Kiva Productions

SAP and Data Warehousing

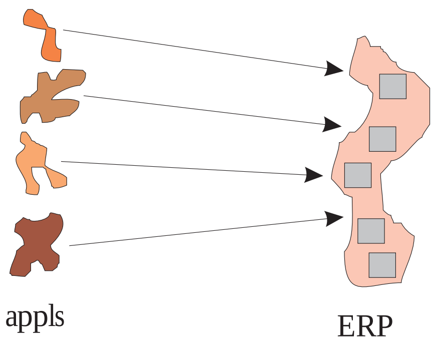
by W. H. Inmon

In the beginning were applications. Then these applications were maintained. And the maintained applications were merged with another company and had to interface with their maintained applications that were never before imagined or designed for working with other applications. And these applications aged and were maintained some more. Then application packages appeared and were added to the collection of applications. Soon there was a complex mess of epic proportions.

More maintenance, more requirements, more time passing, more mergers, more small applications and trying to get information out of the stockpile of applications was an impossibility.

Into this arena came ERP applications such as SAP, BAAN, J D Edwards, and a host of other players. The ERP applications offered to take the Gordian approach and smite the applications stockpile a mighty blow by creating new applications sensitive to current requirements which were also integrated. The appeal to the business person was enormous and soon ERP applications were everywhere. Indeed, as time passed, ERP applications began to make a dent in the older applications stockpile.

Figure 1 shows the appeal of unifying older applications into an ERP framework.



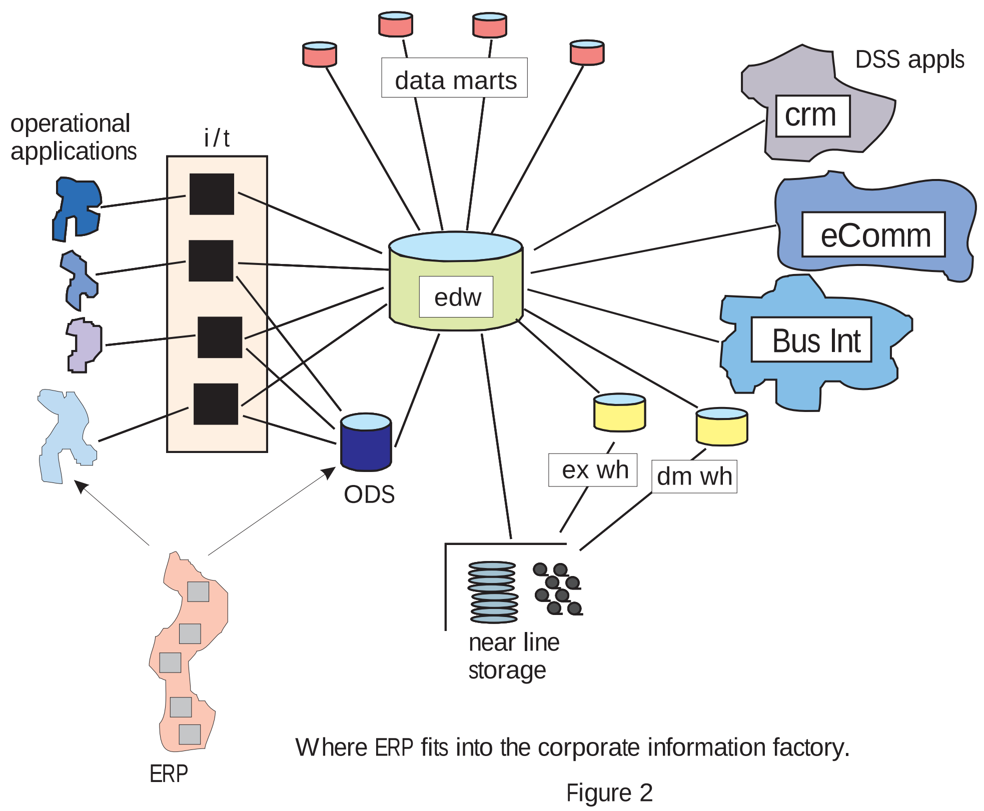
Individual transaction applications are consolidated into ERP.

Figure 1

The appeal was such that many corporations around the world began to buy into the ERP application solution, even when it was known that the ERP solution was not cheap or fast. The odor of the older legacy applications stockpile was such that, coupled with the threat of the year 2000, many organizations could not resist the appeal of ERP, whatever the cost.

The Corporate Information Factory

At the same time that applications were evolving into ERP, the larger body of information systems was evolving into a framework known as the corporate information factory. The corporate information factory accommodates many different kinds of processing. Like other forms of information processing, the ERP solution fits very conveniently into the corporate information factory. Figure 2 shows the relationship between the corporate information factory and ERP.



Where ERP fits into the corporate information factory.

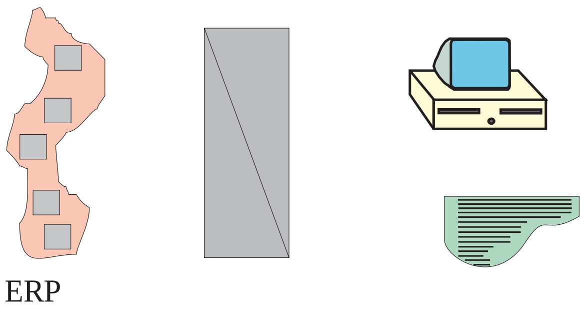
Figure 2

ERP fits into the corporate information factory as either another application and/or as an ODS. In the corporate information factory, ERP executes transactions which then generate data to feed the ODS and/or the data warehouse. The detailed data comes from the ERP application and is integrated with data coming from other applications. The integrated data then finds its way to and through the different part of the corporate information factory. (For an in depth explanation and description of the various components of the corporate information factory, please refer to THE CORPORATE INFORMATION FACTORY, W H Inmon, Claudia Imhoff, John Wiley, 1998.)

The advent of ERP was spawned by the inadequacies and the lack of integration of the early applications. But after implementing part or all of ERP, organizations discovered something about ERP. Organizations discovered that getting information out of ERP was difficult. Simply implementing ERP was not enough.

Frustration With ERP

Figure 3 shows the frustration of organizations with ERP after it was implemented.



Getting information out of ERP is difficult.

Figure 3

Many organizations had spent huge amounts of money implementing ERP with the expectation that ERP was going to solve the information systems problems of the organization. Indeed ERP solved SOME of the problems of information systems, but ERP hardly solved ALL of the problems of information systems.

