Z2 Computer Solutions
Wayne L. Atchison
(303)-999-0701
email: Wayne@Z2cs.com
web: www.Z2cs.com

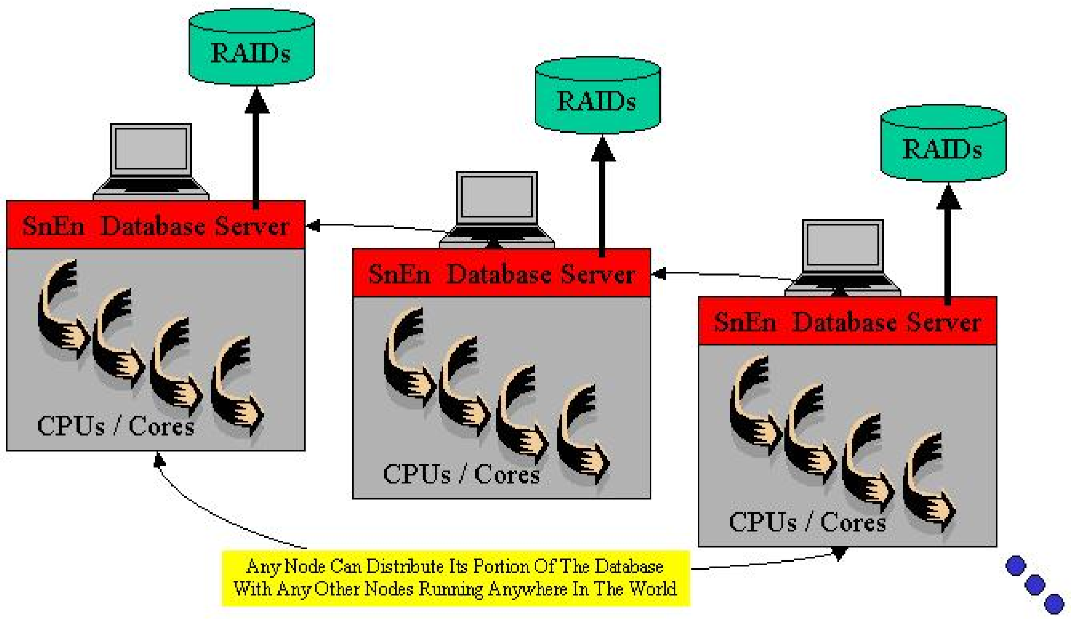
Oracle and the Snippet Engine
Side By Side Comparison
Written March 11, 2008
Edited April 14, 2008
Printed 3/10/2009

