Chapter 1 : Cognos Framework Manager Best Practices

Regardless of the data model complexity there are a number of best practices that should be followed when building models using Framework Manager: Never allow Cognos to create joins automatically when importing data sources. Cognos rarely chooses correctly and this inevitably creates future rework.

Tuesday, 16 April Best Practices for Cognos Framework Manager Development For this blog I am going to focus on the best practices, organisation and overall methodology of Framework model development. It describes the IBM recommended four layer approach, detailing the purpose and reason for each of the four layers and then moves on to discuss the use of data sources and avoidance of stitch queries. Before Development Starts When designing a framework model you want to be sure the model you develop will withstand future changes and development without impacting existing reports which have been built upon it or requiring changes to defined calculations and filters. To achieve this in either a single of multi-language model it is important that at the very start of the modelling process you decide and set a design language. A design language should be a language which is not required within your environment. You should ensure throughout the design of your model that the design language is never used as the active language; by default Framework manager sets the design language to the active language and therefore this must be changed before proceeding. This will help ensure the model you create is durable. Each layer has a distinct function and set of activities. The layers build upon one another, with the database layer being the foundation of the model. Packages, data source connections and parameter maps fall outside of the modelling layers. Database Layer The database layer contains data source query subjects based directly on the objects found in the underlying database. Always try to use unmodified SQL for query definitions in this layer as modifying the SQL disables framework meta-data caching capabilities forcing Cognos to validate the query subject at run time which reduces the efficiency of the end report. The same applies to filters and calculations; avoid adding these to database layer query subjects. Joins, cardinality and determinants should be set at the database level but other than these no further modelling work should be done at the database layer. Extra time spent during development ensuring these are correct can pay off tenfold in report run times later! Modelling Layer The Modelling layer is where the majority of development work is done. A well designed modelling layer will add business context and understanding to the data in the model ready for consumption by business users and report developers. There are a number of tasks which are done in the modelling layer; listed below are some of the most common but this is by no means a complete modelling guide. Start by organising query items taken from the database layer of your model into logical query subjects; it is perfectly acceptable and normal practice to combine elements from multiple tables into single logical query subject. Give query items appropriate names that reflect recognised business terminology. Remember to set descriptions, tooltips and multi-language definitions along with output formats and usage properties. Define any required prompts, prompt-based calculations, calculations and filters. To ensure your model is as durable as possible base calculations and filters on items in the database layer of your model. Organise query subjects in a user friendly manner using folders where necessary to logically group similar query subjects. Dimensional Layer The Dimensional layer is used for presenting dimensionally modelled data to users; it is often mistaken as being the source used for building transformer cube models but these can and should be built off the presentation layer. As with the modelling layer, you should use the dimensional layer to add business context to the data from the database layer by renaming and adding appropriate descriptions and tooltips. To avoid confusion ensure query items which appear in both the dimensional and modelling layer have the same name, tooltip and description. Presentation Layer The sole purpose of the presentation layer is to make it easy for users to find the information they need. No modelling is carried out in the presentation layer; it should contain only shortcuts to items in the modelling layer along with folders and namespaces for organisational purposes. The result is a new namespace in the modelling layer, which contains shortcuts to the selected Query Subjects. The new namespace can then be moved to the presentation layer. Data Sources To ensure your model is efficient pay particular attention to data sources. Even in a well designed model there may be multiple data sources but they should all point to the same Cognos Connection data source. Having data sources within your model which point to different Cognos Connection data sources, even if they ultimately query the same database, will result in Cognos opening multiple connections to the database server, generating unnecessary stitch queries and carrying out basic join work on the Cognos application server instead of the database server which is much more efficient for this type of work. Avoiding Stitch Queries Stitch queries are a expected single query which is broken down into two or more individual queries before being sent to the database. The results from the two or more queries are then joined on the Cognos application server which is in most cases, not very efficient. Stitch Queries occur when a model has multiple one to many relationships usually as a result of data coming from multiple fact tables which can only be joined through a conformed dimension. As an example, consider the model diagram shown below. The Sales fact and Inventory fact tables can only be joined through one of the three conformed dimensions Time, Product and Staff. This will mean any report which takes data from both the Sales Fact and Inventory Fact tables would result in Cognos generating stitch queries. This is the correct approach for this type of query but frequently occurs due to incorrectly set cardinality within the Framework model. It is essential to ensure cardinality is set correctly throughout your model to ensure unnecessary stitch queries which can cause report performance to be very poor. Posted by Unknown at.

