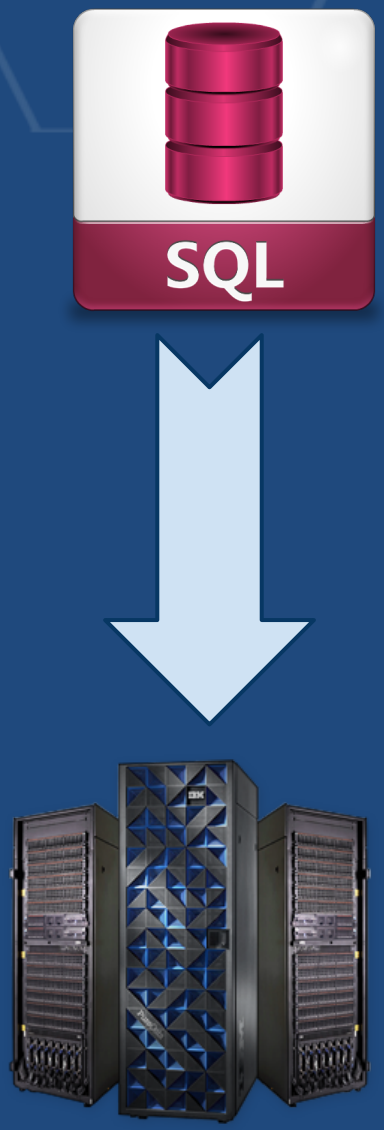
Migration Traps, Guiding Principles, & Best Practices

Migrating from SQL to Netezza

An Overview of Migration



Migration projects are, by nature,

a continuum of discovery.

This process comes with three high-level knowledge risks:

- *What you know*
- *What you don't know*
- *What you don't know that you don't know*

An Overview of Migration

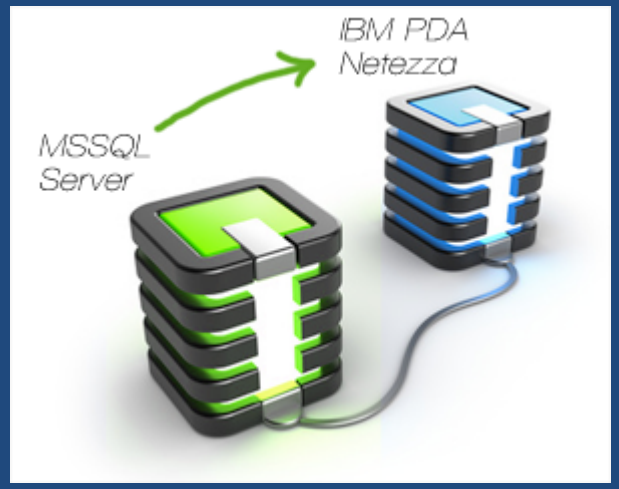
When it comes to...

- *What you know, and*
- *What you don't know*

...there are knowledge-risks that can be mitigated with research.

However, it's *what you don't know that you don't know*, however, might trap you.

Migration Traps to Avoid



Migration Traps to Avoid

1. Thinking You *Can* or *Should* Get it "Right" the First Time

- This is a myth.
- Successful migration cannot begin with the idea that we can "get things right" the first time.
- The problem is that the first time is not the only time that counts. What about the next time, and the next?
- We want to get things right for those times, too.

2. Trying to Etch the Development in Stone Too Early

- PureData for Analytics Netezza technology is so fast that it's easy to quickly dig ourselves out of a ditch.
- This means developers can experiment, prototype, profile, rework, tweak, tune, etc. and implement billions of records, build a model, only to chuck it if they don't like it or need it.

Migration Traps to Avoid

3. Thinking You Need a Go-Live, Cut-Over Calendar Date... IMMEDIATELY!!!

- Your go-live, cut-over date will probably change at least 3x.
- A migration project is a testing project.
- The migration and backfill portions are only surface-level tasks and challenges.
- When you're done migrating, most of your time has been spent testing and re-testing the result.