Side By Side Comparison Between Oracle and the Snippet Engine

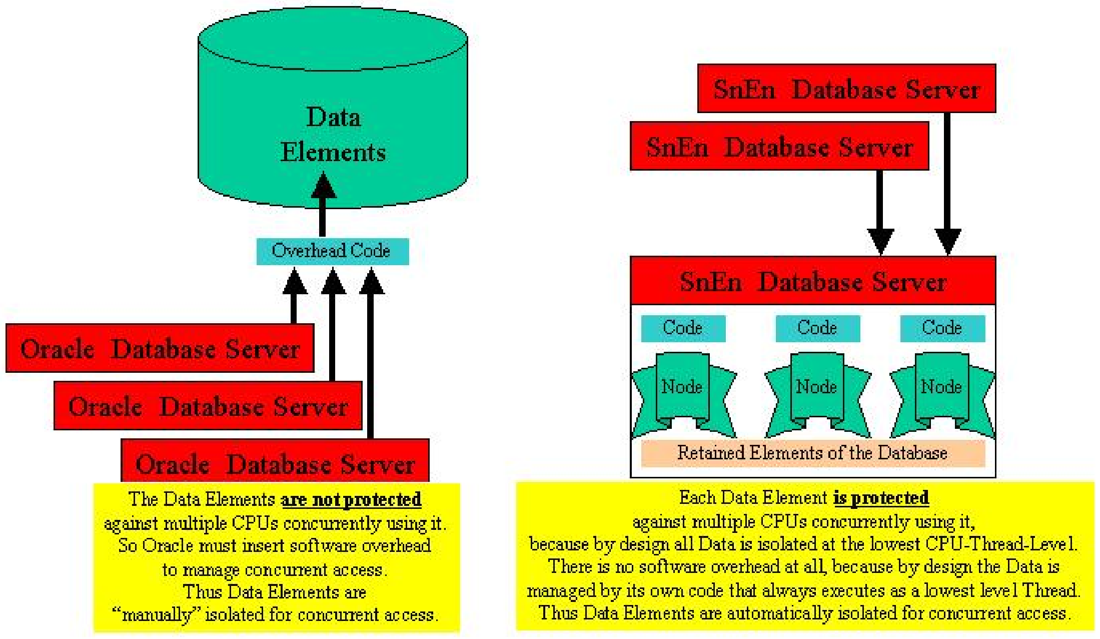
Oracle 9i / 10g Database Server	**Snippet Engine Database Server**
1.) The primary focus of Oracle is to provide a general-purpose Relational Database solution suitable to most applications. The Oracle Database is not the project's application, it is the repository of its data.	1.) The primary focus of the Snippet Engine (**SnEn**) is to provide a very specific application and Database Server solution, and is not inherently a general-purpose Database Server. A SnEn Database is both the project's application and its repository of its data, the application is not separate from the Database, they are one and the same.
2.) The design structure is that the software code associated with being the application, all the code required to provide the intelligence of the application itself, the PHP, Java Script, or ASP code executing within the HTTP Host Server process, this code is executed separately from the Database Server.	2.) The design structure is that the software code typically associated with the application, all the code required to provide the intelligence of the application itself, the PHP, Java Script, or ASP code that Oracle projects would require, most of this code no longer exists, but is now executing as 'C++' code outside of the HTTP Host Server process in the SnEn.
3.) It is the project specific PHP, Java Script, or ASP code executing in the HTTP Host Server process which is the primary provider of the application's intelligence. It is the Stored Procedures stored within the Oracle Database and this software executing within the HTTP Host Server process which provides the intelligence of the system, otherwise the Oracle Database Server is only housing data.	3.) The SnEn Database Server is not just housing data. The SnEn is the primary provider of the application's intelligence too. What little PHP, Java Script, or ASP code that is left to be developed in the HTTP Host Server is primarily there to interface with the SnEn. It is the SnEn which provides both the intelligence of the system and houses the data.
4.) Of course the PHP, Java Script, or ASP code required to manage the Client-side interaction is also part of the HTTP Host Server process.	4.) Of course the PHP, Java Script, or ASP code required to manage the Client-side interaction is still in the HTTP Host Server process.

Comparing the Oracle Database Server to the SnEn Database Server on a capability-by-capability basis is problematic, because they do not share the same philosophy nor target the same audience. Oracle is a general-purpose high powered Transaction oriented SQL Relational-Database, while the SnEn is not. An Oracle Database can be distributed.	The SnEn is designed for creating a Distributed Database having any level of failsafe redundancy. The SnEn methodology follows the "same-form-building-blocks" model. A "block" is a single Web Hosting Server with the SnEn and a RAID-like disk array. To create a project with a database that is distributed anywhere in the world you just add more "blocks". Each "block" adds more capacity, greater failsafe redundancy, load-balancing, and greater performance. As a collection of "blocks" the SnEn is a specific-purpose high powered Command oriented XML Distributed-Database, while Oracle is not. The SnEn allows the Designers to create whatever Database capabilities they require to satisfy the project goals. **To compare the two Database Servers on a capability-by-capability basis, only the mechanism that could be used by the SnEn Developers to accomplish an equivalent capability will be described.**

"Same Form Building Blocks" Distributed Database



Isolation Of Data Elements


Data Elements
Overhead Code
Oracle Database Server
Oracle Database Server
Oracle Database Server
The Data Elements **are not protected** against multiple CPUs concurrently using it. So Oracle must insert software overhead to manage concurrent access. Thus Data Elements are "manually" isolated for concurrent access.
SnEn Database Server
SnEn Database Server
SnEn Database Server
Code
Code
Code
Node
Node
Node
Retained Elements of the Database
Each Data Element **is protected** against multiple CPUs concurrently using it, because by design all Data is isolated at the lowest CPU-Thread-Level. There is no software overhead at all, because by design the Data is managed by its own code that always executes as a lowest level Thread. Thus Data Elements are automatically isolated for concurrent access.

