Real-time DBMS for Data Fusion

Dave McDaniel and Gregory Schaefer

Silver Bullet Solutions, Inc.

4747 Morena Blvd., Suite 350

San Diego, CA 92117

davem@silverbulletinc.com, gschaefer@silverbulletinc.com

Abstract – *As data and information fusion technology and application evolves, the need is increasing to cope with large amounts and diverse types of data. Although there would be many benefits to employment of Data Base Management Systems (DBMS) in automated fusion processes the data access throughput requirements of automated fusion processes have vastly exceeded the performance of off-the-shelf DBMS's. The availability of large random access memories is allowing the development of "real-time" data base management systems. While these are currently being used in financial market and telecommunications applications, their ability to bring DBMS benefits to data fusion applications has yet to be explored. This paper presents results of experimentation with these emergent "real-time" DBMS's for automated fusion applications. We used algorithms, data characteristics, and scenarios from deployed and R&D systems. The applications-dependent data structures were converted to Entity-Relationship models and generated into "real-time" and conventional DBMS's.*

Keywords: Data Fusion, DBMS, real-time, correlation, knowledgebase, embedded

1 Introduction

Data access demands can be very high for fusion processes, particularly those defined as Levels 1-3 fusion. For purposes of this discussion, "high-volume data access" means that the volume of data access is large enough that the data access time could cause the input transaction job or processing queue to exceed operational requirements and that special design consideration must be given to either special data access techniques (e.g., hashing) or managing the job queue (e.g., pruning).

At level 1, it is often necessary to access many association or correlation candidates for goodness-of-fit testing. In a system with 2000 track file capacity, hundreds may fall within an input track report in dense areas. This happens often because for applications such as air traffic control, tracks are clustered in airways and around cities and airports; they are not uniformly distributed across the surveillance area. Even in currently deployed systems, the design must account for thousands of accesses per second [1]. At level 1, it is also often necessary to access many archetypes and currently known instances for target identification processing. For example, in a theater-level system, hundreds archetypes and instances can be required to be accessed per second [2]. Level 2 and 3 fusion processes can also have high access demands as reference and track files are referenced to discern patterns that could lead to level 2 and 3 knowledge.

Because of the high-volume data access for these types of fusion processes, the data must be maintained in computer RAM in applications-dependent data structures and accessed with special hash and search algorithms. Early radar trackers used hash by target range and bearing [3]. One system binned archetypes by their parameter ranges and intersected the bins to lookup the applicable archetypes [4]. As the fusion systems evolve to higher level fusion and the need to input and reference more types of data in ever greater quantities, the use of Data Base Management Systems (DBMS) such as Oracle or Microsoft's SQL Server offers many benefits. However, there have been out of the question for fusion applications primarily because a single disk access can greatly exceed the entire timing budget for accessing all candidates. Note that this does not mean fusion systems do not employ DBMS's; many do. However their use is not in the Level 1-3 fusion processes as described herein; they are used to reference information for human analysis, as an augmentation to a display, and, sometimes, to read-in portions of data to RAM for use by fusion processes. Because of this performance limitation, fusion system developers must build their own data access and handling software. This paper describes why it would be better to use a full-featured DBMS and how new "real time" DBMS's have been recently developed that can support fusion processes such as those described in this paragraph. We also present results of experimentation with one commercially available

product provided to our lab for experimentation. We ran typical fusion data access software with this real-time DBMS and measured performance compared with the traditional custom data access routines. The results are presented in the final section.

2 "Real-time" DBMS

"Real-time" is often used synonymously with what might be called "fast-time", meaning, the processing is done quickly, or more precisely, within some allotted time period [5]. This differs from a more purist definition of real-time computing that would address determinism, that is, that ability of the computing system to respond assuredly at a specified time, to a specific level of precision less than a millisecond. For Levels 1-4 fusion there is rarely a requirement for this type of real-time; rather, 'fast-time' is usually what is required. An example of several-thousand track fusion systems running in non-real-time, or "general purpose", operating systems are the US Navy's Multi-input Tracking and Control System. It receives radar contact reports from 30 radars and track reports from whomever is participating in various tactical networks, tracking the contact reports in a adaptive tracker, correlating all the independent input sources, and estimating track kinematics. It runs in a multi-tasking operating system but not a real-time operating system by the deterministic criterion just described.

In many respects, real-time DBMS' often are fast-time DBMS' by the deterministic criterion. However, as described in paragraph 2 and the Navy example, fast-time is just what is needed for Level 1-3 fusion. So that we don't have to debate what truly constitutes "real-time", from this point forward we will use the term "embedded" to refer to the DBMS under experimentation and "offline" to refer to conventional DBMS that operate interactively with the disk drive for data access.