Chapter 2 : Cognos Report Studio Best Practices in Development

Cognos FM Best Practices There are a number of best practices that should be followed when building models using Framework Manager: Never allow Cognos to create joins automatically when importing data sources.

This reduces the maintenance and makes the model more understandable. Suitable for transactional databases. Can segregate relational objects Query Subjects and dimensional Objects Regular and Measure dimensions properly. Provides better scalability and supports new object inclusion. Database layer should not be used for reporting. Shortcuts should be used wherever possible, instead of copies. Shortcuts make it easier to maintain metadata because changes to the target object are reflected in the shortcut. While publishing the package, base view should not be included. Only the Business Views with Filters should be included. Cardinality describes the association between two query subjects and is set at each end of the relationship. It should be ensured that the cardinality addresses business needs. Following are the considerations to take while defining cardinality: Facts - Dimensions For a inner join specify only For outer joins, In order to define the behavior expected when querying at a one or more levels of time for example, dimensional information is used. Levels are defined for years, quarters, months, and days. A key is defined for each level and in this example, that key is sufficient to uniquely identify the level. There may be following types of security through Framework Manager Model Data Level Security restricts the data relevant to the user logged in. Object Security enables user to access only relevant Report Objects Specify user and administrative access to the package. Avoid having Orphan Tables Filters should be generic so that they may be used across all levels. In case of customized requirement, it should be done at Framework Manager and not at Report Level. Separate namespaces should be used when two objects of the same type having the same name is required. Source code control repository should be used whenever possible to restrict access and track changes to the projects and segments. Separate views for each user group should be created. Many to Many relationships should be avoided as they generate stitched queries. Instead Star Schema grouping should be used. Recursive relationship should be converted to fixed hierarchy using shortcuts. In a fact table query subject, exposing any query items that are not facts measures with an aggregation rule should be avoided. If necessary, they should be exposed as a separate model query subject that looks like a dimension. Whenever possible reduce the number of query subjects. Represent each discrete business concept with a single model query subject. Conformed dimensions should be given the same name in each namespace where they appear. If two fact tables have conformed dimensions, but different levels of granularity for that dimension, then for each fact table only the relevant levels should be included. Only the objects that will be reported on by users should be published in a package. Surrogate keys should be hidden while publishing a package. It is recommended to use determinants to identify certain levels of aggregation within the query subject. This is particularly useful when dealing with multi-fact and multi-grain queries. Avoid loops while creating relationships, by using shortcuts, as they behave unpredictably while reporting. Generic filters or calculations should be created in Framework Manager. Its recommended to create a view that caters to a group of reports, which makes the reporting easier to maintain.

Chapter 3 : Cognos 10 Tips: Best Practices for Cognos Framework Manager Development

Best Practices in Modeling IBM Cognos Semantic Layers in Framework Manager In October , we published the Insight " Best Practices in Modelling IBM Cognos 8 Semantic Layers " on our website. Since then, this page has become the most visited page on our website, with over pageviews since publication, clearly showing the need for the topic.