4. Allowing Your ROI Demand That You Load the Netezza NOW and Test LATER

- NEVER sign up for a migration on a fixed bid.
- This is because 80% of migration is testing, and 20% is development.
- Never underestimate the time needed to test the migration.
- A migration that is not testable is not trustworthy, and will eventually not yield an ROI.

Migration Traps to Avoid

5. Thinking Your Current Data Structures are *Good Enough* "As Is" to Proceed with a Migration

Follow good requirements-gathering principles by assuming *nothing* and asking the following questions:

- What if the existing structures in in your SQL DB are poorly designed or aren't optimized for Netezza?
- What if they eventually really turned out to be redesigned and/or re-engineered shadows of the original design?
- What if they have been pumped with performance props you don't end up needing at all in Netezza?
- What if the existing data model is just a product of organic growth (and has no real thematic design)?
- What if your existing structure isn't even optimized for its current home?
- What if the prior data modeler was just plain lousy at it?
- What if the existing users have been chomping-at-the-bit for new functionality in the existing model, and have been told by people promising to deliver to just hang on a bit longer (which means indefinitely)?
- What if they are all expecting all of the backlogged requests to get included as part of the migration?

Migration Traps to Avoid

6. Settling for a 10-20x Boost When You Could Have 100x +

- Don't make the mistake of trying to move data structures into place for the POC, only to get a 10-20x boost, and then move from there as a primary starting point.
- With a little data-structure tuning, you can get 100x… or even more!
- Otherwise, you're arbitrarily

7. Thinking You Really *Do* Know All of Your Data

- Now that you have Netezza's horsepower at your fingertips, it will be tempting to think you've got it all under wraps.
- However, it's far better to profile the data in all your primary tables first, in order to get to know it better, prior to starting development of your migration.

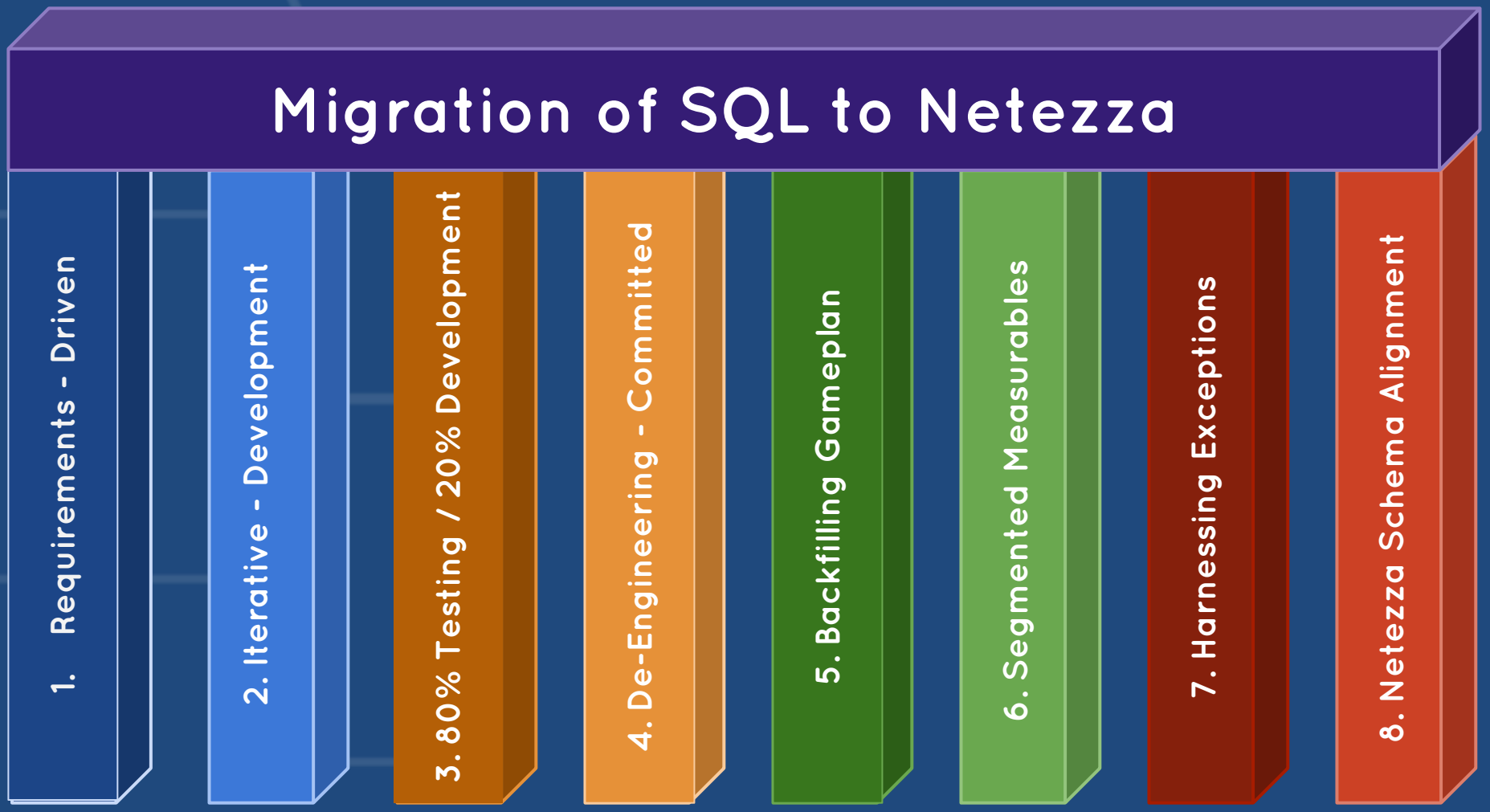
Foundations for Migrating MSSQL to Netezza