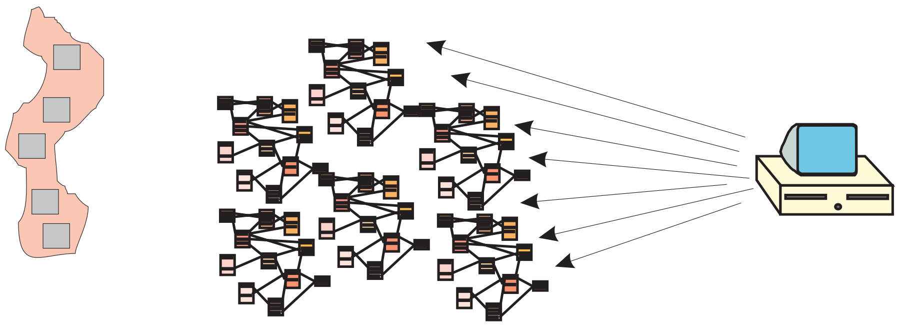
Organization after organization found that ERP was good for gathering data, executing transactions, and storing data. But ERP had no idea how the data was to be used once it was gathered.

Of all of the ERP vendors, SAP was undoubtedly the leader.

Why was it that ERP/SAP did not allow organizations to do easy and smooth analysis on the data contained inside its boundaries? There are many answers to that question, all of which combine together to create a very unstable and uncomfortable information processing environment surrounding ERP/ SAP.

The first reason why information is hard to get out of SAP is that data is stored in normalized tables inside of SAP. There are not a few tables. There are a lot of tables. In some case there are 9,000 or more tables that contain various pieces of data in the SAP environment. In future releases of SAP we are told that there will be even more normalized tables.

The problem with 9,000 (or more!) tables storing data in small physically separate units is that in order to make the many units of scattered data meaningful, the small units of data need to be regrouped together. And the work the system must do to regroup the data together is tremendous. Fig 4 shows that in order to get information out of an SAP implementation, that many “joins” of small units of data need to be done.



The performance implications of doing joins on 9,000 or more tables is tremendous.

Figure 4

The system resources alone required to manage and execute the join of 9,000 tables is mind boggling. But there are other problems with the contemplation of joining 9,000 tables. Some of the considerations are:

- are the right tables being joined?
- do the tables that are being joined specify the proper fields on which to join the data?,
- should an intermediate join result be saved for future reference?
- what if a join is to be done and all the data that is needed to complete the join is not present?
- what about data that is entered incorrectly that participates in a join?
- how can the data be reconstructed so that it will make sense to the user?

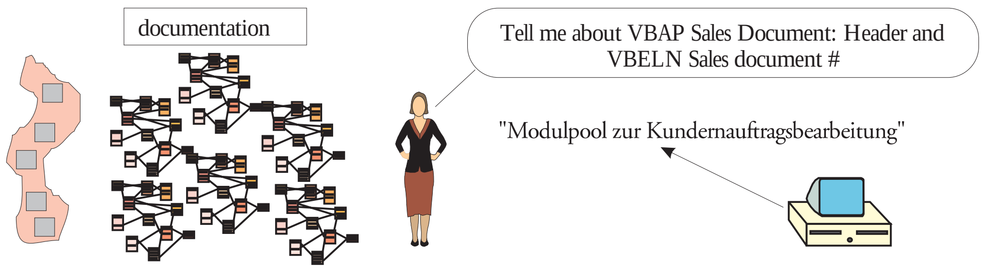
In short, there are many considerations to the task of joining 9,000 tables. While performance is a big consideration, the integrity of the data and the mere management of so many tables is its own large task.

But performance and integrity are not the only considerations. Life and the access and usage of information found in SAP's 9,000+ tables is made more difficult when there is either:

- no documentation, or
- significant portions of the documentation that exists is in a foreign language.

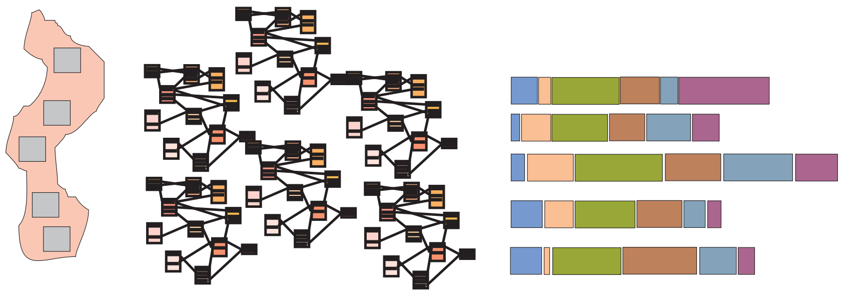
While it is true that some documentation of SAP exists in English, major important aspects of SAP do not exist in English. For example, the table and column names of SAP exist in what best can be described as "cryptic German". The table and column names are mnemonics and abbreviations (which makes life difficult). And there are thousands of table and column names (which makes life very difficult). But the mnemonics and abbreviations of the thousands of table and column names are of German origin (which makes life impossible, unless you are a German application programmer). Trying to work with, read and understand cryptic German table and column names in SAP is very difficult to do.

Figure 5 shows that when the documentation of an ERP is not in the native language of the users of the system then the system becomes even more difficult to use.


documentation
Tell me about VBAP Sales Document: Header and VBELN Sales document #
"Modulpool zur Kundernauftragsbearbeitung"

Important parts of the documentation are not in English.
Figure 5

But there are other reasons why SAP data stored internally is difficult to use. Another reason for the difficulty of using SAP lies in the proprietary internal storage format of the system that SAP is stored in, as seen in Figure 6.

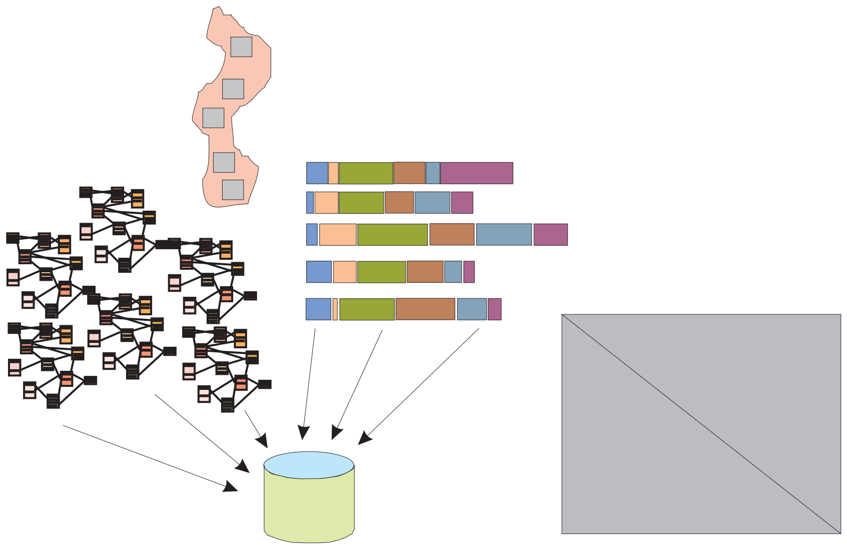


The internal format is proprietary.

