

Structuring the Captell environment

This paper looks at the issues surrounding how to structure a Captell environment and suggests a method of structuring that addresses these issues.

The two major issues are "Backup and Recovery" in the case of a failure of some sort and "Change Control" to manage the development life cycle of Captell report objects.

Most Captell establishments will be of a size that requires dedicated server hardware to run the Captell database, this paper focus on these establishments. In the case of smaller environments where Captell is installed on a single desktop the principals can still be applied however; not necessarily to the same extent.

SQL Server Instance and Database Structure

The major repository for Captell information is the Captell SQL Server database which resides in an instance of SQL Server. A single hardware platform (server or desktop) can support a number of instances of SQL Server, each instance can support a number of databases.

To correctly manage the development, test and production cycle for Captell reports three separate databases are required.

Development

The development database will be used for all development work and will contain a subset (by volume) of the production data.

The development database resides in it's own SQL Server instance.

Test

Report objects will be migrated from the development environment to the test environment for testing. All report objects present in the Production environment are also present in the Test environment.

The test environment contains data for new tables, or those that are being modified; these tables will have an associated schedule and will load data from the production source data files. Tables that are not being modified but are referred to by other report objects are replaced with an SQL query that simply selects all records from the production table. For example a table in Production named [APPL\Response] may be replaced in the test environment with a query named [APPL\Response] that uses the following SELECT statement.

```
Select * from [CAP-PROD].dbo. [APPL\Response]
```

This technique makes the production data available to test without having to duplicate it in the test database.

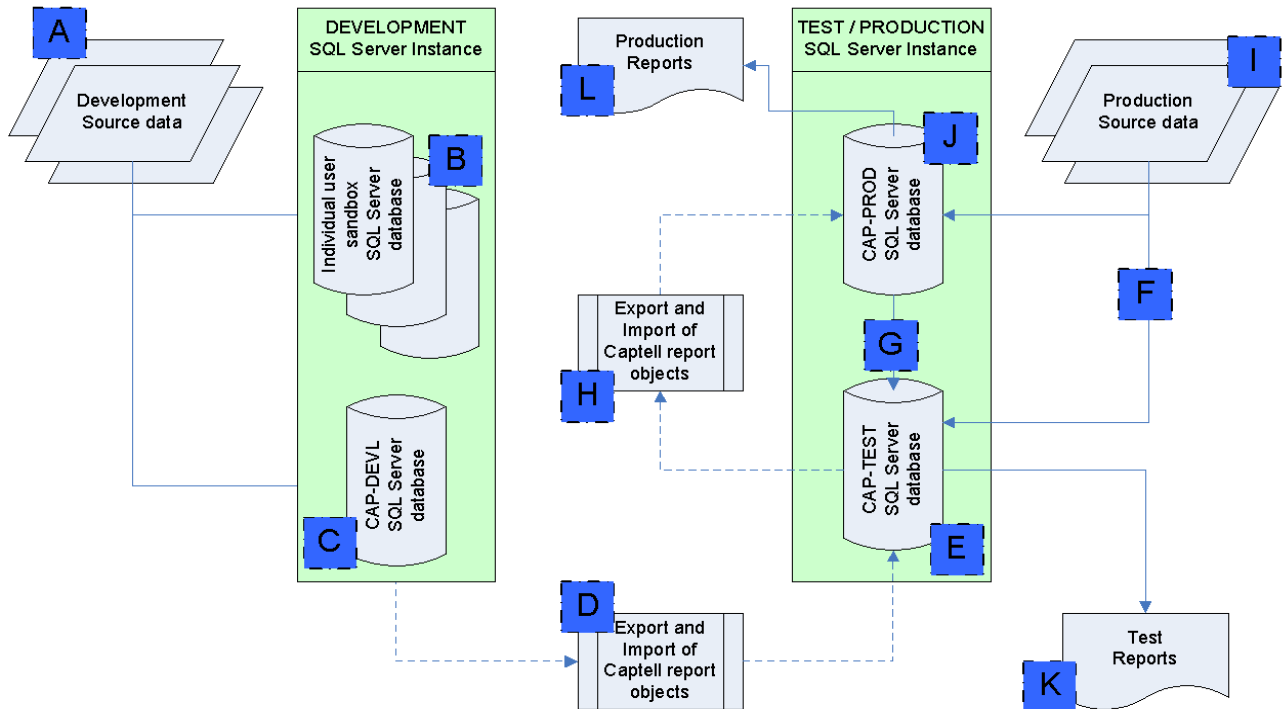
The test database resides in the same SQL Server instance as production to allow the linkage to the production data.

