DEFINING A DATABASE UPGRADE DESIGN METHODOLOGY

Nakul Vachhrajani

ABSTRACT

Databases are they key component in any product or Enterprise system. As multiple versions of these products and systems have been rolled out over the years, a need has been felt to define a database upgrade methodology to overcome the multiple issues that ISVs (Independent Software Vendors) face with ad-hoc upgrade strategies. This paper attempts to summarize the drawbacks of the conventional upgrade approach; recommend the key components of a database upgrade and define their boundaries, roles and responsibilities. Finally, the paper also presents recommendations for changes to the conventional SDLC (Software Development Life Cycle).

Keywords: databases, upgrade, methodology, transformation, product, software development, SDLC, ISV

1. INTRODUCTION

In today's world, business needs and technology changes faster than the blink of any eye. Product upgrades and technology refresh cycles are being performed by ISV's at an average rate as high as one upgrade per product per quarter per organization. New features are being introduced and existing features being made richer and capable to capture the changing needs of a growing business. As a business grows, it spreads across multiple geographic locations and beings to accumulate data from multiple data sources at an ever increasing transaction rate into its data centers.

Ensuring that an organizations' data is upgraded successfully with each technology refresh is essential because the keystone of success for any industry is its ability to use technology to leverage past historical data and project future trends based on domain health to implement and maintain a successful supply chain by taking business critical decisions in real-time and staying "ahead of the curve".

In the quest to achieve greater accuracy and availability of the data on hand, companies invested significantly into ERP (Enterprise Resource Planning) and CRM (Customer Relationship Management) systems in the early 1990s. Ever since they were implemented, they are amassing large amounts of data in the associated OLTP (On-line Transaction Processing) and OLAP (On-line Analytical Processing) systems. Data has started taking central stage and the accuracy, integrity, reliability and accessibility of the data in these systems has becomes more and more important. To make matters more complicated, this

data moves in a closed loop from the various interfacing systems spread all along the production process to the end consumers and external systems ultimately feeding back as trend data into the system. This has led to growth of "invisible" relationships which are difficult to define and control.

Like any other valuable asset, data needs to be well managed if it is to continue to provide business value. Unchecked data growth and distribution has increased risk, storage and management costs. Management of years of production data within a single system also impact service levels and disaster recovery initiatives. [1]

Each product upgrade comes with its own challenges, and while the overall upgrade follows the standardized software maintenance methodology, the lack of a standardized database upgrade methodology causes database upgrade to be handled in an ad-hoc manner and is at the mercy of the understanding and capability of the development team undertaking the upgrade. This is at a time when 50% of the upgrade process relates to customizations and integrations. [2]

This paper attempts to highlight the drawbacks of conventional database upgrade methodologies. Based on these drawbacks, it also recommends the key components of a database upgrade and defines their boundaries, roles and responsibilities to overcome these drawbacks. Finally, to help make these recommendations part of a sound methodology, the paper also presents recommendations for changes to the SDLC.

For the purposes of this paper, we will be restricting ourselves to the Microsoft technologies (more specifically, Microsoft SQL Server), however, the method described can easily be generalized to non-Microsoft technologies as well.

2. THE THREE PHASES OF A DATABASE UPGRADE – AN OVERVIEW

Any database upgrade can be divided into three distinct phases, which can be defined as under:

A. Data Cleanup and Conditioning

No production system is 100% defect-free. Syntactical, boundary condition and security related defects are generally encountered in the normal user workflows;

however logical and business errors manifest themselves as data issues and may lie hidden from the user, sometimes for years. When the associated business rules in the custom code written to handle off-the-shelf requirements are strengthened, additional validation put in, or integrated third party systems change their data conditioning requirements, then these defects come to light in the form of improper data flow or other integration errors.

This phase deals with an attempted rectification of these known errors, so that while the application code would ensure that such data is no longer generated, the pre-existing data is cleansed to prevent replication of these defects via data propagation.

This phase also deals with data archival, purging, and achieving modern PCI (Payment Card Industry) and other data compliance standards.

This phase does not modify the structure of the data, but instead focuses on data manipulation or conditioning.

B. Schema Upgrade

This phase exclusively deals with the transforming the schema, i.e. the structure of the data. Database object modifications are involved.

This phase involves minimal data manipulation, and is ideally independent of the source version.

C. Post Upgrade preparations

Once the database schema is transformed, multiple preparatory steps need to be taken to get the data back on line. This is typically a part of what is referred to as a “Go Live!” process for databases in industry jargon.