The features of an embedded DBMS are simply that, (a) data accesses do not require disk operations, (b) typical DBMS functions are available such as responsive to the Entity-Relationship model, access via SQL and extensions such as Procedural SQL or "record sets", background archiving and backup, and so on. In addition, it is beneficial if the SQL and other data operations software have been optimized for RAM operation rather than disk data access; ordinary DBMS' have been optimized for disk data access.

3 Benefits of Embedded DBMS's to Fusion Systems

DBMS's provide robust built-in features that can improve many fusion system performance attributes such as:

a. System Reliability. System crashes and hangs caused by data corruption can be reduced
b. Presentation Accuracy. Output data is consistent.
c. Object Fidelity. The objects of fusion can be treated with more accurate and complete object semantics.
d. Inference Execution Accuracy. Inference can operate with greater completeness and faithful object inter-relationships.
e. Backtracking and Auditing. Input, state, and output history can be maintained to a very large degree.
f. Inference Design Accuracy. Inference design can be made more logically coherent and complete.
g. Tractable Data Management. Reference, track file, sensor file, etc. can be managed in a systematic manner.

The features of DBMS's that enable these types of benefits are described in the following subparagraphs.

3.1 Structural Data Integrity

The phase "structural data integrity" addresses the accuracy of the database with respect to its design. This means the database should be self-consistent and data values should correspond to the database design. Data integrity problems cause fusion systems to crash, hang, output incorrect data, and confuse operators [7] [8]. Data integrity problems in fusion systems are common and sometimes crippling to operations. DBMS' enable improvements in data integrity compared to flat, unrelated, and unmanaged data structures [6] in several ways including:

a. Non-redundancy. DBMS' support relational normalization techniques such as third normal form that avoid redundant storage of values. By maintaining a value once and once only, there is no chance of inconsistency between the duplicate storing of the same value. The relational model and its implementing DBMS' achieve this by referring or pointing to the single value in its various contexts. Properly implemented, this would prevent one fusion system screen or output from disagreeing with another. Because DBMS' are not used currently, custom software has to be developed to address data inconsistencies on a case-by-case basis [7].
b. Referential integrity rules. Relational DBMS' implement rules that prevent fragmented objects. An object that must have certain components to be complete has its components

related with rules as to the cardinality and null values permitted. An extension of this feature is sometimes called cascaded deletes or triggers. These will cause a complete 'cleanup' of the object once an essential component is deleted. This prevents 'ghost tracks' [8] and software crashes caused when the program tries to access components of an incompletely deleted (or created) track that no longer exists [7].

c.   Data validation rules. These are rules as to the values that be assigned to object properties. They can be as simple as range-of-values and allowed discretes (sometimes called "domain values" or "enumeration types"). They can also be context dependent, imposing rules as to the types of values allowed given other values in existence. The former improve reliability because interfacing software can be assured of the values input. The latter improves object fidelity because illogical object property combinations are prevented.

3.2   Semantic Modeling

DBMS' implement semantically expressive and mathematically rigorous data modeling techniques such as the Entity-Relationship model. As described in [9], this technique supports the complex semantic modeling of a fusion domain in a manner sufficiently rigorous for automatic DBMS data base generation. Because almost all formal database design is now done using this technique and so there is a large body of developed models from which to draw, fusion systems could benefit. More importantly, employing this proven powerful technique in fusion system database design would accrue the same benefits it does for all the other systems for which it is applied: more complete and correct expression of the domains objects, their properties, and their inter-relationships.

3.3   Embedded Knowledgebase

Typical fusion system architectures treat the reference and knowledgebase as independent from the current situation estimate. This is contrary to human reasoning where the a-priori knowledge of an environment blends with the real-time observations. The segregated architecture manifests itself in real system incongruencies, necessitating special software to keep the knowledgebase and the track file in synchronization [7]. In current fusion systems, the knowledgebase is either non-real-time (i.e., resides on a disk drive) or is loaded into applications-specific data structures based on extraction from the complete set on the disk drive. The applications-specific data structure typically does not have the robustness of the

DBMS structure and involves significant effort to construct, operate, and maintain.

3.4   Class-Level Connectionist Structure

As reported in [9], an Entity-Relationship model, from which the DBMS gets its structure, could be used as a basis for class-level inference structure that would provide a framework for more-specific object-level inferences and promote consistency amongst inference rules.

3.5   Embedded – Offline Connection

A feature of some real-time DBMS's is a connection between the embedded DBMS and a mechanical DBMS. This allows less-referenced data (e.g., older track states and sensor reports) to fade out of the embedded DBMS and onto offline storage. This feature also allows for greater completeness of the fusion data without compromising performance.

