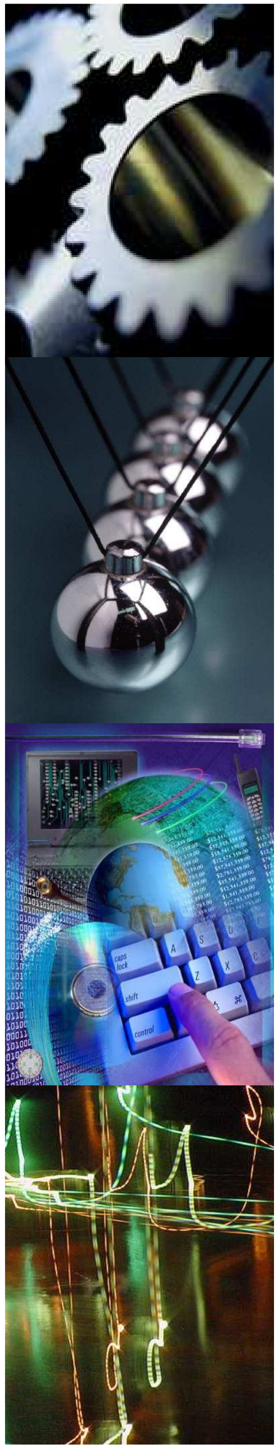


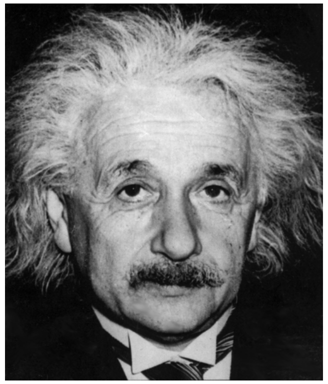
What you need, when you need it

BI4ALL
View Abstraction Benefits

Public Information
Version 3.1: 1/1/2012

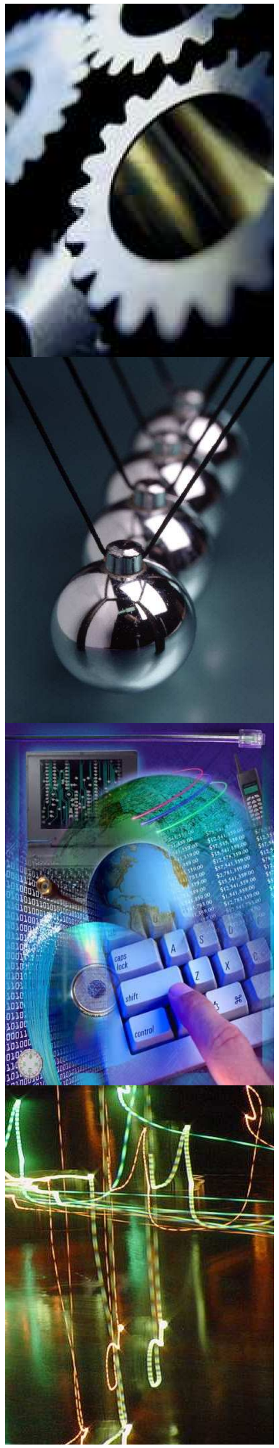
Why Instant Business Intelligence?

“The significant challenges we face today cannot be resolved by the same level of thinking that created them.”



Albert Einstein

instant BUSINESS INTELLIGENCE



Introduction

Public Information
Version 3.1: 1/1/2012

Introduction BI4ALL View Abstraction Layer Benefits

BI4ALL is an innovative model that provides breakthrough benefits in a number of areas such as:

- *2-3x productivity benefit in the customisation and development process*
- *Removing the need for drawing ER diagrams to understand the model*
- *Complete abstraction of the model from the underlying database technology used to implement the database.*

As was the case when we were first introducing large companies to the concepts of dimensional data modeling in the early 90s, we find that the innovation that BI4ALL represents is not well understood by various members of the IT community. This is normal.

In the early 90s many members of the IT community declared 'dimensional models are no way to build a query database'. It was 'new' and 'innovative' and therefore not well understood and dimensional modeling was rejected 'out of hand' in many cases.

Fifteen years later, the same members of the IT community declare 'Dimensional modeling the ONLY way to deliver good end user query systems.'

