

LazyView™
Database Aggregation Capability
White Paper

Lazysoft
technology

www.lazysoft.com

Copyright © 2003 Answerbrisk Ltd

LazyView™ Database Aggregation Capability

Modern businesses frequently have to deal with islands of information residing in different database applications which are often geographically dispersed. For example, disparate aspects of customer data may be held in an accounting system, a CRM system and a sales order processing system. Or a company may find that it can only obtain a global view of its sales pipeline by aggregating data from databases running in New York, London and Tokyo.

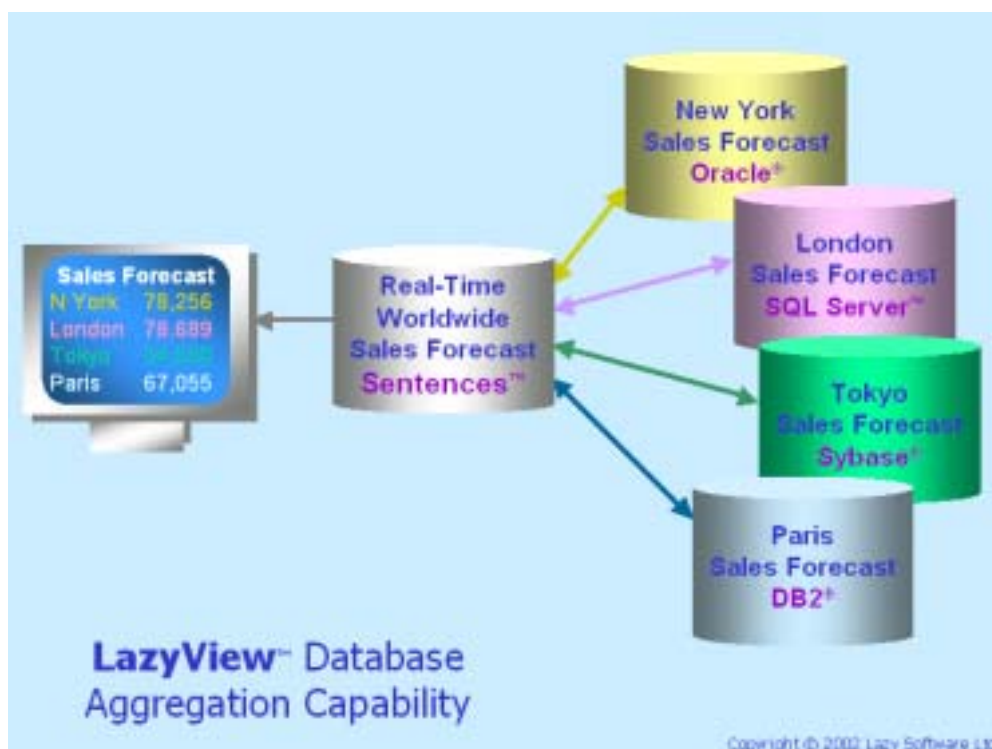
Sentences™ is Lazysoft's innovative multi-user, web-enabled database management system, that is based on the Associative Model of Data.

LazyView™ extends the capability of Sentences, providing real time aggregation of data drawn simultaneously from Oracle®, DB2®, SQL Server™ and other relational databases.

LazyView gives users the unique ability to integrate information from multiple, disparate relational databases quickly and easily without moving data, and without costly data warehousing and migration tools. Here are some examples of where this capability can add value:

- for a single 360° view of everything they know about their customers
- for an aggregated view of assets and asset utilisation across all locations
- for financial consolidation and compliance across geographically dispersed subsidiaries
- to compare raw material prices and availability from multiple suppliers
- to correlate two companies' customer bases on the way to a merger or acquisition
- for a complete view of a sales pipeline across different countries and regions

In addition, Sentences' native ability to read and write XML also enables it to integrate relational data with data from non-relational and web-based sources.