Figure 6

In particular the data found in pool and cluster tables is stored in a proprietary format. Other data is stored in packed variable format. And furthermore, different proprietary formats are used. There is one proprietary format here, another proprietary there, and yet another everywhere. Coupled with the multiple proprietary formats are the proprietary structures used to store hierarchies (such as the cost center hierarchy, which are critical to multi dimensional analysis).

The interrogator or the analyst needs some way to translate the proprietary formatted data and proprietary structured hierarchies into a readable and intelligible format before the data can be deciphered. The key to unlocking the data lies in the application, and SAP has the control of the application code. Unfortunately SAP has gone out of its way to see to it that no one else is able to get to the corporate data that SAP considers its own, not its customers. In short, SAP has created an application where data is optimized for the capture and storage of data. SAP data is not optimized for access and analysis, as seen in Figure 7.



ERP design is optimized for the capture of data and the storage of data, not the access or the analysis of data. No wonder end user analysts are so frustrated with ERP.

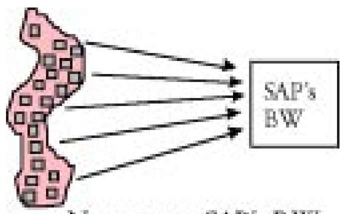
Figure 7

The problem is that it is not sufficient to capture and store data. In order to be useful, data must be able to be accessed and analysed. There is then a fundamental problem with SAP and that problem is that in order for the SAP application to be useful for analysis, the data managed under SAP must be “freed” from the SAP “data jail”.

The problems that have been described are not necessarily limited to any one ERP vendor. The problems that have been described are - in small or large part - applicable to all ERP vendors. The only difference from one ERP vendor to the next is the degree of the problem.

SAP, The ERP Leader

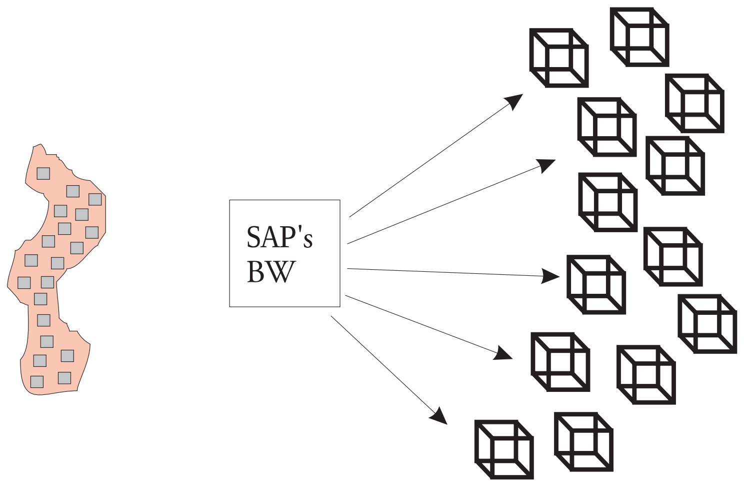
SAP, the leading ERP vendor certainly recognizes the problems that have been created by the placing of data in the SAP “data jailhouse”. In response to the need for information that is locked up in the ERP jailhouse, SAP has created what it calls the “Business Information Warehouse” or the “BW”. Figure 8 shows that SAP has created the BW.



Now comes SAP's BW.

Figure 8

While it is certainly encouraging that SAP has created a facility for accessing and analyzing data locked up in SAP, whether the form and structure of the BW is really a data warehouse is questionable. SAP has created a collection of cubes (i.e., OLAP like structures where the multi dimensionality of data can be explored.) Figure 9 shows the structures that SAP has created.



What SAP calls a data warehouse is a bunch of cubes.

Figure 9

There is no doubt that the cubes that SAP has created are welcome. Cubes make the information available within the structure of the confines of the cube. Indeed, given the lack of SAP reports, these cubes provide a partial replacement for that essential part of the SAP architecture that does not exist.

Do Cubes Make A Data Warehouse?

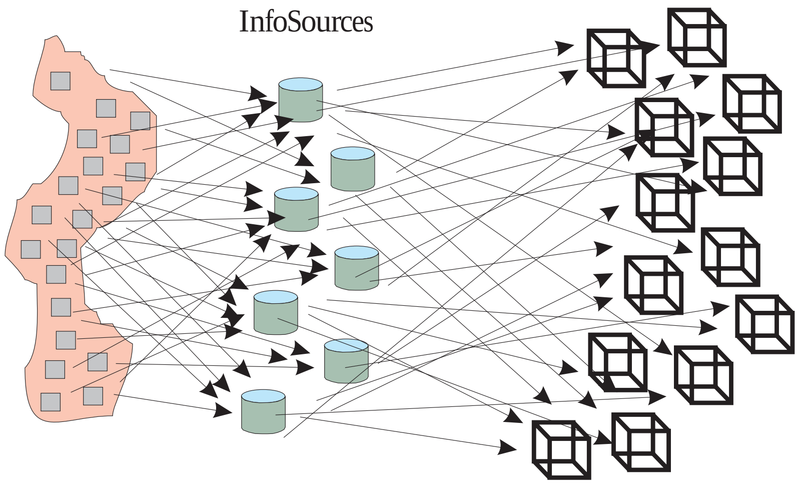
But do cubes constitute a data warehouse? The experience of data warehouse architects outside the SAP environment strongly and emphatically suggest that a collection of cubes - however well designed and however well intentioned - do not supplant the need for a data warehouse.

There are many reasons why a collection of cubes are not a replacement for a data warehouse. This paper will go into some of the more important of these reasons. But it is suggested that there are plenty more reasons why a collection of cubes do not constitute a data warehouse than will be discussed in this white paper.

A Data Warehouse

In order to be specific, what is a data warehouse? (To have a complete description and discussion on data warehousing, please refer to BUILDING THE DATA WAREHOUSE, 2ND EDITION, W H Inmon, John Wiley.) A data warehouse is the granular, corporate, integrated historical collection of data that forms the foundation for all sorts of DSS processing, such as data marts, exploration processing, data mining, and the like. A data warehouse is able to be reused and reshaped in many ways. The data found in the warehouse is voluminous. The data warehouse contains a generous amount of history. The data in the warehouse is integrated across the corporation.

The first reason why a bunch of cubes do not constitute a data warehouse is because of the interface from the cubes to the application. Figure 10 illustrates the problem.



The interface from the many SAP tables
to the staging area to the cubes is circumspect.