Foundations for Migrating SQL to Netezza

8

- There are eight features of migrating from MSSQL to Netezza, which undergird the whole process.
- BIQed is skilled at consulting with clients using these eight guiding principles along the way.

Foundations for Migrating SQL to Netezza



Foundations for Migrating SQL to Netezza

1. Requirements - Driven

Requirements - Driven

These are the types of questions that need answering *now*…

- What are the migrating requirements *right now*?
- So you simply want to migrate the *existing functionality*?
- How will we know (a) the *scope* of the effort, or (b) *when* we are done?

Foundations for Migrating SQL to Netezza

1. Requirements - Driven

Requirements - Driven

There are two categories of requirements for us:

1. High-level, highly-visible, functional solution requirements: the purpose for which the solution will exist in the first place.

2. Detail-level, non-functional solution requirements: operational, administrative, troubleshooting, long-term maintenance and data management.

Foundations for Migrating SQL to Netezza

2. Iterative - Development

Iterative Development

- Requirements-driven migration = iterations to develop & migrate.
- Reality tells us that all finality is actually artificial.
- Release iterations with regularity, making this iterative process an automatic backbone of the migration environment.
- Responsiveness to change will be the new environment, as fewer people are scared of change.
- Reduction of delivery time will be result as Netezza migration iterations are developed and implemented faster than you actually thought possible.

Foundations for Migrating SQL to Netezza

80% Testing / 20% Development

Requirements-driven migration
\+ Iterative development cycles
= 20% of your time developing
and 80% testing what you've developed.

- *Never underestimate* the amount of time needed to test the migration.
- Spend extra time *making it testable*.
- A migration that isn't testable isn't trustworthy.
- There's no "getting it right" the first time.

Foundations for Migrating SQL to Netezza

4. De-Engineering - Committed

De-Engineering Committed

- While "as is" can be good enough for testing, it's a lousy way to jump-start a migration. Here's why:
  - The data in the old system will need to move to the new system prior to migration. We help you plan for it now. *We encourage our clients that they should not underestimate the time this process will actually take.*
  - The functionality in the old system is probably buried under a mountain of performance props.
  - The complexity of the old system is artificial, requiring us to question everything together first. The presence of complexity doesn't dictate its necessity.
  - The former data model is suspect because it carries the constraints & weaknesses of the old system.

Foundations for Migrating SQL to Netezza

4. De-Engineering - Committed

De-Engineering Committed (Cont.)