Production

The production database contains all report objects and all data necessary for the generation of production reports. All tables should be scheduled for update and will obtain their data from the production source data files.

Instance and database design

The optimal structure for Captell is shown in the following diagram, this consists of 2 SQL Server instances, one for development work and the other for test and production.



Notes:

- A** A new source of data is made available for inclusion in Captell, initially it is accessed by a Captell user experimenting. Subsequently a table definition is established in the DEVL environment and development work starts on new report objects. The development source data should be a subset (by volume) of that expected in production.
- B** User databases may be established for individual experimentation and development before moving to full development status.
- C** All development and maintenance work is carried out against the CAP_DEVL database, this database contains a copy of all report objects present in the CAP-TEST and CAP-PROD environments.
- D** Once a new report has been developed sufficiently for testing it is migrated to the CAP-TEST environment by exporting the object definitions from CAP-DEVL and importing them into CAP-TEST

The table definitions will need to be modified to point at the production source data.

- E** The CAP-TEST environment is essentially a copy of the CAP-PROD environment with the inclusion of new or changed report objects from the CAP-DEVL environment.

Tables in the CAP-TEST environment that exist in CAP-PROD are replaced with queries that reference the CAP-PROD data in this way the CAP-TEST environment does not need to contain all the data and test can be run against full data loads.

New tables coming up from CAP-DEVL will be full table definitions pointing at the new Production source data

New tables are updated on a schedule to ensure the database is maintained in synch with CAP-PROD.

- F** Production source data for new tables. See E above.
- G** Queries will be established in CAP-TEST to access the CAP-PROD data. See E above.
- H** Once testing is complete report objects may be exported from the CAP-TEST environment and imported into CAP-PROD, only those objects that are new or have been changed should be migrated.
- I** Production source data in its various guises.
- J** The CAP-PROD database is the source for production report generation. Tables are updated from the production source data on a schedule the only objects that are manually changed in this database are parameter values that are changed when generating reports.
- K** Comparing reports generated from the CAP-TEST database with those generated from the CAP-PROD is the major test for ongoing integrity.
- L** All reporting is generated from the CAP-PROD environment.

Backup

The three databases require different backup strategies, however; in all cases the backups should be made and scheduled by SQL server. The main reason for this is that the SQL Server transaction log is cleared when a backup is made, without a regular SQL Server backup the transaction log will continue to grow and consume all the available disk space.

In addition to the SQL Server backups normal full volume disk backups should be taken on a regular basis.

User sandbox databases

Backup of these databases could be scheduled monthly or left to the individual user to manually backup as required.

CAP-DEVL

The contents of this database will change on a daily basis with new development work it's size should be relatively small as not all the source data will be loaded. A SQL server backup should therefore be taken each night a several generations retained.

CAP-TEST

The CAP-TEST database effectively becomes the main repository for Captell report object definitions, the database's size should be relatively small as it will not contain the bulk of the production data, only that data related to new tables being tested. A SQL Server backup should be made of this database each night and several generations taken.

In addition to the regular scheduled backups a special backup should be made of the CAP-TEST environment immediately before importing any new report object definitions from the CAP-DEVL environment.

CAP-PROD

The CAP-PROD database contains all the production data and report objects. A SQL Server backup of this database should be taken based on the frequency of table updates, e.g. if you are only updating your data weekly then a weekly backup will suffice, if daily then daily backups will be required.