Figure 10

The ERP application contains a lot of tables. The cubes are built from those tables. Each cube must be able to access and combine data from a lot of tables. In order to accomplish this, SAP has created a staging area (in SAP parlance called an “ODS”). The staging area is an intermediate place where data is gathered to facilitate recoverability and the loading of cubes. While a standard data warehouse functionally does the same thing, there are some very important reasons why SAP’s staging area is not a data warehouse:

- the granularity of the data inside the staging area is not consistent. Some data is detailed at the transaction level. Some data is weekly summary. Some data is monthly summary. In short the staging area consists of a bunch of tables which have different levels of granularity. Trying to mix data from two or more tables of different granularity is an impossibility, as DSS analysts have found over the years.
- the data inside the staging area is not directly accessible nor comprehensible to anyone using a non SAP OLAP access and analysis tool. While the staging data exists in Oracle, its structure and content is such that it is not useful for direct access by a standard tool such as Brio, Business Objects, or others. In order to access the SAP data, the OLAP vendor must make the third party software work on top of the SAP OLAP engine using an OLE DB interface. The problem with this approach is that the third party OLAP vendor is subject to the limitations of the SAP OLAP engine. It is fair to say that the third party OLAP tools are much more sophisticated than the SAP OLAP tool. Furthermore, if a third party OLAP vendor does not have an OLE DB interface, then the third party OLAP tool cannot access the SAP data at all. By creating a roadblock to the access of the data, SAP has grossly limited the functionality that can be applied to SAP data. In addition, the ODS does not contain dimensional data (master data) and transactional data cannot be joined with dimensional data.

- the tables (InfoSources) in the staging area are segregated by source or destination and data elements (InfoObjects) need not be consistent across InfoSources.
- there is no consistent and reusable historical foundation that is created by the cubes. In a data warehouse, not only is a stable foundation created, but the foundation forms a historical basis of data, usually transaction data. From this historical foundation of data, many types of analysis are created. But there is no such historical foundation created in the staging area of SAP. It is true that SAP can store data historically. But the storage of historical data is done so that there is no compatibility of structure or release across different units of storage. In other words, if you store some data on Jan 1, some more data on Feb 1, and yet some more data on Mar 1, if the structure of data or the release of data has changed, then the data cannot be accessed uniformly. In order to be historically enabled, historical data must be impervious to the moment in time and the release of the storage of data.

In short, SAP staging area does not provide a basis for access to data by third party tools, does not provide integrated data, does not provide a historical foundation of data, and does not provide transaction level data. Instead, a web of cubes is created that require constant refreshment.

If there were only a few cubes to be built then the complexity and size of the interface would not be an issue. Even if a cube can build off of data that has been staged, the interface is still very complex.

Every cube requires its own customized interface. Once a corporation starts to build a lot of cubes, the complexity of the interface itself becomes its own issue. Furthermore, over time, as the corporation continues to add cubes, the interface becomes more and more complex. One way to calculate how many programs to be created is to estimate how many cubes will be required.

Suppose m cubes will be required.

Now estimate how many individual programs will be needed in order to access ERP tables. Suppose on the average that 36 tables need to be accessed by each cube. Now suppose a program can reasonably combine access to tables by doing a four way join. (If more than four tables are joined in a single program, then the program becomes complex and performance starts to really suffer.)

Furthermore, suppose that a staging area serves ten cubes. In this case the ten cubes would all have the same level of granularity.

Under these circumstances, the number of interface programs that need to be written and maintained are:

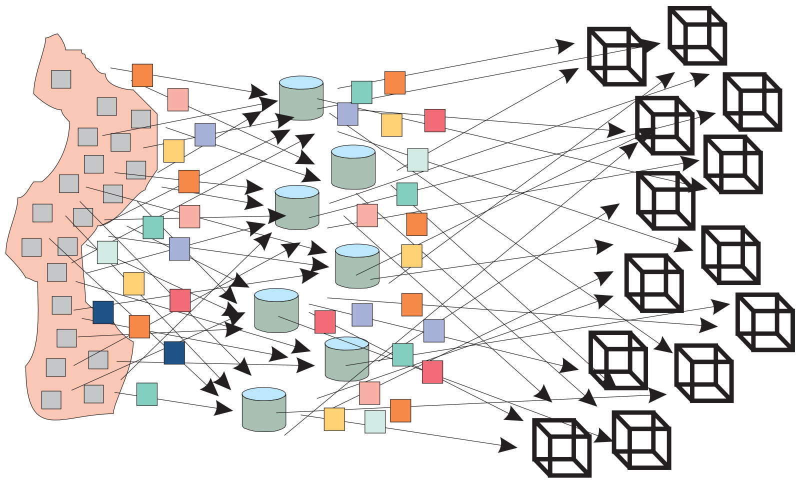
$$((36 / 4) \times m) / 10 = (9 \times m) / 10$$

If there are 25 cubes then (9 x 25)/10 = 22.5 programs need to be written. But if the SAP installation is large and there are a lot of cubes, then as many as 200 cubes may need to be created. The number of programs that need to be written and maintained in this case are:

(9 x 200) / 10 = 180 programs

It does not require a vivid imagination to see that the interface between the cubes and the SAP tables can become its own bottleneck. Both the initial creation of the interface and the ongoing maintenance of the interface present challenges.

Figure 11 shows that the complexity of the creation and maintenance of the interface between the SAP tables and the cubes is its own consideration.

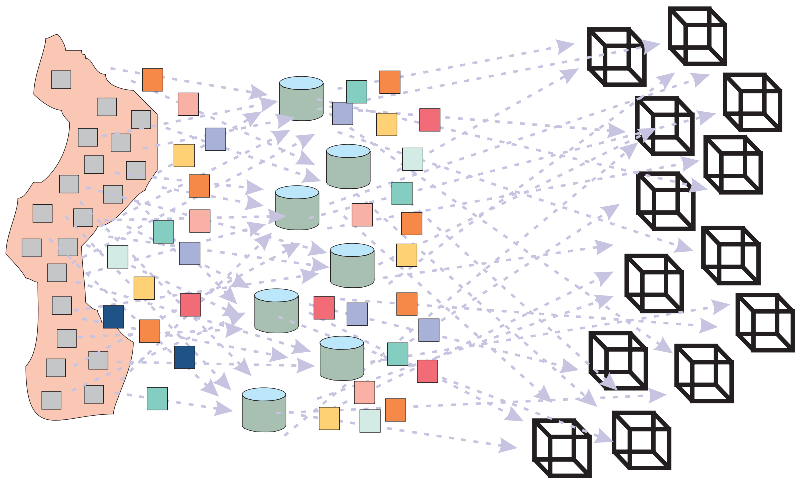


The number of interfaces and the lack of consistency of granularity is daunting.

Figure 11

But the creation and the maintenance of the interface programs is not the only issue. The next issue is that of the resources needed to keep the cubes up to date.

Figure 12 shows that significant hardware resources are required in order to keep the cubes in synch with the latest version of data in the transaction environment.



The sheer amount of hardware resources required for constantly maintaining the cubes is tremendous.