4 Experiment Descriptions

In the interest of time and getting some early results to the community, only two types of experiments were conducted. However, they both employed the same type and volume of data access from actual fusion systems deployed either today or in R&D for possible future systems. Both experiments involved searching though the entire track file for candidates. Sequential searching has always been prohibitive, even in applications dependent structures [4, 10]. In both cases, techniques for associative memory [11] approximations are made, that is, where the objects of interest are referenced by their attributes instead of their primary identifiers.

The sample product with we experimented for this paper is TimesTen [12]. TimesTen has been successfully applied in financial and telecommunications applications but the manufacturer reports this is the first experimentation for data fusion known to them.

Ideally, the embedded DBMS would run under a real-time operating system such as Lynx, VxWorks, or Carrier Grade Linux. Since for these experiments we were interested in relative comparisons between the offline DBMS, embedded DBMS, and applications-dependent data structures, the experiments were conducted under non-real-time Linux.

4.1   Multi-Source Integrator (MSI) Correlation Candidates Retrieval

As mentioned previously, hashing by azimuth and range has been used in radar trackers since the 1970's. For theater-level fusion systems,

← -1024 nm   1024 nm →

Associative Memory
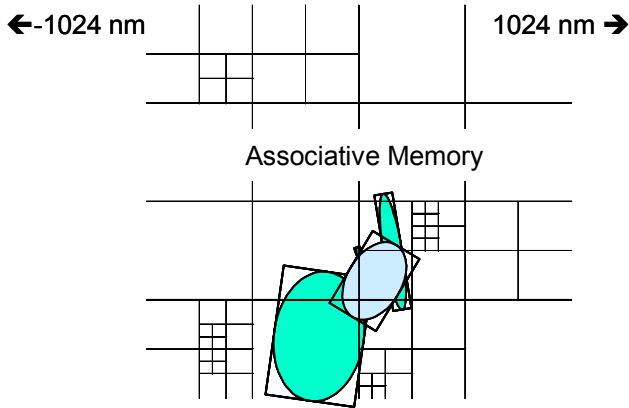
- Start out with 2048X2048 association cells
- 2 bytes per track number = 4 Mbyte AM
- Allow telescoping to 100 Mbyte

Figure 4-2. Multi Source Integration (MSI) Gross Gating

latitude and longitude hashing is more typical. Hashing is a type of associative memory approximation. An approximation is used because the number of attribute values that might be used for a 1,000 nm radius surveillance area would be 40 billion to have resolution cell size of 0.01 nm. To then associate with these with 2 byte track numbers would take a lot of RAM, the basic challenge of associative memory. Most of these would be unused since, realistically, targets are not nearly uniformly distributed over this area; they tend to be clumped

around population centers, airports, shipping and air lanes, seaports, and so on. This presents an opportunity for reducing the associative memory problem by conforming the number of resolution cells to the target density, a technique called "telescoping" [10]. In this technique, resolution cells are created and deleted depending on the number required to their use and need to inscribe a track's 2-D 2-σ ellipse. Criteria are established for creation (when an ellipse can't be adequately described) or deletion (when few tracks need that degree of resolution).

The associative memory returned the head of a link-list. The retrieval candidates were then processed through a record-by-record gate check. It is at this point the DBMS is invoked in the gating, by accessing what we call secondary information for gross-gating, the associative memory in-effect gating by primary data. Secondary data for gating in MSI was:

