The QIX Engine - under the hood
Symbol tables, Bit-stuffed indices and State spaces

HIC – Henric Cronström
27 January, 2016
What is the QIX engine?

- Back-end in QlikView & Qlik Sense
- Same as QlikView engine – but improved

Row-based

QIX Engine

Columnar + some other optimizations

Qlik Sense client

QlikView 12 client

QlikView 11 client
The Seven Principal Engine Components

- Authorization
- Logical Inference Engine
- Calculation Engine
- Rendering Engine
- Export Features
- Script Engine
- Internal Database

Data
What happens during the **script execution?**

**SQL** `SELECT * FROM Orders;`
- Read record by record from source
- Create or Append to **Data** table
- Create or Append to **Symbol** tables

**SQL** `SELECT * FROM [Order Details];`
- Read record by record from source
- Create or Append to **Data** table
- Create or Append to **Symbol** tables

**SQL** `SELECT * FROM Products;`
- Read record by record from source
- Create or Append to **Data** table
- Create or Append to **Symbol** tables

Create **Synthetic keys and Tables**

Define **Keys between tables**

The script execution is **sequential**

If the data model is bad, many **Synthetic keys will be created.**
This takes time and eats memory.
**Data tables and Symbol tables**

<table>
<thead>
<tr>
<th>Product</th>
<th>List Price</th>
<th>In stock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bike Helmet</td>
<td>20</td>
<td>Yes</td>
</tr>
<tr>
<td>Boots</td>
<td>30</td>
<td>No</td>
</tr>
<tr>
<td>Bow tie</td>
<td>10</td>
<td>Yes</td>
</tr>
<tr>
<td>Cap</td>
<td>10</td>
<td>Yes</td>
</tr>
<tr>
<td>Tube Socks</td>
<td>5</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- The **Symbol tables** contain only distinct field values: Bit-stuffed indices and their clear-text meanings.
- The compressed **Data table** contains the same number of columns and rows as the source, but is filled with bit-stuffed indices, and thus extremely compact.

**Data Table**

<table>
<thead>
<tr>
<th>Product</th>
<th>List Price</th>
<th>In stock</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>01</td>
<td>1</td>
</tr>
<tr>
<td>010</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>11</td>
<td>0</td>
</tr>
</tbody>
</table>

**Symbol Tables**

<table>
<thead>
<tr>
<th>Product</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>Bike Helmet</td>
</tr>
<tr>
<td>001</td>
<td>Boots</td>
</tr>
<tr>
<td>010</td>
<td>Bow tie</td>
</tr>
<tr>
<td>011</td>
<td>Cap</td>
</tr>
<tr>
<td>100</td>
<td>Tube Socks</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List Price</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>20</td>
</tr>
<tr>
<td>01</td>
<td>30</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>11</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>In stock</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Yes</td>
</tr>
<tr>
<td>1</td>
<td>No</td>
</tr>
</tbody>
</table>
The Seven Principal Engine Components

- Authorization
- Logical Inference Engine
- Calculation Engine
- Script Engine
- Rendering Engine
- Export Features
- Data
- Symbols
- Internal Database
Implications of **Symbol tables**

- Many distinct values in a field ▶ Longer bit-stuffed index *and* more rows in the symbol table (more RAM usage and more CPU usage)
  - If possible, reduce the number of distinct values. E.g.
    ```
    Time( Floor( Time, 1/24/60 ), 'hh:mm' )
    ```

- Long field values ▶ Longer values in symbol table (more RAM usage)
  - If possible, reduce size of field value. E.g.
    ```
    Autonumber( ProductID & '|' & CustomerID & '|' & Date )
    ```
**Colors and States**

- Two Flags:
  - Every field value is either **Selected** or **Not Selected**
  - Every field value is either **Possible** or **Not Possible**
- These two state flags are *independent* of each other
## Colors and States II