Figure 12

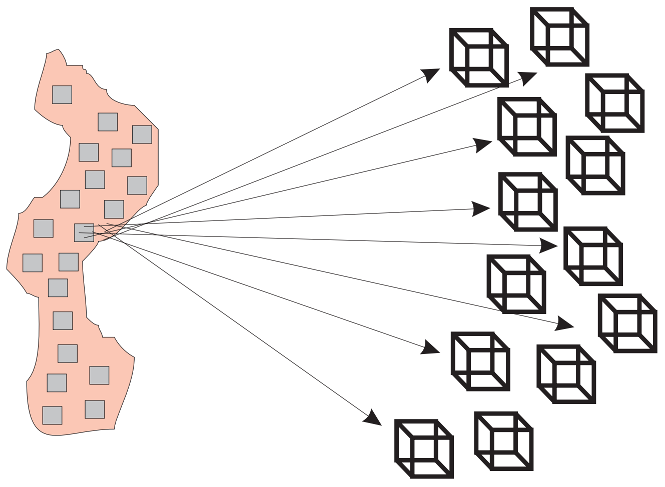
Every time the world of transactions changes, the staging area that the cube accesses needs to be changed. Then the cubes that emanate from the staging area that depends on data from the transaction that changed each require update. If update is not done, then one cube will be operating on and reporting on data from a different point in time than other cubes. In doing so, the consistency of reports coming from the cubes will vary, and in some cases vary considerably.

Hardware Resources

But updating one or more staging areas and then the cubes the staging area services every time a change is made in the transaction environment is a very expensive thing to do. The sheer number of resources required for constant creation and recreation of cube data is intimidating.

Keeping constantly moving data in synch is only one aspect of the problem of managing multiple cubes without a real data warehouse. Another challenge is that of keeping the structural semantics of the data in the cubes in synch as well. For example, suppose the definition and structure of a table in the ERP environment changes. The change has the potential for rippling throughout the staging environment and the cube environment, requiring the alteration of the structure of each cube (or at least many cubes). Each cube that is affected must be destroyed and rebuilt. In some cases this is not too much work. But in other cases this is an enormous amount of work. And in yet other cases the rebuilding of significant portions of the staging areas and the cubes is out of the question.

Figure 13 shows the ripple effect of structural changes throughout the cube environment.



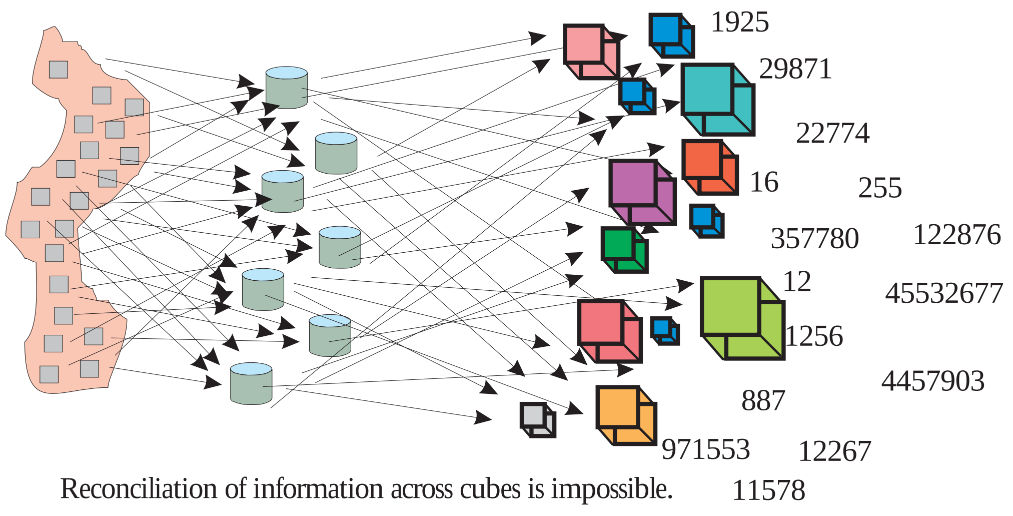
The ripple effect of making a single change in the SAP environment on the cubes is intimidating.

Figure 13

Another very bad downside of the multiple cube approach instead of a real data warehouse is that of the lack of reconcilability of data coming through the staging areas and through to the cubes. Figure 14 shows this downside.

Fig 14 shows that each cube is different from each other cube and yields different results when queried. The cubes are shaped by the requirements of the different members of the end user community. The cubes are different in many ways:

- different granularity,
- different dimensions,
- different calculations of roll ups of data,
- different volumes of data, and so forth.



Reconciliation of information across cubes is impossible.

Figure 14

It is not surprising that cubes are constructed differently because different cubes serve different communities. But this inherent difference in the structure and content of cubes leads to a problem with the lack of integration and the lack of consistency of data found in each cube.

Not surprisingly, one cube provides management with one answer and another cube supplies management with another answer. What is management to do? Who is management to believe?

Furthermore, the ability to reconcile results across cubes is circumspect. There is no "single point of truth" on which to make decisions. When viewed from the organizational perspective, no wonder the organization is so frustrated with the multiple cube approach to making decisions.

But inconsistency of information across cubes is not the only problem. There is no basic interchange of information across cubes as well. One of the powerful uses of OLAP technology has long been the ability to do drill down and drill across processing. But SAP's OLAP product builds and treats each cube as if it were a singular entity, with no ability to communicate or coordinate analysis across multiple cubes.

Other SAP Problems

But there are other problems with the SAP multiple cube approach to DSS processing. The problem is that many organizations already have a major investment in third party OLAP tools. The problem is effectively that SAP only allows its OLAP tool to access the cubes. Third party OLAP tools can access SAP data but only at a very superficial level. If third party tools could get at the transactional data or the cube data found in SAP cubes or staging areas then there would not be a problem. But, for the most part, SAP bars standard third party tools of access and analysis from getting to the SAP staging area or cube data. Figure 15 shows that the SAP solution is a highly proprietary solution.

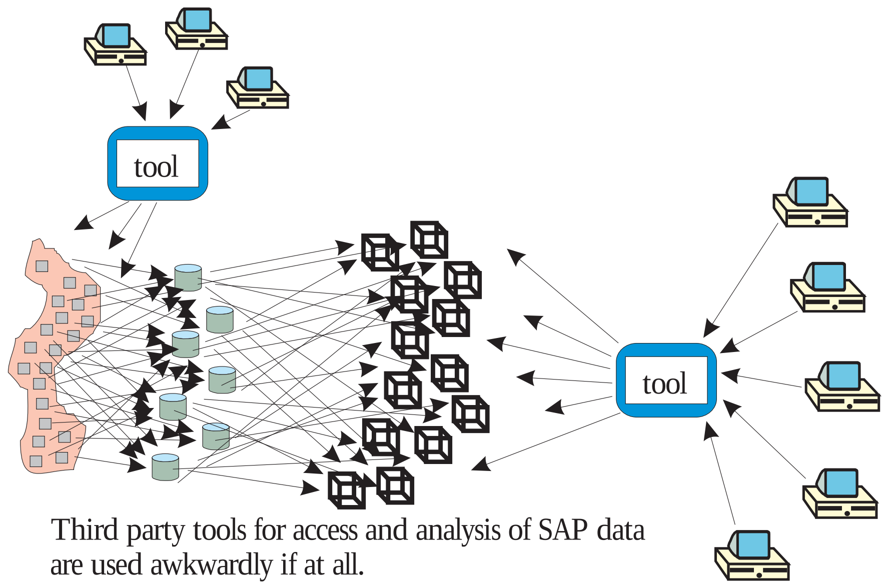

tool
tool
Third party tools for access and analysis of SAP data are used awkwardly if at all.