Provides better scalability and supports new object inclusion When importing from a relational data source, cardinality is detected based on a set of rules that you specify. The available options are: The most common situation is to use primary and foreign keys as well as matching query items that represent uniquely indexed columns. The information is used to set some properties of query items as well as to generate relationships. Create relationships and ensure cardinality rules are appropriate. This is an extremely important component of the modeling process. Framework Manager uses the cardinality rules to assist the query engine in generating the appropriate SQL statements. Different types of cardinalities are: For example, each student has one student number. For example, each teacher has many students. For example, many students have many teachers. Fact tables should be separated by a dimension. Instead Star Schema grouping should be used. The most common type of ambiguous relationships are where multiple valid relationships exist between two tables, reflexive relationships table joins to itself. This can be resolved by creation of alias tables; however, it is not recommended to build deep hierarchies to resolve reflexive relationships. This should be accomplished by flattening out the table. Minimized means that the generated SQL contains the minimum number of tables and joins required to obtain values for the selected query items. The results from the two or more queries are then joined on the Cognos application server which is in most cases, not very efficient. Stitch Queries occur when a model has multiple one to many relationships usually as a result of data coming from multiple fact tables which can only be joined through a conformed dimension. As an example, consider the model diagram shown below. The Sales fact and Inventory fact tables can only be joined through one of the three conformed dimensions Time, Product and Staff. This will mean any report which takes data from both the Sales Fact and Inventory Fact tables would result in Cognos generating stitch queries. This is the correct approach for this type of query but frequently occurs due to incorrectly set cardinality within the Framework model. It is essential to ensure cardinality is set correctly throughout your model to ensure unnecessary stitch queries which can cause report performance to be very poor 2. The first occurs when there are multiple valid relationships. This typically occurs between facts and dimensions. In a fact table, a different date is present: Combining multiple dates in a single query will no longer return results. Another issue occurs when handling recursive relationships. The classic example is the manager â€” employee relation. An employee has a manager. The manager is an employee and also has a manager that again is an employee. These situations can be handled by creating multiple model query subjects for every occurrence. You would however have to reset all the properties of every model query subject created leading to unnecessary work. A convenient solution to this problem is using shortcuts. There are two types: Regular shortcuts will be used while creating the Presentation View. This is a particularly useful feature when dealing with multi-fact, multi-grain queries. When you have a sales fact at day level and a target fact at month level, combining both facts in a single query would lead to incorrect results. The targets would be multiplied several times as they are stored at month level and not at day level. Determinants will change the default behavior of the query. Cognos will recognize the difference in grain and will write 2 queries that will be stitched together to return proper results at the proper grain. A determinant will specify what set of columns will uniquely define a set of columns. Each level is specified identifying the key and attributes that belong to a level. The lowest level is marked unique. Cognos uses the mechanism of stitch queries to perform these types of requests. A stitch query will perform a full outer join to break queries into multiple selects, one for each fact table and then stitch the data back together. Granting or denying access to a package a very effective and easy way to implement a basic level of data security. However, each individual object at any level can be secured. Data security will restrict users to query data they are not allowed to. For example, a district manager will only be able to query data of his district. Row level security can be put in place in two different ways. It is possible to hard code the values for every group. However a more generic approach is to create embedded filters using security macro functions such as the LDAP username 2. During importing, the Usage property is set according to the type of data that the query items represent in the data source. You need to verify that this property is set correctly. For example, if you import a numeric column that participates in a relationship, the Usage property is set to identifier. You can change the property. For relational query items, the value of the Usage property depends on the type of database object the query item is based on Usage Property.

Chapter 4 : What are the best practices while building a Cognos Framework Manager Package ?

Framework Manager is a metadata modelling tool in Cognos in which data is sourced from one or more data sources. By Adding Multi-lingual capabilities, one can create a model which is a business representation of data available to the users for analysis.

