OLAP Data Scalability

White Paper

Ignore OLAP Data Explosion at great cost.

"… many organisations will never know that they figuratively bought a very expensive rowing boat, when they could have traveled business class for less!"

by Johann Potgieter

Executive Summary:

Introduction:

The reality of data explosion in multi-dimensional databases is a surprising and widely misunderstood phenomenon. For those about to buy or use an OLAP product, it is critically important to understand what data explosion is, what causes it, and how it can be avoided, because the consequences of ignoring data explosion can be very costly, and in most cases, result in project failure.

There are very few OLAP vendors who can truly claim to have technically conquered the consequences of data explosion. The claims offered by many vendors about how they manage data explosion make it very difficult to understand what is actually important on this topic and what's not.

For example, one of the problems of data explosion is that it results in a massive database. The size of the database in one product can literally be hundreds and even thousands of times bigger than the same database in another product.

Rather than admit to the problems of data explosion, the vendor with the massive database will argue that his database is handling large data sets, while he will imply that the vendor of the smaller database – a database without data explosion - cannot address large enterprise datasets.

The correct analysis should be to compare sizes with equal volumes of base data, but because the size of the databases are so profoundly different, prospective customers find it hard to believe that such dramatic differences are possible with similar datasets.

The end result is that organisations often commit to what they erroneously believe is the best so-called enterprise solution. This mistake comes at a huge price (see consequences of ignoring data explosion). Ironically, in this way, a vendor's biggest weakness (data explosion) becomes their biggest selling point.

The consequence of ignoring data explosion:

- Massive databases that are literally hundreds and even thousands of times larger than is necessary
- Expensive hardware is required to process and accommodate exploded data
- Load and or calculation times that take hours rather than seconds or minutes
- Large costs to build and maintain these monolithic models
- The hidden cost of failing to provide timely and relevant enterprise business intelligence - there is a great cost associated with the inability to make fast business decisions and the negative culture that prevails because of poor underlying analytical systems
- Real or defacto project failure

Sparsity (antonym – Density):

"Sparsity" and "sparsity handling" are important concepts worth understanding as a precursor to understanding data explosion.

Input data or base data (i.e. before calculated hierarchies or levels) in OLAP applications is typically sparse (not densely populated). Also, as the number of dimensions increase, data will typically become sparser (less dense).

For example, in a 1 dimensional matrix, you can suppress all zero values and therefore have a 100% dense matrix. In a 2 dimensional matrix, you cannot suppress zeros if there is a non-zero value in any element in the two dimensions (see figs 1 and 2).

	YEAR
A	10
C	20
D	8
F	15

Fig 1: 100% dense

	Q1	Q2	Q3	Q4
A	10	0	0	0
B	0	20	0	0
C	0	0	8	0
D	0	0	0	15

Fig 2: 25% dense (4 out of 16 data points populated)

Whilst it is not true in all cases, typically as the number of dimensions in a model increases, so does the data sparsity. For example, if you are storing sales data by product and by month, it is conceivable that you will sell each product each month (100% dense). However, if you were storing sales data, by product, by customer, by region, by month, clearly you would not sell each product to every customer in every region every month.

By adding the dimension "gender" to this model, you would double the possible size of the cube by storing the data by either of the two variables male or female, but the size of the stored data would remain the same. In this case, by introducing another simple dimension, the sparsity will have doubled!

To provide a practical baseline expectation for sparsity, we researched data sparsity on a variety of models with a sample of 7 companies. Each company had a variety of models (eg P&L, Balance Sheet, Cash Flow, Sales Analysis, HR/Labour Analysis, Budgeting & Forecasting, Industry Specific models etc) with differing dimensions.

The Industry specific models included insurance claim analysis, "telco" call analysis and revenue per user analysis and a medical device company's sales analysis. A detailed summary of our research can be found in Appendix A.

Our summary findings were as follows:

1. Data density in all cases was significantly less than 1% - i.e. extremely sparse.
2. As the number of dimensions increases, so did the sparsity of the data (models reviewed had between 5 and 16 dimensions).
3. Extreme sparsity existed in all the "Industry Specific" models (all models had density of less than 1 billionth of a %).

Sparsity Handling

Superficially, any multi-dimensional model needs to provide space for every possible combination of data points. Since in sparse models most data points are zeros, the main issue is how to store all values other than zero values. For example, if the data density of a model is 1% and there is no sparsity handling, the resulting model will be 100 times larger than a model that has perfect sparsity handling. Sparsity handling therefore is the efficient storage of very sparse data.

Don't confuse poor sparsity handling with data explosion – a common myth:

It is important not to confuse poor sparsity handling (the inefficient storage of zero values) with data explosion. Although sparsity handling is an issue for multi-dimensional databases, it usually only accounts for differences of less than ten times between products.