This phase involves ensuring that security around the database objects is applied properly; data organization has been tuned for optimal performance and all disaster recovery measures are in place. It also involves the creation of non-critical maintenance plans, night jobs and other partially independent activities.

This phase involves no data manipulation, but can involve minor schema changes.

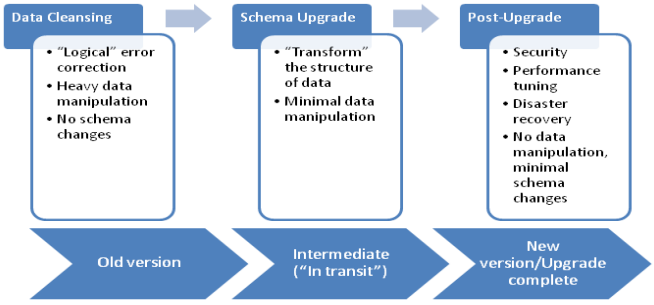

Figure 1. The Three Phases of a Database Upgrade

3. THE CONVENTIONAL UPGRADE APPROACH – AN OVERVIEW

A conventional database upgrade approach may not have the three distinct phases described above. Most upgrade packages generally have the data cleansing and schema upgrade phases combined into one, while others might have the data cleansing phase embedded into the schema upgrade phase.

Database upgrades are conventionally accomplished using a single, monolithic module comprising of all three phases, technically implemented as a set of sequential scripts and/or an upgrade package in the form of a DTS (Data Transformation Service) or a SSIS (SQL Server Integration Services) package.

Not having clear boundaries between each distinct task means that this method has a number of shortfalls. These are described in greater depth by the author below.

4. CONVENTIONAL UPGRADE APPROACH -DISADVANTAGES

A. Larger Downtime requirements

Because the data cleansing and post-upgrade preparations stages not two separate processes, but are included inside the actual upgrade window, the time required to perform the upgrade now varies depending upon the amount of data to be cleansed and conditioned, and also on the geographical distribution of the clients.

Most data cleanup operations do not need a system outage and those that do need can be planned during the weekly system outage cycles.

However, having a single package means that the upgrade planning cannot take advantage of the weekly maintenance windows and also of the times when system resources are occasionally free and available for use (e.g. the database servers are generally very lightly loaded during the night after the nightly jobs are complete and before the users come in for the day in the morning).

Larger time requirements mean that the system needs to remain "offline" for long, from anywhere between 8 hours to 36 hours during an upgrade, which depending upon the geographical reach and nature of business may not be acceptable. Ultimately, the more a business system stays offline, more are the losses.

B. Increased Cost of Technology

The inclusion of the data cleansing and post-upgrade processes into the actual schema upgrade package means that all the three stages need to be executed sequentially, in one go. Data manipulation and schema transformations require the creation of a large number of temporary objects and swap space. This is normally accomplished in the "tempdb" (temporary database) of Microsoft SQL Server.

The temporary space requirements during the upgrade are unusually high, and are not part of the normal workload. Therefore, one needs to plan and invest more in storage space, which otherwise would lie unused.

Also, in order to have more data processed in the shortest possible amount of time means that to decrease downtime, one must have higher computing power available. This computing power may not be in proportion to the normal requirements which are dependent upon the user workloads, which are a function of the number of concurrent users.

The suboptimal use of hardware decreases the return on technology investment.

C. Increased Risk

Because the operations on the data during a data upgrade are not segmented, a single error has the potential to cause the entire upgrade to fail.

For example, data cleansing involves major modification to the data, which means that the database server needs more temporary space for data manipulation. Depending upon the transformation approach, the space requirements on the database server increase even further (sometimes becoming as high as twice the normal requirements), which, if not planned for beforehand, will result in the entire upgrade being aborted by the database server.

Even if no technical challenges are encountered, it is possible that the business rules and transformations are not properly enforced. Even if the upgrade fails to reset a single flag or update a particular column because of faulty data condition

evaluation logic, the entire application may not work as expected, increasing sustenance costs.
Once failed, the only option is generally to start all over again, further increasing the total system downtime.

D. Limited upgrade models

The lack of boundaries in the upgrade phases, means that the entire upgrade can either be conducted either "on-premise" or "in-house".
"On-premise" upgrades mean that the ISV vendor must visit or be provided remote access to the deployment site/data center for executing and monitoring the entire upgrade, reducing automation in the process, and also increasing idle man-hours while the upgrade is in progress.
An "in-house" upgrade requires that the data be shipped in-house from the deployment data center which implies an increased technology cost, increased transit damage and security risk, and also an increased downtime.

