

www.objectivity.com

Corporate Headquarters

640 West California Ave.
Suite 210, Sunnyvale,
CA 94086-2486 US
Tel: (408) 992-7100
Fax: (408) 992-7171

Summary: Hitting the Relational Wall

This summary is based on a white paper written by Dr. Andrew E. Wade, Ph.D., a founder of Objectivity, Inc. and ODMG. He authored ODMG-93 and has also written many articles. To read the full-length paper, please visit our website at www.objectivity.com.

Introduction

Relational Database Management Systems (RDBMSs) have been very successful, but their success is limited to certain types of applications. As business users expand to newer types of applications, and grow older ones, their attempts to use RDBMS encounter the "Relational Wall," where RDBMS technology no longer provides the performance and functionality needed. Attempts to scale the wall with relational technology lead to poor performance, poor scalability, and loss of integrity. ODBMSs offer a path beyond the wall. "Hitting the Relational Wall" measures the wall, explains what model and architectural differences cause it, how to foresee it, and how to avoid it.

Background

Now three decades old, relational technology is showing signs of age. It works well when supporting simple, tabular data models, with short, simple operations and centralized data storage and access, but anything beyond that encounters the "Relational Wall." This term has been coined to describe what happens as relational databases reach the point that they can no longer scale to meet the demands of an application. The only way to add capacity to an RDBMS is add equipment and databases, and the more additions there are, the slower and more unwieldy the system becomes, sometimes to the point of failure.

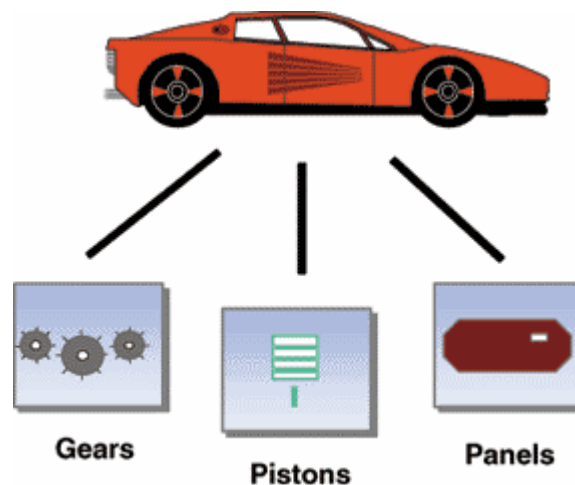
This phenomenon is the result of applying the wrong tool (in this case, RDBMS). One might, for example, try to use a screwdriver to pound nails into a wall, and some nails might actually go in, but in the process many nails would likely break, along with screwdrivers and walls. Instead, using the right tool, a hammer makes the job much easier.

Summary: Hitting the Relational Wall

Differences between an ODBMS and an RDBMS

An RDBMS is the right tool for many applications, but not for those that handle large volumes of complex data. An ODBMS enables scalability, flexibility, and performance when the RDBMS can no longer support the application.

Suppose you wish to store your car in the garage at the end of the day. In a relational system, all data needs to be normalized, meaning that the car must be disassembled down to its basic elements, which must each be stored in separate tables. So, the screws go in one table, the nuts in another, the wheels in a third, and the side panels, and the pistons, etc. In the morning, when you wish to drive to work, you must first reassemble the automobile, and then you can drive off. In an Object DBMS (ODBMS), on the other hand, this is modeled with just one object for the entire car and another for the garage. Run one operation ("store") and you're finished.



The Wall's Effect on your Application

Increased development costs and longer time to market

With a traditional RDBMS, programmers not only have to program their applications but also have to program and maintain a translation layer between the RDBMS and the application data. This maintenance includes adding features, fixing bugs, and improving capabilities, which together can comprise as much as 80% of the lifetime cost of the application. Some users have saved as much as half their development time and cost by switching to an ODBMS because they no longer have to work with the translation layer.

Summary: Hitting the Relational Wall

Potential Integrity loss

Since the mapping from flat RDBMS data to the application data is done within the application, by application programmers, there is the danger (even assuming all programmers program perfectly) that one programmer will do this mapping differently from other another programmer, resulting in a mismatch in their work and integrity violations. Such integrity issues can be subtle and difficult to locate.

In an ODBMS, there is no mapping required. The DBMS understands and directly operates on high-level structures (such as the automobile, the engine, etc.) as well as low-level structures (such as piston, bolt). Complexity of data structures and relationships is captured in the DBMS itself, so all users share the same view of the data and are guaranteed to be compatible at all levels, avoiding integrity risk.

In fact, by using grant and revoke (access permissions), certain users and groups may be limited to access only certain structures, and only certain attributes or operations on those structures. This guarantees that integrity will be maintained.

Performance

At runtime, disassembling and reassembling the automobile from its basic components takes time. The more complex the structure, the more time it takes to run the application. Therefore, the more relationships you have between data objects, the more you'll benefit from an ODBMS.