<table>
<thead>
<tr>
<th>Name</th>
<th>Cylinders</th>
</tr>
</thead>
<tbody>
<tr>
<td>Audi 5000</td>
<td>5</td>
</tr>
<tr>
<td>Audi 5000S DI</td>
<td>4</td>
</tr>
<tr>
<td>Mercedes 300D</td>
<td>3</td>
</tr>
<tr>
<td>Audi 4000</td>
<td>6</td>
</tr>
<tr>
<td>AMC Concord</td>
<td>8</td>
</tr>
<tr>
<td>AMC Concord D1</td>
<td></td>
</tr>
<tr>
<td>AMC Concord DI</td>
<td></td>
</tr>
</tbody>
</table>

- Green with tick mark = Selected
- Dark gray with tick mark = Selected excluded
- Dark gray without tick mark = Non-selected excluded
- Light gray = Alternative
Colors and States III

<table>
<thead>
<tr>
<th></th>
<th>Possible</th>
<th>Not possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selected</td>
<td>Selected</td>
<td>Selected excluded</td>
</tr>
<tr>
<td>Not selected</td>
<td>Optional</td>
<td>Alternative</td>
</tr>
</tbody>
</table>

Future?

<table>
<thead>
<tr>
<th></th>
<th>Possible</th>
<th>Not Possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selected</td>
<td>Selected</td>
<td>Selected excluded</td>
</tr>
<tr>
<td>Not selected</td>
<td>Optional</td>
<td>Alternative</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Possible</th>
<th>Not Possible</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selected</td>
<td>Selected</td>
<td>Selected excluded 1</td>
</tr>
<tr>
<td>Not selected</td>
<td>Optional</td>
<td>Alternative</td>
</tr>
</tbody>
</table>

Excluded 2
Excluded 3
Logical Inference and State Vectors

- Every time you click, the states and colors for all field values are calculated. This is the Logical Inference.
- The states are stored in the State Vectors.
The Seven Principal Engine Components

- Authorization
- Logical Inference Engine
- Calculation Engine
- Rendering Engine
- Export Features
- Script Engine
- Data
- Symbols
- State Vectors
- Cache
Logical Inference

The state evaluation propagates from table to table. The state is stored in the state vectors.
Calculations vs Logical Inference

1. Logical inference on Data model

2. Aggregations in Transient Hypercubes (Objects)

QIX Internal Database
Multithreading

One or several threads per user

One thread per object

Several threads per chart

Selection

Logical inference

List box

Find values or combinations

Calculate aggregation

Chart

Logical inference

Calculation engine

Single-threaded! Possible bottle-neck!
The Calculation Engine and State Vectors

The Calculation Engine calculates most objects, which contain expressions with aggregation functions.

- Aggregations are functions that loop over the possible (State=1) rows in the Data table: `Count()`, `Sum()`, `Avg()`, `Concat()`, `Only()`, `Min()`, `Max()`, etc.
- Some system functions use the Symbol tables and their state vectors: `GetSelectedCount()`, `GetPossibleCount()`, `GetExcludedCount()`, etc.

```
<table>
<thead>
<tr>
<th>Product</th>
<th>List Price</th>
<th>In stock</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>00</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>001</td>
<td>01</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>010</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>011</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>11</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>List Price</th>
<th>Value</th>
<th>Selection</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>20</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>01</td>
<td>30</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
```
How does the **Calculation Engine** work?

- **Sum**\( \text{Quantity} \times \text{ListPrice} \) = ?
  - Answer: 42. \( (1 \times 3 + 2 \times 5 + 3 \times 3 + 4 \times 5) \)
  - Calculated in a *temporary* table: Transactions \( \bowtie \) Products.

Considerable time single-threaded! Possible bottle-neck!
How does the **Calculation Engine** work?

- The QIX Engine determines which table to use for the aggregation.