Figure 15

SAP will say that third party OLAP tools can access SAP data. This is true in the sense that SAP allows access through SAP interfaces. But the third party OLAP tools have to access the SAP data through an SAP interface. Stated differently, the third party tools have no direct and independent access to the SAP data. In short, the third party OLAP tools outside of ERP are out of luck. Unfortunately for SAP, the community of third party OLAP access and analysis tools are well entrenched and are already well accepted. There is a mature audience of users of third party OLAP access and analysis tools that cannot directly use SAP data. Many corporations have a significant capital and intellectual investment made in these third party OLAP tools and would like to see a constructive interface to SAP, if SAP would allow it.

Redundant Data

But there are other reasons why the SAP multiple cube approach instead of a real data warehouse is expensive. There is a tremendous amount of redundant data across the cubes that are created. Figure 16 shows this redundancy of data.

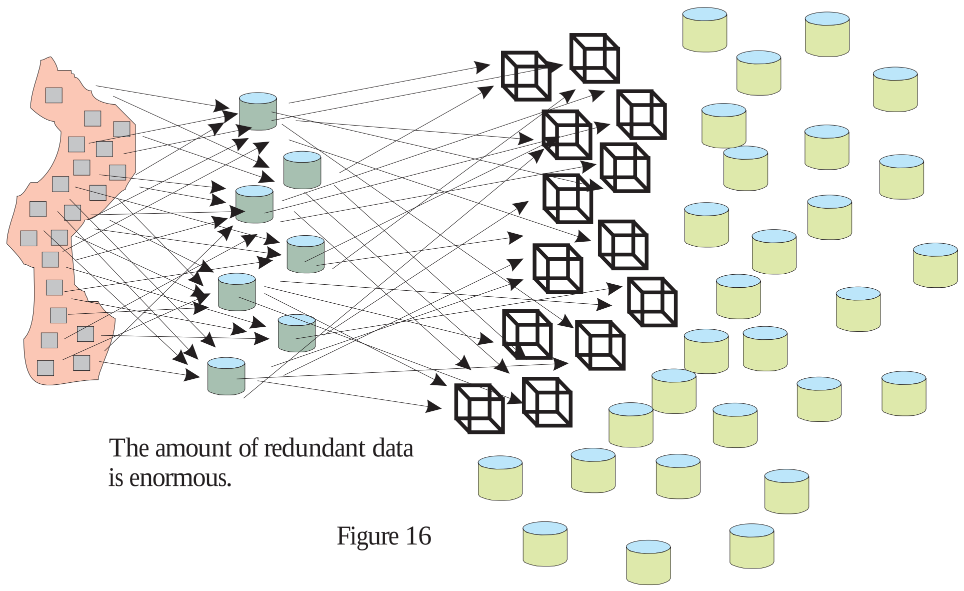


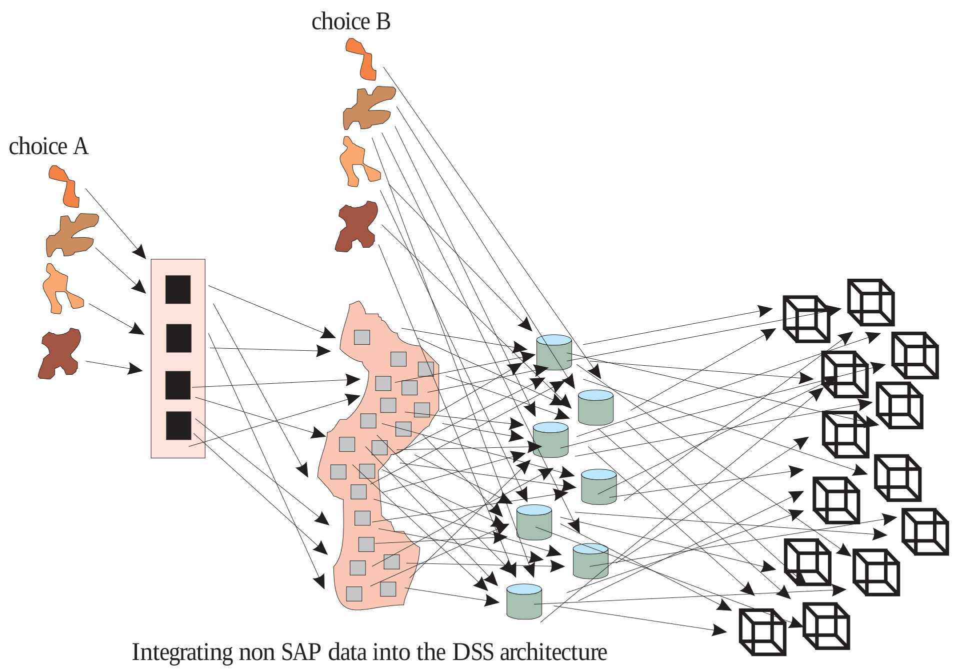
Figure 16

The redundancy of data occurs because each cube must effectively capture, restructure and store the same data that each other cube stores. (Strictly speaking this may not be true. Under some circumstances for a small number of tables, there may not be much redundancy of data. But over many tables over normal circumstances, it is normal for much redundant detailed data to be repeated in cube after cube.) The creation of redundant data implies that the cubes are much larger and much more expensive than they have to be.

The Challenge Of Including Non SAP Data

Another problem of the SAP multiple cube approach is that of what to do about non-SAP data that needs to be included in DSS analysis. Figure 17 shows this dilemma.

Fig 17 shows that there are two choices. One choice is to integrate the non-SAP data into the R/3 portion of SAP and then include this data into the SAP cubes.



Integrating non SAP data into the DSS architecture is a real trick.

