CNET Case Study

- Our Problem
- How Does MySQL Fit?
- How CNET Made a Search Service
- Strengths
- Our Issues
- Conclusions
CNET Networks, Inc.
The Problem

• Search Vendor Changes
• Product End-of-Life
• Very High Replacement Costs
• Learning New Tech and Integration

• Yep. We need it “Faster, Better, Cheaper”.

MySQL®
Some of Our Requirements

• Sorting and Good Text Searching
• Fast Content Updates and Distribution
• Standalone Service Speaking HTTP and Providing XML
• Adequate Capacity and Scalability
• Inexpensive
• Rapid Development
Why Think of MySQL?

Technology
- “Fulltext Searching” and Sorting
- Replicate Content
- Source Code
- Robust and Customer-Hardened

Economics
- Search Vendor RFP
- Low Cost
- Easy Learning Curve
Our Internal Product

ATOMICS

Apache TO MySQL In CNET Search
How CNET Made a “Search Service”
Why Apache At All?

Mostly all “CNET” Reasons:

• Our HTTP Approach
• XML Out
• Logging and Status
• Other Goodies
How Do We Query?
Atomics Admin (artistindexdb)

Atoms

c10-gdl-at-qry3.cnet.com:8531

[STATUS]

[REAL APACHE]

c10-gdl-at-qry3.cnet.com:8531

[HARDWARE] [TRAFFIC] [PROBLEMS] [PROC STATUS] [INFO]

[MYSQL Server]

c10-gdl-at-qry3.cnet.com:11010

[HARDWARE] [TRAFFIC] [PROBLEMS] [PROC STATUS] [STATUS] [MYSQL HISTORY]

Make a Query

[SCHEMA EXPLORER] [TABLE STATUS] [CONFIG] [OPEN TABLES]

[FULL SQL INTERFACE]

SELECT * FROM artistindex ORDER BY docId DESC

Search

Assistance

[DOCUMENTATION] [FILE A BUGZILLA] [SEND EMAIL]

Fri Apr 15, 2005 03:37:38 PM
**/select mode**

<table>
<thead>
<tr>
<th>SQL Statement</th>
<th>SELECT SQL_CALC_FOUND_ROWS productID, addDate, name, shortDescription, MATCH (name, expressURL) AGAINST ('love song') as score FROM artistindex WHERE MATCH (name, expressURL) AGAINST ('love song') ORDER BY score DESC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Return Number Found</td>
<td>(SQL SELECT statement should contain SQL_CALC_FOUND_ROWS)</td>
</tr>
<tr>
<td>Protocol Version</td>
<td>1</td>
</tr>
<tr>
<td>Start Row</td>
<td>0</td>
</tr>
<tr>
<td>Maximum Rows Returned</td>
<td>20</td>
</tr>
</tbody>
</table>

**/raw mode**

<table>
<thead>
<tr>
<th>SQL Statement</th>
<th>SELECT COUNT(*) FROM artistindex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Protocol Version</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Search</td>
</tr>
</tbody>
</table>
### Atomics Search Results

**Status:** 0  
**Records Found:** 222  
**Number of Fields:** 5  
**Records Returned:** 20  
**Query time:** 5 ms

<table>
<thead>
<tr>
<th>productID</th>
<th>addDate</th>
<th>name</th>
<th>shortDescription</th>
<th>score</th>
</tr>
</thead>
<tbody>
<tr>
<td>100049511</td>
<td></td>
<td>Paal Nilsen-Love</td>
<td></td>
<td>8.8227500915527</td>
</tr>
<tr>
<td>100041826</td>
<td></td>
<td>Marcus Love &amp; Love Inc.</td>
<td></td>
<td>8.8227500915527</td>
</tr>
<tr>
<td>100040402</td>
<td></td>
<td>Love Spit Love</td>
<td></td>
<td>8.5022525787354</td>
</tr>
<tr>
<td>100588944</td>
<td>2005-01-06 19:46:23</td>
<td>Song Destruction</td>
<td>A guy playing an instrument. A guy recording some music. A guy that prefers to record instrumental music. A guy that just does this as a hobby.</td>
<td>8.2230520248413</td>
</tr>
<tr>
<td>100214124</td>
<td></td>
<td>The Culling Song</td>
<td>The Culling Song is a 5-piece rock band from Oxford, Ohio.</td>
<td>8.2230520248413</td>
</tr>
<tr>
<td>100375464</td>
<td></td>
<td>a streetlight song</td>
<td>alternate prog rock band... 3 piece... emotional and epic songs... use of standard rock and experimental methods and instruments...</td>
<td>8.2230520248413</td>
</tr>
</tbody>
</table>
This image shows a web page with a search result for a product. The search results are displayed in a table format with columns for `productld`, `addDate`, and `shortDescription`. The table includes rows with specific product IDs and dates.

The lower section of the image contains an XML code snippet. The XML code is a response from a database query, indicating a successful search with 222 records found and 5 fields. The query parameters include a `productId` and `productDate`, with a query time of 5 ms.

The XML code includes fields such as `productId`, `addDate`, and `shortDescription`, with values like `100270965` and `100049511` for product IDs, and dates like `10/09/65` and `10/04/26`.

The search results also mention songs from The Smiths, such as "Love Spit Love" and "How Soon Is Now?".
How Do We Populate DBs?

Our Index Building (from Scratch):

- ContentDB => MySQL Import File => Import, Build DB Indices

Our Incremental Changes:

- JDBC-based to MySQL which does INSERT, UPDATE, DELETE, or REPLACE
- Only to a Single Master Server
- Replication Distributes
Strengths

• Searches Can Be Blindingly Fast
  • “Natural Language” and BOOLEAN MODE

• Parallel Service
  • Content Replication to Client-Slaves
  • Auto-Recovery of Changes
  • HTTP Load Balancing Makes Capacity Upgrades Linear with Hardware Additions

• No Huge Vendor Capital Expense
Strengths (2)

• It’s a Database!
  • Changes are Reflected Immediately
  • Ultimately Flexible
  • Queries are Transparent and Clear

• We Get Upgrades
  • Apache and MySQL

• It Does Not Crash*
Strengths (3)

• Widely Known Leveraged Tech
  – Open Components and Specifications
    • SQL, Schema Design, and MySQL
    • Apache, Apache modules
    • HTTP, XML, XPP3
    • Easy to Expand and Add Features
Our Issues

- Extra Work and Thought
  - Creativity
  - Transformation Example: Stemming
- “Natural Language” vs. BOOLEAN MODE

$$\begin{align*}
(4 \times (\text{MATCH (title) against ('warez' in boolean mode})) + \\
2 \times (\text{MATCH (author) against ('warez' in boolean mode})) + \\
\text{MATCH (title, dek, body, author) against ('warez' in boolean mode})) \\
\end{align*}$$

as score
Our Issues (2)

• Complexity, Space, and Speed
• Retrieval Blind Spots
  • Minimum Word Lengths
• Non-Alpha-Numerics (plus ‘_’)
  • C++ vs. C#
  • AT&T
  • Wi-Fi
Conclusions?

- Atomics is a Workhorse and Very Capable
- Retrieval Issues
- Meets >80% Our Search Needs
- Scoring Algorithms – and Boolean Mode vs. Normal Mode
- Very Flexible