Overview of the Data Repository

The Data Repository (DR) is a copy of your Meditech LIVE and TEST data maintained on a separate server and in a different database engine (SQL Server). It is intended to supplement, but not replace, your existing NPR and/or RD reporting environment.

Because this copy of your data lives outside of Meditech itself, you should consider it a new application deployed in your enterprise. It has new administrative requirements, needs new skills and requires new tools to manage and exploit. Your users will access it in a different way and your IM/IT department needs to treat it on par with Meditech itself as an application needing maintenance, backups and upgrades.

This document details our suggested approach for implementing a reliable, easy-to-maintain and speedy DR system.

Data Flow from MT to the DR

As your users add, edit and delete records in Meditech LIVE (and TEST) that activity is tracked in an xfer (Transfer) index maintained by the filer in each module (NUR, ADM, BAR, etc.). Each xfer log is accessed by your DR server continuously, to fetch a record of what has changed in Meditech. The DR server then updates its copy of the LIVE or TEST data with those changes. Note that the transfer log is a queue … entries are always being added into the queue by Meditech, and then taken out by the DR.

Changes do not replicate from LIVE to the DR instantly!

If your DR environment is running smoothly, you probably have no more than a 15 minute delay in the movement of edits from Meditech into the DR. As a result, you cannot replace all NPR and RD reporting with the DR. Many reports can be delivered by from the DR, but not anything requiring up-to-the-minute point of care data.

A side effect of the queuing process used by the DR is that the xfer logs can get a stuck sequence (due to varying factors in Meditech). When a module queue stalls, no new updates flow to the DR from that module, the queue backs up and the DR falls out of sync with Meditech.

Clearing a stuck sequence and restarting the queue almost always requires a ticket with Meditech, and that means it takes time to resolve. While that is happening, the queue is continuing to back up! What does that mean to you and your users? Any reporting downtime caused by a stuck queue takes as long to clear as the queue is stalled.

Queue stuck for a day? Reporting could be down for that module for two days.
Site Requirements to Implement the DR

Deploying the DR is implementing a **new application**, with new requirements:

- The DR queues have to be checked daily by someone in your organization.
- The DR database server needs to be backed up, re-indexed and rebooted regularly.

Beyond that, you need someone to actually build the new DR-based reports. You need a way to deliver them to your end users. All of this means implementing and maintaining a new application infrastructure.

Meditech made one key design decision for the DR which drives your response to many of these issues. They chose to build the DR **only** on the Microsoft SQL Server database engine. You are **not** locked in to using only Microsoft-delivered reporting solutions with the DR, but that is often the easiest path to take. Once you implement DR-based reporting in your enterprise, your users are going to rely on those reports in administrative, clinical and ancillary workflows.

*If you use the DR, you need to treat it like you treat Meditech itself.*

It’s easy to sideline the DR as a secondary system that does not need monitoring, maintenance, updates and expansion. But as soon as DR-delivered reports are used in your workflows, and the DR goes down, it can be as bad as LIVE going down.

*If you are going to use the DR, treat it like a Tier One Application.*

At the least, that means:

- DR backups are validated regularly.
- There is at least one analyst assigned to the DR as a primary responsibility.
- A SQL Server DBA is employed by your organization.
- Your report developers are trained in SQL-based development.

Meditech’s Expectations

Meditech provides a page of expected DR-related customer responsibilities:

[http://www.meditech.com/prdr/Pages/DRbbASCustomerResponsibilities.htm](http://www.meditech.com/prdr/Pages/DRbbASCustomerResponsibilities.htm)

In summary, your enterprise is expected to:

- Monitor the transfer jobs daily; reviewing errors and watching for delays in processing.
- Monitor the background job(s) that manage reading M-AT data and turning those audit files into activity index entries to be processed by the DR xfer system.
- Monitor SQL Server errors and event logs; confirm backups are working.
Monitor space usage in the DR (it will only ever grow…)
Monitor concurrent users to size the connection pool properly.
Watch for un-needed indexes and user logins.

These requirements do not include report development. These are the tasks for the system admin(s) and/or SQL DBA to undertake. Yes, additional staff and staff skills are required when you implement the DR. Meditech has a note about this:

https://www.meditech.com/prwdr/pages/draiascoreteam.htm

It is notable how many organizations have implement the DR without happening to review this page… and that is a **minimal** recommendation. If you are serious about implementing the DR and using it to improve your reporting workflow, business practices and clinical care you are going to need:

- 1 SQL Server DBA (1 FTE)
- One half-time DR Team Lead (0.5 FTE)
- At least one full-time DR Report Writer (1 FTE)

The skillset for implementing, managing and using the DR for report development and delivery is **not** the same as the skillset for NPR, except…

**Business Knowledge is King**

Hiring a SQL report developer off the street will get you almost nothing, if they do not understand the business of healthcare, medical nomenclature and (preferably) where the data is stored in Meditech.