- state variables
- IFF Modes 1, 2, 3, 4
- Identity
- Category

Given this construct, the associative memory identifies the retrieval candidates in effect near-instantaneously. However, the associative memory is updated as each track is updated. Since a track may not be updated for a while and it is important that the associative memory always present a current estimate
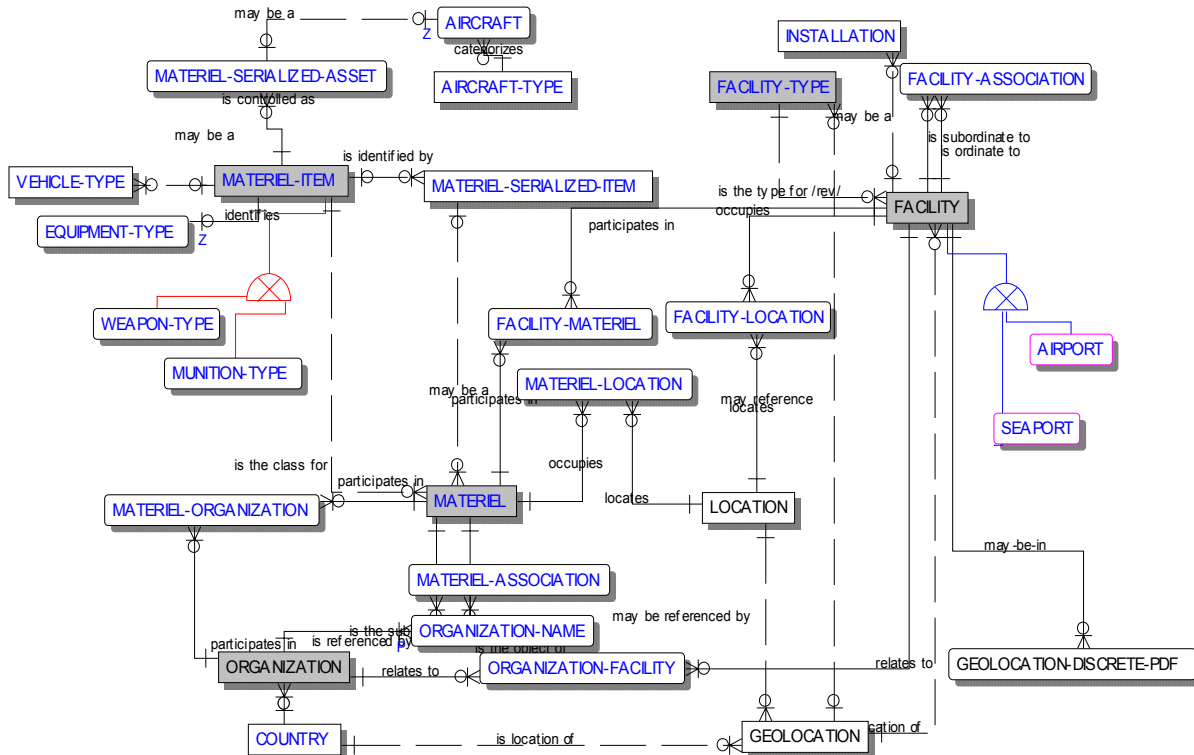

Figure 4-1. MSI Data Structure for Correlation Candidacy

of possibilities, the associative memory has to be refreshed periodically, to account for significant covariance propagations. This is done periodically, as a background task, scanning the track file to determine if the associative memory needs updating, that is, that the 2-D covariance is significantly out of date. The decision to propagate is when the 2-D covariance has eroded by 5% or approximately,

$$t \geq \sqrt[4]{\frac{4.41\left(\sigma_x^2 \sigma_y^2 - \sigma_{xy}\right)}{\sigma_a^2\left(\sigma_x^2 + \sigma_y^2 - \sigma_{xy}\right)}} \qquad (1)$$

where σ_a, the sd for acceleration in the process noise is a set of system constants selected based on track Category (Air, Surface, Subsurface, Land/Mobile, Space)

The performance drivers of the data manager are the retrieval (or access to) the candidates by the gross gate function and the maintenance of the associative memory. The number of retrieval candidates required varies, depending on the 2-D covariance of the input track report, the density of tracks about the input, and their 2-D covariances. The track sensor update rate characteristics and densities from the system's expected operating environment, the same ones used to determine the system's requirements and later to parameterize timing analyses. We used these to determine the

table sizes (number of instances), track input arrival rates, and numbers of candidates expected to derive from the associative memory.

The data structure was made equivalent to that for the deployed system but using a normalized E-R model as shown in Figure 4-1, as derived from the U.S. standard for command and control, the Command and Control Core (C2 Core) data model [13] as would be necessary to accrue the benefits described in paragraph 2, herein. Only the portions of the model that would be required for gross gating and associative memory maintenance are included since only those would be subject to high data rate retrieval.