Today, we find that we are facing the same concerns with the fundamental concepts of the BI4ALL models. The two key features that deliver the benefits above are:

- The use of meaningless names in the underlying tables.
- The use of views rather than a case tool in the development process.

And these two approaches are being commonly challenged during the education process for prospective clients.

The purpose of this document is to articulate the benefits of these two fundamental mechanisms of model design and why we have decided to build the BI4ALL models using these techniques rather than reuse the techniques we have been using for many years.

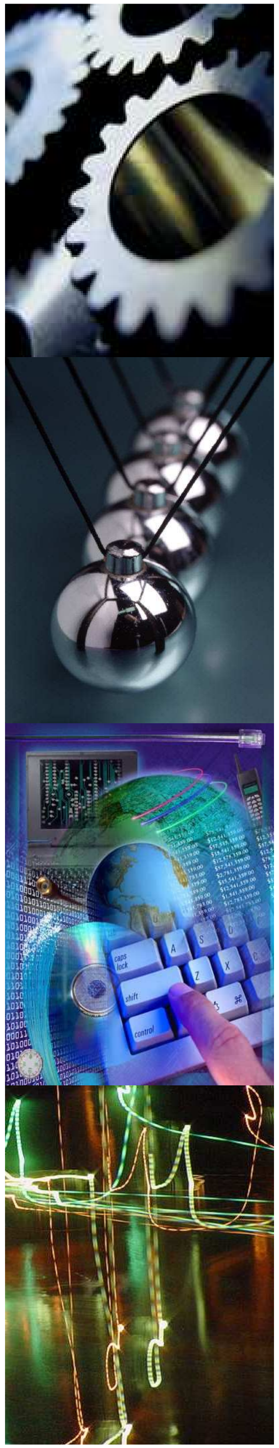
As you consider BI4ALL as a model to adopt for your company we ask that you keep in mind the design points that were employed in the development and design of the models:

- During the customisation and implementation process it must provide at least a 3x productivity benefit over the previous 'best practice' methods of implementing such models.
- It must be possible to design and develop models with at least a 5x improvement of productivity.
- It must be possible to provide the model to the market at a price at least 50% lower than any other comparable model.
- It must be completely independent of the underlying database technology.

Having defined these design points, we spent a great deal of time and effort reviewing all our experience and proposing many ideas as to how we might deliver on these design points.

As Einstein said *"The significant challenges we face today cannot be resolved by the same level of thinking that created them."*

As a result of challenging the 'conventional wisdom' of always using case tools and building physical tables we have been able to deliver on the design points. This paper explains why and how. We trust that it explains clearly and concisely the benefits of using views as a completely abstracted model.



History of Relational Modeling Tools

History of Relational Modeling Tools

Charlie Bachman. This is a name familiar to anyone who is familiar with the history of relational database modeling. That familiar red bow tie icon, the symbol of Charlie Bachman, used in so many parts of what was the worlds leading data modeling tool in the late 80s, the Bachman suite.

Why did we need the Bachman suite of tools to design and implement relational databases in the late 80s? A number of reasons come to mind:

- Relational Database modeling was a rare skill in the late 80s.
- Relational databases (namely DB2) were very expensive and very slow compared to using the other IBM database IMS.
- Relational Database tuning skills for large systems were almost non-existent. Making 'the database go faster' was not well understood.
- The number of IOs required to perform the same function on DB2 was between 10 and 100 times more than for IMS. This huge increase in IOs meant design of the tables was extremely important yet skills in this are were very rare.
- Diagrams representing relational models were needed by people new to ER modeling to understand the relationships between tables. In IMS the relationships between segments was far better understood because of the hierarchical nature of the database. You knew parent child relationships based on the physical implementation of the model. These are not immediately and easily apparent in a relational model.
- The data types that were available to people were new and there needed to be increased verification of data types.

All in all, as the emergence of the relational databases happened in the late 80s there was a great need for tools that would assist the data modeler design and build the database because of a chronic lack of knowledge about what the database was actually going to do. And Charlie Backman and his team embedded that knowledge into the Bachman suite.

Interestingly enough, one of the 'absolute rules' of the early generation of relational database designers seems to have been completely forgotten now. That was "never allow access to the base table, always use a view to access data."