A namespace creates a qualifying container for objects, avoiding naming conflicts. Within a namespace, the modeler can use folders to group standalone filters or query subjects. Namespaces will structure frameworks. In a namespace a number of query subjects are added. They represent the tables in a framework. There are three different types of query subjects: Data Source query subjects: Standalone filters are pre-designed filters that can easily be re-used in the reporting tools by the author. A Measure Dimension contains a collection of numeric values. The Regular Dimension provides the accompanying set of descriptions and identifiers. The Measure and Regular Dimensions are linked with Scope Relationships to define the level at which the measures are available for reporting. Creating Durable Models While creating a model it is important to create a proper structure. The use of a multi-tier structure will shield the end-user from changes at database level such as migration to a different database technology, or changes to column or table names. By creating an efficient layered structure, relational models can be modeled into virtual star schemas, providing predictable and reliable query results to the end user. The first step in creating the Framework Model is importing the metadata. This can be handled by using the Metadata Wizard. It is good practice to create a separate namespace for every data source that is needed in the framework. On top of the namespace for the data source, a global namespace should be created: Query subjects are linked together using Relationships. When all data source objects are imported, the model should be scrutinized to verify all relations between the query subjects are correct. It is good practice not to blindly import the relationships. By manually creating the relationships, a much higher level of control is achieved. Relations should always and only be created in the Database Foundation View. Mixing relationships at different levels will only cause confusion and incorrect results. For maintenance purposes, it is however recommended never to make any changes in freehand SQL. If you do, the query subject has to be manually adjusted if changes are made at database level. When changes are made in the database, importing is by far the easiest way to update the query subjects. You can also use the Update command in the Tools menu to update a single query subject. Although it is possible to import data from different data sources, the reflection should be made that there is a performance penalty in doing this. When the data sources are on different servers or use different technologies, IBM Cognos will not be able to write SQL-statements that will contain objects from both data sources. Therefore it is highly recommended to use only 1 data source per physical database platform. In the Data Foundation view some other tasks need to be done. By using the proper tab pages, calculations and determinants can be added to the query subjects without making changes to the SQL code. The usage of a field can be an identifier, attribute or fact. Facts are numeric, usually additive or semi-additive data. All indexed columns or columns containing business keys should be set as identifier. Attributes are typically all other strings. For every fact column, the aggregate should be set. Other options that should be set are the format, screen tip, description These properties are inherited by derived objects at a later stage in the modeling process. Model for predictable results The greatest challenge for the model developer is creating a model that returns proper query results at all times, no matter what columns were selected in the report by the user. When importing from a relational data source, cardinality is detected based on a set of rules. IBM Cognos uses the following rules: This is typically the case with snowflake dimensions. This situation can be handled by using model query subjects. The model query subject will logically condense the snowflake into one object, thus enforcing the correct context in every query. However, there is a performance drawback. Condensing multiple tables in a single model query subject will force Cognos to retrieve the entire snowflake even when no fields are needed from the underlying tables. Therefore it is better not to condense the snowflake using a model query subject. Instead, model the snowflakes with 1: Tables in the snowflake can be joined using 1: This will allow the usage of Minimized SQL, retrieving only the objects that are needed and ensure the proper usage of the query subjects. The first occurs when there are multiple valid relationships. This typically occurs between facts and dimensions. In a fact table, a different dates are present: Combining multiple dates in a single query will no longer return results. Another issue occurs when handling recursive relationships. The classic example is the manager employee relation. An employee has a manager. The manager is an employee and also has a manager that again is an employee. These situations can be handled by creating multiple model query subjects for every occurrence. You would however have to reset all the properties of every model query subject created leading to unnecessary work. A convenient solution to this problem is using shortcuts. There are two types: Regular shortcuts will be used while creating the Presentation View. Multi fact multi grain queries A determinant is needed to identify levels of aggregation within the query subjects. This is a particularly useful feature when dealing with multi-fact, multi-grain queries. When you have a sales fact at day level and a target fact at month level, combining both facts in a single query would lead to incorrect results. The targets would be multiplied several times as they are stored at month level and not at day level. Determinants will change the default behaviour of the query. Cognos will recognise the difference in grain and will write 2 queries that will be stitched together to return proper results at the proper grain. When retrieving 2 measures from two different fact tables using a different granularity, Cognos can determine the correct aggregation when determinants are specified. A determinant will specify what set of columns will uniquely define a set of columns. Each level is specified identifying the key and attributes that belong to a level. The lowest level is marked unique. This will enable the report developer to create a report showing revenue at week level versus month figures without double counting the lowest grain fact. Cognos uses the mechanism of stitch queries to perform these types of requests. A stitch query will perform a full outer join to break queries into multiple selects, one for each fact table and then stitch the data back together. Determinants are specified at the Data Foundation Layer. Consolidate When all data related issues and reporting traps are handled, the next step in the modeling process is creating a Consolidation View. The consolidation view usually is split up into two namespaces: The first is used for normal reporting and generates SQL that is fired to the database. The second is used in multi-dimensional analysis and resembles an OLAP cube. The DMR-model is a virtual way of modeling the data source and does not physically stores data. The structure will, unlike the Data Foundation View, not resemble the database. The main goal of this layer is to provide an easy to understand structure that is recognisable to business users. It is perfectly okay to combine a snowflake into a single model query subject as this would be a logical point of view of the business users. Normally technical meaningless objects such as load dates or sequence numbers like primary keys and foreign keys to dimensions should be removed or hidden. Facts should only contain measures and degenerate dimensions. All foreign keys to dimensions should be hidden to the business user. While debugging reports, it can be quite handy to be able to include the primary key of a table to identify exactly which record has issues. In this layer, no relationships between query subjects should be laid, ever. A model query subject is also the best place to use macro functions and parameter maps to handle multilingual tables. By using the calculations in Model Query subjects, modelers can avoid entering freehand SQL in data source query subjects, which should be avoided at all times for maintenance purposes. The data is presented in an OLAP-style, but is not physically stored on the server. Instead at every user request, a query is executed. This caching mechanism will provide similar performance as a physical cube when primed correctly. A Regular Dimension consists of one or more defined hierarchies containing levels, keys, captions and attributes. Level information is used to roll up the measures. Each level should have the key and caption defined.