In addition to the regular scheduled backups a special backup should be made of the CAP-PROD environment immediately before importing any new report object definitions from the CAP-TEST environment.

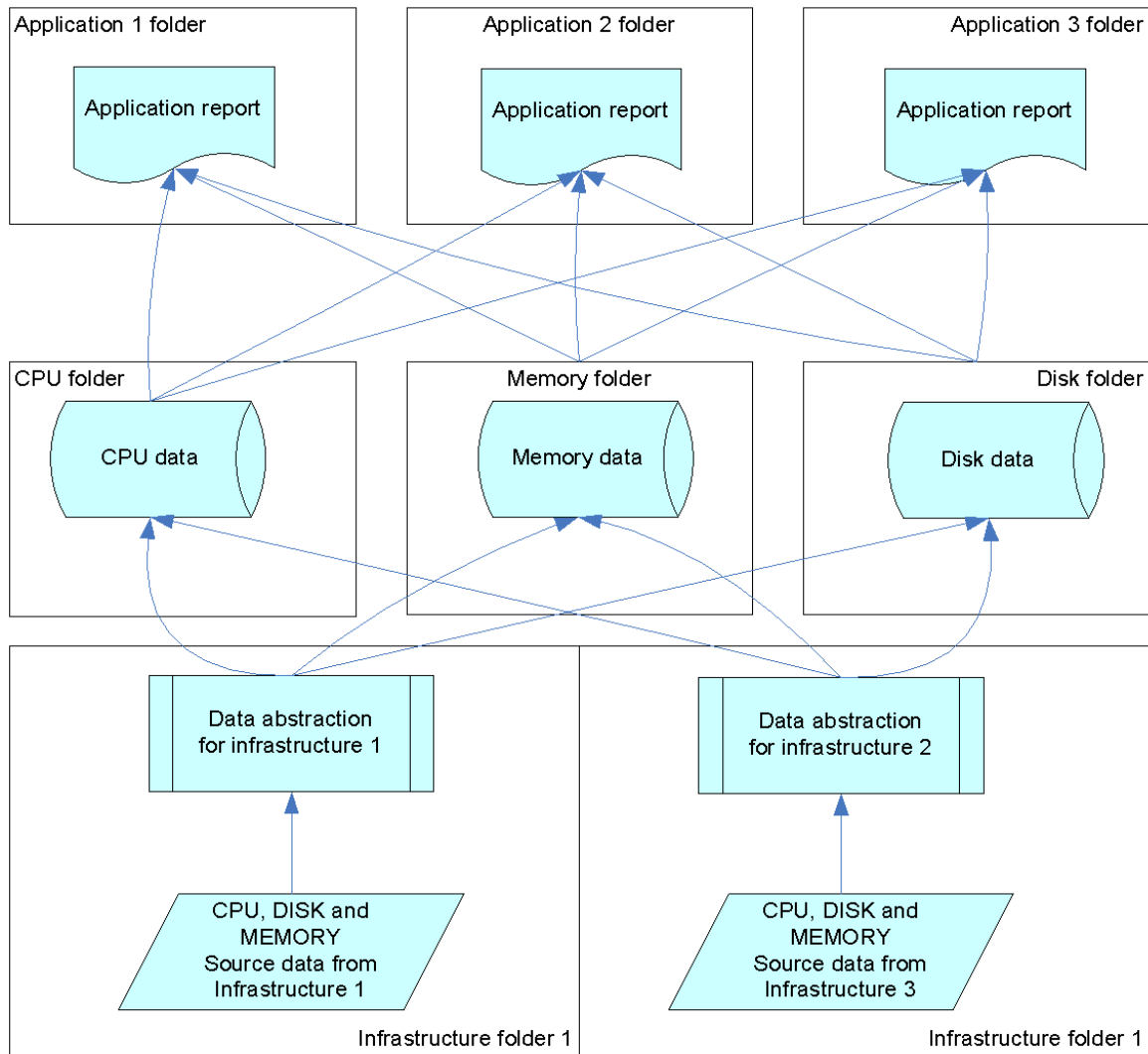
Recovery

Database backups can be used to replace an existing database or can be restored to a new database. The new database options would be used when individual report objects needed to be restored after their inadvertent deletion. The latest backup being restored to a new "temporary" database, the missing objects exported and then the "temporary" database deleted.