- De-engineering looks like…
  - Taking the time to build out the desired target models you want to keep, and aligning them with the physics built into Netezza.
  - NOT migrating the data “as is” from the old machine to the new, no matter what you’ve been told.
  - Starting from the target and working backward to the source, which is the most effective way to guarantee that everything you build will have a purpose and do what the users want.
  - Mapping how every column arrived - in flow form, which is the most effective way to know that every target column has a place, as well as when to leave source columns behind rather than transport them into the new, only to throw them away later.

Foundations for Migrating SQL to Netezza

5. Backfilling Gameplan

Backfilling Gameplan

- We do not assume that you will backfill your data only once. Moving historical data from your old platform to the new one cannot be underestimated in terms of time or level of difficulty.

- On the surface, backfilling is as simple as…
  - Identifying which tables to move
  - Leaving the rest behind
  - Extracting the data and loading it in the Netezza

- However, there's one important feature missing: requirements-driven, iteratively developed migration necessarily *TESTS ALL data*.

Foundations for Migrating SQL to Netezza

5. Backfilling Gameplan

Backfilling Gameplan (Cont.)

- Backfill *may* be a one-time execution in production.
- But when it comes to testing, iteration is the gameplan.
- Therefore, the extract-load process should be optimized so that it can be repeated, making iterations efficient. This process means that you…
  - DON'T be afraid to have hundreds of extracts.
  - DO try to find an incremental, operational boundary that will allow you to pull the data only once, especially for historical activity.

Foundations for Migrating SQL to Netezza

6. Segmented Measurables

Segmented Measurables

- The first steps toward a successful migration are to..
  - Put it all on the table and identify what's actually doable.
  - Understand fully what you are about to "bite off."
  - "Bite off" the migration of data in "chunks."
  - Divide and conquer those "chunks" with your team resources.
  - Manage the teams so that you can migrate in stages: learn, refine, repeat.
  - Remember migration is an environment and not a project.
  - Prepare yourself for the long haul with set measurables, and manageable goals.
- Failing to remember these important guidelines are the reasons why many migrations lurch along.

Foundations for Migrating SQL to Netezza

7. Harnessing Exceptions

Harnessing Exceptions

- Treat complexity as an artificial, unnecessary symptom.
- Developing in an iterative environment will allow the migration to efficiently flag and handle exceptions and harness them.
- Deal with performance issues as optimizations, and not as engineering exercises.
- As you discover exceptions, decide that they will be…
  - Logically inactivated
  - Dealt with administratively
  - Harnessed with rules
  - Never allowed to dictate or become the rule.

Foundations for Migrating SQL to Netezza

8. Netezza Schema Alignment

Netezza Schema Alignment

- Make no mistake: *the schema of your Netezza model must align with Netezza's physics.*
- *You purchased* the machine because of its *performance.*
- The physical model of your migration is far more important than the logical or conceptual model.
- Remember why you purchased the Netezza in the first place. Otherwise, you may end up putting emphasis on the logical model only to find out that it doesn't translate into the physical model that naturally unlocks Netezza's power.

Benefits of Migrating with Netezza

Benefits of Migrating with Netezza

- *Netezza is fast enough to allow developers to make mistakes… even significant ones.*
- *It allows developers to build out whole data models, fill them with terabytes of data, test them, and toss them out - and all that in only a day or two.*
- *With any other data warehouse technology, by the time the data model is in place and filled with data, weeks or months will have passed, and there's no time to start over.*
- *Netezza's power allows developers to really analyze the data and find out what they want to know…. iteratively and quickly.*

Therefore…

Don't lock your model down too quickly.

It will be too brittle and functionally unworkable.

If you get yourself in a ditch, you can get out quickly.

Benefits of Migrating with Netezza

Don’t Have a Formal Data Transportation or ETL Environment?

Just get your data within “lasso-distance” of the Netezza, either by directly landing it into a flat-file located on a mounted volume within the Netezza host’s visibility, or following other viable intake scenarios.

Likewise, if you have a system that can understand nzload to push data into Netezza, we can mix and match any number of intake protocols to affect the data assimilation.