Chapter 5 : IBM Developer : cognos proven practices documentation

Best Practices in Cognos 8 Framework Manager Model Design: Part 2 - Advanced Modeling Issues | The Ironside Group says: August 11, at pm Last month's article described the overall approach for best practice model design using the 4 layer architecture.

From crosstabs to graphs, details or summaries, sales reporting to financial statutory reporting, Report Studio is both powerful and versatile. Due to its expansive feature set and extensibility, every report developer will have their own approach to development. When inevitably under deadlines, even the most experienced developer is pressured to skip important steps that may seem unnecessary at the time. Developing a consistent and disciplined methodology around the creation and modification of reports is a habit that can actually save time and effort. Consider the following suggestions when developing new reports or modifying others to make the most of your time and to avoid the frustration that comes with modifications and requirement changes. These can prove invaluable to recall changes when returning to a report after a period of stability. Commenting out a previously used formula, dating changes or even noting the business user requesting the change can be useful inclusions. Version your work Which copy of your work is current? Unless you have a robust version control system in place, this question causes headaches. Versioning your report copies with the date as shown below has a couple of distinct and concrete advantages. The most recently dated version is the current one. This naming and date convention also sorts reports appropriately within Cognos Connection. Secondly, previous known good versions exist to which you can revert in the event of irreconcilable errors. I typically have an archive folder to which I immediately move old versions in case I need to revert. Finally, and perhaps most importantly, habitually creating a well-labeled new version when presented with new requirements establishes an implicit release process. This view is essentially where the user always goes to for their most up to date version. The name or location of this never has to change, providing a consistent user experience. Organize and label reports and objects. If report objects are well labeled and kept orderly, they will be easier to locate and change. This is especially true as none of us are working on just a single report at a time! Returning to a well-organized report saves the mental switching costs of re-familiarization. Use mock query items. Create placeholder query items with expression value of 0 or some other neutral value that are used solely for organizing the query data. Visual landmarks are created, making the job of quickly identifying groups of query items easier. Use mock filter items. The above technique can also be applied to filters. Well labeled filters allow quick navigation and will make the job of modifying your work, either by you or others, much more straightforward. How many times have you disabled a filter only to forget whether it was originally required or optional?! Copy and paste the following within a query in Report Studio to get the mock filters shown in the image above. If the users need the reports to be accessible from alternate locations, views are employed. It undeniably adds time when moving a report through the release cycle; however the benefits of stable testing and insulating users from changes are invaluable. This multiple phased package approach is also useful as Framework Manager will also automatically document the last published date for each, helping keep package versions under control as well. Document report changes One of the first things I do when creating a new report is to create a documentation page within the report to record feature requests and requirements changes. Copy and paste the following text into the Page Explorer in Report Studio to include a change log template. Use a Boolean variable to control its display. I am able to view the content in the documentation page, while report consumers cannot. While these tips are not entirely comprehensive, they hopefully represent some guideposts that can keep your development cycles clean and humming without the added noise of rework and expensive mental switching costs. And maybe â€" just maybe â€" they can keep you sane in light of ever changing business requirements!

