

The Case for a New Type of Database

A Plugged In Software White Paper
David Wood, CTO
Bernadette Hyland, CEO
Tate Jones, VP R&D
22 July 2002

Abstract

This White Paper discusses why Plugged In Software produced the Tucana Knowledge Store, a new type of database. Data representation using hierarchies, the relational model, the object model and the directed graph model are discussed and compared. A real-world example is used to illustrate the comparisons. Finally, usage of the Tucana Knowledge Store as a metadata repository is discussed and compared with metadata storage using relational databases.

Comparing Data Models

Human beings are excellent at categorizing information. We sort, file and compare everything we see during every waking hour. Beyond that, we draw (often artificial) boundaries around collections of things and thus create higher-level data representations of those things. Consider, for example, looking at a cloud. A cloud is not a "thing", it is an area in which a gaseous solution is in a certain state, yet we have the ability to treat it as such. This is incredibly useful because it allows us to refer to these constructs when we communicate. It adds an immense richness to our ability to express ideas, especially by analogy. Our ability to perform these functions is one of the hallmarks of the human brain.

When dealing with highly abstract information such as we might find in a typical office setting, two data models have clearly stood the test of time: The humble hierarchy and the relational model. We are all familiar with hierarchies. Electronic file systems and document management systems organize collections of documents in hierarchies. Hierarchical databases are considered old fashioned, but are the basis for modern directory services. We even represent the contents of XML documents that way; XML is a hierarchical format. A good example of a hierarchy is an organizational chart:

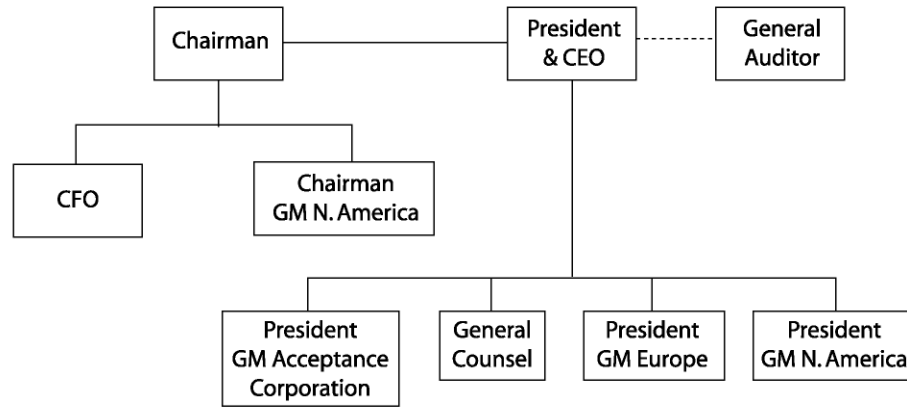


Figure 1. A Simple Organizational Chart

The relational model is the basis for relational databases. Relational databases have become so popular that even experts in the field simply call them "databases". In reality, there are hierarchical, object-oriented, relational and directed-graph databases. The relational model, as its name suggests, describes the *relationships* between two pieces of information. The allowable relationships must be defined in the database's schema; that is, the descriptions of each data type must be fixed in place before any information is entered. Knowledge of the schema is essential before one may query the database. No information may be gleaned from a relational database without *a priori* knowledge of those pre-defined relationships. For example, one might have a table of employees each of which has a name, a title and a salary:

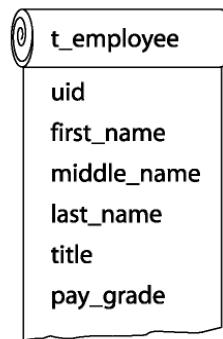


Figure 2a. A Simple Relational Table Description

uid	first_name	middle_name	last_name	title	pay_grade
0	Ralph	Paul	Deed	SE	SE1
1	Janet	Gloria	Stein	TL	SE4
2	April	Mary	Gomez	SE	SE2

Figure 2b. A Simple Relational Table with Sample Content

Relational databases have been tremendously successful because many real-world data relationships do not change and may thus be pre-defined. Every employee has a name. Every employee is paid a salary. Because these relationships are static it is easy to describe them in a relational database. Knowing these relationships it is simple to query the contents of a database. A simple query might be, "Give me a list of all employees whose salary is over \$45,000". Relational databases excel at this type of challenge and response where the relationships are both known and unchanging.