E. Synchronization issues with interfacing systems

Once any upgrade is in progress, all interfacing systems are offline because the database itself is generally under exclusive access.
Having data conditioning as part of the conventional monolithic upgrade process means that interfacing systems may be unaware of the changes that take place as part of the upgrade.
For example, the Event management system of a hospitality enterprise system typically consists of an interfacing system called a "reader board". A reader board is a 3rd party system that provides a display of the event name, event times, and event location for all guests in a convention center. Typically, this data is buffered for a period of time to avoid loading the feeding OLTP system.
Assume that the parent (i.e. "feeder") system had a bug wherein the system allowed the end time to be less than the start time. Once discovered, a business decision is taken to update all such occurrences to now have an end time is updated to be equal to the start time. This data manipulation is done as part of the upgrade.
However, because the reader board interface is offline, it fails to pickup this change and may continue to display incorrect data based on whatever data the system had in its cache.
Similarly, there are many other scenarios wherein the interfacing systems simply fail to detect a change done during the data conditioning phase only because all interfaces were offline when the change was made.

F. Data warehousing impact

Accuracy of data in a data warehouse is of utmost importance because corporate strategy and planning, operations, decision making, marketing, selling, customer satisfaction and business growth all hinge on the MIS (Management Information Systems), which is powered by the data warehouse. If the data conditioning and schema upgrade operations involve rectification of float values or change in the precision of decimal values, the warehouses will reflect a change in historical revenue values immediately after the upgrade.

What becomes difficult is to point out the exact nature of the changes which the management should expect and also what operation caused the change - the data conditioning or the schema change.

Another impact is that immediately after the upgrade, the nightly ETL (Extract-Transform-Load) process is left with a large amount of data to process, which might possibly lead to the warehouse marts being under process well outside of their intended nightly window.

G. Business rule changes impact Cost of Quality

Data conditioning is required because the data currently in the system is technically correct, but logically incorrect from a business sense. Because data conditioning is not separate from the upgrade process the addition or change in a single business rule means that the entire upgrade package needs to be changed.

A change in the upgrade package directly mandates the execution of a full quality assurance cycle on the upgrade package, increasing the cost of quality per change/amendment in the business rules.

H. Logistical issues

ERP products nowadays are global products. An ISV in the United States may need to deploy the system on the other side of the Earth in the Far East. Software transport costs directly depend upon the mode of transport. A physical media transfer on a compact disc attracts taxation issues, whereas an onsite transfer is restricted by the available bandwidth.

Similarly, if the upgrade is to be carried out “in-house”, then shipping the data from one country to another has its own additional challenges as the transfer must comply with the various data compliance and privacy norms.

Although the choice of upgrade mode is determined on multiple points, larger upgrade packages mean that either party involved in the transfer needs to pay higher media cost and/or higher bandwidth cost.

I. Source Version dependency

The destination schema resulting from the upgrade of an enterprise system is a constant irrespective of the source version. However, if the upgrade does not have the clear boundaries between data cleansing and schema upgrades, this is not a possibility.

Data issues generally arise because of a defect in the code base of a particular product version or release. These issues need to be rectified before upgrading to the next version. Most enterprise systems today have seen about 10 different releases (one per year, on an average). Conventional upgrade packages of these systems are now burdened with the responsibility of taking care of those data issues that can only been seen in the early versions of the product.

Because the data conditioning phase is still dependent on the source version the entire upgrade is now dependent upon the source version.

J. Archival/purging window [3]

IDC (International Data Corporation) estimates that 45 gigabytes of data currently exists for each person on the planet: that's a mind-blowing 281 billion gigabytes in total. While a mere five per-cent of that data will end up in enterprise data servers, it is forecasted to grow at a staggering 60 percent per year, resulting in 14 Exabyte of corporate data by 2011.

Compounding data growth challenges are the global data retention regulations that require the management and storage of different types of data for extended periods of time to satisfy the demand for documented authenticity and accuracy of the records. For example, the HIPAA (Health Insurance Portability and Accountability Act) and SOX (Sarbanes-Oxley Act) require that data be retained for a minimum of 5 years before it can be purged from a system.

Maintaining historical data involves partitioning and archiving it to segregate and remove historical data safely from the production environment. Finally, after the required retention period, this data needs to be retired and completely eliminated from the system, freeing up valuable capacity for primary business needs.