Chapter 6 : IBM Cognos Proven Practices: Durable Models Best Practice for OEMs using IBM Cognos BI

Metadata Modeling Best Practices with IBM Cognos Framework Manager is a three-day course. This course ensures participants become proficient with the Framework Manager modeling tool following a best practices approach. In this fully interactive course, participants practice performing the skills with the instructor's guidance.

Framework Manager Model Design. For those starting out with Cognos 8, or even those who have worked with it for quite a while, Framework Manager model design can seem like a daunting or even overwhelming task. This is the first in a two-part article that will focus on the best practices, organization and overall methodology of Framework model development. As is often the case with complex tools, even developers who have worked with Cognos for a while can be confused on at least some of the concepts and considerations of model design. The focus of this first article will start from the beginning, and describe an overall 4 layer approach, detailing the purpose and reason for each of the four layers. Four Layer Approach One of the basic tenants for best practice model design is to segment the model into four specific sections or layers Data, Logical, Presentation and Dimensional. Each layer has a specific function and set of modeling activities. Generally, the layers build upon one another, with the data layer being the foundation of the model. Other areas, such as packages and connections are also important, but fall outside of the layers. Many of the best practices can be thought of in terms of what development activities should or should not be performed in each of the layers. Data Layer The data layer, also called the import layer, contains the data source query subjects, based directly on the underlying database objects. If you modify the SQL code or add a filter or a calculation to the data subject, it eliminates Framework Managers meta-data caching capabilities. This means that Cognos 8 will validate the query subject at report run time, adding overhead to the report load time. In some circumstances, this may be worth the trade-off, but should be avoided when possible. Add joins, cardinality and determinants at this level. Cardinality, the definition of how joins should behave, is critical to a well developed model and can be confusing, even to seasoned veterans. In lieu of a lengthy discussion within this article, consider other IBM documents which offer a thorough discussion of cardinality, and the typical types of situations you may see. Expect 20 to 40 percent of the model development to take place in the data layer. Logical Layer The logical layer adds the bulk of the meta-data to the model, and consequently is where most of the work is likely to occur. It provides business context and understanding to the data objects. Tasks include but are not limited to: Organizing elements into logical query subjects. This may mean combining elements from multiple tables, as in a snowflake dimension. Renaming element names, including descriptions and tooltips, and adding multiple language definitions if needed. Assign standardized and business-defined terms to the database columns, giving them business context. Add calculations, and filters including stand-alone filters, embedded filters and security-based filters. Base these on the underlying data layer objects to make them less susceptible to errors when copying and reusing the query subject. Arrange query subject items in a user-friendly manner. Make use of folders to group similar items, or when there are too many items. I suggest 12 or fewer objects per folder, although I have used more when the logical breakout calls for it. For example, if there are many dates in a query subject, it might be more intuitive to have all the dates in a single large folder, than to try to subdivide each into smaller, random folders. Arrange the contents of the folder in an intuitive manner, such as alphabetic, or with the most commonly used items at top. Assign output formats and usages to the reporting elements. This is easier if you create a small folder of trivial calculations, used only to provide standardized object format templates. The formats can easily be copied to target items by multi-selecting, and dragging the topmost format through the list. Add prompts, including cascading prompts, and prompt-based calculations. Roughly 50 to 70 percent of the modeling work occurs in this section. Presentation Layer The importance of making information easy for report writers to use is frequently underestimated, but is a critical component to driving user adoption. Fortunately, this can be a simple step. The presentation layer is used only as an organization structure in order to make it easier for report writers to more easily find their needed information. This layer includes only shortcuts to existing items in the logical layer, plus organization items such as folders and namespaces. For example, create a namespace called Orders and include shortcuts to the ten or twenty relevant query subjects, out of perhaps a hundred or more query subjects in the logical layer. Also include shortcuts to relevant filters. Commonly used query subjects such as Items or Customers will appear in multiple areas. Rename the shortcuts to something which provides helpful business context. For organizing major groupings, you can use either folders or namespaces. However be aware that namespace names must be unique, and that items within