The PHP, Java Script, or ASP code executing in the HTTP Host Server process, makes external Tcp connections to the Oracle Database Server. <u>There is no reason why a project could not use both Oracle and the SnEn.</u> The typical interface format between the Host Server code and the Oracle Database Server is SQL. The primary reason for these connections is to store and retrieve its own data so that the Host code can process the fetched data and figure out what to do in order to provide the project's intelligence.	The PHP, Java Script, or ASP code executing in the HTTP Host Server process, makes external Tcp connections to the SnEn Database Serve, just as with the Oracle Database Server. <u>There is no reason why a project could not use both the SnEn and Oracle.</u> The typical interface format between the Host Server code and the SnEn Database Server is XML (or any variation). The primary reason for these connections is tell the SnEn code what the Client-side user just did, so that the SnEn code can figure out what to do in order to provide the project's intelligence.
The interface is in the form of commands telling the Oracle Database Server how to store or fetch the project's data, not how to process and utilize the data to effect the	The interface is in the form of commands telling the SnEn Database Server what the Client-side just did, so that the SnEn software will process and utilize its own

project's intelligence. The amount of Host Server code development that is required to do this is significant, it represents most of the development effort. For example, if the Client wants to fetch all of the data required to build a house, then the Host Server code would have to: ===== make multiple Tcp connections to send multiple SQL commands to the Oracle Database Server, to fetch the data required item by item, as it, within the Host Server process, figures out the full construction of the house, and then sends back the resulting XML showing the constructed house in the Client's Browser. ===== Notice that the Host Server platform is burdened by all of the processing time needed to do all this, it is therefore not free to handle as many other HTTP requests.	data to effect the project's intelligence. The SnEn figures out what to do next, and provides the XML required to send back to the Client-side Browser. The amount of Host Server code that is needed to be this interface between the Client-side and the SnEn is minimal. The software development of the SnEn represents most of the development effort. For example, if the Client wants to fetch all of the data required to build a house, then the Host Server code would have to: ===== make just one Tcp connection to send just one XML command to the SnEn Database Server, which will internally within itself fetch the data required as each self-aware self-managed SnEn Node knows how to provide its subset of the solution, as it, externally of the Host Server process, figures out the full construction of the house, and then sends back the resulting XML showing the constructed house in the Client's Browser. ===== Notice that the Host Server process is not burdened by the processing time needed to do all this, it is therefore free to handle other HTTP requests.
The Oracle Database is primarily designed to store Tables of data in a record-field-Relational manner.	The SnEn Database can be designed as, but is not necessarily, a Relational-Database too. The SnEn allows the Designers to create any kind of Database they require. The SnEn Database may be any combination of any kind of data storage methods. For example all of these may be in the same SnEn Database: a. Traditional Relational data Tables like Oracle provides, fixed length records

	having fixed field definitions consistent between each record. b. Inconsistent Relational data Tables, variable length records having dissimilar field definitions inconsistent between each record. c. Object-Lists, variable length data-objects. d. Index Sequential, sorted list of indexes pointing to large partitions or entities of other kinds of data. e. Btree++, binary-tree-oriented lists of indexes pointing to large partitions or entities of other kinds of data. No matter what kind of data storage method is needed, a SnEn Node is created which knows how to do it.
The Oracle Database Server provides very powerful and very flexible queries, so that it may fetch nearly anything that was stored, and have it returned in almost any format required. For example, if the Client-side Browser needs a Table filled out with interrelated dynamic data, the PHP, Java Script, or ASP code executing in the HTTP Host Server process will make multiple connections to the Oracle Database, and within the Host Server process itself figure out the Table's contents based on the user's security settings, and return the XML required.	The SnEn Database Server typically does not provide for flexible queries, such capabilities would have to be developed as part of the project. But from another point of view there is no reason to do so, as the Host Server code is not the code which processes the data. Both the project's data and the software logic required to process the data are both contained in the same EXE-process. The SnEn 'C++' code will perform whatever internal-queries between its Nodes that is required to fully process the commands coming from the Client-side Browser. For example, if the Client-side Browser needs a Table filled out with interrelated dynamic data, the SnEn will get a single command, and executing on its own platform will internally interact between its own Nodes, figure out the Table's contents based on the user's security settings, and return the XML required.
The Oracle Database Server can be used "as is" to provide its capabilities to many diverse kinds of applications.	The SnEn Database Server can be used "as is" to provide its capabilities to many diverse kinds of applications. But this is more of a foundation for creating specific