*It is easier to teach an NPR developer SQL than a SQL developer NPR.*

If you have an NPR development team, or even staff doing NPR development off the side of their desk, you will get the most bang for your buck by training them in writing SQL code and SQL-based reports. Hands down, that is the fastest road to getting DR-savvy resources internal to your organization.

The data in the DR models the data in Meditech. If you know where something is in Meditech, you can find it in the DR. But if you don’t know where it is in Meditech…

**The DR Reporting Toolkit**

With all that we’ve covered, we haven’t gotten to being able to create and deliver a report to our users!

*Meditech does not provide you with a report development and delivery toolset.*
This part of the DR infrastructure is left to your enterprise to identify, purchase and deploy. And that is a jarring departure from how NPR and RD work. The reporting environments delivered within Meditech provide everything. A way to build a report, test a report, and then deliver a report to your users. You get a security model and even version control (in RD 6.x).

But not for the DR. You will need to decide on:

- A SQL coding environment for your report developers to use.
- A SQL-based report builder for your report developers to use.
- A delivery mechanism for those reports.
- Version control software to manage the code and the reports.

As it happens, if you have licensed Microsoft SQL Server **Standard** edition or better, you have gotten **almost** all of those tools you need for report development and delivery:

- **SQL Server Management Studio** will let you build and test queries.
- **Business Intelligence Development Studio** (or Report Builder) will let you turn that query into a formatted report using SQL Server Reporting Services.
- **Business Intelligence Development Studio** will also let you create extracts using SQL Server Integration Services.
- **SQL Server Reporting Services** will let you deliver your reports via a web-interface to your users, either all by itself, or in cooperation with SharePoint.
- **SQL Server Integration Services** will let you schedule file extracts (or database loads) via SQL Agent.

But you won’t get a version control system – you need to license Microsoft **Team Foundation Server** separately, or use an open-source solution.

One more caveat – **SQL Server Management Studio** is not a terribly efficient place to build and test SQL queries. There are **much** better tools that will make your report developers markedly faster in responding to data requests.