4.2 ESM/ELINT Similar Source Integrator (SSI) Classification and Correlation Candidates Retrieval

In this test case, based on [2], multi-source ESM and ELINT reports are input for a theater-wide area. This experiment is based on an application developed for multi-source ESM and ELINT fusion against theater-level Electronic Order of Battle (EOB). The objective of the system is to correlate the input parameter reports against an EW reference file and the theater order-of-battle. Te theater-level OB provided the initial track file loads of all expected
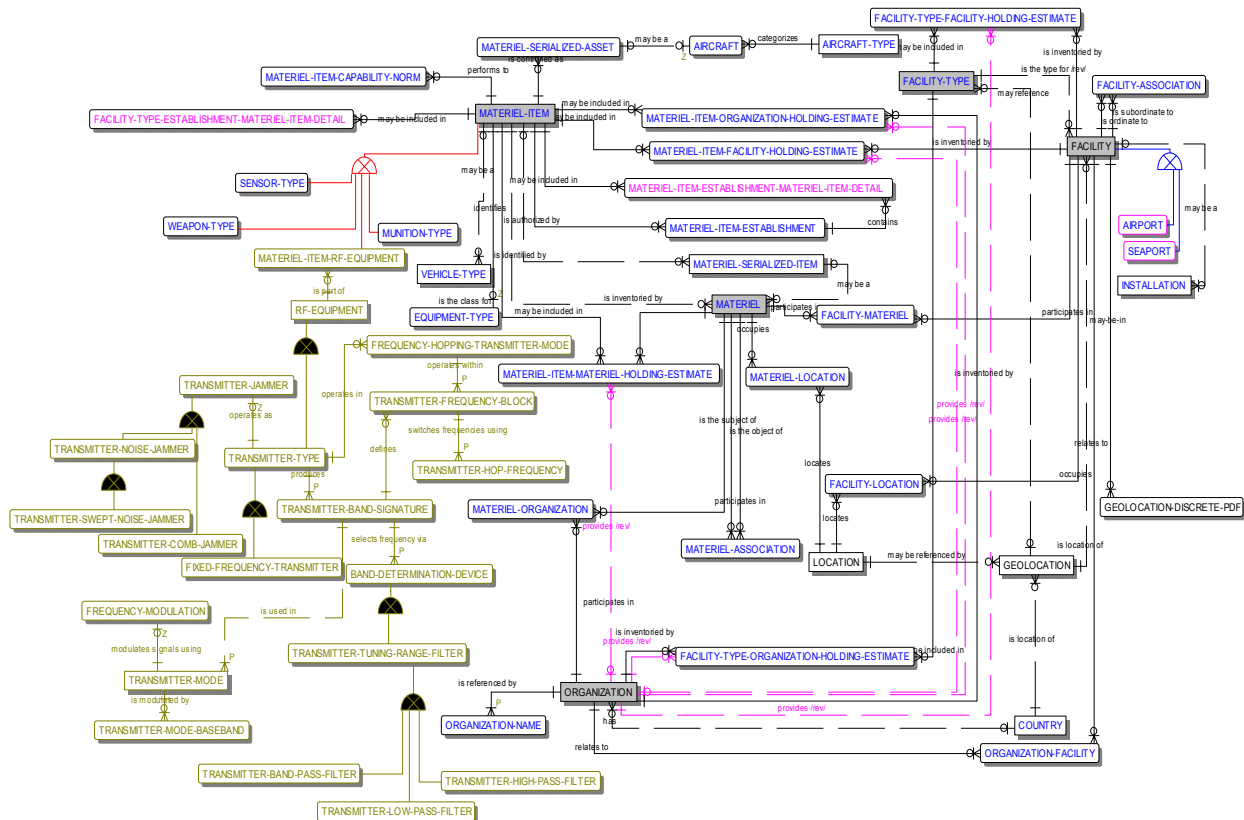

Figure 4-3. ELINT SSI Data Structure for Classification and Correlation Candidacy

but not currently observed tracks. As sensor data was input, the reports were correlated against this a-priori knowledge as well as tracks heretofore received. A multi-hypothesis structure was the basis for a Bayesian net. All significant hypotheses were maintained.

All data was maintained in RAM in applications dependent data structures. The arrival rates of ESM and ELINT reports were 5/sec. For each report, the initial lookup involves comparing the report to emitter reference file variables. The EW reference file has around 3,000 types of emitters. Once emitter hypotheses are formulated, then the order-of-battle candidates are retrieved. There are often 30,000 platform and facility candidates corresponding to:

a. Naval ship home ports / anchorages
b. Aircraft types at airbases and airfields
c. Air bases
d. Radar and SAM sites
e. Headquarters and other military facilities with emitters
f. Merchant ships
g. Civilian and general aviation
h. Current live track file

5 Results

As a result of the procedures described in the previous paragraph, the MSI and ESM databases were sized as shown in Figure 5-1. These sizes were used to populate the data structures shown in Figure 4-1 and Figure 4-3. This would be a very large fusion database compared to most fusion systems in existence today. The procedure also resulted in data update types and rates as shown in Figure 5-1. High and low arrival rates were estimated and the scenario operated for 3600 scenario seconds, enough time to gain associative memory hits for most of the update types.

