# QlikView and Qlik Sense - DIRECT DISCOVERY

Technical Addendum

Published: November, 2012

Version: 6.6

Last Updated: April, 2015

Version History

| Version | Change | Sections |
|---------|--------|----------|
| 6.4 | Can and cannot do | Section 2 |
| 6.4 | Multi-Table scenarios and restrictions | Section 3.8 |
| 6.5 | Sense inclusions | Section 3.3<br>Section 3.3.3<br>Section 3.6 |
| 6.6 | Sub Queries vs in clause SQL generation | Section 3.4<br>Section 3.8 |

# Contents

# 1 Overview

This document provides a technical overview of the Direct Discovery feature.

Direct Discovery capability expands the potential use cases for Business Discovery, enabling business users to conduct associative analysis on larger data sources. It provides complete associative experience on top of data coming directly from external larger data sources, and enables users to combine that big data with data stored in memory. With Direct Discovery, business users can leverage any data useful for analysis without scalability limitations.

The following part of the paper provides a technical overview of implementing and using Direct Discovery. It also provides information on the best practices and limitations of this feature with this release.

# 2 What is Direct Discovery

The Direct Discovery capability combines the associative capabilities of the in memory dataset with a query model where not all of the source data is directly loaded into the data model, instead the aggregated query result is passed back to the user interface. The direct discovery data set is still part of the associative experience where the user can navigate both on the in-memory data and the direct discovery data associatively.

- Can be used to build aggregated charts on **homogeneous** large data sets
- Can be used to look at detail records in a table box on large data sets
- Can reflect **updated** records without reloads (not new records)
- Can support more than one Direct Discovery table in certain scenarios

- Not as fast as in memory apps
- Will always be slower compared to SQL query run times on source due to associative model calculation times
- Not a solution for scalability/performance issues in the underlying source
- Not designed to convert all tables in apps into Direct Discovery mode
- Not a real time solution

# 3  Technical Details

With the unique associative experience in combination with Direct Discovery users are able to navigate and interact with data by a combination of methods.

## 3.1  How Does Direct Discovery Work?

QlikView/Qlik Sense determines which data resides in-memory and which data is direct discovery data by using the special script syntax, "DIRECT QUERY". This syntax allows certain data elements not to be loaded into the data model during the script reload process, but still available for query purposes from the user interface and to be combined for analysis with the in memory dataset.

Once the direct discovery structure is established, the direct discovery fields can be used with certain QlikView and Qlik Sense objects. When a direct discovery field is used in a chart the corresponding SQL query runs on the external data source. The result of the query will be displayed in the chart. When selections are made in the application the associated data values of the direct discovery fields will be used in the WHERE conditions of the queries. With each selection, the direct discovery charts will be recalculated.  It is possible to use calculation condition feature of the QlikView charts to set a condition indicating when the chart should be calculated. Until that condition is met, QlikView will not run queries and the chart will not be calculated. Please note that QlikView will execute SQL queries on the data source for some of the list boxes that use direct discovery fields (filter box in Qlik Sense). This is required to achieve the associative navigation capability.

## 3.2  Data Loading

Within the script editor a new syntax is introduced to connect to data in direct discovery form. Traditionally the following syntax is required to load data from a database table:

```
ODBC CONNECT TO AdWorks;

    LOAD CustomerID,
       SalesPersonID,
       SalesOrderID,
       OrderDate,
       month([OrderDate]) as OrderMonth,
```

```
    year([OrderDate]) as OrderYear,
    SubTotal,
    TaxAmt,
    TotalDue
    DueDate,
    ShipDate,
    AccountNumber,
    CreditCardApprovalCode,
    rowguid,
    ModifiedDate
SQL SELECT
    CustomerID,
    SalesPersonID,
    SalesOrderID,
    OrderDate,
    SubTotal,
    TaxAmt,
    TotalDue
    RevisionNumber,
    DueDate,
    ShipDate,
    AccountNumber,
    CreditCardApprovalCode,
    rowguid,
    ModifiedDate
FROM AdventureWorks.Sales.SalesOrderHeader;
```

Figure 1. Traditional Load Script Syntax

To invoke the direct discovery method, the keyword "SQL SELECT" is replaced with "DIRECT QUERY" and additional keywords – DIMENSION,MEASURE,DETAIL and NATIVE are introduced:

```
ODBC CONNECT TO AdWorks;

    DIRECT QUERY
      DIMENSION
        CustomerID,
        SalesPersonID,
        SalesOrderID,
        TerritoryID,
        OrderDate,
        NATIVE('month([OrderDate])') as OrderMonth,
        NATIVE('Year([OrderDate])') as OrderYear
      MEASURE
        SubTotal,
        TaxAmt,
        TotalDue
      DETAIL
        DueDate,
        ShipDate,
        AccountNumber,
        CreditCardApprovalCode,
```

```
        rowguid,
        ModifiedDate
FROM AdventureWorks.Sales.SalesOrderHeader;
```