	project solutions, not as a general purpose server.
The project's intelligence must be provided in Stored Procedures and PHP, Java Script, or ASP code executing within the HTTP Host Server process environment.	The project's intelligence must be provided in SnEn Nodes, typically coded in 'C++', executing externally to the HTTP Host Server process environment. However, the result of this development is the creation of SnEn Nodes, and Nodes are typically immediately reusable in future projects. Building a library of SnEn Nodes will over time dramatically reduce the amount of software-development needed in future projects. For example, once a Node is created which knows how to talk to a weather satellite, it can be immediately reused in the next weather project.
The Oracle Database Architectural Philosophy is much different than is the SnEn Philosophy. <u>The Oracle philosophy is focused on using traditional OOD capabilities as it processes "Transactions"</u>:	The SnEn Database Architectural Philosophy is much different than is the Oracle Philosophy. <u>The SnEn philosophy takes advantage of managing a database around the SnEn's OOD capabilities, and utilizing verb-object Nodes as it processes commands</u>:
Oracle is designed around the idea of having a single large central farm of disks;	The SnEn is designed around the idea of having a single mirrored RAID-platform of disks;
managed by a number of Database Server Platforms (the Oracle software);	managed by a single Database Server Platform (the SnEn software);
which are surrounded by an army of application Server Platforms (HTTP Servers);	which are surrounded by only one to ten application Server Platforms (HTTP Servers);
which are making thousands of Tcp connections to the many Database Server Platforms.	which are making only around 1024 Tcp connections to the single Database Server Platform.
The typical design is an army of HTTP Web Hosting Servers (computers)	The typical design is one HTTP Web Hosting Server (computer) connecting to

surrounding a large number of Database Servers (computers) managing a single large disk-farm of data.	one Database Server (computer) managing a single RAID disk-farm. And then this base-configuration is replicated any number of times to manage a large distributed network of Database Servers.
The way that the Oracle Database scales upward to meet growing demand is to add more CPUs and more HTTP Web Hosting Servers and place load-balancing mechanisms in front. As Internet traffic increases the load-balancing evenly distributes the load across the many HTTP Web Hosting Servers.	**The way that the SnEn Database scales upward** to meet growing demand is to add more CPUs and more base-configurations, and place load-balancing mechanisms in front. As Internet traffic increases the load-balancing evenly distributes the load across the many HTTP Web Hosting Servers.
The Oracle Database can be implemented as a Distributed Database Architecture using “Distributed SQL”, which allows the terabytes of data to be programmed to be disbursed among the multiple Oracle Database Server Platforms.	The SnEn by design creates a Distributed Database Architecture, as the terabytes of data are easily disbursed among the multiple SnEn Database Server Platforms. The ease of data disbursement is because any Node can talk to any other Node anywhere in the world. A Node designed to manage a specific portion of the database isolates that data portion to be managed at the CPU-thread-level. Nodes which manage data portions involving multiple terabytes of data, are designed to split the data-volume across any number of other sister-Nodes which are running anywhere else in the world. Such an arrangement between self-aware self-managing Nodes can also include any level of data-redundancy and load-versus-volume balancing between them.
The PHP, Java Script, or ASP programmer is provided “Location Transparency”, as he does not care where the data Table is located physically, the “Distributed SQL” command will know how to find it.	The application programmer is provided “Location Transparency”, as he does not care where the data is located physically, the SnEn Node managing that portion of the database knows how to find it.
Each piece of this architecture can be stationed anywhere in the world, so that a robust worldwide distributed database can be developed.	Each piece of this architecture can be stationed anywhere in the world, so that a robust worldwide distributed database can be developed.

The Oracle Database Server is immediately available on most Platforms	The SnEn Database Server is only available on a Windows Platform. However, the SnEn core code is mostly in Berkley Standard format with the Windows specific code isolated. Thus the SnEn can be easily ported to other Platforms.
Has a Multi-Version Consistency Model providing developers with only one option: “Writes don’t block Readers and Readers don’t block Writers”.	Has a Multi-Version Consistency Model providing developers with two options: “Writes don’t block Readers and Readers don’t block Writers” (just as Oracle). “Readers and Writers are blocked until done”.
Multi-Version Consistency Model is accomplished by providing readers with another copy of the data-image while the real data is in the write-commit-procedure.	Multi-Version Consistency Model is accomplished by providing two types of writing, “Write-Concurrently” and “Write-Exclusively”, at any level of the data, from a Node to a data-record to a data-field. “Write-Concurrently” allows both Readers and Writers to concurrently use the same larger-area of real-data without blocking. Data integrity is preserved because the Designers organized the data area to logically prevent concurrent read/write of the same fields. For example, hundreds of terminals are concurrently reading/writing to the same List of airline reservations, but the design prevents any one of them concurrently reading/writing to the exact same reservation. “Write-Exclusively” blocks both Readers and Writers until something is finished. When the design itself cannot prevent concurrent read/write of the same real-data, or when everyone else needs to wait until something else is finish first, then “Write-Exclusively” is used.

