ConceptsDB

A Novel Database Engine
Uniquely Combining Property Graphs and Hypergraphs

Product Description

ConceptsDB is a new database engine under development by Linguistic Technology Systems (LTS). **ConceptsDB** uniquely combines the benefits of Property Graph database engines and Hypergraph database engines, both of which have emerged as popular **NoSQL** solutions for flexible, heterogeneous data storage. **ConceptsDB** supports key-value properties and property schema on nodes and edges, like a property graph. At the same time, **ConceptsDB** supports multi-node hyperedges and contextual groupings of nodes and edges, like a hypergraph.

ConceptsDB introduces a Hypergraph Data Model Protocol (**HGDM**) which allows individual files and multiple kinds of databases to be accessed as if they were Property Hypergraphs. As such, **ConceptsDB** technology does not need to be used *solely* with databases stored via the **ConceptsDB** back-end. Instead, with special "adapter" libraries, other databases — or even individual files — could be queried, traversed, and integrated using **ConceptsDB**-style code.

What distinguishes **ConceptsDB** from other database engines is that it is designed from the ground up to prioritize standalone, native application development with responsive, multi-media User Experience. **ConceptsDB** can be bundled directly as **C++** files into applications and code libraries, unless special optimizations are needed. Furthermore, **ConceptsDB** directly supports **Qt**-based **GUI** programming, with the option of saving **Qt** objects and application session/state data directly in the database.

Given its flexible integration of disparate database models and its emphasis on rich desktop-style application development, **ConceptsDB** stands apart from most database engines on the market, which tend to be backend-only (no explicit support for **GUI** views) and locked in on specific, limited data models (such as **SQL**, **RDF**, or "big column").

The "VersatileUX" Design Philosophy behind ConceptsDB

The software design paradigm behind **ConceptsDB**, and its associated tools, can be summarized as "Versatile User Experience" (VersatileUX), a particular LTS coinage. VersatileUX connotes a development methodology which emphasizes self-contained native applications with few external dependencies; custom-implemented **GUI** components; and heterogeneous data models that can integrate information from disparate data sources. All of these features of an adaptable application development environment make the resulting software applications more versatile.

By embracing the VersatileUX philosophy, software applications based on **ConceptsDB** can integrate data from multiple sources instead of being limited to a single **ConceptsDB** instance as a single back-end. Each data-source connected to an application can be visualized as a "vine" carrying information into a central arena (the term *vine* is derived from Carnival, a biomedical data integration project of the Perelman School of Medicine, University of Pennsylvania). Each vine can be equipped with its own **GUI** library and object models — this approach is "versatile" because

data models and GUI design are *adaptively* tailored to individual data sources, rather than rigidly imposed by a central software application. A ConceptsDB application then becomes a modular synthesis of multiple vines and their associated GUI components.

The *versatility* of ConceptsDB applications derives, in part, from the *flexibility* of ConceptsDB's "Property Hypergraph" data representations. While Hypergraphs and Property Graphs have emerged as leading industry paradigms for heterogeneous data integration, they have, heretofore, never been combined in a fully rigorous manner. Prominent examples of projects based on property graphs include Carnival (mentioned above), CANDEL (the Cancer Data and Evidence Library) from the Parker Institute for Cancer Immunotherapy (based in San Francisco), and AllegroGraph "Semantic Data Lakes." Hypergraphs, for their part, are also used for projects such as SECO, which provides interactive "digital notebooks" backed by HYPERGRAPHDB. What ConceptsDB accomplishes is that it takes these industry trends to the next level, by synthesizing the unique features and capabilities of both property-graph database engines (such as NEO4J, Apache TinkerTop, Datomic, and AllegroGraph) and hypergraph database engines (such as HYPERGRAPHDB, WHITEDB, Microsoft Graph Engine, and the Trinity Specification Language).

Expressive Data Models for Application Development and the Role of ConceptsDB

Within the overall database market, NOSQL engines (which cover a range of non-relational solutions) have emerged as a significant alternative to SQL-based relational database engines. The NOSQL sector is further divided into different categories, such as graph databases, document-oriented databases, and "big-column" data stores. The push toward NOSQL is driven by a desire for application-development efficiency and the need for expressive data models. When compared to relational databases, NOSQL data models are conceptually closer to real-world information. One of the benefits of expressive data models is that they require less development time to translate data used by an application into data that can be stored in a database. The resulting savings in development costs make NOSQL solutions an attractive option for custom software.