Whilst some might say that a difference of size of up to ten times is important, it is nowhere near as important as the differences that arise as a result of data explosion. As stated previously, these differences can be hundreds and even thousands of times between good and bad databases. Also, whilst sparsity handling was more of a problem a few years ago, most vendors now have a reasonable solution for this.

Importantly, this is another classic area of vendor deception. When challenged on the topic of data explosion, some vendors divert the argument to sparsity handling. Because both sparsity handling and data explosion are poorly understood, a vendor's handling of sparsity may be incorrectly accepted as an ability to adequately address and avoid data explosion.

Data Explosion - what it is and what causes it – the facts

Data explosion is the phenomenon that occurs in multidimensional models where the derived or calculated values significantly exceed the base values. There are three main factors that contribute to data explosion.

1. Sparsely populated base data increases the likelihood of data explosion
2. Many dimensions in a model increase the likelihood of data explosion
3. A high number of calculated levels in each dimension increase the likelihood of data explosion

Let's extend our previous example to explain this. In fig 3, the 100% dense 1 dimensional model with 2 levels, has base data that exceeds calculated data in a ratio of 4:1. There is no data explosion here. In fig 4, the 25% dense 2 dimensional model with 3 levels in 1 dimension, has base data of 4 values that "explodes" to 15 calculated values.

	YEAR
A	10
C	20
D	8
F	15
Total	**53**

Fig 3: No data explosion

	YEAR	Q1	Q2	Q3	Q4
A	**10**	10	0	0	0
B	**20**	0	20	0	0
C	**8**	0	0	8	0
D	**15**	0	0	0	15
A+B	**30**	**10**	**20**	**0**	**0**
C+D	**23**	**0**	**0**	**8**	**15**
A+B+C+D	**53**	**10**	**20**	**8**	**15**

Fig 4: 3.75 times data explosion

By increasing sparsity, and/or adding dimensions, and/or adding calculated levels in dimensions in the above example, the data explosion rate will increase.

A frame of reference for data explosion in practice:

In practice between 5 and 12 dimensions are very common. Highly sparse models are also typical. Density factors of 1% or less are very common and should be assumed unless proven otherwise. A typical product dimension has between 4 and 9 levels and an account dimension has between 8 and 16 levels. The other dimensions in typical models often have between 2 and 6 levels.

To provide a practical frame of reference, we researched data explosion in a variety of models from our sample of seven companies. A detailed summary of our research can be found in Appendix A. Important findings were as follows:

1. Data density in all cases was significantly less than 1% with extremes in all the "Industry Specific" models.
2. The average number of levels in each model was between 3 and 6.
3. The highest number of levels for any dimension in each model was between 6 and 16 levels.
4. The core "industry specific" models had in all cases between 10 and 16 dimensions.
5. The basic P&L models researched each had between 12 million and 40 million base data points while the "industryspecific" models researched each had between 80 million and 366 million base data points.
6. A data explosion factor of between 30 and 500 times was recorded in the basic P&L models researched resulting in between 600 million and 21 billion exploded data points.

7. Precalcualted data exploded to be between 4,000 and 66,000 times larger than the base data in the "industry specific" models researched resulting in between 356 billion and 23,972 billion exploded data points.
8. After data explosion, the "industry specific" models would require disk storage or memory of between 2 and 150 Terabytes versus between only 1 and 5 Gigabytes for models unaffected by data explosion.
9. To load and fully precalculate incremental data, the "industry specific" models would require between 11 hours and 242 days versus between only 1 and 27 minutes for models unaffected by data explosion.

These results are astounding and hard to believe. The reality is that many vendors are relying on you not believing this. Furthermore, the above points only begin to suggest the limitations experienced by those effected by data explosion.

Clearly, there are some models that cannot be contemplated if they are precalculated. For example, a load and precalculation time of 242 days or even 6.5 days is not feasible. As you can see from our research, these models are invariably the important "industry specific" models – such as sales analysis, customer analysis and product analysis. These are the real enterprise models with large data sets.

Furthermore, there are many typical OLAP applications where precalculation times of a few hours or even only ten minutes will render these models unpractical. For example many "read/write" applications, such as "budgeting and forecasting data collection", "what if analysis" and "scenario modeling", typically require instant calculation and or consolidation of any result in the model. Turn around times of tens of minutes or hours are totally unacceptable especially where there are many users.

Beware of Deceptive Vendor Benchmarks and Explanations!