This means that Readers see old data, maybe even wrong data, while crucial Writes are being made.	This means that Readers are held off until crucial Writes are finished. They will only see good data.
"Oracle Real Application Clusters" or "Cache Fusion": allows two or more Tcp-connection-instances to be concurrently Reading/Writing the exact same data. Oracle Database Server automatically handles this in RAM without either application knowing about the other. This means can add more HTTP Web Host Servers into the same cluster without changing code.	This is the "Multi-Version Consistency Model", looking at it from a different point of view. The SnEn also does this: automatically handles this in RAM without either application knowing about the other. This means can add more HTTP Web Host Servers into the same cluster without changing code.
The Write-Commit-Procedure and Roll-Back is automatic, and therefore cannot be optimized by project or circumstance.	The Write-Commit-Procedure and Roll-Back is under the Designer's control, as it is part of a Node's interaction with other Nodes. Therefore it can be optimized by project and circumstance. Because Nodes are command-based self-aware self-managed entities which inter-relate with other Nodes, when and how to commit its own data is a design decision. The Designers decide when and how the changes to real-data are committed to disk, and how to roll it back if something fails.
Data Tables are the fundamental construct within an Oracle Database	Data Tables are a design decision and thereby are an optional construct within a SnEn Database. A SnEn Database is the aggregate of, and the interaction between, all of its Nodes. Each Node may have any type of data-storage-construct, including dissimilar blogs, objects, images, as well as typical relational data Tables.
Database Administrator has direct control over Sorting Tables	The SnEn defines a coding-construct called the "Ordered List Construct" (**OLC**). This construct allows SnEn Nodes to have their own Sorted-Lists of Variable-Length-Records. These constructs may be dynamically created in RAM to effect the resorting of data into any format of Tables.

Database Administrator has control over Cache Memory Allocation, settable at 8KB Blocks and 64KB Pages	The SnEn manages its own memory cache having a fixed size. Memory is managed in blocks which may be any size in multiples of 2 above 64 bytes per block. The Database Administrator can only change this by recompiling the software and recreating the Database from its last Check Point File. This limitation is offset because of the performance gain obtained by managing its own memory cache. The SnEn manages its own RAM in a manner which completely avoids the O.S. kernel. Looking at the CPU's internal cycles, avoiding "hitting" the kernel saves over 600 cpu-cycles per RAM access. Since RAM is accessed constantly, this represents a major performance gain.
Large Tables can be partitioned into Range Partitions with Range Indexes for faster performance	Nodes may employ any combination of OLC, or Index Sequential, or Btree++ or any other indexing methods to effect the same performance capabilities
Public and Private Synonyms for data Schemas	The SnEn allows Schema definitions which may be used by any Node. Public and Private Schemas is a matter of what the Designers allow. Schemas may also be used to define security settings on a field-by-field basis, and for defining report formats.
Allows "Independent Sequence Objects" of code	Everything is an "Independent Sequence Object". All Nodes are executed asynchronously and autonomously, and may interact with any other Node.
Allows "Collections of Procedures and Functions" using the language PL-SQL stored within the Database. Stored Procedures are pre-compiled.	Everything is a "Collections of Procedures and Functions". All Nodes are nothing but a collection of pre-compiled functions using the 'C++' language.
Allows "Event Triggers": "Select", "Insert", "Update", "Delete", and "DDL Events". Can be set to fire "Before" or "After" on a transaction's single row or else all of its rows.	Allows "Event Triggers" of any kind at any time, as any Node may trigger any other Node running anywhere in the world both "Before" or "After" any type of Event at any level of the data. Further, each OLC record may optionally specify a Node which knows how to