- It is determined by the field references inside the aggregation:
  
  \[
  \text{Sum( Quantity )} \rightarrow \text{The table with Quantity}
  \]
  
  \[
  \text{Sum( ListPrice )} \rightarrow \text{The table with ListPrice}
  \]
  
  \[
  \text{Sum( Quantity } \times \text{ ListPrice )} \rightarrow \text{A temporary table}
  \]

- If fields from different tables are used inside the same aggregation, lookup tables or a temporary table is created. This is a single threaded process.
The Calculation Engine oddities

- $\text{Sum}(1) = \, ?$ (No field references!)
  - Answer: Not determined which table to use!
- $\text{Sum}(0*\text{UnitPrice} + 1) = \, ?$
  - Answer: 4: Calculated in the Transactions table
- $\text{Sum}(0*\text{ListPrice} + 1) = \, ?$
  - Answer: 2. Calculated in the Products table
- $\text{Sum}(0*\text{UnitPrice} + \text{ListPrice}) = \, ?$
  - Answer 16: Calculated in a temporary table: Transactions $\bowtie$ Products
Implications of the **Calculation Engine**

- Field references from different tables
  - More RAM usage, more CPU time
    - For performance, you may consider moving a number to the "correct" table, so that the creation of a temporary table is avoided.

- Field references from different tables
  - Possibility for **incorrect calculations**
    - Avoid If()-functions. Use Set Analysis instead. E.g. avoid
      ```sql
      Sum( If( OrderMonth=5, InvoiceAmount ) )
      ```
      Instead, use
      ```sql
      Sum( {$<OrderMonth={5}>} InvoiceAmount )
      ```
Set Analysis

- Set Analysis is merely an additional set of state vectors, created by the Logical Inference Engine.
Set Analysis

\textbf{Sum( Sales )}

- Aggregate over the records, using the normal state space
Set Analysis

\textbf{Sum}( \{\ldots\} \textbf{Sales} )

- Aggregate over the records, using a different state space
- This first pair of curly brackets define a Record Set: A set of records in a data table.
Set Analysis

\[
\text{Sum}\left( \{ \$ \text{<Month>...> } \} \text{ Sales}\right)
\]

- Set modifier: This new state space should be created from the normal state space, but modified by a selection in Month
Set Analysis

**Sum( \{${<Month= \{\ldots\}>}$ Sales )**

- This new pair of curly brackets denotes the *Element Set*: A set of values in a symbol table.
Set Analysis

\( \text{Sum} \left( \{ \text{\$<Month={"..."}>} \}\right) \text{ Sales } \)

- Double quotes denote a search for field values
- Do not use single quotes!
Set Analysis

\[ \text{Sum}( \{\$<\text{Month}=\{" \leq \ldots "} \} \} \text{ Sales } ) \]

- A search string with a relational operator means a numeric search.
Set Analysis

\[ \text{Sum}( \{ \$<\text{Month}={"\leq \$(...) "}\} \} \text{Sales} ) \]

- A dollar expansion means that something is inserted \textit{before} the parsing of the expression.
Set Analysis

\[ \text{Sum} \left( \{\$<\text{Month}={"\leq\$( =\ldots )"}\}\} \text{Sales} \right) \]

- An equals sign inside the dollar expansion means an expression instead of a variable
Set Analysis

\[ \text{Sum} \left( \{<\text{Month}=>\text{Max}(X)\} \right) \text{ Sales} \]

- This expression needs to be an aggregation
Implications of **Set Analysis**

- Set Expressions are evaluated by the *Logical Inference Engine*, to create an alternative state space.

- It is done *before* the aggregation made by the *Calculation Engine*. Hence, a Set Expression cannot be evaluated row-by-row in the chart.
Set Analysis

\[ \text{Sum(} \{ \text{Month} \leq 4 \} \cap \text{Sales}) \]

Order of events:

1. Dollar expansion
2. Parsing of expression
3. Set Analysis expression evaluation
4. Cube expansion and aggregation
The Seven Principal Engine Components

QlikView® C-client:
- All rendering is Client-side

QlikView® Ajax:
- Tables are rendered client-side
- Graphs are rendered server-side

Qlik® Sense:
- All rendering is Client-side
Questions & Answers
Thank You