The results of the MSI experiment are shown in Figure 5-2. This shows that the embedded DBMS is considerably slower than an applications dependent data access but generally keeping up with real time. That is, there are very few intervals in which the data access time exceeds 1 second. In contrast, the conventional DBMS almost always exceeds real-time. The two 0 values visible in the chart probably result from measurement imprecision. The embedded DBMS access time per update has mean of 0.98 ms, with sd of 0.17 and range of values of [0,10] compared to the conventional DBMS access time mean of 15 ms with sd of 0.015. The high value probably results from an AM hit for a particularly demanding update type, e.g., an order of battle update. A regression analysis of number of AM hits against total access time shows a Multiple r of 0.99 suggesting a linear loading on the embedded DBMS.

|	# Objects	MSI Update Period	MSI AM Hits	ELINT/ESM Update Period	ELINT/ESM AM Hits	Aircraft	Aircraft Type	Weapon	Ship	Vehicle	Airport	Seaport	Garrison Site	Missile Site	Electronics Site	other Facility	Aircraft Type - Facility Association	Ship - Facility Association	RF # per object	Materiel Typical	# Materiel Low	# Materiel High	# Types
Fire Control (JCTN and Fires Net)																							5000
TBM	4	0.25 sec	5	0	0				4										0.0	0	0	0	
Supersonic Aircraft and Missiles	50	1 sec	5	4 sec	30	25	25												2.0	50	15	50	
Subsonic Aircraft and Missiles	500	4 sec	5	10 sec	30	450		50											2.0	500	200	500	
Gen Av & Helo	400	4 sec	5	10 sec	30	400													2.5	500	250	500	
Land Mobile	100	4 sec	5	10 sec	50					100									0.3	9	5	14	
Land Fixed	100	10 sec	5	10 sec	20								25	25	25	25			0.5	34	30	37	
Tactical Radar																							
ATC	250	4 sec	10	10 sec	30	225	25												2.5	313	63	313	
Surface & Nav	100	10 sec	5	30 sec	50	8	2	90											4.0	400	360	440	
Tactical Decision Link (JDN)																				0	0	0	
Air	500	10 sec	10	30 sec	50	450		50											2.0	500	200	500	
Surf	250	30 sec	10	1 min	100				250										4.0	600	0	0	
Sub	15	30 sec	5	30 sec	30				15										0.5	2	0	0	
Land Fixed	400	10 min	5	10 min	50								40	10	100	75	100	75	0.3	68	34	68	
Land Mobile	200	30 sec	20	60 sec	200					200									0.3	19	9	19	
Operational Decision Net (JPN)																							
Surf	500	1 min	40	1 min	50				500										4.0	1200	0	0	
Sub	30	5 min	40	1 min	50				30										0.5	3	0	0	
Land Fixed	600	12 hr	30	1 min	30								70	30	100	100	200	100	0.3	101	0	0	
Land Composite Mobile	300	1 hr	40	1 min	100					300									3.0	338	56	338	
Merchant Shipping	10000	1 wk	40	1 wk	40				10000										1.0	5000	100	5000	
Order of Battle Net																							
Airbases	400	1 wk	20	1 wk	30								400					2400	2.0	800	800	800	
Aircraft Types per Airbase	6	1 wk	200	1 wk	100		1000												1.0	100	100	100	
Naval Ports & Anchorages	100	1 wk	20	1 wk	30							100							1.0	100	100	100	
Ship / Boats per Naval	24	1 wk	100	1 wk	100				2400									2400	4.0	9600	9600	9600	
Electronic Sites	4000	1 wk	20	1 wk	30										4000				2.0	8000	8000	8000	
Military Ground Sites	7000	1 wk	20	1 wk	30								7000						0.3	1750	1750	1750	
Missile and Gun Sites	3000	1 wk	20	1 wk	30									3000					1.0	3000	3000	3000	
Civil Aviation Flight Plans	200	4 hr	20	0 hr	0	200													0	0	0	0	
Airport Flight Scheds - Airports	50	1 wk	10	0 wk	0						50								0	0	0	0	
Active Aircraft Flight Scheds per Airport	100	1 wk	30	0 wk	0	5000													0	0	0	0	
		473	3368	112	4512	6758	1000	156	13285	600	560	140	7225	3200	4325	200		2400		32886	24572	31027	5000
		Updates and AM Hits Per Sec													**Table Sizes**								