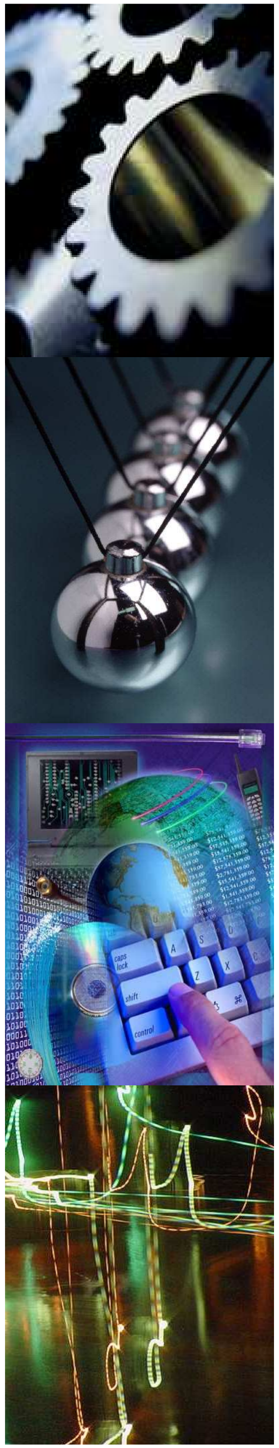
In the succeeding years relational databases became more and more complicated. ERP systems with tens of thousands of tables emerged and these systems defeated any single individual ability to know how they 'hung together'. Case tools which could provide some platform for understanding how the data in these systems is related to each other. And it is perceived that Case tools and repositories meet this need.

Even worse. As the need for integrated data for decision making emerged in the 90s there arose a need to integrate data from many disparate internal systems built over many years, as well as data from suppliers, customers and other third parties. The rise and rise of Data Warehousing to meet this need has meant that the number of systems, files, tables and fields coming into the EDW far outstrips the capacity of the single individuals brain to be able to comprehend the meaning of all the data let alone the ability to design a data model to house said data. And it has been perceived that the case tool has been a tool that can assist the EDW Architect capture and store this understanding over time.

And so, we are where we are today. Where a variety of vendors push their case tools to say "this is the 'state of the art' in capturing and deploying data models for Data Warehouses today."

We have worked with the best of these ER tools over their entire life. From Bachman, to Excelerate, to Erwin, to the best available today, PowerDesigner.

Our experience is that using such a tool is great if one is relatively inexperienced in building EDWs. But if one is a very experienced and knowledgeable person, these tools slow one down during the development process.



View Abstractions in BI4ALL

Public Information
Version 3.1: 1/1/2012

View Abstractions in BI4ALL

View Abstractions in BI4ALL

In BI4ALL (plus all verticals) there are only 20 physical tables. These tables have meaningless names such as TD0001 and TF0001. The fields in the tables also have meaningless names that merely denote the data type of the fields.

We have reproduced a small portion of TD0001 on the opposite side of this page.

The way that these tables are used is that views are placed over the top of the 'meaningless' columns and a 'table number' is assigned in the where clause of the view.

This allows the views to be placed into a single table and to be differentiated by the table_number.

Today in the combined models for BI4ALL (not including the work views for the segmentation engine) there are in excess of
840 tables and 28,000 fields.

When the designers of BI4ALL were considering how to design and build the BI4ALL models consideration was given to building the models using Erwin or Powerdesigner. These are the leading two case tools for developing and maintaining data models.

The designers of the BI4ALL models have a vast amount of experience in using these two design tools and they also have a vast amount of experience in implementing similar models using case tools.

It is also worth mentioning that when initial consideration was being given to designing and building the BI4ALL models it was forecast that the models would eventually exceed 1,000 tables and 20,000 fields.

From the combined experience of the designers it was agreed that to build such a sophisticated suite of models and maintain them in either Erwin or PowerDesigner would consumer large amounts of development resources.

It was felt that we could not achieve the design point of "2-3x productivity benefit in the customisation and development process" by using a case tool, no matter how good that design tool was.