Captell Folder Structure

Where possible the Captell folder structure should be replicated across the development, test and production databases. This will facilitate the movement of report objects between these environments.

As the high level reports will be fairly common across platform and “application or area of interest” and the metrics coming from the various platforms will be similar in function but may be structured differently, a general philosophy should be that the folder structure should allow for the merging of functionally similar data at a low level to facility cross platform reporting. The following diagram illustrates.



Here the data from different infrastructure that is similar in function is manipulated to a common form in the data abstraction process. This would be implemented in Captell as a table definition to hold the raw source data, a query to perform the abstraction and a summary table to hold the “normalised” data.

A good starting point for the folder structure is to create root folder for each subject area and then separate folders within the root for each Captell report object type. Some additional root level folders will also be required for global or common objects. Subject Area may be an application or functional area of interest and/or they may be technical areas of interest. The following diagram illustrates:

1. GLOBAL
 - 1.1. Tables
 - 1.2. Queries
 - 1.3. Parameters
 - 1.4. Pivots
 - 1.5. Charts
 - 1.6. Traffic lights
 - 1.7. Documents
2. APPLICATION 1
 - 2.1. Tables
 - 2.2. Queries
 - 2.3. ...
3. APPLICATION 2
 - 3.1. Tables
 - 3.2. Queries
 - 3.3. ...
4. NETWORK UTILISATION
 - 4.1. Tables
 - 4.2. Queries
 - 4.3. ...
5. RESPONSE TIME
 - 5.1. Tables
 - 5.2. Queries
 - 5.3. ...

In this basic example root node folders are separated into functional areas such as application or metric

In the case of table data is loaded from the source files for the specific area of interest, this may be split out by an external process or by utilising the Captell Input SQL facility

Where your reporting is across multiple platforms additional folders may be required, this provides for the separation of report objects at the functionally different platform but allows for the coming together of the data at the application level

1. GLOBAL
 - 1.1. Tables
 - 1.2. Queries
 - 1.3. ...
2. PLATFORM 1
 - 2.1. Tables
 - 2.2. Queries
 - 2.3. ...
3. PLATFORM 2
 - 3.1. Tables
 - 3.2. Queries
 - 3.3. ...
4. APPLICATION 1
 - 4.1. Tables
 - 4.2. Queries
 - 4.3. ...
5. APPLICATION 2
 - 5.1. Tables
 - 5.2. Queries

5.3. ...

This example introduces an extra root node to cater for data coming from different platforms such as Windows, Unix or Mainframe.

The raw data (say for UNIX) would be all be loaded at the PLATFORM level then queries at the platform level could summarise as required. At the application level Queries would refer to the PLATFORM level data selecting data for the specific application

You may also wish to consider introducing a level under the TABLE folders to separate out the summary data from the "raw" data.

6. GLOBAL
 - 6.1. Tables
 - 6.1.1. Summary
 - 6.2. Queries
 - 6.3. ...

- 7. PLATFORM 1
 - 7.1. Tables
 - 7.1.1. Summary
 - 7.2. Queries
 - 7.3. ...
- 8. PLATFORM 2
 - 8.1. Tables
 - 8.1.1. Summary
 - 8.2. Queries
 - 8.3. ...
- 9. APPLICATION 1
 - 9.1. Tables
 - 9.1.1. Summary
 - 9.2. Queries
 - 9.3. ...
- 10. APPLICATION 2
 - 10.1.Tables
 - 10.1.1. Summary
 - 10.2.Queries
 - 10.3....