Figure 5-1. Update Rates and Database Sizing
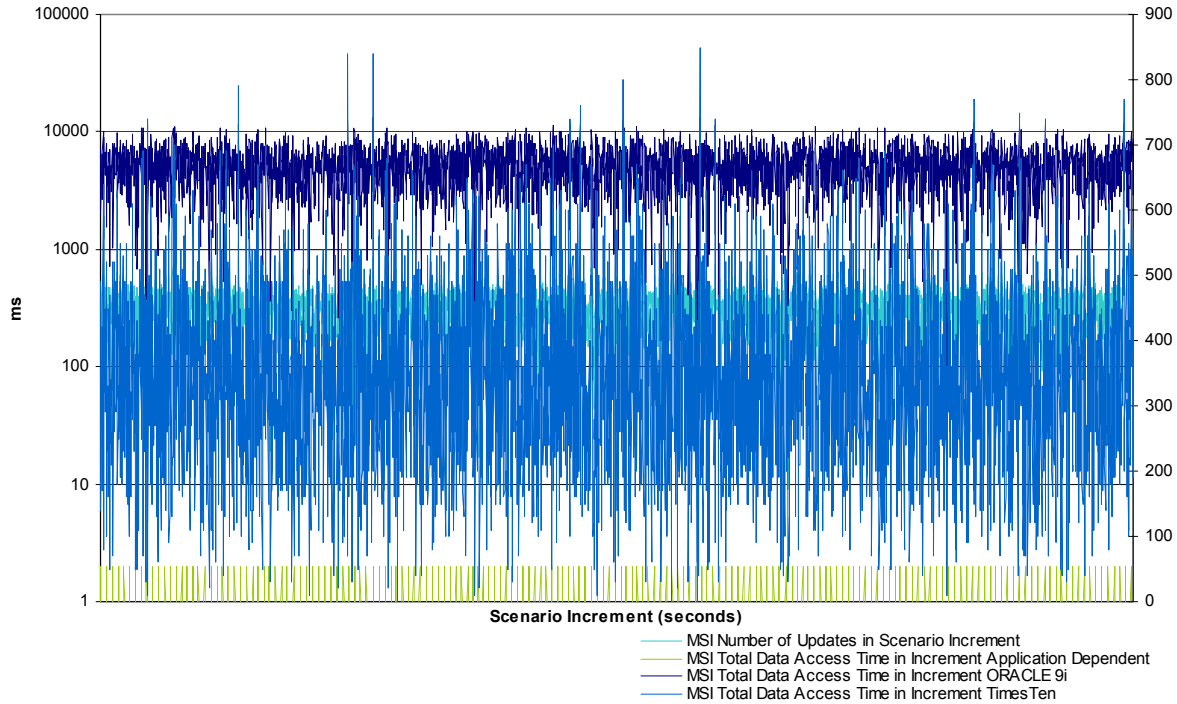
Figure 5-2. Access Time Comparison for MSI Gross Gating

Results from the ESM/ELINT SSI experiment are shown in Figure 5-3. As can be seen the total time required in any one increment is considerably less than for MSI, due to the lower update rate for ESM (see Figure 5-1). The embedded DBMS access time per update have mean of 9.1 ms with an sd of 0.43 and range of [7.69, 11.48] compared to mean of 0.011 and sd of 0.005 and range of [0.005, 0.077] for the application dependent structures and mean of 137.4, sd 0.11 for the conventional DBMS. Again, the regression analysis shows linear performance, with a Multiple R of 0.98.
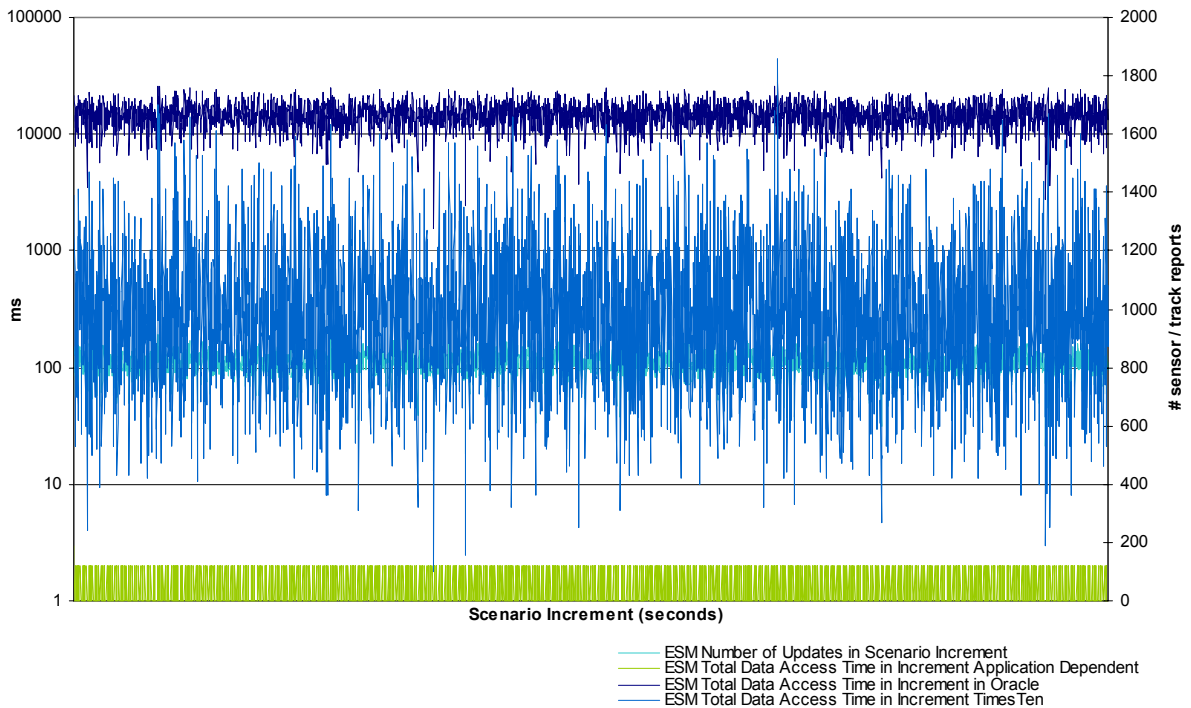
Figure 5-3. Access Time Comparison for ESM/ELINT SSI Classification and Correlation Candidates Retrieval