Figure 2. QlikView Load Script Syntax for Direct Discovery

In the example above, the source data table "SalesOrderHeader" is loaded with the use of "DIRECT QUERY" and "DIMENSION" keywords, only columns CustomerID, TerritoryID, SalesPersonID, SalesOrderID and OrderDate are loaded into memory as symbol tables. Other columns following the "MEASURE" and "DETAIL" keywords exist in the source data table within the database and they are not part of the in memory data model. Both measure and detail fields are fields that QlikView/Qlik Sense is aware of on a "meta level". The actual data of measure/detail fields reside only in the database but the fields may be used in expressions. More information on the measure/detail fields are provided in the next section. Please note that preceding load cannot be used with Direct Discovery as it only relates to the data that is loaded in memory.

The script would produce the following SQL passed through to the source database:

> *SELECT DISTINCT "CustomerID" FROM "AdventureWorks"."Sales"."SalesOrderHeader"*
> *SELECT DISTINCT "TerritoryID" FROM "AdventureWorks"."Sales"."SalesOrderHeader"*
> *SELECT DISTINCT "SalesPersonID" FROM "AdventureWorks"."Sales"."SalesOrderHeader"*
> *SELECT DISTINCT "SalesOrderID" FROM "AdventureWorks"."Sales"."SalesOrderHeader"*
> *SELECT DISTINCT "OrderDate" FROM "AdventureWorks"."Sales"."SalesOrderHeader"*
> *SELECT DISTINCT month([OrderDate]) FROM "AdventureWorks"."Sales"."SalesOrderHeader"*
> *SELECT DISTINCT year([OrderDate]) FROM "AdventureWorks"."Sales"."SalesOrderHeader"*

The direct discovery data can be joined with the in-memory data with the common field names (Figure 3). In this example, Territory, Customer and SalesPerson tables are loaded in memory and joined to the direct discovery data via various dimension fields. This allows the user to associatively navigate both on the direct discovery and in memory data sets.
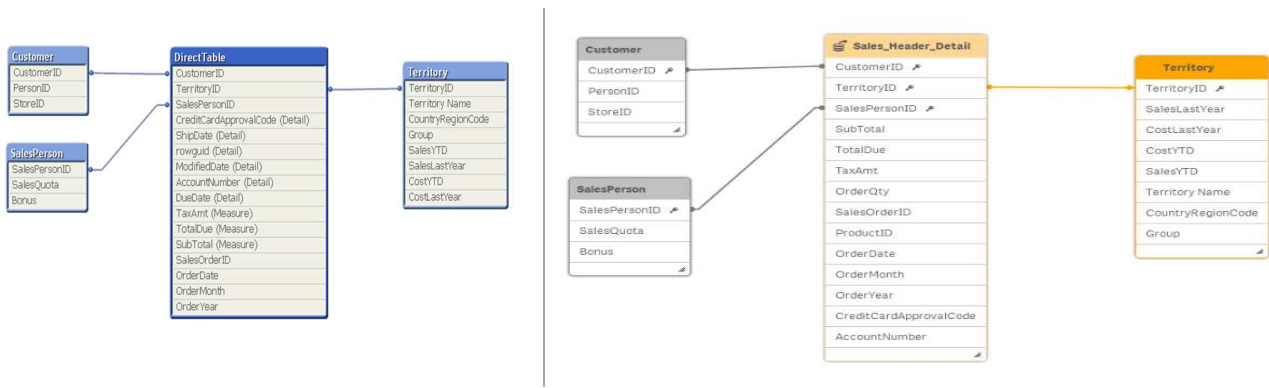
Figure 3. Associative Data Model for Direct Discovery and In-memory data sets