Efficient application development is facilitated by ConceptsDB, which has a flexible and expressive data meta-model, even more so than typical NOSQL solutions. Many database engines (of all varieties) prioritize supporting high-volume throughput and extremely large overall data storage capacity, both of which tend to limit the expressiveness of data models stored in the database. Often, however, this is an unnecessary trade-off: for medium-sized database instances there is no need to implement restricted data models which require complex application-to-database translations. As such, ConceptsDB does not force programmers to use oversimplified data models so as to optimize volume which the database does not need.

Vertical Markets for ConceptsDB

With its unique VersatileUX philosophy and expressive data models, ConceptsDB potentiates a broad market reach, with product applications in biotech, scientific publishing, fintech, manufacturing, engineering, military, infotainment, and e-commerce. Because database engines are among the most ubiquitous genre of software tools within Information Technology, the market is propitious for introducing new database engines that facilitate streamlined application development.

LTS has focused on biomedical applications for ConceptsDB as its first potential vertical market. In biomedical applications, expressive data models are more important than size/throughput optimization. In addition, biomedical data is often linked to files in special scientific formats. Much of the raw data in a biomedical data set is encoded in special files which require specific parsers and

code libraries. In these contexts, the database engine itself has to work with a range of different file formats and special-purpose libraries, all interoperating with records stored within the database proper. As an embeddable C++ engine with a flexible data model that can support file-system operations and encoding meta-data referring to primary data stored elsewhere, **ConceptsDB** is particularly well suited to the unique architectural requirements of scientific and biomedical computing.

Tools Related to the ConceptsDB Engine

In addition to the **ConceptsDB** database engine itself, LTS's development tools include the following:

1. Transparent Hypergraph Query Language (**THQL**): A scripting and graph-query engine which includes and significantly extends the Gremlin Virtual Machine. **THQL** could be used both for processing graph queries and for embedding scripting capabilities within native software. A priority for **THQL** is that both query and script processing are done with few external dependencies, where the engines can be embedded in other applications and distributed in source code fashion. This would be a significant departure from normal scripting/query engines (there are database engines, such as **WHITEDB**, which can be embedded in source fashion — as LTS proposes via **ConceptsDB** — but these do not have separate query languages). **THQL** is referred to as "transparent" because all parsing, scripting, and query-processing is directly accessible to the host application (**THQL** engines would not rely on precompiled libraries that are opaque to source code analyzers and debuggers). In short: **THQL** offers modular scripting and database query engines *embedded entirely as source code* in native applications, as opposed to being dependent on external libraries.

2. Multi-Media Data-Set Explorer (MdsX): A toolkit for building applications which serve as interactive data-set presentations or "digital notebooks." User-visible MdsX resources will, by design, include a mixture of graphics, text, and code, so that MdsX in the native-application domain covers use-cases similar to **SECO**, **KAGGLE**, and **JUPYTER**.

3. Native-Cloud-Native (**NCN**): A technology for developing "Cloud-Native" container services (such as Docker or Kubernetes) as a back-end for standalone native applications. **NCN** would permit cloud backup, user/session persistence, and inter-application messaging among native "VersatileUX" applications. In the context of biomedical data-sharing, for instance, **NCN** components could route information between institutions participating in data-sharing initiatives and/or implement the central aggregators where multiple institutions' data is joined together.

4. Dataset Creator (dsC): Tools for composing publications and preparing data sets for open-access data-sharing. The crucial focus of dsC is ensuring that data sets and the associated journal articles are properly interoperable, with detailed annotations applied to publication text, and links (within the data set) to corresponding sections of the publication. In the context of a biomedical data-sharing initiative, for example, dsC would help scientists publish their research data on scientific hosting platforms and reference this data in their papers more effectively. This would thereby allow other scientists who are reading those papers to conveniently access the author's data for further analysis and/or replication.

For more information please contact:
Amy Neustein, Ph.D., Founder and CEO
Linguistic Technology Systems
amy.neustein@verizon.net • (917) 817-2184