	manage the "ripples" associated with changes made to that record. This type of "Triggering" is automatically handled by the SnEn core, the programmer does not have to do anything to make this happen.
Provides "Dead Connection Detection"	Also provides "Dead Connection Detection". The SnEn core gracefully handles a connection when it fails. Nodes using dead connections may react any way they decide, for example "Die" or "Reconnect".
Transaction Processing: each transaction is written to a "Redo Log Buffer", then advanced to "Redo Logs" (disk) and then advanced to the "Data File Buffer" (RAM). "Database Reconstruction": after a Software or Hardware Failure is accomplished using "Distributed Transaction Recovery" from the Archive Logs which are Check Points of the most recently updated (committed) memory blocks. The database may be reconstructed from these files.	The SnEn allows the Designers to implement their own Transaction Processing mechanism as they require. The SnEn is not necessarily "Transaction-Oriented". It is better to describe it as "Command Oriented", where a command might require numerous internal Node-to-Node Transactions. Only the project Designers will know at what level and at what stage in a command-process they want to make Transaction Log entries. The database may be re-constructed from such Transaction Log files, if that is what the project Designers decide to do. The SnEn core periodically creates Check Point Files as specified by the Designers. Each Check Point File is a simple text representation of the entire Database. The Database can be reconstructed from these Check Point Files.
Backup Procedure: the Database's internal data files may be backed up even while online.	The Check Point Files are created while online. Such files may be backed up or mirrored even while online. While online the SnEn Database itself is also a disk file, and this file, may be mirrored or replicated even while online.
Recovery Procedure: restore the data files, verify control files are synchronized, which may be accomplished by an automated process.	Recovery Procedure: only one task needs to be performed, and may be automated. That is one of the Check Point Files needs to be moved into the working local

	directory and the SnEn-Application restarted. The Database will automatically be recreated from this Check Point File.
Database Security: Administrator defined User Logins have adjustable security settings allowing and preventing access to specific data	The SnEn allows the Administrator to define security settings for each User Login. Further, security settings may prevent some users from access to certain sections of code, access to some Nodes, or access to some capabilities.
The typical Oracle Database will have **28 internal-files** spread across various disks within the cluster's subsystems	The typical SnEn Database will have **5 files** plus the Check Point Files spread across any disk subsystems designed
Oracle uses its own advanced file-system, the "Oracle File System", giving much better performance than the conventional file systems	The SnEn uses the provided file system of the Operating System. However, it manages its own memory cache in a manner which completely avoids depending upon the file system, that is, the file system does not effect the performance of the SnEn.
Storage Concepts: Tables are stored in Table Spaces, which contain one or more data files, which may be Partitioned with Indexes	Each Node is its own data repository, and may employ any Storage Concept required. Data Tables are typically stored as an OLC within a Node. There is no limit to the number of Nodes, nor the logical size of an OLC. Each Node may interface with other sister-Nodes to effect any level of data disbursement, redundancy, and load-balancing, using any type of indexing into "Partitions".
Fail Over and Availability: is very high when using Server-Clusters and RAIDs	The SnEn allows the Database to be disbursed across any number of Server-Clusters and RAIDs located anywhere in the world. Therefore Clusters and RAIDs have the same advantage.
Exporting Data requires writing code and must be in SQL Plus	The Designers define how data is exported, then it is implemented as 'C++' code within a Node.
Each User connection requires about **1MB** of the Oracle Database Server's memory	Each User connection requires about **50KB** of the SnEn Database Server's memory
Oracle Database Server's RAM Memory used for Caching is allocated dynamically and does change	SnEn Database Server's Memory used for Caching is allocated statically during startup, and does not change
Supports about 15,000 active User Connections	The SnEn is limited only by the O.S. and platform configuration

When Would You Use The Snippet Engine Instead Of Oracle

1. To create a very specific application and Database Server solution:

- Some projects require a proprietary solution. Some projects do not want a traditional Database. The SnEn creates proprietary Database Servers which are singularly unique to each project. This uniqueness significantly increases the security of the Database. Outsiders would not have any idea how to access the data residing within SnEn Nodes. Their hacker and SQL skills will not give them any opportunity to see the data. And the internal staff would have to be SnEn Programmers to even know what to do.

- Everything from the network's hardware design to the software implementation, everything from the level of performance to the level of failsafe redundancy, all aspects of the hardware and software are under the project's direct control.