Note that discovery of new relationships is prohibited by the mechanism of a relational database. Without knowing the relationships in advance, queries against a relational database are impossible.

Object-orientation is a very different approach to storing data. Software "objects" include both some data and the logic that manipulates that data. This encapsulation is convenient for abstracting higher-level logic away from underlying information and implementation details but comes with significant performance penalties. Information systems with high numbers of objects are highly inefficient, keeping object databases far below the scale of the largest relational systems.

Directed graph data models assume very little about the data they hold. Each piece of information is connected to others via an arbitrary relationship. There is no limit as to how many relationships two pieces of information may share, nor how many other pieces of information one may connect to. The information and relationships thus form a graph, or web, of interconnectivity. Directed graphs are "directed" in that a relationship applies from one piece of information specifically to another. Bi-directional directed graphs have relationships that operate in both directions. Redrawing Figure 2 in a directed graph would look like this:

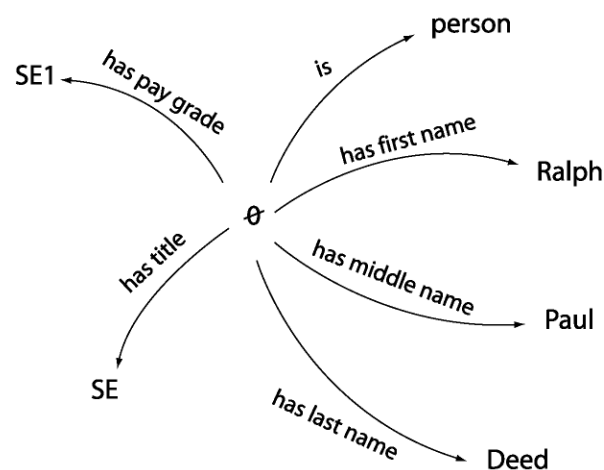


Figure 3. A Directed Graph

Each of the above data models has advantages and disadvantages. The largest single problem with hierarchies is that they do not represent duplicate information well. That is, if a particular piece of information has to exist in two places in the hierarchy the model breaks down. In practice tracking changes in duplicate information is very problematic. Relational databases solve this problem by allowing tables to hold references to entries in other tables. When the database is queried, the tabular information is joined to express a result. Unfortunately, performing table joins is a computationally expensive operation. Many joins result in slow responses to queries. This is known as the "join problem" and is shared by all relational database implementations.

A database implementing a directed graph could also encounter the join problem. In the Tucana Knowledge Store, however, the database was designed to hold information in the form of subject-predicate-object statements. Each statement thus contains

some data elements and a relationship between them. That is, all information entering the database is in a regularized form. Plugged In Software chose to use the World Wide Web Consortium's Resource Description Framework (RDF) standard as the basis for its data representation. This makes sense, since any type of information and information relationship may be described in such a form. Massively indexing all statements (i.e. sorting all statements by subject, again by predicate and again by object) effectively flattens the data structure and allows any query constraint to be satisfied by a simple range in one of the indices. One pays a penalty for holding more information in an RDF statement than one holds in a relational record, but advantages are gained at query time. The join problem is thus avoided by creating a database effectively optimized for joins.

Query times in relational databases can grow exponentially with interrelatedness of the data. Query times in the Tucana Knowledge Store grow roughly linearly in proportion to the amount of data.

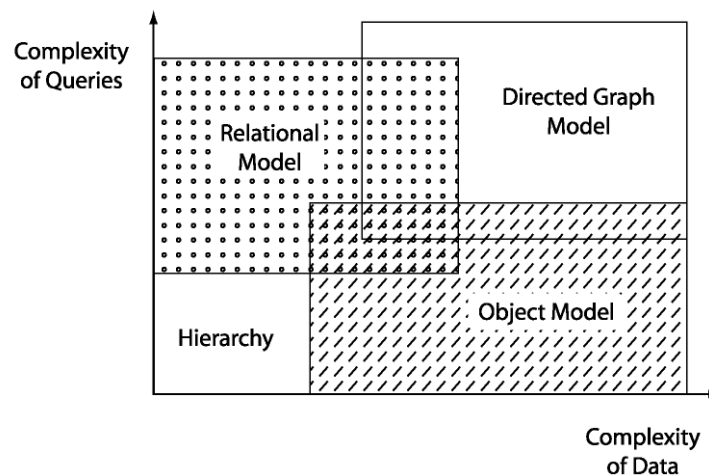


Figure 4. Choosing a Data Model

Figure 4 shows the resulting recommendations. Hierarchies are suited for simple data structures which do not need to be queried with any complexity. Object models may be used for complex data but not for complex queries (that is, the queries available are effectively encoded in the objects). Relational databases excel when queries may be very complex but become limited when the data is also very complex. The directed graph is suited when both queries and data sets are complex. The Tucana Knowledge Store implements a directed graph data model to allow for such complex environments.

