
BExtPath from BlobStoreCheckIssue where Issue = 1 order by "Class", "ID"
          FOR READ ONLY
      DO
        if (select byte_substr(xp_read_file(BExtPath,1),0,1)) is NULL // ie
error reading file
            then update BlobStoreCheckIssue set "Issue" = 2 where "Class" =
"BClass" and "ID" = BID // still a problem
            else update BlobStoreCheckIssue set "Issue" = 0 where "Class" =
"BClass" and "ID" = BID // now OK
        end if;
    END FOR;
  -- check existence of remaining issues
    FOR BlobLoop3 as BlobCursor3 NO SCROLL CURSOR
      FOR select "Class" as BClass, "ID" as BID, "ExternalFilePath" as
BExtPath from BlobStoreCheckIssue where Issue = 2 order by "Class", "ID"
          FOR READ ONLY
      DO
        set "@Folder" = null;
        set "@Folder" = left(BExtPath,locate(BExtPath,'\',-1)-1);
        if exists (select * from sp_list_directory("@Folder",1) where
file_path=BExtPath and file_type ='F')
           then update BlobStoreCheckIssue set "Issue" = 3 where "Class" =
"BClass" and "ID" = BID // file present in folder - must be locked or
insufficient rights to read
           else update BlobStoreCheckIssue set "Issue" = 4 where "Class" =
"BClass" and "ID" = BID; // file NOT present
        end if;
    END FOR;
select "Class", "ID", "ExternalFilePath", "Issue", case when "Issue" = 3
then 'File present but not accessible' when "Issue" = 4 then 'File missing'
```

```
end as "Description" from BlobStoreCheckIssue;
END;
```

The validation is run by SELECTing from the stored procedure:

```
select * from pears.ValidateExternalBlobStoreFilesExist();
```

which gives a result like:

Class	ID	ExternalFilePath	Issue	Description
O	TI01FQSS161120130006	i:\iqxdocs\2013\11-16\OTI01FQSS161120130006.dat	4	File missing

Orphaned Blobs in the file system can be identified with this stored procedure:

```

CREATE PROCEDURE "pears"."ValidateExternalBlobFilesExistInDatabase"(IN
@AlternativeLocation long varchar DEFAULT NULL)
RESULT ( "FilePath" long nvarchar ,"FileSizeInKB" unsigned bigint ,"Created"
timestamp with time zone ,
"Modified" timestamp with time zone ,"Accessed" timestamp with time zone )
BEGIN
    DECLARE @StandardLocation long varchar;
    DECLARE LOCAL TEMPORARY TABLE BlobFiles("ID" bigint default
autoincrement,"Location" char(1),FilePath long nvarchar ,FileSize unsigned
bigint ,Created timestamp with time zone , Modified timestamp with time zone
,Accessed timestamp with time zone, PRIMARY KEY(ID)) NOT TRANSACTIONAL;
    -- regularise Standard and Alternative locations
    SET "@StandardLocation" = (SELECT "BlobExternalRootFolder" from params);
    IF "@AlternativeLocation" is NULL THEN SET "@AlternativeLocation" =
"@StandardLocation" END IF;
    -- @AlternativeLocation now holds the actual location of the files we
want to check
    IF "right"("@StandardLocation",1) != '\' THEN SET "@StandardLocation" =
string("@StandardLocation",'\') END IF;
    IF "right"("@AlternativeLocation",1) != '\' THEN SET
"@AlternativeLocation" = string("@AlternativeLocation",'\') END IF;
    -- Find the files litsed in the BlobStore
    INSERT INTO BlobFiles("Location",FilePath)
        SELECT
'T',replace("ExternalFilePath","@StandardLocation","@AlternativeLocation")
from BlobStore where ExternalFilePath is not null;
    -- Find the files in the file system
    INSERT INTO BlobFiles("Location", "FilePath" ,"FileSize"
,"Created","Modified" ,"Accessed")
        SELECT
'D',file_path,file_size,create_date_time,modified_date_time,access_date_time
```

```
from sp_list_directory(@AlternativeLocation,3) where file_type = 'F';
    CREATE INDEX A ON BlobFiles("Location",FilePath );
    -- select those files only listed in the file system
    SELECT FilePath ,FileSize/1024 ,"Created", "Modified" ,"Accessed" FROM
BlobFiles join (
         (select max("ID") as "ID" from BlobFiles  where Location='D' group
by "FilePath" having count(*) =1 )) as MissingBlobs on BlobFiles.ID =
MissingBlobs.ID order by FilePath;
END;
```

The validation is run by SELECTing from the stored procedure:

```
select * from pears.ValidateExternalBlobFilesExistInDatabase();
```

which gives a result like:

FilePath	FileSizeInKB	Created	Modified	Accessed
i:\\iqxdocs\\2003\\06-19\\OTIMS5212151906200300.dat	89	2013-11-15 23:04:47.000+00:00	2013-11-15 23:04:47.000+00:00	2013-11-15 23:04:47.000+00:00
i:\\iqxdocs\\2013\\08-20\\OTI1JAASS200820130002.dat	9	2013-08-20 18:54:10.000+00:00	2013-08-20 18:54:10.000+00:00	2013-08-20 18:54:10.000+00:00

Each of these procedures can take an optional parameter of an alternative BlobStore folder, if specified, this will be compared with the contents of the BlobStore instead. This can be useful for checking that the backup of the external BlobStore is being correctly replicated.

From:
https://iqxusers.co.uk/iqxhelp/ - **iqx**

Permanent link:
https://iqxusers.co.uk/iqxhelp/doku.php?id=sa28-00&rev=1512146147

Last update: **2017/12/01 16:35**