6 Conclusion

Powerful information modeling, database, and Object-Oriented tools and techniques have evolved over the past 10 or 20 years that can provide a foundation for large-scale fusion systems. Because this foundation is relatively stable, being grounded in the fundamental semantics of the enterprise, and because it is transparent, many types of fusion algorithms can be unified within it and operate in an integrated manner over a broad spectrum of information. Algorithms can be updated or changed in a modular manner, with good assurance that they will interoperate with the rest of the system. This enables a broad community of participation in the fusion system. Due to the broad scope of modeling that is possible and the ability to employ custom but modular algorithms, most suitable to the belief at hand, this approach will allow the development of very-large fusion systems.

The results of these preliminary experiments provide good evidence that, as would be expected, high-performance embedded DBMS's have much higher data access times than application dependent data structures. However, the results do show that even under intense scenarios and with massive fusion on a general purpose medium performance off-the-shelf computer using a non-realtime operating system, the embedded DBMS can perform adequately. The MSI experiment had average sensor updates of 473/sec resulting in requirements to access and average of 3,368 complex object structures per sec. Similarly, the ESM/ELINT experiment had average 112 updates per sec. resulting in the requirement to access 4,512 complex objects per sec. These accesses were against a track, situation awareness, and intelligence database with information on over 130,000 complex objects. With more powerful processors, a realtime operating system, and a distributed computing environment, the embedded DBMS could be expected to perform even better. A key enabler in the experiments was the associative memory. This kind of layering, of application-dependent associative memories, with high-performance embedded DBMS's, followed a conventional off-line DBMS for amplifying display and historical data, can enable the types of fusion benefits described in [9] and paragraph 1, herein.

7 References

[1] McDaniel, D., Tooey, M., "Advanced Combat Direction System (ACDS) Block 1 Tactical Information Integration and Control Group (TIICG) Algorithms Timing Analysis", *Technical Report*, American Defense Systems, Inc., San Diego, CA, 1987

[2] McDaniel, D., *Electronic Warfare IDentification (EWID) Small Business Innovative Research Final Report*, Space and Naval Warfare Systems Command, 1996.

[3] "A Simplified Turn Logic", *Note N-1090*, Fleet Computer Programming Center, Pacific, San Diego, CA 1963

[4] Rawlings, T., "Report on Parameter Search Methods for EW SSI", *Block 1 Combat Direction System Engineering Notebook*, Hugest Aircraft Company, Fullerton, CA, 1986

[5] Stankovic, J., Hyuk Son, S, Hansson, J., "Misconceptions About Real-Time Databases", *IEEE Computer*, June, 1999

[6] Ramamritham, K., "Real-Time Databases", *International Journal of Distributed and Parallel Databases*, Vol. 1, No. 2, 1993

[7] *Functional Description Document for Track and Relational Data Synchronization, Version 1.0*, Prepared for Defense Information Systems Agency (DISA) by FGM, Inc., August 26, 1998

[8] *NTDS Functional Allocation Study*, Chief of Naval Operations, 1980

[9] McDaniel, D., "Multi-Hypothesis Database for Large-Scale Data Fusion", *Proceedings of the Fifth International Conference on Information Fusion*, International Society of Information Fusion, Sunnyvale, CA, 2002

[10] Rawlings, T., "Track Position Correlation System", *Block 1 Combat Direction System Engineering Notebook*, Hughes Aircraft Company, Fullerton, CA, 1986

[11] Simpson, P., *Artificial Neural Systems*; Pergamon Press: Oxford, 1990

[12] The TimesTen Team, "High Performance and Scalability through Application-Tier, In-Memory Data Management", *Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, 2000, Cairo, Egypt,* Morgan Kaufmann, 2000

[13] Walker, R., Loaiza, F., Simaitis, E., "Development of a Far-Term Data Model for the Global Command and Control System (GCCS)"*, IDA Paper P-3047*, Institute for Defense Analyses, Alexandria, VA, 1995.