BALEEN: *The Fast, Scalable and Flexible Data Warehouse*

What is it?

- Baleen is an **orthogonal data warehousing system** that allows for the storage of any clinical or financial data record from any kind of hospital system. It achieves this by adopting a true ‘post relational’ approach to data warehousing.
- Baleen never needs to be defined. **The data itself IS the definition**.
- Baleen lets **YOU choose the data models** YOU wish to employ, **Baleen is model agnostic**.
- Baleen requires little or no index building as-is. The **true cost of indexing** data is built into the core structures.
- Baleen **supports many custom data storage topologies** and can be hosted on any kind of database, irrespective of the vendor.
- An **owner of baleen can build out custom structures** or use baleen’s own middle tier integration process for data display and data mart delivery via REST XML and other standard/custom formats.
- Baleen is **fully customizable** and recursively simple, an owner of Baleen can grow and expand it and share those changes with other users. Baleen is currently being implemented in Microsoft .NET, but since the underlying data model is the power of Baleen, any kind of middle tier can be used. Baleen’s source code is sold with the product the end user is free to customize the code – in accordance with agreements – as much as they wish!
- Baleen **allows owners to make scientific and predictable choices about growth**. Hidden growth costs are never completely avoided, but Baleen avoids most of them and gives

the CTO/CIO the power to make strategic data storage choices and to manage the complex budgets associated with data warehousing.

- Although Baleen is being marketed primarily to health care organizations, it is not limited to this domain. Baleen's power is in its generic storage model, so **any industry or domain could take advantage of Baleen's features**.
- Baleen allows customers to implement a new data model, but **RE-USE current business rules with minimal modification.**
- Because Baleen coalesce dimensions instead of conforming them, **it can load data quickly without suffering from data quality issues**. Baleen will attempt to treat any token as a type, unless the parse fails.

The Baleen Universes of Information

1. **Domains** – data as it is organized into a multi-server environment.
2. **Sets** – data as it is classified by off line classification algorithms for grouping and analysis.
3. **Ontology** – data as it exists within a 'theory of knowledge' model.
4. **Facts** – data as it is stored and protected for the long term.

The core values which motivate Baleen

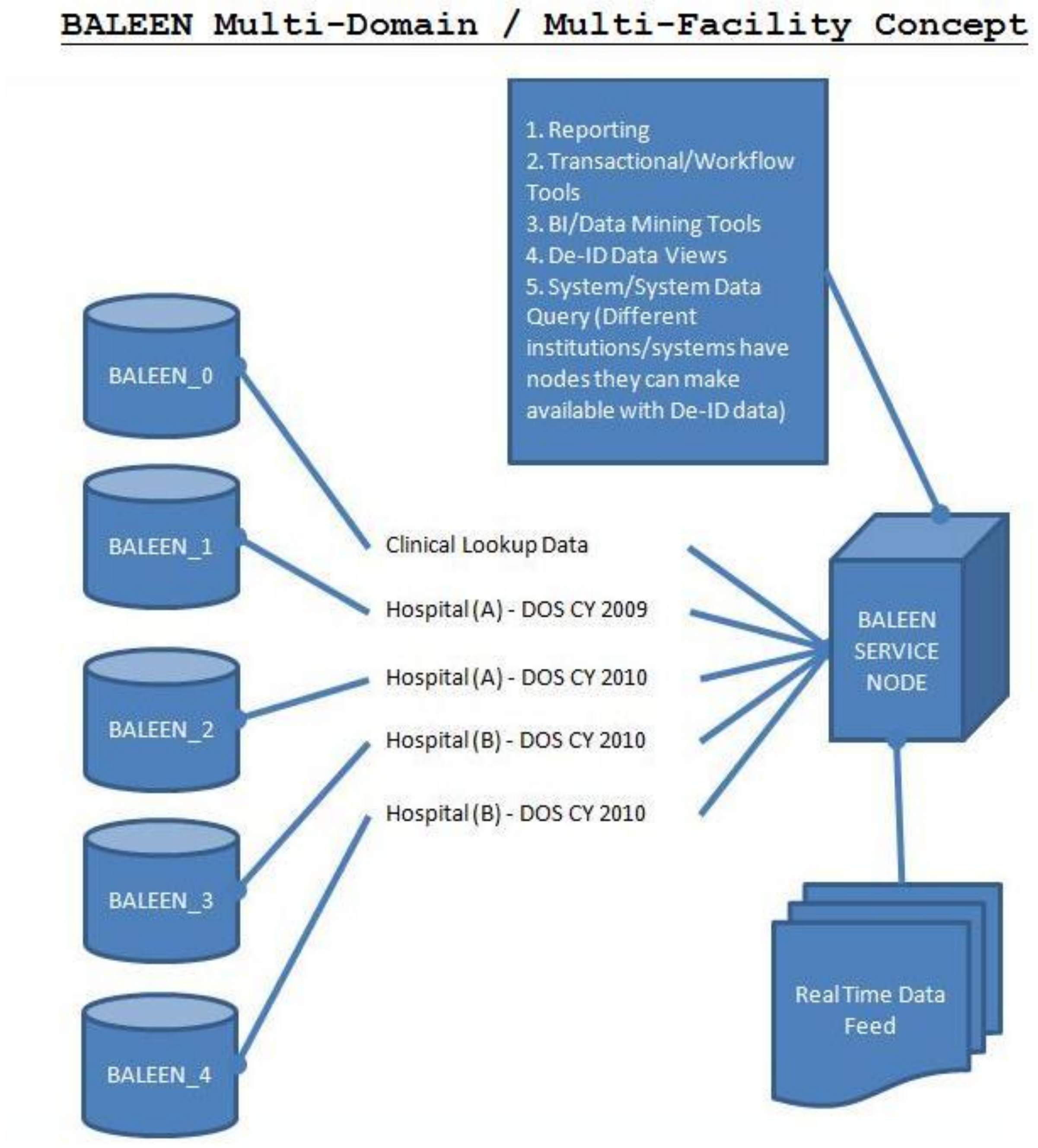
1. **Maximum fill:** a data warehouse MUST be designed to reduce (to a minimum) the number of empty 'holes' given some universe of 'pigeons'.
2. **Finite:** to the extent possible, it only stores unique facts - if necessary it will keep track (via counting) of unique facts.
3. **Compact:** it does not store repeating or duplicating data and where possible will slice these repeating sequences off from a parent and create a new child table/entity. This can be done automatically based upon the entropy state of the table itself.
4. **Targeted:** This is tricky, but basically we want to reduce the number of 'moves' a fact makes to an absolute minimum while moving to its final destination in the warehouse -- and then reduce how much churn happens after that. This means we reduce file i/o, network traffic and other 'general' resource allocations so that other processes can more effectively share these resources too.
5. **Frugal:** Related to targeted. A good DW will only use the resources from the CPU it must and will not store or spawn resources that are not necessary to the function of the DW. A good data warehouse will only save a unique object once (related to compactness and maximum fill)
6. **Decentralized/Federated:** a good data warehouse is naturally federated into smaller db/server deployments. For example, a hospital system might have 100 facilities, you can use this information plus a temporal feature (like date of service) to organize your DW into smaller cells or units. This will also make your analysts happy, because their queries will run faster.