As figure 4 demonstrates, the business users can associatively make selections on the data sets visualise charts containing direct discovery fields.
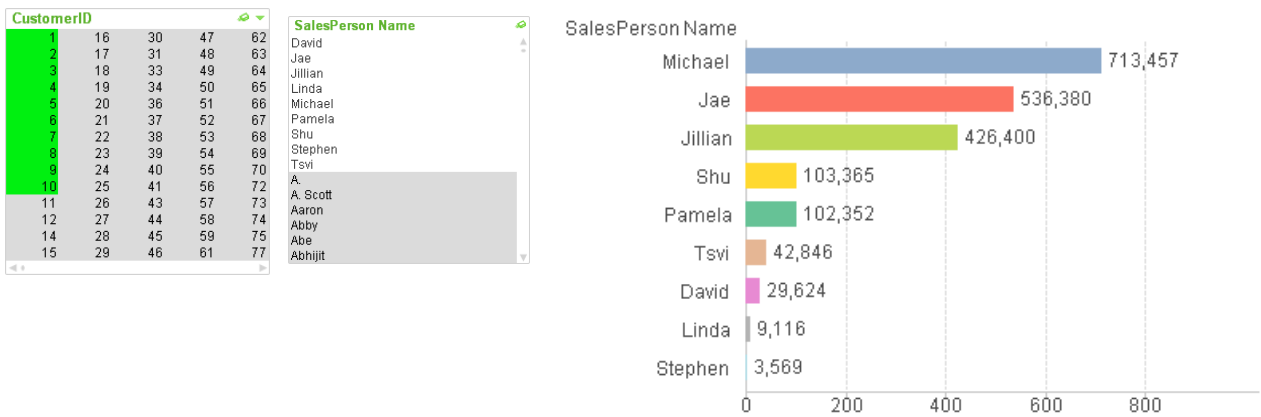


Figure 4. Associative Business Discovery on direct discovery and in-memory data sets

It is also possible to execute source Database SQL functions with the direct discovery table on the load script by using the keyword NATIVE which should be used within single quotation marks.

```
NATIVE('month([OrderDate])') as OrderMonth,
NATIVE('Year([OrderDate])') as OrderYear
```

Statements containing single quotes will require additional quotes:

```
NATIVE('CAST(DAY(SYSDATETIME()) AS VARCHAR(2)) + ''/'' + CAST(MONTH(SYSDATETIME())
AS VARCHAR(2)) + ''/'' + RIGHT(CAST(YEAR(SYSDATETIME()) AS VARCHAR(4)), 2) as
[D/M/YY]
```

Figure 5. Use of the NATIVE function with the direct discovery table in the load script

It is also possible to use a WHERE statement on the load script with the direct discovery table.

### 3.2.1 Background and definition of measure fields

A measure field is a field that QlikView/Qlik Sense is aware of on a "meta level". When the keywords "DIRECT QUERY" in conjunction with "MEASURE" are in the load script, only the metadata of the fields is loaded from the source table. The actual data of a measure field resides only in the database and can be used in expressions. The idea is that a measure field will be treated as any other field when the user works with expressions. The measure fields are required to have an aggregation function when used on the user interface (except in Table boxes). When the chart generates queries including measure direct discovery fields, it uses a GROUP BY on the SQL statement with the corresponding dimension.

The reason to introduce the concept of measure fields is for larger data sets forcing an aggregation to be done on the database and have the result set passed back.
For example, if the measure expression Sum(SubTotal) is used in a chart the database will do the sum. The result is that the usage of the direct discovery fields from the database will be transparent for the user.

### 3.2.2 Dimension direct discovery fields

Dimension direct discovery fields are the fields that are listed after the DIRECT QUERY and DIMENSION keywords in the load script. The unique data values of these fields are loaded into the in memory symbol tables. The main use case for dimension fields are;

- To define dimension chart values
- To create the association SQL between the in-memory data and the direct discovery data
- To define list boxes with direct discovery data

Please note that when the dimension direct discovery fields are used in list boxes, the data values displayed in the list boxes will not get updated UNLESS the application is reloaded.

### 3.2.3 Detail direct discovery fields

Drill to details is a new capability introduced in 11.2 SR5 and allows "detail" records to be displayed in a QlikView table box only.  A detail record is one to which no aggregation is applied and effectively just generates a select statement from QlikView and renders the appropriate values.  To cater for the potential large volume of data which could be rendered an additional Set variable is introduced:

```
SET DirectTableBoxListThreshold = 100000;
```

**The default value is 1000.**

As part of the drill to details functionality additional syntax has been introduced this follows the DIRECT QUERY statement:

```
DIRECT QUERY
  DIMENSION
    CustomerID,
    SalesPersonID,
    SalesOrderID,
    OrderDate,
    NATIVE('month([OrderDate])') as OrderMonth,
     NATIVE('Year([OrderDate])') as OrderYear
  MEASURE
    SubTotal,
    TaxAmt,
    TotalDue
  DETAIL
    DueDate,
    ShipDate,
    AccountNumber,
    CreditCardApprovalCode,
    rowguid,
    ModifiedDate
FROM AdventureWorks.Sales.SalesOrderHeader;
```

Figure 6. Use of DETAIL syntax in the load script

DETAIL - These are the fields that the user may want to display in a drill-to-details table box but that should not be involved in any chart expressions.  The data will be displayed at the lowest level without aggregation within a table box.