However, before data is archived, it needs to be conditioned and audited to ensure that it is free from logical defects. Conventional upgrades which do not have separate data conditioning phases cannot allow for such auditing intervals and hence do not normally have archival/purging windows.

K. Data Compliance/Privacy [4]

The PCI (Payment Card Industry) Security Standards Council is an open global forum, launched in 2006, that is responsible for the development, management, education, and awareness of the PCI Security Standards, including the PCI DSS

(Data Security Standard), PA-DSS (Payment Application Data Security Standard), and PTS (PIN Transaction Security) requirements.

With banking changing over the years towards electronic fund transfers and payment cards (Credit Card/Debit Card), more and more systems are now accumulating the sensitive, personal banking and payment card information. These businesses put the personal information of millions of customers at risk in case the information security around their databases and networks in compromised.

Compromised data negatively affects consumers, merchants, and financial institutions because all it takes is only one incident to severely damage the business's reputation. Account data breaches can lead to catastrophic loss of sales, relationships and standing in the community. Other consequences are decreased share prices, lawsuits, insurance claims, cancelled accounts and heavy government fines.

The PCI DSS defines a set of 12 requirements for any business to help prevent security breaches and theft of payment card data.

The 3 pivotal points around which these 12 requirements are organized are:

•**Assess** – "to identify all technology and process vulnerabilities that pose risks to the security of cardholder data that is transmitted, processed or stored by your business"

•**Remediate** – "Remediation is the process of fixing vulnerabilities – including technical flaws in software code or unsafe practices in how an organization processes or stores cardholder data"

•**Report** – "Regular reports are required for PCI compliance; these are submitted to the acquiring bank and global payment brands that you do business with"

All the three are ongoing processes because as data compromise becomes more and more sophisticated, it becomes more and more difficult for an individual business to stay ahead of the curve. They are multi-pass, iterative processes, which cannot be embedded into the single, monolithic one-pass conventional upgrade model.

L. Troubleshooting issues

Last, but not the least important is the fact that troubleshooting an upgrade process with no or minimally strict phase boundaries means that troubleshooting failed upgrades become a complex process of deduction from a possibly large log.

Each of the three phases can encounter different types of issues:
• **Data Conditioning** - mostly data integrity issues
• **Schema Upgrade** – mostly structural, syntactical and technology issues
• **Post-upgrade** – mostly security, performance and disaster recovery issues

Each phase needs different kinds of possibly mutually exclusive troubleshooting tools and skills, which make troubleshooting a one-step upgrade a complicated problem – one which has no generic solution.
Also, because a monolithic upgrade does not allow for conditional flow control, troubleshooting failed upgrades translates into difficulties with reproducing the scenarios and increased time because each attempt needs a start from scratch.

5. REVISITING THE THREE PHASES OF A DATABASE UPGRADE

Now that we have an understanding of the multiple issues that arise out of not having clearly defined boundaries between the database upgrade phases let us attempt to define the roles and responsibilities of each phase.

TABLE I. BOUNDARY DEFINITION

Parameters	Upgrade Phases		
	Data Conditioning	Schema Upgrade	Post-upgrade preparations
Primary Role	Data Cleansing	Schema transformation	Prepare for use
Data Manipulation	Major changes	Minor changes	No changes
Schema Changes	No changes	Major changes	Minor changes
Production Downtime?	No	Yes	Partial
Upgrade mode	On-premise	On-premise/In-house	On-premise
Ideal System time utilization	Yes	No	Yes
Integrations/ Warehouses	Online	Offline	Partial
Impact of Business chagnes	Yes	No	No
Automation	No	Yes	Partial

Parameters	Upgrade Phases		
	Data Conditioning	Schema Upgrade	Post-upgrade preparations
Independent Process	Yes	No	No
Risk (Business)	High	Low	Low
Risk (Technical)	Low	High	Medium
Source Version dependency	Yes	No	No
Data Archival/Purging	Yes	No	Yes
Data Compliance/Privacy	Yes	Yes	Yes

As can be seen from the table above, the clearly defined boundaries provide the following benefits:

A. Higher system uptime

The key phase influencing downtime is essentially the schema upgrade. If the schema upgrade is made an independent phase, system uptime dramatically improves, theoretically by as much as 66%.

The Data conditioning phase can theoretically be executed when the system is online and in production. The additional benefit is that data cleansing can be divided into multiple sub-phases such that the execution is spread over multiple time slots taking advantage of the system "idle time".