  The SnEn does not impose any specific hardware or software architecture. For example the SnEn does not impose "Server clusters", nor does it impose buying extra software licenses to create a Distributed Database.

2. To create a better environment for development and future enhancements:

The burden of software development does not go away. For each project requirement there will have to be some code created somewhere by somebody to make it happen. Whether the coding be done in PHP, Java Script, ASP, or 'C++', there is software to be developed, and software which must be changed for future enhancements.

- The intelligence of the project's overall application, which is usually manifested by the software developed as PHP, Java Script, or ASP code executing within the HTTP Host Server process, most of this code is now no longer needed. Instead the application's intelligence is developed in 'C++' that runs outside of the Host Server process.
  - Most Client-Browser requests are performed outside of the Host Server process' "Managed Code" Environment. Just as Oracle's core software is written as "Unmanaged Code", so is the SnEn written as "Unmanaged Code". Why? "Unmanaged Code" provides far better performance. By moving all of the software development of the project's intelligence outside of the slower "Managed Code" environment into the faster "Unmanaged Code" environment, the net result is far better overall performance than with "Managed Code" solution.

- This means that the SnEn will figure out the Client requests in a fraction of the CPU-cycles compared to the number it would take in PHP, Java Script, or ASP. The SnEn code does not have any of the restrictions or overhead that is placed on PHP, Java Script, or ASP "Managed Code".
- With the SnEn the interfacing PHP, Java Script, or ASP code only makes one Tcp connection per Client request to the SnEn Database Server. The Oracle-solution requires multiple connections, one for each data Table query and save operation. Further, the ASP code does not have to figure out the XML either, as the SnEn returns the answer as XML that is ready to upload to the Browser.
- Very complicated and time-consuming Client-Browser requests are performed outside of the HTTP Host Server process.
    - The User does not have to wait, he can keep working. The ASP code can tell the Client-Browser to keep working while the SnEn asynchronously figures out a time-consuming request. When the SnEn is finished the Client-Browser is updated.
    - Client requests that require interfacing to external devices, such as temperature sensors, and synchronizing with external computers, such as Web Host Services, to figure out the answer can be performed asynchronously "in the background" by the SnEn. For example <u>a Distributed Database is by definition an environment whereby Client requests cannot be satisfied without synchronizing with other external computers</u>. Such external communications may take long periods of time to accomplish. The SnEn handles unexpected delays and inconsistent external communications "automatically", it is the designed way Nodes work together.
    - It is much easier to develop complicated and interrelated capabilities in 'C++' than in PHP, Java Script, or ASP. The advantages of .NET services and capabilities, as well as others, are just as available in 'C++' as they are in ASP. There is no reason not to use C++.
- The Host Server Process is now freed up and can focus on handling more HTTP requests from Client-Browsers.
    - Because the overall number of CPU-cycles required to figure out Client requests is dramatically reduced, the Host Server platform can now handle many more Client-Browser requests than with an Oracle-solution.

<u>3. The project requires more than just simply fetching and displaying data:</u>

- If all that is required is to fetch data rows and columns from the Database and simply producing the XML for the Browser to display it as is, then the amount of

PHP, Java Script, or ASP code developed may be minimal. But the more post-processing of the data that is required, the more advantageous it is to off load that complexity and processing time to an external process like the SnEn. The more data inter-dependencies that exist, the more complications there are, the more advantageous it is to manage the data using SnEn Nodes. Examples are:

- Complicated statistical analysis
- Complicated and dynamically changing rules
- Complicated and dynamically changing inter-dependencies
- Complicated and dynamically changing ramifications upon changing values
- Complicated and dynamically changing external investigations using external Web Host Services to figure out the answer
- Complicated and dynamically changing inter-dependant inventories
- Complicated and dynamically changing inter-dependant distributions
- Complicated and dynamically changing inter-dependant external device controlling (robotics)
- Complicated and dynamically changing inter-dependant gaming / war-room strategies

<u>4. The project requires sophisticated security on User access to the data:</u>

- If all that is required is to fetch data rows and columns from the Database and without regard as to who can see what simply show it in a Browser, then the amount of PHP, Java Script, or ASP code developed may be minimal. But the more sophisticated the security issues on the data and the more complicated that security is on which code sections can or must be executed, the more advantageous it is to off load that complexity and processing time to an external process like the SnEn. Using the SnEn:
    - Security settings are defined for each User Login, or anything else required.
    - Security settings are at any level required: field, record, List, Table, Node, even access to specific software capabilities.
    - Security settings include designating what a User can see, change, create, and delete.
    - Security settings are easily changed, all being defined in a simple text file.
    - Security access is employed by the SnEn so that the interfacing PHP, Java Script, or ASP code only gets back what the User can actually see and do next.