a namespace must likewise be unique. For this reason in particular, I generally prefer to use separate namespaces. Dimensional Layer The dimensional layer is required only for models which include dimensionally modeled data. Leaving aside the trivial situations where cubes are simply imported into the model, this includes dimensionally modeled relational data DMR. Specifically, this is for creating Dimensional and Measure Dimension Query subjects. Much like the presentation layer, this layer also is built upon the logical layer, which leverages the effort put into that layer. Apply the element renaming, descriptions, tooltips in the logical layer, and they can be reused in the dimensional layer, with some help from the search tool. Data Sources One area which many modelers neglect is the data sources. A key part of the data source is the Content Manager Data Source, which references a data source configured via the web interface in the Administration area of Cognos Connection. You should have as few of these as possible, ideally only 1 per database, unless there are very specific reasons such as security. Even so, many security or logon issues can be addressed via multiple connections or signons within a data source. Framework Manager will automatically create multiple data sources, one per schema, for example , but they should reference the same Content Manager Data Source. The reason minimizing these is because multiple Content Manager data sources will open separate connections to the database and force Cognos to perform simple joins within the reporting engine instead of in the database where they belong. Relationships and Cardinality The area which causes much confusion and difficulties with modeling is relationships and cardinality. There are a number of documents which discuss the handling of data modeling traps, such as the Guidelines for Modeling Metadata document included in the documentation, and so I will not duplicate those discussions here, instead addressing other topics. A common concern is when to alias a table in the data layer. In general, common dimension tables used across multiple fact tables, or contexts, should not be aliased. This is important in creating conformed dimensional reports. For example, a parts dimension is the same when used for reporting against sales or inventory fact tables. When a table has separate usages, or meanings, it should be aliased. For example, a general purpose address table, which contains branch, customer and employee addresses. These are often identified by having multiple join paths or filters which can signal different usage. Another situation to use aliases is when there are multiple usages of a dimension table. The most common occurrence of this is the general-purpose date dimension table. Applications such as health and insurance can have many different joins to a date dimension, each of which has a different context. This can be complicated to track, because one usage, such as Admission Date must apply in a conformed manner to many fact tables, while a different usage Treatment Date must also be conformed across many of the same dimension tables. Obviously a meaningful naming convention is required! Controlling Stitch Queries One frequent point of confusion, is how the model causes reports to create stitch or multiple queries. To resolve this requires a firm understanding of the cardinality in relationships. Also realize that stitch queries are sometimes best way of issuing a query. This is a smooth progression from dimensions to facts. It can easily generate facts with correct aggregations with this simple scenario. Remember that Cognos will treat the query subject at the end of one or series of 1: N relationships as fact tables. An easy visual check is to make sure that the 1: In a query where i nventory fact and sales fact are compared at a product dimension level, a stitch query is the correct approach. However, if this is an unintentional or poorly modeled relationship, incorrect or inefficient queries will be generated. Understanding the implications of cardinality in relationships is the best defense against this. Naming Conventions and Nomenclature Another area of confusion is naming conventions and nomenclature, and areas surrounding these subjects. Each element name is uniquely identified within the package by a 3 part name: The element itself, the parent query subject or shortcut name, and the parent namespace to the query subject. Element names must therefore be unique within the query subject, and similarly for query subjects within the namespace. However, the namespace name must be unique within the entire model. Notice that folder names do not come into play when defining names. They are used only for organizing other items.

Managing Name Changes The result of all this, is that names within the model are very important, especially once report writing begins, because they are difficult to change without invalidating existing reports. So, in order of preference, maintain consistent names across models by: Get it right the first time. If multiple areas of the organization have a stake in naming conventions, involve all organizations early in the design process, even if the first stages do not apply to some of them. Identify situations where a renamed item is used. Keep in mind, though, that if your model references a development server, it will not identify reports or queries which exist only in your production environment. Therefore it may not adequately identify issues in every environment.