It is not our intention to suggest that the Tucana Knowledge Store (or any other directed-graph database) be used to replace relational databases. Figure 4 shows that relational databases are the best tool for environments where complex queries need to be run against structured and relatively simple data.

Figure 4's overlapping areas infer that many real-world problems may be solved using more than one approach. The next section will use some real-world data to illustrate when one data model may be chosen over another. Later we will discuss how Plugged In Software uses the Tucana Knowledge Store particularly to handle large volumes of distributed metadata.

An Extended Example

Consider a real-world organization chart. Figure 5 shows another level of the chart shown in Figure 1. Note that some people have reporting lines to more than one person. The structure is no longer a hierarchy!

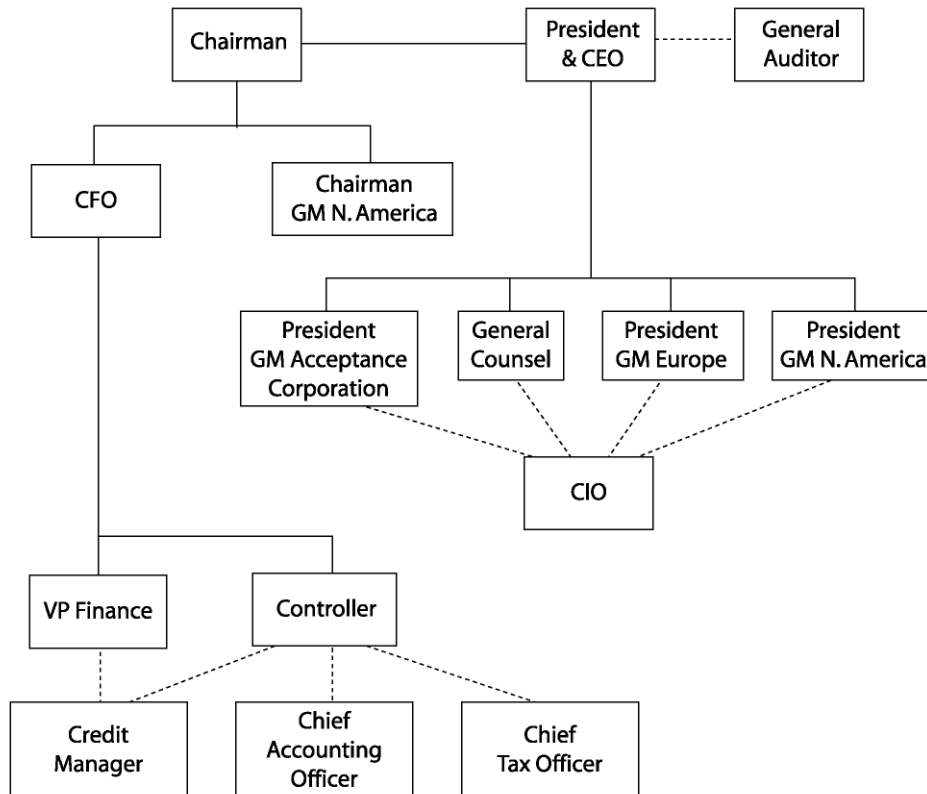


Figure 5. A More Complex Organization Chart

Luckily, we can still use a relational database. A real-world organization would have a more complex employee table. Perhaps it would look like this:

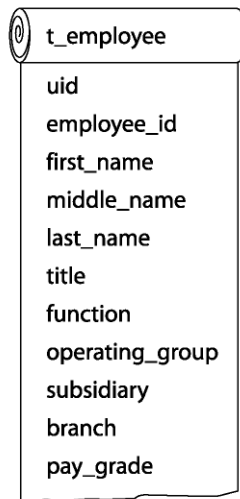


Figure 6. A Relational Employee Table

The expanded employee table results in more tables. Briefly, these tables hold structured information about employees. We want to hold this information in tables so we can query the contents. Figure 7 shows the tables needed.