TD0001 - General Utility Dimension Table
This dimension table is to store the more general dimension tables which have sensible numbers of fields and have a mix of character, varchar, money, integers etc.

```
create table dbo.TD0001
  ( pk_TD0001  integer  not null  default 0
  , level_col     varchar  (0010) not null  default 'unknown'
  , dim_char_ky_fld    varchar   (0255)  not null  default 'unknown'
  , date_from                   datetime
  , date_to                     datetime
  , current_flag                integer
  , table_number                integer       not null  default 0
  , row_del_frm_src_ind         char  (0001)
  , seetl_batch_number          integer not null  default 0
  , seetl_file_cycle_number     integer  not null  default 0
  , seetl_view_name     varchar    (0255)  not null  default 'unknown'
  , audit_timestamp_01          datetime
  , char1_01                    char (0001)
  , integer_01                  integer
  , money_01                    money
  , decimal_01                  decimal (38,10)
  , float_01                    float
  , date_01                     datetime
  , timestamp_01                datetime
  , varchar255_01               varchar  (0255)
  , dim_key_ag1     integer     not null default 0
)
  on [PRIMARY]
;
```

It was also felt that we could not achieve the design point of "Complete abstraction of the model from the underlying database technology used to implement the database" by using a case tool because the case tools maintain data about the physical implementation of the tables in order to create the tables.

It was therefore decided that we would not use a case tool to implement the BI4ALL models.

instant business intelligence

A

White Paper

by

Instant

Business

Intelligence

www.instantbi.com

View Abstractions in BI4ALL

Having made the decision that we would not use a case tool to implement the BI4ALL models that left us with a decision to make as to what technology we should use.

This generated a lot of discussion and ideas. Such ideas where put forward as:

- Placing the models into spreadsheets and generate them from there.
- Placing the definitions of the models into a database and generate them from there. (And indeed earlier versions of models were generated from tables and these utilities are still available in SeETL.)

All the ideas that were proposed have both positive and negative aspects to them.

One of the biggest problems with the idea of 'generation' of the models that was clear to the designers was that in the early years of model development there would be a very high degree of change that would occur to the models. Therefore any generation process would significantly increase the development time and effort required to produce the first few versions of the models. Experience showed this would be so.

In the end we recalled an idea from the early 90s. This idea was to create meaningful tables and place over those tables meaningless views which were then used by early versions of SeETL to perform the ETL processing. This idea of 'meaningless names' for SeETL to use during ETL processing was very useful at the time. What was done was the 'cobol generator' was told how many fields of what type were on a particular table and code was generated to move those fields. By doing this we were able to dramatically reduce ETL development time in the early 90s.

The 'breakthrough' was to reverse this idea. Rather than have meaningful tables and columns with meaningless views it was proposed to have meaningless tables and columns with meaningful views and view column names.

There was a secondary consideration which was also taken into account when this discussion was taking place. Over the last 5 years we had great success with the concept of 'presentation views'. In these cases we had built the ETL subsystems to the physical tables and then presented different views of the underlying physical tables through the presentation views. We had so much success with this idea that we had standardised on only ever presenting data to the user out of the presentation views.

Armed with these two ideas we attempted some experiments on exactly how this might work for developing a large and sophisticated model.

What we found was that the development effort and the customisation effort for such models had a productivity profile of more than 5x compared to developing the same models in PowerDesigner during the development stage.

In short, the idea of completely abstracting the model from the physical tables by using views made it commercially viable for a small consulting to develop a suite of data models and bring them to market at a price point that would be far below any of the existing models from existing vendors.

The main reason for the significant difference in the list pricing of BI4ALL models and models from other vendors is the breakthrough approach of using the view abstractions.

If this breakthrough was not found it would not have been possible to fund the development of a similarly sophisticated suite of models by a small software company such as Instant Business Intelligence.

Productionisation of the BI4ALL Models

Even though the models are developed using the view abstraction layer this does not mean that the database must be implemented with so few as 20 tables in production. Nor does it mean that no case tool can be employed when the models are productionised.

We are 'agnostic' as to how exactly the physical database is constructed when productionising the BI4ALL models.

Recall the design point:

"Complete abstraction of the model from the underlying database technology used to implement the database."

There are a number of "EDW Specific" databases that are gaining good acceptance in the marketplace and we did not want to 'rule out' the use of BI4ALL on any of these new and emerging technologies.