**Summary of Implementation Gaps**

<table>
<thead>
<tr>
<th>DR Component</th>
<th>Meditech Expectation</th>
<th>Possible Issue or Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database requires dedicated server</td>
<td>Site will purchase DR hardware and license Microsoft SQL Server</td>
<td>Default Meditech install of the DR, has structural issues.</td>
</tr>
<tr>
<td>Database runs on Meditech SQL Server</td>
<td>Site will hire a SQL Server DBA</td>
<td>No SQL Server DBA on staff or FTE in budget</td>
</tr>
<tr>
<td>Reports are SQL-based</td>
<td>Site will hire one or more SQL-savvy reporting resources</td>
<td>No SQL skills among site reporting staff, or FTEs in budget; SQL developers who know Meditech are very rare – so hiring an FTE may be quite difficult.</td>
</tr>
<tr>
<td>Reports cannot be delivered through Meditech itself</td>
<td>Site will implement SQL Server Reporting Services or Crystal Enterprise or Cognos to deliver</td>
<td>No SSRS/CE/Cognos administrator, no server to host web-site, no skillset in how to manage the</td>
</tr>
<tr>
<td>DR Component</td>
<td>Meditech Expectation</td>
<td>Possible Issue or Gap</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>--------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>New reports must be developed</td>
<td>Site will develop or hire one or more SQL-savvy reporting resources</td>
<td>No SQL skills among site reporting staff, or FTEs in budget.</td>
</tr>
<tr>
<td>Not all reports can be from the DR</td>
<td>Site will know where to build reports appropriately</td>
<td>Site has been told by Meditech sales that all of their reporting can come from the DR; or site does not know which modules transfer to the DR, or what the latency is on the xfer process.</td>
</tr>
<tr>
<td>Transfer process must be monitored</td>
<td>Site will check transfer queues regularly and open tickets with Meditech as issues arise</td>
<td>No analyst assigned to the DR, little understanding of DR issues or how to resolve them; little visibility (in pre-6.x) to what is happening in the transfer queues.</td>
</tr>
<tr>
<td>Transfer process is accurate</td>
<td>Site will review data between the DR and NPR/Focus and report errors to Meditech</td>
<td>Has no comprehensive way to test data validity (in pre-6.x environments); data validation process in 6.x is astoundingly slow.</td>
</tr>
<tr>
<td>New Data model</td>
<td>Site will explore and learn how the data model fits together and maps to NPR/Focus structures.</td>
<td>Site may have NPR-savvy reporting resources who know nothing of SQL, or SQL resources who know nothing of NPR.</td>
</tr>
<tr>
<td>Data model not indexed</td>
<td>Site will identify tables needing indexes and create them as needed</td>
<td>Has no SQL Server DBA to create indexes for them, is unfamiliar with indexing best practices; does not know what indexes will be needed.</td>
</tr>
<tr>
<td>Data model incomplete</td>
<td>Site will identify tables in the DR which exist, but are not turned on to receive data from NPR/Focus, and request they be turned on.</td>
<td>Has no idea what tables they will need in the future, so they only turn them on as needed – which means data may have purged out before the tables are turned on.</td>
</tr>
<tr>
<td>No example reports</td>
<td>Site will find example SQL on Meditech KB or internet and turn into reports using their reporting tool of choice.</td>
<td>Has no idea where these examples are; the examples may or may not work with their configuration; no actual report examples (everything from Meditech is just queries).</td>
</tr>
<tr>
<td>No best practices</td>
<td>Site will develop own DR-related processes, procedures and best practices.</td>
<td>Has no idea (having no SQL experience) how they should build queries, what tools they should use, how reports should be managed and tracked, etc.</td>
</tr>
<tr>
<td>Server Maintenance</td>
<td>Site will know how to backup and maintain a SQL Server instance</td>
<td>Has no SQL database administrator, so they are unlikely to know that the database needs re-indexing jobs running weekly, that the server needs to be rebooted monthly and backups need to be validated regularly.</td>
</tr>
<tr>
<td>New Implementation</td>
<td>Site will let Meditech know how the database containers should be named, what initial tables (beyond the core set) will be turned on.</td>
<td>Has no idea how the database containers work, or what they should be named, or which servers they really need in their DR.</td>
</tr>
<tr>
<td>DR Component</td>
<td>Meditech Expectation</td>
<td>Possible Issue or Gap Environment</td>
</tr>
<tr>
<td>------------------------------</td>
<td>---------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>PCS data is difficult to access</td>
<td>Site will figure out that some key Focus data elements do not come to the DR properly and devise workarounds</td>
<td>Has no SQL reporting resources who could identify the issue, figure out where the data is, and write the necessary functions/stored procedures to get the data out effectively.</td>
</tr>
</tbody>
</table>

**Requirements**

Let’s recap what you need for a successful DR environment:

- Hardware to run the DR application on.
- The SQL Server database engine to hold the data.
- The DR databases on the server to hold LIVE and TEST data.
- A DBA to install, maintain and manage that infrastructure; including deploying custom database objects and indexes.
- A SQL coding tool to build and test queries in.
- A SQL report development tool to build and test reports in.
- A version control system and server to manage the queries and reports.
- A delivery mechanism to get those finished reports into the hands of your users.
- A hardware platform to host the reporting web site.
- Appropriate workflow to tie all of this together efficiently.

All of these components fit together to provide the complete picture. Now let’s go through our main options for each piece, and Iatric’ recommendations for them:

**Hardware**

Remember that we’re talking about a Tier One application here:

**Minimum Hardware Platform**
- Production SQL database Server [physical]
- Test and Development SQL database server (named instance for Dev or second copy of the SQL engine) [virtual]
- Reporting Services web server [virtual]
- Production DR Background Job server [virtual]
- Test DR Background Job server [virtual]

**Recommended Hardware Platform**
- Production SQL database Server [physical]
- Test SQL database Server [virtual]
- Development SQL database Server [virtual]
- Production DR Background Job server [virtual, properly sized]
- Test DR Background Job server [virtual]
✦ Reporting Services integrated into an existing multi-server SharePoint farm
✦ or Reporting Services Production web server [virtual] and Reporting Services Test web server [virtual]

**Server Specifications**

✦ Microsoft Server 2008 R2 x64
✦ Microsoft SQL Server 2008 R2 x64
✦ Minimum of 16GB of RAM per server, prefer 32 GB. All data storage SAN connected with extendible partitions/LUNs.