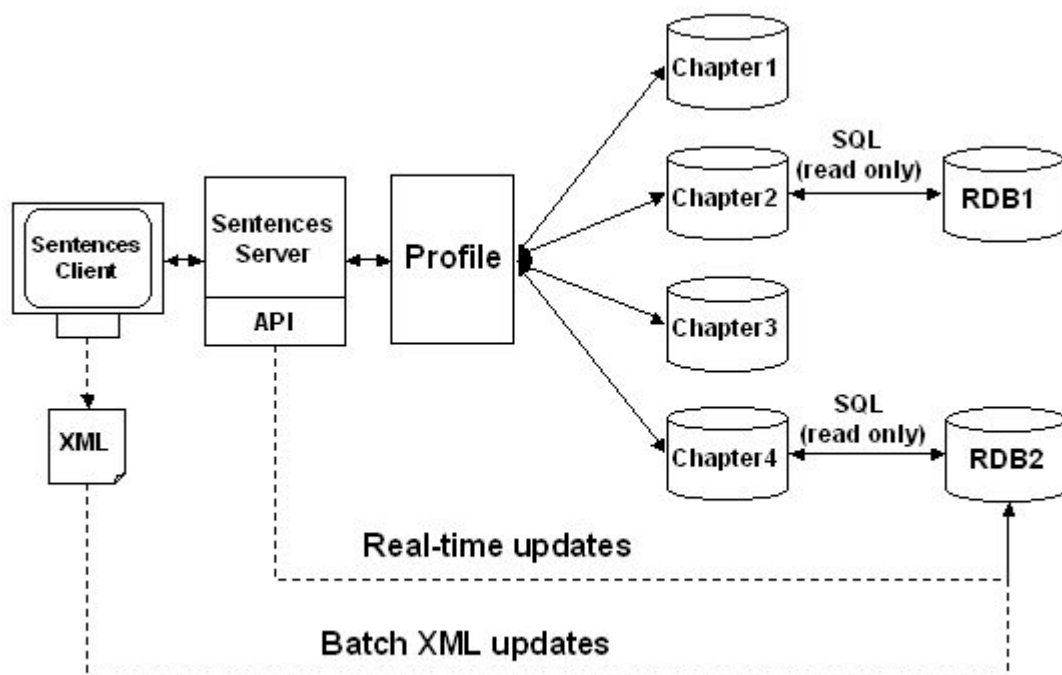


Technical Information

The Sentences Associative DBMS has a naturally distributed structure that makes it an ideal base for the LazyView functionality. A Sentences database comprises any number of individual chapters, each of which is a single file that may reside on any server. A chapter may contain data, or schema, or both. The chapters that together comprise a user's view of a database and to which they have access is determined by the Sentences profile that they use to log onto the Sentences server. Chapters may be added to, or removed from, a profile to alter the users' view of the database. Each profile may be used by any number of users. The fact that data or schema may have originated from separate chapters is transparent to users.

To create LazyView, we have developed the concept of a "pseudo-chapter". A pseudo-chapter appears to the Sentences server exactly like a regular Sentences chapter, so all the normal Sentences functionality is available. However, requests for data to these chapters are converted to SQL requests that are run against the target database using JDBC. Each relational database is represented by its own pseudo-chapter, so LazyView can retrieve data from multiple databases simultaneously. The relational data appears in the Sentences user interface in the same way as ordinary Sentences data. Users may choose which data from the target databases is visible in Sentences.

It's important to recognise that LazyView operates in real time. It does not copy data from its target relational databases into its own chapters, so there is no need to transport large volumes of data from relational databases into Sentences. As a consequence, LazyView can readily work effectively with extremely large relational databases. Also, updates made to the target relational databases from other sources will be automatically visible in Sentences.



LazyView Capabilities

Native Sentences Functionality

Once data from multiple target databases is aggregated using LazyView, the full range of Sentences' native capabilities may then be brought to bear on it. These include:

- Multi-user, web-based data access and maintenance
- Instant, 360° view of any item or value in the database, showing all its inward and outward references
- Sophisticated associative query capability, with automated XML and XSL output and presentation
- Instant search and selection on any attribute, with wild card capability
- Instant data change and capture capability through "omnicompetent" dataforms
- User interface options include Sentences DE (Deployment Environment), embedded Java applets and dynamic HTML

Augmenting and Overriding Data

Using LazyView, data drawn from relational databases may be augmented or overridden locally by the addition of new data that is then stored in an ordinary Sentences chapter.