Good examples of EDW specific databases that are seeing good acceptance in the marketplace are:

- Sybase IQ
- Sand
- Netezza

Similarly, we did not want to rule out the use of the BI4ALL models on the standard databases like SQL Server, DB2 and Oracle. (Though we have used names longer than 30 characters so we do need to maintain a separate model for Oracle.)

We have even placed the BI4ALL model onto 'free' databases such as MySQL to provide the choice for clients to use such databases.

One of the issues with all these databases is that the vendors are making every effort to 'lock in' customers once the database is selected and therefore the physical implementations for each of these databases are all different enough that if a case tool was used to deliver the BI4ALL models there would be a significant customisation effort to update the delivered models in the case tool for the particular database the client had selected.

When productionising the BI4ALL models the only 'rule' is that the same suite of views are presented to the ETL subsystem and the applications. Therefore it is possible to do any of the following:

1. Create tables which are a 1-1 representation of the views and then place the views over those tables.
2. Create tables where there is some level of co-habitation of multiple views in the same table depending on the DBAs assessment and testing of performance on the tables.
3. Retain a significant level of co-habitation of views in the same table if the database engine will support such co-habitation and it does not have a significant impact on performance.

These choices reside with the DBA team who will productionise the models once they are developed.

With some databases such as Sybase IQ, Sand and Netezza the co-habitation of multiple 'logical' tables in the one physical table will have an extremely small effect on the performance of the queries that run on the database. So small that it would not be worth the effort of undoing the co-habitation.

If the DBA team decide to create tables which are very close to a 1-1 representation of the views at productionisation time it should be clear that the underlying table names could be customised to be 'real' names and the underlying columns could be customised to be 'real' names. There is no reason why this cannot be done.

Further, should a client have a standard case tool into which data models must be published it is clear that once the underlying tables have been built and the views repointed to those tables there is the opportunity to import the physical database design and the views into the case tool.

It is important to understand:

There is no 'rule' or 'assumption' that says BI4ALL must remain and a model in which views have a high level of 'co-habitation' and that the fields must remain 'meaningless' at productionisation time.

EDW Models and the Sheer Number of Objects to Manage

People who have not dealt with sophisticated EDW models like BI4ALL are usually not aware of the explosion in the number of objects to manage when implementing these models onto databases such as SQL Server, Oracle and DB2.

We will use Oracle as an example because it is provides the ability to create more objects and more parallelisation than SQL Server because of it's more sophisticated partitioning mechanisms. It is expected that SQL Server will implement more sophisticated partitioning in the future, especially if Microsoft want to compete with Oracle. The partitioning features we mention are already available in MySQL.

When creating a large fact table in Oracle it is recommended that bit mapped indexes are placed on each integer key that will be used to join the fact table to a dimension table and that the fact table be partitioned by some element of time.

Taking the 'extreme' case where 10 keys a commonly used to join to fact tables and the partitioning will be monthly and 5 years of data will be retained we get the following number of local bit mapped indexes. 10 x 12 x 5 = 600.

That is, there would be 600 local bit mapped indexes to define for the fact table. If using a case tool, each of these indexes would need to be entered via the case tool GUI. If we had 10 such tables, which might be common in a large telco, we would suddenly have 6,000 bit mapped indexes to create by entering their details into the case tool GUI. This number of indexes soon defeats the DBA teams ability to type them into the case tool!

Similar explosions in objects to manage occur for the partitions of the tables. However, the explosion in the number of indexes to create is the one that causes the most grief when using sophisticated models like BI4ALL on databases like Oracle and SQL Server.

Our experience is that the only sensible and productive way of creating this many partitions for tables and indexes is to generate them through scripts of create a template and perform 'change all' commands to the template to get the finished DDL to create the partitions and indexes.

What we have found from experience is that if a template is created and a mechanism is used to number objects according to the table they are referring to it is very simple to generate all the partitions and indexes required even though the numbers of objects becomes very significant.

Once the objects have been created it is then possible to reverse engineer them into the case tool to maintain them in the case tool should the client choose to.

This becomes a much faster and easier exercise than trying to type all this information into the case tool in the first place and generate the partitions, tables, and indexes from the case tool in the first place.

