Contents

In Memory Overview ........................................................................................................... 3
Benefits ................................................................................................................................. 3
Agile development & rapid delivery ................................................................................ 3
Data types supported by the In-Memory Database ............................................................. 4
When to use in-memory ..................................................................................................... 4
Sizing ................................................................................................................................... 6
Overview ............................................................................................................................ 6
Data Compression Examples .............................................................................................. 6
Server Requirements .......................................................................................................... 7
Performance testing & results ............................................................................................ 8

Yellowfin, and the Yellowfin logo are trademarks or registered trademarks of Yellowfin International Pty Ltd. All other names are trademarks or registered trademarks of their respective companies.

While every attempt has been made to ensure that the information in this document is accurate and complete, some typographical errors or technical inaccuracies may exist. Yellowfin does not accept responsibility for any kind of loss resulting from the use of information contained in this document.

The information contained in this document is subject to change without notice. This text contains proprietary information, which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced, stored in a retrieval system, transmitted in any form or by any means, or translated into another language without the prior written consent of Yellowfin International Pty Ltd.

The incorporation of the product attributes discussed in these materials into any release or upgrade of any Yellowfin software product – as well as the timing of any such release or upgrade – is at the sole discretion of Yellowfin.

This document version published April 2010 Copyright © 20010 Yellowfin International Pty Ltd.
In Memory Overview

Yellowfin’s in-memory database is a columnar database where the data is stored in memory and not on disk as with traditional database tools. A column-oriented DBMS is a database management system (DBMS) which naturally stores its content by column rather than by row. This has advantages for databases such as data warehouses and library catalogues, where aggregates are computed over large numbers of similar data items.

Benefits

1. Column-oriented systems are more efficient when an aggregate needs to be computed over many rows but only for a notably smaller subset of all columns of data, because reading that smaller subset of data can be faster than reading all data.

2. Column-oriented systems are more efficient when new values of a column are supplied for all rows at once, because that column data can be written efficiently and replace old column data without touching any other columns for the rows.

3. Data compression - Column data is of uniform type; therefore, there are some opportunities for storage size optimizations available in column-oriented data that are not available in row oriented data.

Agile development & rapid delivery

One of the key benefits of in-memory analysis is the ability to use it as part of your agile development process. In an agile environment you can use the in-memory database to build rapid proof of concepts, or throw away analytical applications that can be used to solve particular business problems without the need to invest heavily in a full blown data warehouse environment.

In this way Yellowfin shortens the development life cycle by removing some key steps from the process. Doing so end users can start to use the data you have provided to rapidly deliver reports and insight into the business.
Data types supported by the In-Memory Database

The Yellowfin in-memory database is designed to allow you to access the following data types for analysis and reporting.

- All standard numeric data types (eg. int, decimal, float, binary)
- All standard character data types (eg. char, nchar, varchar, nvarchar, text)
- All standard time data types (eg. timestamp, datetime)
- GIS data (GIS points, GIS polygons)
- BLOB & CLOB data

When to use in-memory

In-memory analysis is a specific data warehousing solution that should be used only in appropriate use cases. It will not solve all of your data storage and reporting needs. However, used effectively it add significant benefit to BI projects.

The figure below illustrates that as data complexity and volumes grow the appropriate solution for handling them will change.

Where in-memory analysis excels is:
1. **Departmental BI projects** – few data sources, low level of data transformations.

2. **Rapid Integration for ISV’s** – the ability to quickly integrate reporting on top of an OLTP application.

3. **Short term BI analysis** – throw-away analytical projects that need rapid access to data in an agile manner.

4. **Data Source is slow** – when the data source being accessed is slow due to the complexity of the schema being reported against.

In-memory Analysis is not appropriate when:

1. Real time reporting is needed
2. When Merging data from multiple data sources
3. Terabytes of data being accessed
4. If your data store is already fast – or can be made to be
Sizing

Overview

Based on the columnar in-memory database architecture of Yellowfin there are a number of considerations that need to be made when determining the size of the servers you need to meet demand.

When considering the use of Yellowfin’s In-Memory Database, the following factors should be examined:

1. The more unique the data in the view that’s being cached, the more memory it will take as it will compress less than repetitive data.
2. The more long strings in the data of the view, the more memory is used to cache it.
3. The more complex the query of the view is, the longer it will take to cache

Data Compression Examples

The size and complexity of your data set will determine the amount of memory is required to load the data into memory. The following example is used to highlight just how much variation can exist. For example a simple view with multiple repetition can be significantly more compressed than a view with low repetition and a large amount of unstructured data.

<table>
<thead>
<tr>
<th>Rows</th>
<th>View Query</th>
<th>Data</th>
<th>RAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>36,000,000</td>
<td>Simple</td>
<td>Repetitive</td>
<td>1 GB</td>
</tr>
<tr>
<td>4,000,000</td>
<td>Complex</td>
<td>Unique</td>
<td>10 GB</td>
</tr>
</tbody>
</table>
Server Requirements

The In-Memory database is written in Java and is cross-platform, this means that it will run on any Java enabled platform. It will make use of the memory allocated to Java. Yellowfin will support both 32-bit and 64-bit servers. However, for large databases it is assumed you will need a 64bit system to address RAM above 4GB.

The following minimum server requirements are recommended.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>2GHz or faster</td>
</tr>
<tr>
<td>RAM</td>
<td>Min 2 Gb</td>
</tr>
<tr>
<td>Operating System</td>
<td>Windows 2000 or later&lt;br&gt;Linux (Red Hat Enterprise Linux or SUSE Linux Enterprise Server recommended)&lt;br&gt;Solaris 7 or later&lt;br&gt;Mac OS X</td>
</tr>
<tr>
<td>Java</td>
<td>JSE 1.5 later</td>
</tr>
</tbody>
</table>
Performance testing & results

A critical step in the planning of your Yellowfin deployment is estimating the capacity requirements. This task approximates the initial computing resources needed to meet delivery objectives and service level agreements. Capacity estimation can be achieved through either an informal or a structured approach and should include consideration of the key activities undertaken by users. Since there is so much variation in the possible sizing of your in-memory database we highly recommend a structured performance testing phase. The following elements should be considered.

1. Test the time taken to initially cache the view. This information will help you to determine the appropriate way to maintain data in the in-memory database – such as refresh schedule times and incremental load versus full load.
   - Incremental loads mean less work for the system
   - Choosing to refresh overnight if the initial load is time consuming can mean better performance during the day

2. Time taken to run reports will help you decide whether to cache report results as well as the view
   - If the reports are going to be run regularly and the queries are complex it may be worth caching the results as well as the view.

3. How many concurrent users will be accessing the in-memory database. The more users you have running more complex queries the greater the amount of memory required will be.