The post-upgrade preparations can also be made partially online by only choosing to carry out the bare minimum preparations, like applying security and disaster recovery measures offline. The performance tuning operations can be partially online thanks to the technological advances made in Microsoft SQL Server 2008.

Overall, a clear definition of the phase boundaries results into higher system uptime, and the usage of system idle time also results in reduced cost of technology.

B. Reduced risk

Data manipulation, as discussed earlier, is a result of "logical" defects or changing business rules. Assume that we have a business rule like "A sales order must have a sales owner associated with it." Later on, we have an amendment added that says "If the original sales owner has retired or is no longer in the system, a

default sales owner needs to be assigned." This change mandates a change to the data conditioning routines. Most business risk is thus only associated with the data cleansing routines.

The technical risk associated with system resources and configuration is more concentrated on the schema upgrade and post-upgrade preparation phases.

This risk can further be reduced by segmenting the schema upgrade phase into two: an "upgrade advisor" phase and the actual upgrade phase.

In the conventional approach, the business and technical risks are applicable equally to all the three phases. Having independent phases means that the risks are now isolated within any given phase and do not cascade/distribute to other phases.

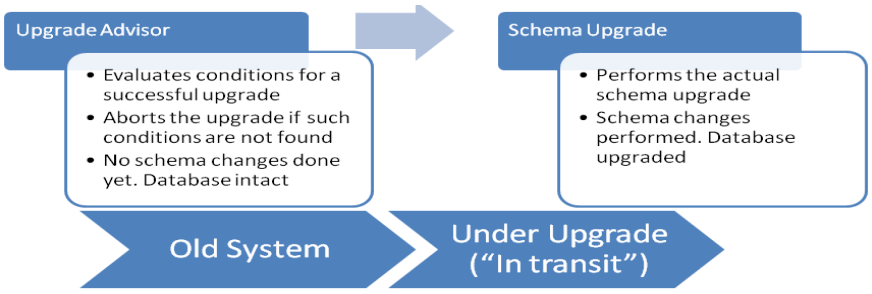

Figure 2. Reducing technical risk in the schema upgrade phase

C. Possibility of Hybrid upgrades models

Because all phases were conventionally packaged into a single upgrade package, the entire upgrade could either have been done "on-premise" or "in-house". A hybrid solution was not a possibility.

However, with the separation of boundaries amongst the phases, it is now possible to have a hybrid approach. On premise model is now the most suitable for the data conditioning phase. Depending upon the nature of schema transformation and the space requirements, this phase can be either executed on-premise or in-house. The only phase that cannot be done in-house is the Post-upgrade preparations.

If the end customer location (in case of an on-premise installation) does not have a dedicated IT staff working with them, a hybrid approach is a much easier option for them and the ISV undertaking the upgrade.

D. Improved interface synchronization

The data conditioning can take advantage of the system idle time and can be performed with the system and its interfaces online. This means that all

peripheral applications will be notified of any data changes that occur in the source system.

E. Improved warehouse performance

The business will now be able to see an immediate effect of the data conditioning procedures in the MIS reports and on the interfaces. Because the data conditioning can also be controlled and planned, any issues can be identified immediately and corrections done before the system is upgraded and rolled out.

Also this indicates that there should be no data change during the schema upgrade.

Because all data change is now distributed into the data conditioning phase and not in the schema upgrade phase, the amount of data to be processed by the nightly ETL jobs also distributes itself. This leads to efficient ETL processing and the marts are cubes are ready for use when the business leadership arrives at the office.

Please note that what improved interface synchronization and warehouse performance also implies is that the data conditioning needs to be a very well tested operation because all data changes made will be propagating across the entire system.

F. Reduced impact of business rule changes

Business rule changes only impact the data cleansing phase, which can further be split up into multiple sub-phases depending upon the source version for which the rule applies.

Taking our example from point #IV-B ("Reduced Risk"), if the business rule to have a default sales owner was introduced in version #n, only upgrades from versions #n-m to #n are impacted. Any upgrades from version #n to version #v are not impacted.

The containment of the business rule changes means that the business rule changes may only end up mandating a quality assurance cycle of the data cleansing phase. Rest of the upgrade process is not affected, and therefore does not need to be tested, increasing the "go-to-market" time.

G. Improved Logistics

Smaller, more modularized upgrade phases imply that they can easily be transferred from one point to another. While conventionally, one would have to wait for the entire module to be transferred resulting in process downtime, the necessary modules for the schema upgrade and post upgrade processes can easily

be transferred over the background when a separate team is working on the data conditioning phase, reducing process downtime.