For example, when using LazyView to consolidate a global sales forecast pipeline from regional data, it would be possible to record global budgets and forecasts locally in the Sentences database, and also to override local forecasts to construct "what if" scenarios. Similarly, when pooling customer data from several sources, it would be possible to maintain a list of customer relationship managers centrally in Sentences and assign to them responsibility for individual customers.

When relational data is augmented or overridden in this way, the primary keys of affected rows are stored by LazyView within ordinary Sentences chapters to ensure that the new data is correctly persisted. (It is important to recognise that only the primary keys of affected rows are stored: LazyView does not keep any other data from affected rows, or any reference to unaffected rows, so the current state of the target database, including ongoing updates, will always be accurately reflected by LazyView.)

Update Strategy

For security reasons, direct access by Sentences to the target databases is read-only. If there is a requirement to generate updates to the target databases as a consequence of decisions taken under LazyView, this may be accomplished in any one of three ways:

- A Sentences trigger may be written to initiate an update to the target database in any one of a number of ways, depending on the target database itself.
- A Java program may be written using the Sentences API to check for appropriate conditions and initiate the update similarly.
- Sentences can generate updates in the form of an XML document.

Useful Associative Constructs

When combining and correlating databases using LazyView, the unique capabilities that Sentences owes to its associative architecture become very powerful.

Table Equivalence is the ability to record that two tables both refer to the same or a similar thing in the real world. For example, Database A may contain a table called "Client", and Database B may contain a table called "Customer". When LazyView connections are established to Database A and to Database B, it is possible to assert that "Client" and "Customer" are equivalent. Selecting either will then aggregate and list instances of both. This sets the scene for other important capabilities.

Row Equivalence provides a similar capability at row level. Having established that Client and Customer are equivalent, LazyView may then record that the Client called "IBM" in Database A is in fact a reference to the same real-world entity as the Customer called "International Business Machines" in Database B. Sentences will then aggregate their attributes, so that in selecting either "IBM" or "International business Machines", the attributes of both are visible.

As a refinement, it is also possible to say that one instance supersedes another, rendering the second invisible. This could be used to consolidate the attributes of both "IBM" and "International Business Machines" under the "IBM" entity, rendering "International Business Machines" invisible.

Column Equivalence then comes into play. This may be used to assert that the column called "Credit limit" in Database A's Client table is equivalent to the column called "Maximum balance" in Database B's Customer table. This ability can be used to simplify the combined view, and also to identify inconsistencies between two databases. If, for example, "IBM" has a Credit limit of \$250,000 in Database A, and "International Business Machines" has a maximum balance of \$100,000 in Database B, this becomes immediately apparent. This in turn is managed by:

Attribute Supersedence. Under these circumstances, it is then possible to use attribute superseding to say that "International Business Machines' \$100,000 Maximum balance is superseded by IBM's \$250,000 Credit limit. Henceforth, only the Credit limit of \$250,000 will be visible.

This table summarises the capabilities of equivalence and superseding:

	<i>Equivalence</i>	<i>Supersedence</i>
<i>Tables</i>	The table "Client" in Database A is equivalent to the table "Customer" in Database B. Both types have the attributes of both.	N/A
<i>Columns</i>	The column "Credit limit" in Database A's Client table is equivalent to the column "Maximum balance" in Database B's Customer table. Both types have the attributes of both.	N/A
<i>Rows</i>	Client "IBM" is equivalent to Customer "International Business Machines". Both continue to be visible; the attributes of both are pooled and are visible on both.	Client "IBM" supersedes Customer "International Business Machines". "International Business Machines" is no longer visible; the attributes of both are pooled under Client "IBM".
<i>Attributes</i>	N/A	"IBM has Credit limit \$250,000" supersedes "International Business Machines has Maximum balance \$100,000".