7. **Rational:** does not create dimensions based on field names, but rather on the nature of the data - those characteristics that make it open to heterogeneous data sources. It recognizes that the value space is a slowly changing dimension - in terms of time and space. The value space is increasing at a decreasing rate.
8. **Focused:** not everything belongs in a data warehouse, there are MANY algorithmic operations which would be better implemented outside the scope of the DW, in another process.
9. **Friendly:** design a system which makes the life of the interface/report (both input and output) designer easier - not more difficult
10. **GO Like, Not Chess Like:** while the value space may be relatively finite, the relational space is potentially infinite - especially if we allow for multiple connectedness (or more than one arc between nodes). The DW should therefore allow for open-ended generative growth through flexible version and not restrict itself to a confining solution space. Chess designed databases work, but only in finite and specific domains.
11. **Testable/Verifiable/Audit-able:** a well designed data warehouse is open to inspection, audit ready (given an appropriate amount of resources), testable (both in terms of ongoing, but more importantly at the start to validate design -- you should ALWAYS validate design!) By being open a data warehouse is allowed to evolve, by having simple open-ended rules, it can evolve flexibly -- because the world is always changing. Data is not perfect, but you can have an accurate representation of the imperfection.
12. **Fast:** it is our belief, that if you implement the above, you will achieve speed. But, there is something important about calling out this feature as being not simply 'important' but probably the MOST important feature.
13. **Accepting:** a good data warehouse will attempt to store data 'as is' without obliterating or molesting or messing with the essential form of the source data -- ETL is butchery!
14. **Don't re-invent the wheel:** if there are features/behaviors that the OS or other systems are ALREADY doing, don't duplicate it if it works -- integrate with it.
15. **Safe Navigation:** Allow for SAFE high level investigation, but prevent drilling into sensitive/PHI/HIPAA protected data. De-ID should be seen as a process of hiding and showing sub-trees within the super tree of the ontology.
16. **Fluid:** The distinction between FACTS and DIMENSIONS is somewhat arbitrary and context sensitive. As with living language, living data must accept this and learn to manage this. Put another way: one person's fact is another person's dimension.
17. **Coalesce versus Conform:** Data should be allowed multiple separate interpretations (dimensions/types), and allow the report/view designer to choose the appropriate one.

Baleen Data Warehouse Layers

1. First Layer: **Federation -- Storage of Knowledge**; Break Databases into DOMAIN as <<Facility/Start DOS Year/Start DOS Month>>
2. Second Layer: **Ontology -- Theory of Knowledge**
3. Third Layer: **De-ID -- Safe Access To Knowledge**
4. Fourth Layer: **Facts -- Origination of Knowledge**

Baleen Distributed Domain Model


BALEEN Multi-Domain / Multi-Facility Concept
1. Reporting
2. Transactional/Workflow Tools
3. BI/Data Mining Tools
4. De-ID Data Views
5. System/System Data Query (Different institutions/systems have nodes they can make available with De-ID data)
BALEEN_0
BALEEN_1
BALEEN_2
BALEEN_3
BALEEN_4
Clinical Lookup Data
Hospital (A) - DOS CY 2009
Hospital (A) - DOS CY 2010
Hospital (B) - DOS CY 2010
Hospital (B) - DOS CY 2010
BALEEN SERVICE NODE
Real Time Data Feed

Who Are We?

1. **Daniel John Sullivan** – Medical and Health Informatics Professional with several years experience with both the warehousing and the analysis of healthcare and insurance financial data.
2. **Wynn Burke** – Medical Informatics Professional currently employed by the University of Washington.
3. **Any Questions?** – Please contact Daniel Sullivan @ 206-658-5965 or danielsullivan2002@comcast.net