**Database Configuration**

*Do not accept* the default DR database configuration that Meditech delivers. You want a custom setup that looks like this (using a Meditech 6 environment as an example):

**Production Database Server** (connected to LIVE ring)

✦ *fdb* database container
✦ *ndb* database container
✦ *zcus* database container

**Test Database Server** (connected to TEST ring)

✦ *fdb* database container
✦ *ndb* database container
✦ *zcus* database container

**Development Database Server** (weekly copy of Production, not connected to an MT ring)

✦ *fdb* database container
✦ *ndb* database container
✦ *zcus* database container

Why do we want this arrangement? To make the lives of our report developers easier, reduce errors when moving code from test/dev to live, address performance issues earlier and keep updates from breaking our reports:

In this arrangement, a report developer writes a SQL query against a common set of database container names (*fdb, ndb*) and stores his query (as a stored procedure) in *zcus*. Since *zcus* is site-controlled, it will not be wiped out by an update or upgrade delivered by Meditech.

Query and report development can occur using either TEST (if there are dictionary or CDS entries needed by the report that are not yet in LIVE) or DEV (if the report is using what is in LIVE). Then promoting the report to LIVE means just changing the database server name in the report (or using a shared data source), as opposed to changing all of the database names in the code itself. With DEV in the mix, you can develop and test your code against real production data, without affecting production.
Iatric Recommended DR Server Layout

**Meditech LIVE Ring**

- Audit Reader Background Job Servers (2) (MAT/NPR to Activity Log to DR)

**Meditech TEST Ring**

- Audit Reader Background Job Server (MAT/NPR to Activity Log to DR)

**LIVE DR**

- **LIVE-DR-1**
  - ndb
  - fdb
  - mdb
  - zcus
  - Physical server, 8(+ CPU cores, 16(+) GB RAM, SAN attached disk in RAID10

- **LIVE-DR-2**
  - ndb
  - fdb
  - mdb
  - zcus
  - Weekly Full Restore of LIVE to DEV
  - sql queries

**TEST DR**

- **TEST-DR**
  - ndb
  - fdb
  - mdb
  - zcus

**DEV DR**

- **DEV-DR**
  - ndb
  - fdb
  - mdb
  - zcus

**LIVE Reports**

- **LIVE-RS**
  - http
  - ssrs
  - VM Server, 4(+) CPU cores, 8(+) GB RAM, SAN attached disk

**TEST Reports**

- **LIVE-RS**
  - http
  - ssrs
  - VM Server, 4(+) CPU cores, 8(+) GB RAM, SAN attached disk

**DEV Workstations**

- **DEV-PC(s)**
  - BIDS
  - Developer workstations, 4 CPU cores, 16 GB RAM, dual 19" monitors

**Example site upgraded from Magic to MT6**

- VM server, 2(+) CPU cores, 8(+) GB RAM, SAN attached disk

- Developer workstations, 4 CPU cores, 16 GB RAM, dual 19" monitors
**M/AT Background Job Servers**

Edits move from your NPR and M/AT modules into the DR via a multi-step process:

1. Edit occurs in Meditech; transaction representing that change is written to an audit log.
2. The M/AT audit log is processed by a background job, turning the audit information into a SQL statement.
3. The xfer job then executes that SQL statement against the DR to apply the update to your LIVE or TEST DR database.

Because there is an intermediary step handled by a background job, the client machines the job is running on needs to be properly sized (in terms of CPU cores, RAM and working disk space). If the background job slows down reading the audit logs – then updates are not flowing to your DR.

Additionally, if you have more than two facilities in your health system – you want the DR work spread across multiple background job servers.

**Data Compression**

In SQL Server 2008 and above, if you purchased the Enterprise edition of the server license, you can turn on Data Compression which will reduce your disk footprint by approximately 50%.

Iatric recommends that you turn on Row-based Data Compression if you have 4 or more CPU cores on your DR servers.

Note that if you turn on Row-based Data Compression after initial creation of the database you will need to recompile all stored procedures in zcus as the engine metrics will change. Turning on page data compression for all tables after initial installation is possible via a script. See [http://blog.niall.com/2011/02/compressing-all-tables-in-a-sql-server-2008-database/](http://blog.niall.com/2011/02/compressing-all-tables-in-a-sql-server-2008-database/) for an example.

**Tables to Turn On**

Though Meditech will by default only turn on a “core set” of tables that they have identified as being key to reporting – hard experience has shown that:

- There are dozens to hundreds of additional tables you will need activated to do the reporting you need to do.
- If you ask, Meditech will turn on all tables for you.