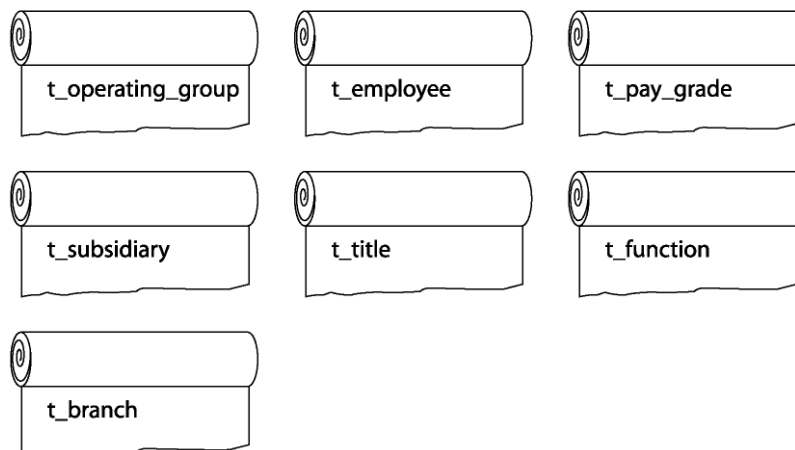


Figure 7. Tables Needed to Support the Employee Table

This example shows that representing the most basic real-world corporate organization chart in a relational database could require a join across a minimum of seven tables. In a large company like General Motors (whose top-level organization chart we used), with 395,000 employees worldwide, operating in 70 countries and with 260 major subsidiaries, an employee lookup via a Structured Query Language (SQL) query has a reasonable overhead. Consider the more typical business query involving greater complexity, more data (e.g. documents, spreadsheets, images, email messages, etc) and the suitability of the relational model becomes too complex for any single team of database administrators and application developers to document and maintain in real time. New applications can take many months to develop, as can modifications to existing ones.

If we held this same information in a directed graph it would soon be difficult for a human to read (see Figure 8). Fortunately for us, this is not true for machines. The directed graph holds semantic information. That is, the information is described in a way that the machine can read. A person is known to be a person and a title is known to be a title. The computer no more "understands" the semantic information than it understands the column headers in a relational database. It can, however, return that information at the request of a querying client. The graph may be "walked" to discover new content and new relationships. User interfaces may be built that have no *a priori* knowledge of the data's schema.

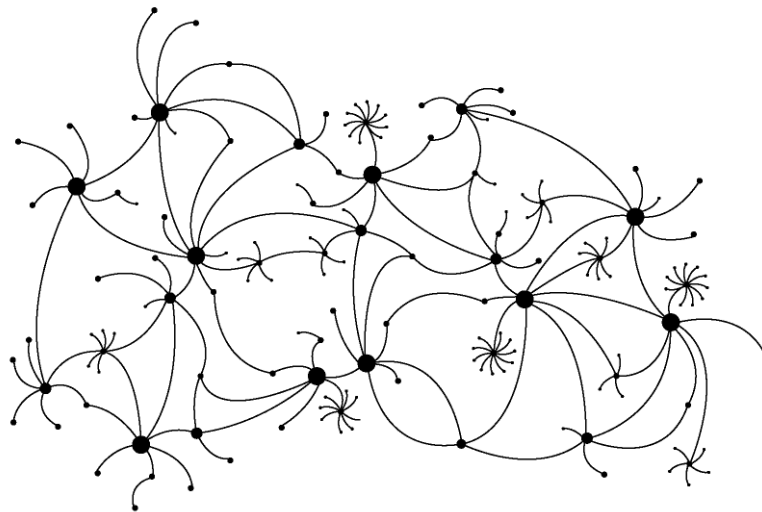


Figure 8. A Complex Directed Graph Showing its Web-like Nature

Storing and Retrieval of Metadata

The Tucana Knowledge Store has been optimized for the storage and retrieval of metadata, or information about data. The process of metadata management consists of complex queries performed against highly interrelated information (metadata statements). This makes metadata management a good application area for a directed graph database.