Benchmarks

The extended "Hitting the Relational Wall" white paper has in-depth results of benchmark studies between Objectivity/DB and an RDBMS, which were designed to test whether or not the Relational Wall exists. The benchmark content was generic, and an independent third party was hired to run the benchmark to ensure maximum impartiality.

Key findings from benchmark implementers:

Creating Records

"The results of the part creation phase show that even with a small number of parts, Objectivity is much faster than [RDBMS] in creating records. When the number of records rises, the difference in time between Objectivity and [RDBMS] is staggering.

Summary: Hitting the Relational Wall

For the benchmark that created 6,558 parts, Objectivity ran over 30 times faster. A separate program was written to validate that the records were actually created, because we couldn't believe Objectivity could create records so quickly. The other program printed all the parts, validating the test."

Updating Records

"For a very small number of parts, the RDBMS surpassed the performance of Objectivity [for record updating]. The RDBMS performance seems fairly linear based on the number of parts being updated. Objectivity's update performance increases dramatically as the number of records updated increases, with the final test showing Objectivity outperforming the RDBMS by over 7 times."

Deleting Records

"The RDBMS out performs Objectivity when the number of records being deleted is very small. Objectivity overtakes the RDBMS in the second test with 268 records, and dramatically outperforms the RDBMS when a large number of records are deleted. In the last test Objectivity was more than five times faster than the RDBMS in deleting records."

How to Avoid the Wall

The Relational Wall dramatically reduces performance, with execution time rising rapidly with increasing complexity, ultimately performing 100x slower than ODBMS. The Wall also causes increased software development and maintenance cost, and dramatically reduced flexibility and extensibility.

This wall can appear suddenly and unexpectedly when users take prototype applications and attempt to deploy them. The resulting growth in information, users, and complexity can bring them face-to-face with the wall. The result is typically failure.

The solution is simple: use the right tool for each job. Applications with the following traits are most likely to suffer the effects of the Relational Wall:

Use of Objects (C++, Java, Smalltalk, etc.), which requires mapping to RDBMS tables. Mapping increases development cost and makes maintenance dramatically more expensive.

Summary: Hitting the Relational Wall

Many-to-Many Relationships. The RDBMS model lacks support for many relationships. Simple one-to-one and one-to-many relationships may be modeled, but at the cost of extra application code to maintain them and loss of performance. Many-to-many relationships typically become too complex to be useful.

Varying Sized Data structures. Structures that track time-series and history information are poorly modeled by fixed size tables. In an ODBMS they're directly supported, making development and maintenance far easier and performance much higher.

Complex data structures. The RDBMS requires and supports only simple, flat, rectangular tables. Any other shape of structure, any complexity, and any nesting will require complex modeling tricks in the RDBMS, making implementation difficult, error-prone, and slow. In an ODBMS, these are supported directly.

Distributed (non-centralized) Deployment environment. The RDBMS borrows its architecture from the mainframe, storing data and performing all operations on a central computer (server). If your environment includes multiples PCs, workstations, servers, the ODBMS will prove far superior by providing a single logical view over all databases on all servers.

Heterogeneous Computers, operating systems, languages, and networks. RDBMSs typically run in one environment only and, at best, provide only remote access from different architectures. With the ODBMS, users can leverage existing investment in hardware infrastructure and training, to incrementally add new technology equipment alongside old, and to support all levels of users.

Flexibility and Extensibility, in both physical and logical design. The RDBMS offers only one physical structure, while the ODBMS allows any logical structures, dynamically mapped to the desired physical structures, in multiple databases, with multiple levels of clustering, across multiple types. All of this is hidden from users and applications by the single logical view.

Integrity. RDBMSs support integrity only at the primitive, flat, tabular level. All data structures above that are left to the application, and integrity is at the whim of each of the applications. The ODBMS supports integrity at all levels, with object operations

Summary: Hitting the Relational Wall

automatically maintaining integrity rules, and enforcing them across all applications. The ODBMS can also limit end users and groups of users to only certain objects and operations.

Scalability in an RDBMS is limited by the single, central-server bottleneck, and by the capacity of the server machine, which must support all users. The distributed, scalable ODBMS architecture avoids bottlenecks by allowing developers to add unlimited numbers of users without slowing down others, add inexpensive servers incrementally to improve performance as needed, and track up to millions of tera-objects, which each may be up to many gigabytes in size.

The final consideration is one of risk. Some, because they are familiar with the older RDBMS companies and products, might deem them to be lower risk. If, however, your application has any of the above characteristics, or might grow to include any of them, then the risk with the RDBMS ranges from having major problems to total failure. If you use objects with an RDBMS, that risk is magnified by the need to add a layer mapping from objects to tables, by the maintenance cost of that layer, and by leaving the application-level structures and integrity maintenance open to individual applications.

Please visit our website, www.objectivity.com, for the full-length white paper.