H. Partially independent of Source version

The only phase dependent upon the source version should be the data cleansing phase as described in point #IV-E ("Reduced Impact of business rule changes").
The post-install preparation is completely isolated from the source version because it comes in sequence after the schema upgrade process.
The schema upgrade process needs to be generic enough to assume that when compared with the destination schema; any object if not existing will be created. Transformation can be done based upon a set of data views which expose information in a standardized format irrespective of the source version.
We will be discussing a few approaches to the schema upgrade phase in the upcoming section.

I. Data Conditioning is now a completely independent process

Data conditioning may be required periodically depending upon the nature of the application and also depending upon the stability of the code.
Conventionally, because data conditioning was a part of the entire upgrade process, the frequency of custom data cleanup requests increased. Now that we have a clear boundary to the data conditioning phase, it can be executed independent of the upgrade cycle, improving data quality.

J. Data Archival/Purging window

Any data earmarked for archival and purging must audited before the archive/purge operations are performed. Auditing can only be performed once it is ensured that the data is free from system introduced defects (i.e. data issues introduced due to code defects). This is achieved during the data conditioning phase.
The data conditioning phase allows for separate data conditioning and auditing intervals. This phase also allows for an iterative archival/purge and allows the archive/purge to be propagated to various interfacing and data warehouse systems.

K. Data Compliance/Privacy

The 3 pivotal points of data compliance/privacy can easily be mapped to the three phases of a database upgrade.
• Assessment and remediation of storage vulnerability is achieved during the Data Cleansing phase

• Remediation also extends into the Schema upgrade phase in case schema changes are required to support PCI compliance
• Report – The post-upgrade phase involves preparation of a report on the PCI vulnerabilities assessed and remediated during the entire upgrade process
Besides, because all are ongoing processes, the assessment and remediation from the data conditioning sub-phase and the reporting sub-phase can be repeated as many number of time as applicable.

L. Improved troubleshooting abilities

Each of the three phases can now be designed keeping in mind the individual troubleshooting needs of each.
The data cleansing phase may need a detailed log of each of the data modifications done and also need the ability to configure for step-by-step or point-in-sequence execution.
The schema upgrade needs that the nature of the schema changes and their progress be logged.
Wrapping each block of code in both stages within a transaction means that restarts from a failed point are now possible.
The post-upgrade preparation phase generally needs an all or none approach and must be logged accordingly.

6. TWO APPROACHES FOR SCHEMA TRANSFORMATION

The Schema transformation phase can be implemented via the following two approaches depending upon the business need. The benefits and drawbacks of each have also been mentioned.

A. Full tranformation

A full transformation means that the source schema is not modified at all. A parallel new schema is created and objects created under the same. Data is then transformed (or "pumped") into the new schema from the old schema. The old schema is finally destroyed.
Benefits of such an approach are:
• Such a transformation maintains a fixed schema structure.
• Hence, structure verification at any point in the life cycle is possible.
• Code development can take advantage of the schema structure.
• This approach has simple documentation needs.

Drawbacks of this approach are:
• Such an upgrade modifies all database objects.

• Because it modified all database objects, this approach is slow and needs a larger downtime window.
• Because an identical schema is created and a copy of the data created before the older schema is destroyed, this upgrade approach needs larger disk space (about twice the original database size).
• Schema customizations created as part of custom performance and/or availability exercises are not preserved through the upgrade .

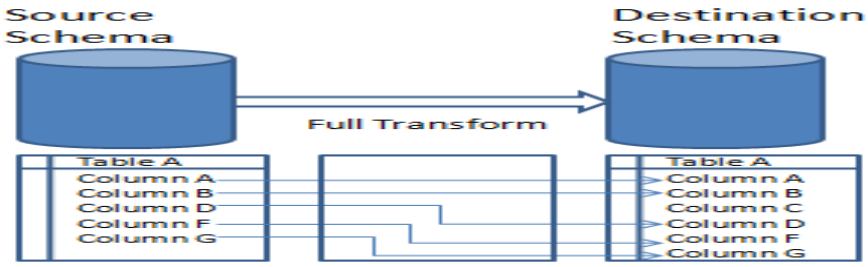

Figure 3. Conceptual overview – The full transform method

B. "Delta" transformation

A "delta" transformation only applies the necessary changes to the original source schema. Thus, if a schema change is not present, the change is applied. If a change is partially present, the appropriate schema rectifications are done.
Benefits of such an approach are:
• Because only required changes are applied, this upgrade approach requires less disk space.
• Only applying the required changes means that this upgrade is a faster upgrade because the amount of data to be moved/modified is proportional to the number of schema changes.
• There are fewer points of failure because only the required database objects are modified.