Figure 17

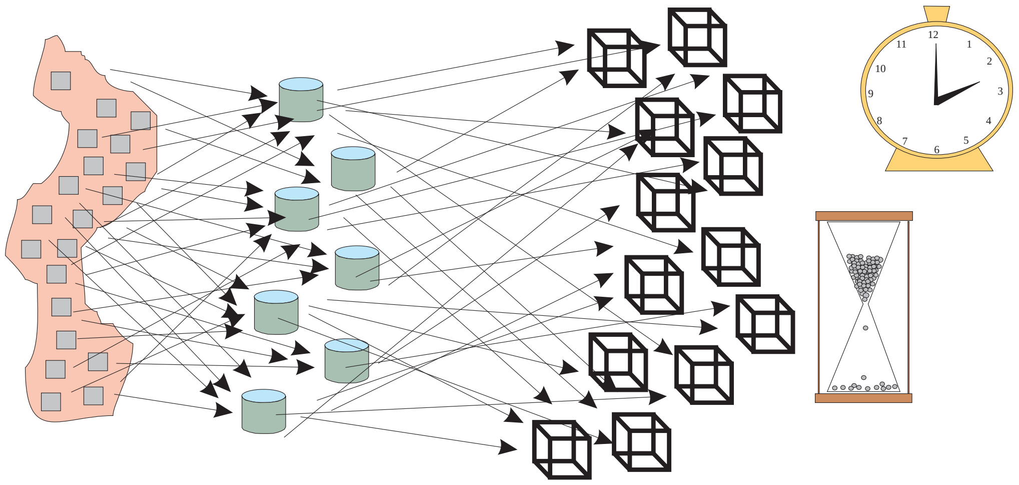
The second choice is to try to incorporate non-SAP data directly into BW without bringing that data into SAP R/3 at all. This direct approach is very appealing. The problem is that in order to do this, customers must write complex ABAP programs to reconcile SAP and non-SAP data. This may be able to be done for some very simple sorts of data, but for the general case, this is a very, very difficult thing to do.

Creating A New Cube

Another important issue is that every time a new cube is defined in the SAP environment, the cube needs to be fed from the raw data found in SAP or from a staging area. While there are tools for the creation of cubes, the fact that there is no intermediary data warehouse means that all cubes must start from scratch if a staging area doesn't exist as a foundation. If in fact there were a real data warehouse, the new cube could be built directly from the data warehouse, not from the raw transaction data. By using a data warehouse as a foundation for new cubes, the designer bypasses huge amounts of work that are required for integrating data coming from a transaction foundation or a staging area/transaction foundation.

Complicating matters is the fact that SAP only allows inserts of data into the cubes, not deletes and updates. In a perfect world there is no need for periodic refurbishment of data. But in a real world there is need for such activity.

A final issue of the SAP cube approach is that there is no satisfactory place for historical data. Figure 18 depicts this shortcoming of SAP.



Another issue is that there is only a handful of historical data that is available.

Figure 18

The SAP R/3 application is good for holding a modicum of historical data. But when it comes to years and years of historical data, the SAP application is hardly the optimal place for historical data. Holding significant amounts of historical data in the SAP application impairs the running of day to day transactions.

And holding large amounts of historical data in the cubes created by SAP BW is not the thing to do either. A cube is limited in its ability to optimize the structure of data for more than one client at a time. For this reason a cube may be good for one user and useless for another user. If historical data were to be placed in the cube environment, the historical data would have to be placed in many cubes. The problem is that placing historical data in many cubes costs a lot.

A second problem with placing historical data in cubes is what to do with the historical data when the cube must be reconstructed. It is one thing to reconstruct a cube when the base data is all in one place to begin with. It is quite another thing to reconstruct a cube when only part of the data is available for reconstruction. And reconstructing a cube when there is a lot of data already residing in the cube is not a pleasant prospect either.

One approach to the management of historical data is that of putting the historical data in the staging area. But there are many problems with this approach. The first is that the staging area contains data at many levels of granularity. A data warehouse bypasses this problem by placing historical data in the warehouse at the most granular transactional level.

What Does The Multiple Cube Approach Do?

What then does the SAP multiple cube approach do? In order to answer this question, there must be a discussion on management reporting. There are many facets to and types of management reports. Management reporting can be done for:

- "what if" analysis,
- exception reporting,
- critical factor analysis,
- statistical analysis,
- exploration analysis,
- data mining,
- periodic standard reports, and so forth.

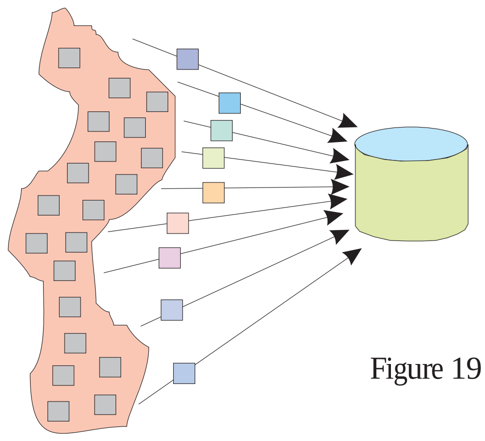
Management reports serve many different levels of the company, from the president to the newly minted MBA. The reports that are available from SAP barely scratch the surface for management's needs for information. The multiple cube approach barely provides a sliver of the spectrum of reports that are needed.

Given that SAP R/3 effectively has no operational reports (as opposed to management reports), the multiple cubes offered by SAP BW make a first attempt at providing the necessary information.

In order to be most effective, SAP's transaction detailed data needs to be stored in an integrated, historical manner where easy access can be made by any tool desired by the end user. SAP's current solution is light years away from this structure.

A More Rational Approach

A much more rational approach for the execution of informational processing is to create a proper foundation of integrated, detailed, historical transaction data. In order to create this foundation, it is necessary to pull detailed data out of SAP and create a real data warehouse, not a multiple cube imitation of a data warehouse. Figure 19 shows that a real data warehouse can be created from data pulled from the SAP environment. Fig 19 shows that the detailed transaction data is pulled out of SAP, integrated and/or reintegrated, and placed into a real data warehouse.



A much more rational approach is to pull the data out of SAP and into a real data warehouse.

Figure 19

Once the real data warehouse is built:

- the detailed foundation of data can be reused by many users,
- can be reconciled,
- can have detailed data stored in a single place,
- can have historical data stored,
- can have non-SAP data easily integrated into the data warehouse,
- can be accessed by standard tools, and so forth.

In other words, the problems of the multiple cube approach espoused by SAP are solved by pulling the data out of SAP into a real data warehouse, not a multiple cube facsimile.

The multiple cubes of information that SAP has created can easily be created from the data warehouse, as seen in Figure 20.