Chapter 7 : Cognos Framework Manager

Cognos Framework Manager Best Practices. Building metadata models using Framework Manager can be a simple or complex one. It all depends on the underlying database structure and how relevant it is to the reporting that is required.

Never allow Cognos to create joins automatically when importing data sources. Cognos rarely chooses correctly and this inevitably creates future rework. Creating joins manually gives full control and avoids cardinality and join issues. Always separate your model into the following three namespaces. Separating into three layers may seem difficult initially, but this organization makes the model easier to maintain. If the underlying data structures change, you will only need to reflect these changes in a single location. Database Layer Namespace â€” This namespace simply contains all the query subjects brought in from your data source. No joins or renaming of query items occur in this layer. By leaving this layer untouched, future model changes will only need to be changed in this layer. All other layers can remain untouched. Logical Layer Namespace â€” Relationships between query subjects should be created in this layer. If query subjects need to be merged, the merge should be created here. Any model filters should be applied and query subjects and query items should be translated to business names in this layer. All query subjects in the Logical layer should reference back to the database layer. Presentation Layer Namespace â€” This layer contains shortcuts to the logical namespace and organizes the data for the business user. Assign query usage in the database layer. Understanding the query usage property and setting it in this layer will ensure that it is always set correctly throughout the rest of the model. Each query subject that is created from the base layer will inherit these properties. Never expose query subjects to the user that could result in cross product query. As an additional precaution, use the governor setting to deny cross product queries. Use the governor settings to limit the query execution time. This will prevent users from developing run-away queries that can impact database performance. Maintain a history of your models using a source control system. CVS is an open source versioning system so cost should not be an excuse!. When modeling your data, avoid creating situations where multiple join paths or loop joins can occur. To prevent these situations, try aliasing the query subject multiple times to force a single path through the data. In a perfect world, this is done for you in the data model and the underlying ETL processes, but in reality we rarely get this lucky. It is possible to create a virtual star schema by merging query subjects to de-normalize data and creating new query subjects that contain only measures. The drawback of this method is that there can be significant performance impacts. Understanding the queries that are being created against the data is critical to making a good design decision here. Set security at the highest level allowable, starting with the package, then the object and if needed the data. Applying security at a high level and then drilling down to more detailed security levels will save maintenance and troubleshooting time. Keep security as simple as possible while still meeting the security requirements and you will thank yourself at the end of the day. Use data level security settings in Framework Manager only when absolutely necessary. Data security is a powerful feature but maintenance and troubleshooting with row level security become complex quickly. If you must use it, make sure to have it well documented for future developers and administrators. If you do choose to use data level security, be aware that it supercedes all other security settings in Cognos Connection. Check cardinality for accuracy and test cardinality to make sure it is behaving as expected. Avoid many to many relationships in your model. If you cannot avoid them, organize data so that the user is less likely to create a report with a many to many relationship. Name query subjects and query items using business terms. As simple as this seems, many organizations leave the original table and column names, creating confusion for the end user. Only publish data that is in the presentation layer, and only expose query subjects that are useful for the end user. This is another simple task that is often overlooked,or more likely skipped to save time. Even if the model developer will be maintaining the model, six months from deployment the developer will not remember all the details of the model. Proper documentation expedites making future changes and fixing bugs Document each layer, the joins associated, and any calculations or merges that were done.

Chapter 8 : Cognos Expert: Cognos Framework Manager Best Practices

Best Practices in Cognos 8 Framework Manager Model Design: Part 2 - Advanced Modeling Issues August 1, / 7 Comments / by Ralph Baker Last month's article described the overall approach for best practice model design using the 4 layer architecture.

Chapter 9 : IBM Cognos Framework Manager â€“ Proven Practice - Perficient Blogs

There are some practices that make the process of creating Metadata Model simple when you use Cognos Framework Manager. 1) When you create a mode, you must make a document that contains all the important details of the Model, so that you can refer the document when asked about the model after few months of its deployment.