Drawbacks of this approach are:
• The upgrade package becomes complex to troubleshoot due to the increased number of decision paths involved.
• This approach needs more detailed and complex documentation.
• This upgrade cannot maintain a pre-specified column order (new column additions appear towards the end of the table), and hence cannot be used for an application which is dependent on column order.

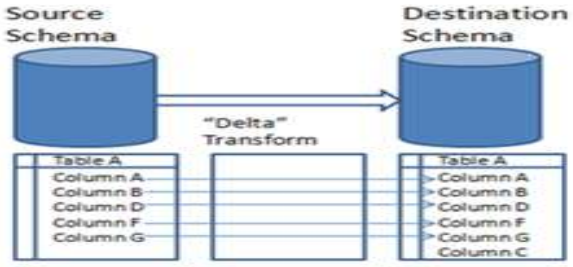

Figure 4. Conceptual overview – The "Delta" transform method

7. FEEDBACK TO THE SDLC

Application designers have traditionally believed that creating a sound core database design is the most important task. Database upgrades have traditionally been handled in terms of design and development methodologies solely by the database developers and administrators. Thus, database upgrades have not enjoyed the privilege to be an integral part of application design and development.

Now that we know the drawbacks of an ad-hoc approach, and also have some guidance on overcoming them, database upgrade design needs to be embedded into the standard SDLC (Software Development Life Cycle).

A. Integration into the business design phase

Any RDBMS (Relational Database Management System) system needs to ensure the 4 basic properties of data, which are popularly abbreviated as ACID (Atomicity, Consistency, Isolation and Durability). However, only the business needs can help engineers ensure Validity of the data in a system.

However, in spite of being able to play such a pivotal role, most business requirements are defined without keeping in mind database upgrades. This ultimately results in multiple iterations and revisions to the business design, cascading changes into already developed features and also custom logic being introduced into the database upgrade.

The business needs to be considered with respect to the nature of data modifications to be done during the data conditioning phase such that the schema upgrade is free from data modification should thus be a part of the business design deliverables. This also helps to ensure that all data changes are communicated to interfacing systems well in time to allow for timely and planned changes in those systems as well.

B. Database upgrade development is a parallel process

Often large enterprise products are developed following the iterative process. As modules roll out from the development teams, they are individually tested against a standard set of pre-defined data in the first iteration. In later iterations, these are tested together as a group during integration testing. The concept behind such an approach is to be able to design the database schema for any feature from scratch. This allows the design to be developed independent of the restrictions that might have been placed by attempting to keep the older design pieces intact. Later on, once the design stabilizes, the migration is worked out.

While this approach has a very visible and valuable advantage of being independent during the design phase, it is not advisable to follow this strategy during development. This is because it tends to increase the cost of quality by increasing the number of testing iterations.

It is recommended that just as a particular unit/feature of the application is released for quality assurance, the associated data migration modules should also be released in parallel.

This does not mean that the quality assurance strategies be compromised. Both static data and migration testing for a module can now be completed in parallel and defects fed back into the development cycles early on, ultimately reducing the cost of quality and also improving the overall stability of the upgrade process.

8. QUESTIONS PENDING RESEARCH

This is a working paper, and hence there are multiple questions that the author is currently researching. A summary of these questions is provided below.

A. Suggested backup/disaster recovery points during the upgrade

More than normal data activity usually occurs during an upgrade cycle. The cycle can be as concise as a single day of downtime, or may be spread across multiple days. It becomes very important to have proper backup sets to allow for complete recovery to either a state before the upgrade was started off, or to anywhere during the upgrade. It is also important to have a complete restart of the backup set creation and other routines once the upgrade is complete.

B. With respect to Microsoft SQL Server, what are the best database recovery models for each phase of the upgrade process?

Microsoft SQL Server offers three distinct database recovery models, with their own pros and cons – Full recovery, bulk-logged recovery and simple recovery.

Each of these three stages of the database upgrade process can be paired up with one of the three recovery models available. The author is trying to research if using different recovery models during each upgrade phase is actually a good idea.