Tucana works by representing large amounts of information with a (typically) smaller amount of metadata. In the case of a word-processing document or an electronic mail message, for example, metadata might include the author, the recipients, the subject, keywords, concepts addressed, people named, dates or places mentioned, etc. Metadata is stored in a *lingua franca* to enable sharing across applications and geographic boundaries: the World Wide Web Consortium's Resource Description Framework (RDF), an international standard for the representation of metadata. RDF is part of the W3C's Semantic Web project, an important next-generation project to facilitate the understanding of information by the machines which store it.

Original data (e.g. a word-processing document or an email) is stored separately from the Tucana Knowledge Store and referenced by uniform resource locator (URL). These data may be stored on a simple file system since they are accessed by machine. This avoids storing large documents in relational databases or object stores and drastically speeds access to the underlying information.

The Semantic Web project has seen the creation of other RDF databases. Unfortunately, most of these have been useful only to academics pursuing RDF research. Of the commercial products launched only the Tucana Knowledge Store was created from the ground up as a native RDF database and not based on existing relational database technology. This is important to the scalability of TKS. Tucana represents the first commercially available building block of the Semantic Web which may be used in large-scale real-world applications. Current research at Plugged In Software is focusing on scaling TKS beyond the limits of relational database technology.

A comparison of metadata storage in relational databases and the Tucana Knowledge Store readily highlights the advantages of TKS.

Storage of metadata in a relational database typically requires a design with a few (one to six) very narrow, deep tables. Each table might consist of two to five columns, most of which are indexed more than once for performance. A basic query would require all tables to be joined with additional table aliases for nested joins. Large numbers of individual queries are required to perform more complex queries, resulting in slower response times.

Updating metadata statements in a relational database is expensive since all columns are indexed. This would require data pages and indexed columns to be locked. The Tucana Knowledge Store allows inline writes into its indices and does not have the concept of data pages.

Updates in a relational database are slow when multiple data locks attempt to obtain the same page lock (a "hot spot"). These types of locks also effects reading performance and maintenance. The Tucana Knowledge Store uses phase trees, so updates do not effect read operations already in progress. This allows long-running transactions.

Relational database query optimizers do not perform well when faced with narrow, deep tables with varying amounts of data distribution. The optimizer's distribution pages (costs) do not contain enough detailed information per index. This occurs because columns storing a variety of data (as in metadata statements) are diluted in favor of columns with high uniqueness (such as a column of phone numbers). Additionally, some RDBMS optimizers are designed to examine all possible execution plans based on rules and cost. Since metadata queries tend to be self recursive and have numerous paths, optimizers battle to determine the best execution plan. In some cases determining the execution plan takes longer than executing the actual query. The Tucana Knowledge Store has an optimizer specifically designed to handle metadata queries.

Caching of information anticipated to be relevant to subsequent queries is also an issue. Most relational databases cache the most recently accessed tables. This does not suit databases holding metadata since the relevant table are narrow and deep. Therefore, RDBMS caching mechanisms are often prohibited from coming into play and can even hinder performance. The Tucana Knowledge Store has a caching mechanism designed for metadata queries.

The last major problem with storing metadata in relational databases is one of security. Additional tables and columns would have to be added to the design above to secure metadata at the table and column levels. This would again increase the number of joins performed and the complexity of the queries. Security in the Tucana Knowledge Store is addressed in a manner which does not compromise performance. Statements are recursively grouped according to their security restrictions. This group information is indexed along with the rest of the statement representation.

Conclusion

Many, but not all, real-world data relationships may be represented in structures such as hierarchies, object models and relational databases. The directed graph data model may be used to represent those which cannot or should not be represented using other models. The Tucana Knowledge Store implements a *distributed, secure, directed graph of metadata*. Thus, the Tucana Knowledge Store takes its place alongside other commercial databases to offer a modern alternative for the storage and retrieval of distributed metadata.

Resources

1. Plugged In Software's Home Page: <http://www.pisoftware.com>
2. Plugged In Software's Products: <http://www.pisoftware.com/products>
3. Tucana FAQ: http://www.pisoftware.com/products/tucana_faq.html
4. Tucana Integration Guide: http://www.pisoftware.com/downloads/tucana_integration_guide_US_letter.pdf
5. Understanding Tucana: http://www.pisoftware.com/products/understand_tks.html
6. Resource Description Framework. World Wide Web Consortium. <http://www.w3.org/RDF/>
7. Semantic Web. World Wide Web Consortium. <http://www.w3.org/2001/sw/>
8. This White Paper: <http://staff.pisoftware.com/dwood/publications/TKSWhitePaper.html>