Vendors' attempts to conceal their inability to beat data explosion may now seem obvious in light of the causes of this phenomenon. Unfortunately many organisations will still never realise that they have been figuratively sold a very expensive "rowing boat", when they could have been traveling "business class" for less. Some of the more common and often extremely well camouflaged methods of deception are outlined below:

1. Concealing data explosion flaws by marketing them in such a way as to make them appear like strengths:
    (i) Deliberately confusing sparsity handling with data explosion.
    (ii) Claiming that massive databases reflect an ability to address enterprise datasets, rather than honestly comparing database sizes using similar based data.
    (iii) Using marketing jargon such as "best practice" and "best of breed" without providing any corroboration or evidence of these claims. Unless an independent source is sited for claims of superiority, it is highly likely that such self-appointed "best practice" and "best of breed" products are anything but.
2. Publishing "benchmarks" or "capability demonstrations" using a combination of any or all of the following to avoid revealing evidence of data explosion:
    (i) Using few dimensions
    (ii) Using non-sparse data
    (iii) Using few calculated levels in dimensions
    (iv) Using small datasets

Conclusion

1. Beware of deceptive and hollow marketing jargon and claims.
2. Make every effort to truly understand data explosion and the effects it can have on your database.
3. Do not confuse "sparsity handling" with "data explosion."
4. Carefully review vendor "benchmarks" and "capability demonstrations" for evidence of an attempt to suppress any of the four main causes of data explosion. The following should serve as a useful checklist:
   (i) Models should have at least 6 or more dimensions
   (ii) Base data density should be 1% or less.
   (iii) An average of 3 or more calculated levels in all dimensions and 5 or more levels in at least 2 of the largest dimensions should be required. At least 1 dimension should have 1,000 or more elements and another with at least 100 elements.
   (iv) Model should have at least 10 million base data points. Ensure this is in fact base data and not a total number of fully pre-calculated data points (base data of as little as 10 thousand data points can easily result in 10 million fully pre-calculated data points. The difference in size would be 1,000 times).
5. Insist on vendors performing a realistic proof of concept, or benchmark against other vendors. Ask for a money back guarantee against the risk of data explosion. Any protest will provide a hint as to their capabilities in this area.
6. Perform reference checks and ask detailed and specific questions during this process. For example, if a referee doesn't know that it is possible to calculate a consolidated view of refreshed or changed data in seconds, they might consider overnight recalculation of their cubes to be reasonable. If asked the question, how is performance, they may respond with "OK", or ""Good" without purposely intending to deceive you. However, if you ask the specific question "how long does it take to recalculate a consolidation after data is refreshed?", the answer "9 hours", will certainly provide a different perspective.
7. Whilst there are many different OLAP architectures (ROLAP, MOLAP, DOLAP etc), any architecture that uses either a partial or full precalculation approach, will almost certainly be affected by the consequences of data explosion.

Choose wisely and you can take the ability to fly for granted!

Appendix A

	Model description and Industry						
Actual metrics from production models using software NOT impacted by data explosion	**Claim Analysis (Insurance Industry)**	**Call Analysis & Revenue per User (Telco)**	**Sales Analysis (Medical Devices)**	**Labour Budget (Utility)**	**P&L (Manufacturing)**	**P&L (Energy)**	**General Ledger (Logistics)**
Number of dimensions	12	16	12	11	9	6	5
Average number of calculated levels per dimension	3.6	3.2	3.3	2.7	5.8	4.2	5.6
Number of elements in largest dimension	805,879	1,178	38,595	1,990	6,792	71,486	38,418
Memory used for base data	1,993 MB	4,572 MB	1,087 MB	96 MB	504 MB	156 MB	246 MB
Number of populated base data	159,435,000	365,780,000	86,926,130	7,665,872	40,320,000	12,494,784	19,651,170
Time in minutes to load base data	318.9	731.6	174	15	81	25	39
Time in minutes to load incremental data specific to this model	11.8	27.1	1	0	2	1	3
Total possible data points	24,858,694,491,128,300,000,000,000 Billion	11,309,917,308,311,400 Billion	83,461,733,138,297,600 Billion	1,143,212,526 Billion	84,451,026 Billion	32,416 Billion	340 Billion
Sparsity	0.00000000000000000000000064%	0.0000000000000032%	0.000000000000000010%	0.00000000067%	0.0000000048%	0.000039%	0.0058%
Calculated metrics emulating software impacted by data explosion							
Compound growth factor	2	2	2	2	2	2	2
Exploded data	653 Billion	23,972 Billion	356 Billion	16 Billion	21 Billion	799,666,176	628,837,440
Number of times exploded data is larger than base data	4,096	65,536	4,096	2,048	512	64	32
Memory required for exploded data	4,081,536 MB 4.08 TB	149,823,488 MB 149.82 TB	2,225,309 MB 2.23 TB	98,123 MB 98.12 GB	129,024 MB 129.02 GB	4,998 MB 5. GB	3,930 MB 3.93 GB
Time to load and precalculate	891 days 7 hrs 41 mins	32718 days 6 hrs 24 mins	485 days 23 hrs 2 mins	21 days 10 hrs 16 mins	28 days 4 hrs 13 mins	1 days 2 hrs 11 mins	20 hrs 35 mins
Incremental load and precalculation time specific to this model	33 days 17 mins	1211 days 18 hrs 54 mins	2 days 7 hrs 55 mins	10 hrs 42 mins	14 hrs 5 mins	32 mins	1 hrs 42 mins
Time to load and precalculate (16 processors 5 partitions)	178 days 6 hrs 20 mins	6543 days 15 hrs 40 mins	97 days 4 hrs 36 mins	4 days 6 hrs 51 mins	5 days 15 hrs 14 mins	5 hrs 14 mins	4 hrs 7 mins
Incremental load and precalculation time specific to this model (16 processors 5 partitions)	6 days 14 hrs 27 mins	242 days 8 hrs 34 mins	11 hrs 11 mins	2 hrs 8 mins	2 hrs 49 mins	6 mins	20 mins