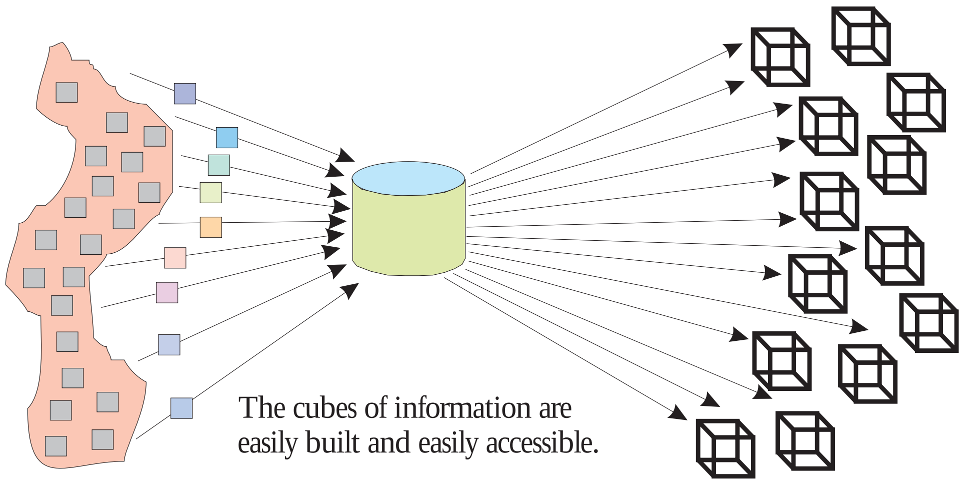
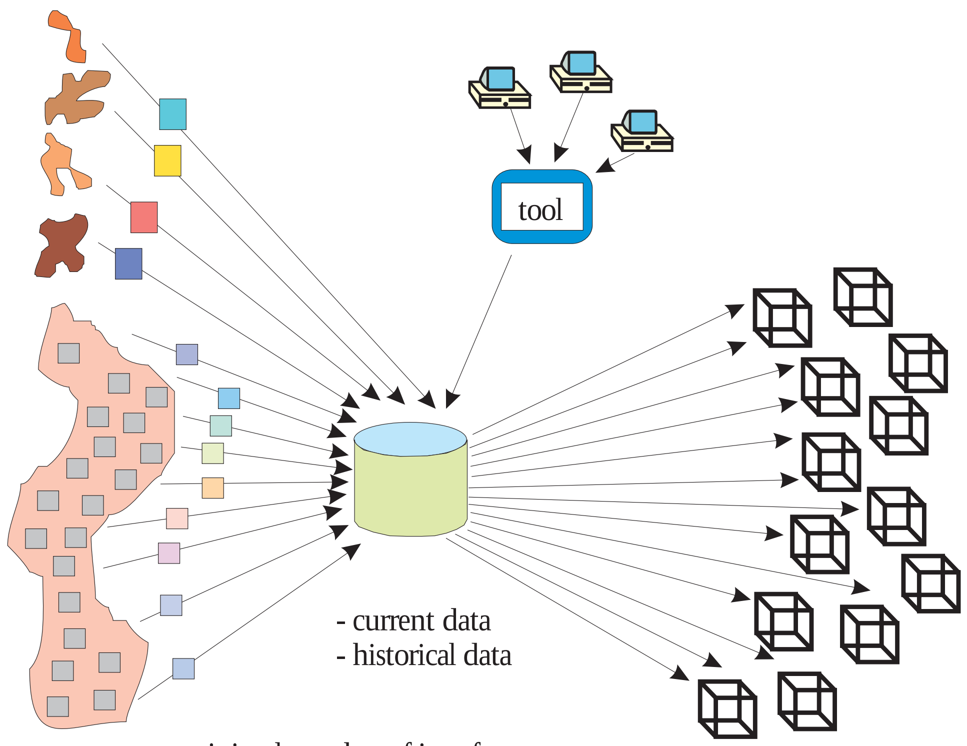

The cubes of information are
easily built and easily accessible.

Figure 20

There is no loss of functionality by going to a real data warehouse outside of SAP.

The environment that is created is shown by Figure 21



- minimal number of interfaces
- current and historical data
- non redundant data
- a single source for reconciliation
- accessible by third party tools
- able to be fed by non SAP data
- ability to add cubes with ease
- a data value needs to be changed in a single place
- documentation for tools and dbms is in English

Building a real data warehouse with SAP data outside of SAP.

Figure 21

Another way to contrast the differences between a real data warehouse and the cube approach shown by SAP is illustrated by Figure 22.

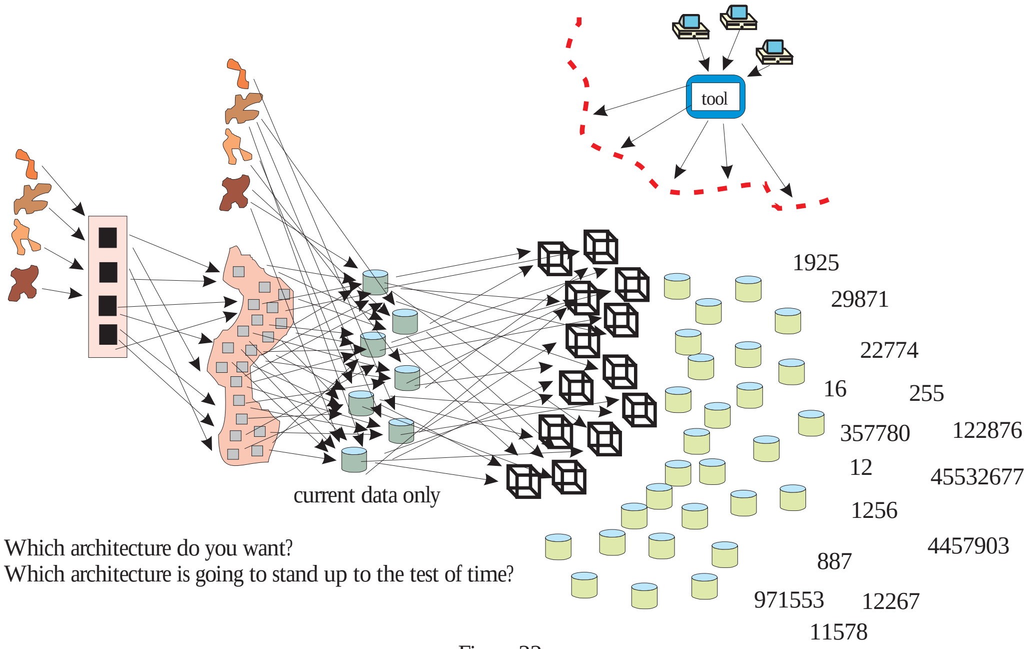


Figure 22

Figure 22 shows that there are major architectural differences between the two approaches.

Conclusion

SAP's BW is not a real data warehouse. Its intended purpose is to provide an infrastructure for developing specific management reporting applications. In contrast, a true data warehouse architecture will support a complete range of decision support functions such as:

- operational reporting as well as management reporting,
- root cause analysis at the detailed transaction level with drill down, drill across, and drill through based on a single and consistent source of data that accounts for the need for reconciliation across multiple business functions.

The true data warehouse architecture will leverage the business investment in best of breed query, OLAP and business intelligence tools without limiting their capabilities. Business that requires the robust and valuable amount of information that resides in a data warehouse will implement a true data warehouse outside of SAP by pulling data out of SAP and integrating the data, or will build a true data warehouse along side the SAP cubes.