C. Suggested quality assurance models for each upgrade phase

Each upgrade phase has different needs and mechanisms of quality assurance. For example, the data conditioning phase needs to be centered on validating the changes in the stored data, whereas the schema upgrade need to be focused on the changes to the structure and organization of data.

The author is studying the various database testing strategies available to identify the recommended quality assurance models for each of the database upgrade phases.

D. Security migration/upgrade

Every database is fortified with various security measures to prevent intrusion. These security measures may also be affected by the database upgrades.

Most common security impacts during a database upgrade are seen with respect to change in the operating permissions on a particular set of database objects.

While all security arrangements may not be automated for upgrade, some can. The author plans to research on how security principals such as database users and logins can survive an upgrade.

E. Coupling infrastructure upgrades with database upgrades

More organizations would prefer to upgrade their hardware and/or the underlying software (operating systems on the servers, SQL Server versions, etc) along with the scheduled product upgrade in order to reduce downtime.

The author finds this concept intimidating as he believes that undertaking infrastructure upgrades during a product upgrade increases the number of variables in the environment, thus increasing the chances of a failure. The author intends to refer to past case studies and interact with industry experts to understand the experience of the years.

9. CONCLUSION

Information is the lifeline of business today. As newer and better analytical systems are being developed, and data from enterprise systems being extracted to business intelligence systems, it is of utmost importance to ensure the accuracy and logical consistency of data in any database. The success of an enterprise system upgrade or technology refresh depends solely on the upgrade of its

underlying database. Allowing inconsistent data to remain in a system and creating new designs to work around bad data will only cause new upgrades to become more complex and will have a higher risk associated with them.

It is time for the Software Development and Maintenance Life Cycles to appreciate the importance of database upgrades, and treat databases at par with the application components. Regular cleanups, tune-ups and audits can only be enforced if the underlying database upgrade process is sound and well defined.

With this paper, the author has a summary of the issues that can be encountered with ad-hoc database upgrade strategies. The author has also attempted to overcome these by identifying the 3 major phases of any database upgrade and assigning boundaries, rules and responsibilities to them. The author plans to continue his research on the open questions pending research in order to propose a complete database maintenance methodology.

ACKNOWLEDGMENT

This paper has grown out of countless hours of troubleshooting failed enterprise database upgrades, application failures due to bad data, and trying to figure out the root cause of the failures to ensure that they do not happen again. Careful evaluation and debate over each of these failed upgrades on long walks with my mentor, Mr. Mukesh Nagda has helped a clear picture of an ideal software and database sustenance methodology.

Under the guidance of my mentors, Mr. Mukesh Nagda and Mr. Nikunj Patel, I have learnt a lot about enterprise systems (both legacy and new), how domains function, how industries work and how every piece of information in the database is important to the management. They have been my mentors not only in the industry, but also in my personal life. I continue to draw inspiration from them, and whatever I am today in the industry is because of them. I thank them for all their hard work and patience and the trust and confidence that they have shown in me.

I would like to extend sincere thanks to manager, Mr. Ashish Buch, who has always been supportive of my academic pursuits. I also thank my colleagues at Patni Computer Systems Limited for being great participants in any conversation and also for being patient with me in my many whims.

Finally, I thank my family members for putting up with me whenever I am up at the computer for endless hours in the night and on the weekends. Their support, trust, encouragement and comfort provide me the energy to scale even greater heights.

REFERENCES:

1. Christy Pettey, Ben Tudor, “Data Growth as the Largest Data Center Infrastructure Challenge”, Gartner Survey , http://www.gartner.com/ it/page.jsp?id=1460213
2. Markus Zirn, "Application Upgrades and Service Oriented Architecture", Oracle Corporation, April 2008, http://www.cio.com/white-paper/646494/Application_Upgrades_and_ Service_Oriented_Architecture
3. “Control application data growth before it controls your business”, IBM, September 2010
4. PCI Standards - https://www.pcisecuritystandards.org/security_ standards/index.php

AUTHOR’S PROFILE

Nakul Vachhrajani is a Technical Lead and systems development professional with iGATE Patni having a total IT experience of about 7 years. He has comprehensive grasp on Database Administration, Development & Implementation with MS SQL Server and Visual C++ & C#.

A guest columnist for SQLAuthority.com and SQLServerCentral.com, he contributes to academia and has presented a Microsoft Virtual Tech Days webcast on "Underappreciated Features of Microsoft SQL Server".

Blog: http://beyondrelational.com/blogs/nakul/default.aspx