This generation of objects using scripts or creation of templates and performing change all commands is a process that can be reduced to a few days if the physical tables that will be implemented retain their 'meaningless column names'.

Should the client wish to include translating the meaningless column names to meaningful column names so that plans produced will note the column names of the tables extra effort is required to do so.

In the future BI4ALL models will be placed into spreadsheets and they will be generated. However, this additional feature is yet to be scheduled into the development process as it is seen as a 'nice to have'. Once this is done, the ability to change the underlying field name to the view field name will consist of no more than cutting/pasting cells in the spreadsheet.

At no point in the future will we pursue the idea of writing tools that will generate the create statements for the physical database. There are plenty of tools available to generate the creations statements for physical objects, and each of the databases is evolving new options at such a rate that we would not be able to keep up such development effort. We will stick to our design point:

"Complete abstraction of the model from the underlying database technology used to implement the database."

instant business intelligence

A

White Paper

by

Instant

Business

Intelligence

www.instantbi.com

Removing the Need to Print Diagrams of the Model

One of the big problems that has occurred in the last 10 years for EDW models is that they have become more complicated than a single human brain can retain with relative ease, no matter how good that brain.

With previous versions of such models it was not uncommon for pictures and documentation of the model to exceed 1000 pages of paper, and still we could not really understand all the connections.

Indeed, some model vendors proudly boast how many pages of documentation they have for their models. We see this as an admission that the models are so complex and so difficult to understand that all that documentation is needed by the development staff.

We have taken a different approach. Another ‘breakthrough’.

Us humans do not have ‘memory expansion slots’. We are born with the brain we have and we must make the best use of it. And our brains evolved well before the advent of sophisticated dimensional data warehouse models!!

In the face of ‘no expansion slot’ we are faced with the problem.

“How can we enable ourselves to remember all the details of a sophisticated model like BI4ALL?”

And the breakthrough is “Don’t”.

We have introduced the concept of “everything is linked to everything” which could be more accurately stated as “if two tables should be linked, they will be linked.”

By doing so we have removed the need to ‘remember’ what tables are linked to each other. With BI4ALL it is now an ‘act of faith’ to assume that “if two tables should be linked they will be linked”.

Once one assumes that “if two tables should be linked they will be linked” then one no longer needs to remember the linkages.

The way this works in practice is that to learn the models and to remember how they fit together one simply needs to read the fact tables and recall the data that is stored in the fact tables.

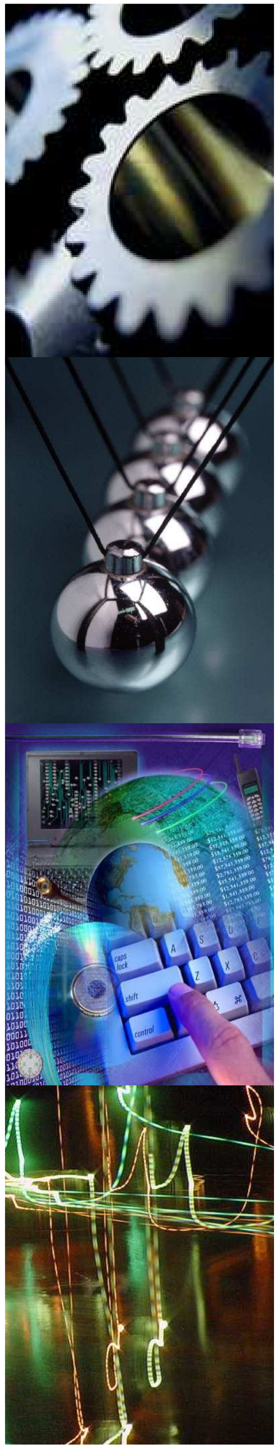
This approach has proven to be remarkably useful. It has made the development and customisation of the models far easier than previous generations.

The techniques used are independent of whether the model is stored in a case tool or just in the provided text document.

The important point to remember is that one does not need the model in a case tool and does not need to produce diagrams of the model in order to learn the model.

The days of learning models by reading diagrams of the model are over. And not a day to soon in this authors opinion!

instant BUSINESS INTELLIGENCE



Thank You For Your Time!