<u>5. The project requires the application to tell the User what to do next:</u>

- Most website interfaces follow the fundamental user-interface philosophy imposed by Microsoft Windows. That is to assume that the User knows what to do next and therefore provide the User with scores of option-icons and sub-menus. Menu driven in this philosophy means that the User is the intelligence, and will click to tell the application what to do next.

  Some projects cannot assume that the user knows what to do next. Some projects need to direct the user step by step without allowing diversion with scores of option-icons. In these projects the traditional Microsoft Windows philosophy cannot be used.

  In this scenario the SnEn provides significant advantages. Because the SnEn is both the application's intelligence and the project's Database, the SnEn knows how to figure out exactly what the user needs to do next, and thereby can drive the user-interface accordingly. The SnEn will provide only the option-icons and sub-menus needed for the user to accomplish the immediate next step. When complicated by complex database security issues, figuring out what can or must be done next may be quite involved. Figuring out these things is much better coded in C++.

  For example, when filling out a form: the user can be lead through the process one field at a time. The SnEn allows the Designers to implement a user-interface which does not assume that the User knows what to do next. Sometimes you must assume the user is very unsophisticated, and must be lead step by step to prevent errors and omissions. Menu driven in this philosophy means that the SnEn application is the intelligence, and will only show the User what needs to be done or decided next.

<u>6. The project requires that some portions of the database are locked out for long periods of time:</u>

- Some projects require a user to edit and monitor a portion of the database in real time. They are in essence locking out that portion of the database from anyone else while they are working with it. This editing and monitoring may take minutes or even hours before completed. It is often not practical to allow a portion of the database to be locked out for long periods.

  The SnEn provides a simple solution. Any portion of the database can be isolated so that a single Node is the data's only interface to a Client-Browser menu-scenario. Now you have a self-aware self-managing thread-level code section isolating access to this portion of the database. Additional Users can also access the exact same portion of data using their own instances of this same Node-type. Now access to this portion of the database is not locked out, as this Node-type knows how to internally interface with all the other instances of itself, and thereby allow multiple users to concurrently edit and monitor the same portion of data with real-time refresh to all. Such a design may include only allowing one User to be the "master" over one field at a time, or prevent certain edits while another is changing certain fields. Nodes are code sections, therefore each can be coded to make sophisticated decisions about who can do what to what and when.

7. The project requires significant post-processing when some data elements are changed:

- Some projects require significant post-processing when certain key values of the database are changed. The concept of "Triggers" are used to signal another process to manage the ramifications when data has changed.

  With Oracle such "Triggers" fire up "Stored Procedures" which are written in the language PL-SQL. This is not easy programming. If the change-ramifications "ripple" extensively to other distributed computers the impact on the Database Server's overall performance could be significant.

  With the SnEn "Triggers" fire up Nodes which are written in 'C++'. This is much easier programming If the change-ramifications "ripple" extensively to other distributed computers the impact on the Database Server's overall performance could be significant, but not as significant as with Oracle. Because Nodes execute asynchronously and in parallel, and because they can easily talk to other Nodes on the distributed computers, the impact of "rippling" will not take as long to accomplish.

When Would You Use Oracle Instead Of The Snippet Engine

1. When you want a typical Database Server solution:

- Some situations demand choosing the conventional approach. Legacy, the programming skills of the development team, contract restrictions, you do not want a proprietary solution, and management-conservatism may be overriding considerations.

2. When the project is primarily a Transaction-Oriented scenario:

- Oracle is specifically designed to be a high speed high volume Transaction-based Database Server. In projects that are primarily solved by processing Transactions and simply managing data Tables, it may not be advantageous to develop a SnEn solution.

3. When most of the project is easily satisfied by Oracle's capabilities:

- Not all projects require a significant software development effort to provide the intelligence. When the PHP, Java Script, or ASP code is relatively straight forward, and heavy reliance on the existing capabilities of Oracle is actually an advantage, then it may not be advantageous to develop a SnEn solution.