The first table above contains actual data and metrics from a sample of 7 companies that were using software that does not use partial or complete precalculation and is not impacted by data explosion. The second table contains derived metrics calculated to illustrate data explosion with software based on a precalcualtion approach. This table illustrates that the first 3 models would not be feasible based on database sizes of in excess on 2.2 terabytes and incremental load times of more than 11 hrs.

Definitions and Assumptions:

1. **Memory used for base data:** We found that 80,000 data points were consistently stored in 1 MB of RAM in the OLAP product we used for this research. We therefore derived this number by dividing the number of populated base data points by 80,000.
2. **Number of populated base data Points:** To derive this number we counted all or a significant part of the base data in each model, then extrapolated this to reflect a full, but reasonably populated model. For example, if we counted 6 months of data, we doubled it to reflect 12 months. However we did not use this approach in dimensions such as customer, product etc as it would not fairly reflect possible data in the models.
3. **Time in minutes to load base data:** We found that we could consistently load 500,000 base data points per minute into the OLAP database. We therefore derived this number by dividing the number of populated base data points by 500,000.
4. **Time in minutes to load incremental data specific to this model:** The incremental data was specific to each model. For example if the model was changed monthly and had 12 months of data, an incremental load would be one twelfth of the time in minutes to load base data. If the model had 12 months and 2 years, an incremental load would be one twenty-fourth (12 X 2) of the time in minutes to load base data.
5. **Total possible data points:** To derive this number we multiplying together the total number of elements in each dimension.
6. **Sparsity:** The "Number of populated base data Points" divided by "Total possible datapoints."
7. **Compound growth factor** (CGF): CGF is a term coined by the OLAP Report (http://www.olapreport.com/DatabaseExplosion.htm) used to calculate data explosion. It is the data growth factor per dimension. The OLAP Report states that it is reasonable to assume a CGF of about 2 for typical applications, which are moderately sparse with semi-clustered data. We have very conservatively used a CGF of 2 in all our calculations although the OLAP Report would suggest a CGF of 2.5 or perhaps even as high as 3 would be reasonable based on these models being extremely sparse and having 6 or more dimensions.
8. **Exploded data:** The "Number of populated base data Points" multiplied by "CGF" to the power of the number of dimensions in the model.
9. **Number of times exploded data is larger than base data:** "Exploded data" divided by "Number of populated base data Points."
10. **Memory required for exploded data:** We derived the number of data points stored per MB from published white papers siting projects storing precalculated or partially precalculated data. We conservatively used the highest derived number, which was 160,000 data points stored per MB. That this was twice the actual size we found for unexploded base data, gave us comfort that we were not overstating our case against data explosion. We therefore derived this number by dividing the number of "Exploded data" points by 160,000.
11. **Time to load and precalculate:** We derived the load and precalculation times from published white papers siting projects loading and precalculating data. We found that 508,800 records were being processed per minute. That this was very close to the 500,000 data points per minute for unexploded base data, gave us comfort that these were reasonable numbers. We therefore derived this number by dividing exploded data points by 508,800.
12. **Incremental load and precalculation time specific to this model:** The incremental data was specific to each model. For example if the model was changed monthly and had 12 months of data, an incremental load would be one twelfth of the time in minutes to load base data. If the model had 12 months and 2 years, an incremental load would be one twenty-fourth (12 X 2) of the time in minutes to load base data.
13. **Time to load and precalculate (16 processors 5 partitions):** We derived the load and precalculation times from published white papers siting projects loading and precalculating data. We found that 5 partitioned parallel processes using a total of 16 CPU's processed 2,544,000 records per minute. We therefore derived this number by dividing exploded data points by 2,544,000.
14. **Incremental load and precalculation time specific to this model (16 processors 5 partitions):** The incremental data was specific to each model. For example if the model was changed monthly and had 12 months of data, an incremental load would be one twelfth of the time in minutes to load base data. If the model had 12 months and 2 years, an incremental load would be one twenty-fourth (12 X 2) of the time in minutes to load base data.