Benefits of Migrating with Netezza

Do You Already Have a Formal Data Transportation or ETL Environment?

BIQed can consult with you to make some strategic decisions as to how we'll partner together to leverage it in the future.

Likewise, we can assist you in pointing it to our intake schema with one caveat question:

How much set-based processing in the ETL environment is better-hosted inside Netezza?

Benefits of Migrating with Netezza

Do You Have a Low-Powered Environment?

BIQed can help you review the implementation plan in order to discover any…

- row-level cleansing required
- set-based transformations

We will want to separate these up front and hybridize the technologies for the best fit for…

- Fast & maintainable flow
- Orchestrated activity rather than interface-only interaction
- Synergy among all the moving parts

Best Practices for Migrating MSSQL to Netezza

Migration Best Practices

1. Inside the machine, BIQed will setup a configuration control environment, possibly including separately configured schemas for hosting the static snapshots of test data, intake, transformation, reference, data, master data, operational data, and application / consumption marts.

2. BIQed will review the existing outputs, working backwards from there.

Supporting existing outputs *is required*, and perhaps these may be the only reason for the warehouse's existence in the first place.

Migration Best Practices

3. BIQed will set up test cases on the Netezza appliance.

From the existing environment, we capture a snapshot of the source information and the final outputs.

Hosted on the Netezza machine in its own schema, our testing will execute in MPP form rather than on the original platform.

4. BIQed will review the data transportation mechanism.

We will either setup a simple intake protocol or point the existing ETL to the Netezza machine.

Ideally intake would happen to a schema that is dedicated to the source.

Migration Best Practices

5. BIQed will adapt our data modeling around performance necessities, pursuing practical considerations over purist ones.

6. BIQed will adapt your approaches to consumption so that the users don't directly access physical tables.

7. Without an ETL environment, BIQed will send raw data to Netezza into the given intake schema.

8. BIQed will perform row-level cleansing and integration inside the Netezza.

9. With an ETL environment, BIQed will break apart the row-level cleansing and the bulk integration, and re-host the bulk integration inside the Netezza.

10. BIQed will keep data transportation and row-level cleaning inside the ETL environment.

Migration Best Practices

11. BIQed takes a holistic view to data transportation. We will copy (or re-host) any data that is part of the integration stream (lookups, cross-references, master or reference data, etc.) into Netezza for this purpose.

(*It's better to plan for how these schemas will look now, because we'll need to focus a deliberate set of housekeeping and reconciliation functions on these data stores separately from the primary flows of information.*)

12. For continuous operations, BIQed will setup the appropriate Netezza data structures to support frequently arriving inbound information.

13. When converting ETL-based bulk integration, BIQed will initially replicate the functionality of what it does today. Then we will refactor the work for a more holistic approach to all internal "ELT" flows.

Migration Best Practices

14. When porting the functionality of stored procedures, BIQed will refactor cursor-based operations into set-based operations.

We will then test their outputs against the baseline. Likewise we will take serial SQL-statement sequences of insert / select functions, consolidate and port them functionally into Netezza.

15. BIQed will not knee-jerk port any cookie-cutter or cut-and-paste procedures. Rather, we will find the common pattern and functionally reproduce it.

16. As this is moving, BIQed will take the time to refactor the rehosted ETL/ELT integration into a holistic flow, “trimming the fat” and redundancy.

Migration Best Practices

17. BIQed believes you should consider continuing to support cubes and summaries with the explicitly stated plan to deprecate them in favor of a Netezza-centric solution.

This may require some creative consumption- point mechanics, but we believe it's worth moving away from the cubes and summaries.

<u>In the final analysis…</u>

- *This kind of factoring can occur with incredible speed and turnaround*, so it's important that this activity be executed by a smaller, focused team rather than a very large one.
- Once the flows are smoothed out and the patterns captured, delegate any additional conversions or testing and push the functionality toward closure.

BIQed is ready to consult with you on

Migrating from SQL to Netezza

visit www.biqed.com or
call 770-714-4148 to
get started today