This is an enormous savings in time and effort for you. Request that all tables in all modules be turned on. This saves you from:

1. You do not have open dozens of tickets as you find new tables that have to be turned on.
2. You won’t have to wait for the IL’s for each new table to complete before you can complete your reporting build and validation.

3. You will not lose data to purging because an unknown, but key, table was not turned on.

The cost in time and effort by Meditech to turn on all tables is vastly outweighed by not losing your data, and not having to go through this process each time to find a table that needs to be activated.

The SQL Coding Environment

The Microsoft-provided tools for actually creating SQL queries are very basic. You can make your report developers more productive by providing them with:

- A third-party SQL editor; like Toad for SQL Server, Rapid SQL or SQL Prompt Pro, provide rich environments for building and testing SQL code. It is physically faster to write and debug SQL using one of these tools than SSMS.
- Dual monitors. Having your coding editor up on one monitor and your internet browser on the other will get you at least a 20% efficiency improvement.

We’ve mentioned that your developers will be saving their queries as stored procedures – that is a fancy way of saying each query is given a name and stored in the SQL Server engine, just like a %Z program is. Why would you do that? Stored procedures are faster when executing, easier to maintain, searchable and can be reused.

The Report Development Environment

Unlike NPR or RD, which are solely the provenance of Meditech, there is a wealth of SQL-based reporting tools. This decision comes down to:

- Do you have Crystal Reports, Cognos, or SSRS already in use?
- How much money do you want to spend on licensing?

None of these tools are terribly useful unless you can deliver reports to your users – and each of them has its own proprietary web-server software for that. So the development tool determines the report delivery mechanism.

- Business Intelligence Development Studio from the SQL Server Developer Edition is the cheapest to deploy at US $40 a seat.
- Crystal Reports and Cognos are more expensive (vastly so in some cases) and more robust as a reporting environment.

If you have the dollars, Business Objects (Crystal Reports) gets you the most capability and most efficient report development. Cognos comes in a close second, though less expensive. And SSRS is waiting if you’re short on capital.

However – you only want one (1) reporting tool. Just one. Not five. Not even two!
**Version Control**

This is probably the hardest part of the reporting environment to fit into your budget. You can absolutely write SQL, develop and deploy reports without a version control system. But as soon as you overwrite some good code with bad, or lose your working directory, or collide with another developer working on the same file... then you are **wasting time and money** you could have saved with some planning.

You have two main options here, given we have a mostly Microsoft-centric environment.

- **Microsoft Team Foundation Server** – an expensive, but full-featured team project/development system that has version control, document check-in-check-out; collaboration tools, a ticketing system...everything. But it takes another server, another SQL license and capital dollars to implement.
- **SubVersion** – an open-source, almost-free, version control system that tracks all of your changes, allows rollback and forking of code; robust; and integrates with Visual Studio and essentially all third-party tools. Requires a server, but it can be a Linux VM and very low cost.

**Report Delivery**

The report development tool also decides what web-based report delivery mechanism you’re going to use:

- **SQL Server Reporting Services** has its own web-server or can use **Sharepoint**.
- **Crystal Reports** can be delivered either through **Business Objects Enterprise** or **Crystal Enterprise Server**.
- **Cognos** reports can be delivered through **Cognos Enterprise**.

Each solution requires a dedicated hardware platform.

- SSRS can use an existing Sharepoint installation, or its own web server.
- Business Objects / Crystal Enterprise and Cognos Enterprise require their own server.

Properly deploying Business Objects requires a trained analyst/administrator, so if you go down that road, that is another FTE to plan for.

**Summary**

A successful DR implementation requires:

- A new hardware platform with multiple servers.
- New server-related software licensing.
- New or retasked FTEs with appropriate training.
- New development and management software.
A new place for users to get reports from.
New report development.

So what do you get from all of that?

- Easier HITECH/ARRA reporting.
- Faster historical reporting (over quarter+ spans of time)
- Business and clinical data held past the purge setting in Meditech.
- Reporting that can combine Meditech and non-Meditech sources.
- Vendor-file exports with integrated SFTP delivery
- More attractive reports with graphing, charting, in-report filtering…
- More delivery options for your reports – by email, by smartphone, via web-pages or RSS, direct export to Excel, Word and PDF
- Ability to investigate data issues with ad-hoc querying
- Ability to combine data across modules which are not normally “open” to each other
- Data Quality reporting and alerting
- Ability to define your own indexes and get at data (particularly in BAR and PP) that is often very slow to access

(end)