### 3.2.4 Detach direct discovery fields

The Detach keyword has been introduced to flag certain dimension fields NOT to be part of the associative experience BUT to be part of the query generated passed to the data source. The main use case for this is for large volume granular dimension values to increase the chart rendering time due to bypassing the associative capability; section 7 describes this in more detail.

As part of the DETACH functionality additional syntax has been introduced to flag the dimensions to be used in this mode:

```
DIRECT QUERY
  DIMENSION
    SalesPersonID,
    OrderDate,
    NATIVE('month([OrderDate])') as OrderMonth,
     NATIVE('Year([OrderDate])') as OrderYear
  MEASURE
   SubTotal,
   TaxAmt,
   TotalDue
  DETAIL
    DueDate,
    ShipDate,
    AccountNumber,
    CreditCardApprovalCode,
    rowguid,
    ModifiedDate
  DETACH
    SalesOrderID,
    CustomerID
FROM AdventureWorks.Sales.SalesOrderHeader;
```

Figure 7. Use of DETACH syntax in the load script

In the above example SalesOrderID and CustomerID would not be part of the associative experience but will form part of the WHERE clause passed to the underlying data source if selected in a list box.

### 3.3 Creating QlikView Objects with Direct Discovery Fields

Due to the interactive and SQL specific nature of QlikView Direct Discovery, only certain QlikView objects can use measure direct discovery fields. Statistics boxes are not supported with measure direct discovery fields.

On the field list, additional information is provided to notify the user that the field is a measure field (figure 8). This is useful information as some of the measure fields from the big data sources may have a billion values and using these fields on the user interface may slow down the user experience. Detailed attention should be paid with the use of measure fields. Although the data values for the measure fields are not loaded in memory, they still consume memory and CPU once they are used on the user interface of a QlikView application.
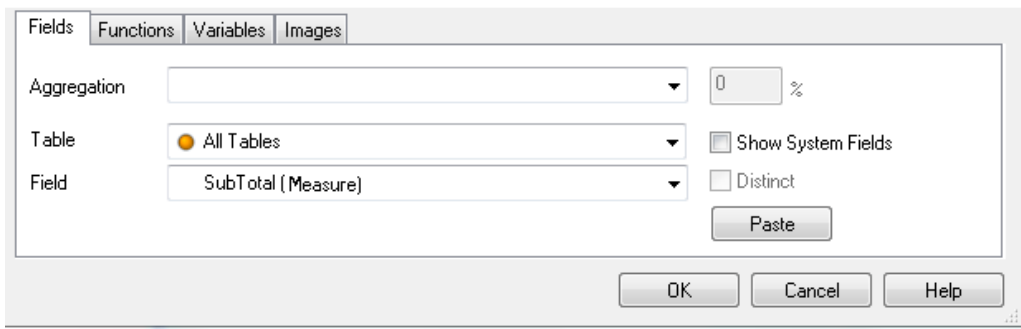


Figure 8. The direct discovery fields are marked with the keyword "Measure" on the field list.

In Qlik Sense direct discovery fields are populated into the library and are indicated with an icon as shown in figure 9:
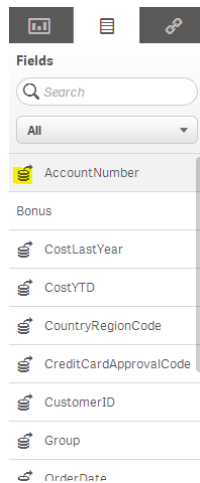


Figure 9. Direct discovery fields in Qlik Sense.

### 3.3.1 How to create list/filter boxes with direct discovery fields

The traditional way of creating list box is used to create list/filter boxes with the dimension direct discovery fields (fields that are listed after "DIRECT QUERY DIMENSION" keywords in the load script).

In the previous example CustomerID, SalesPersonID,SalesOrderID ,TerritoryID, OrderDate, OrderMonth and OrderYear fields can be used in list boxes as these are the dimension direct discovery fields.

Please note that to achieve the associative navigation capability QlikView will execute SQL queries on the direct discovery data source when there is a selection made on the list box using a direct discovery field.

It is possible to use the expression option of the list box with a measure direct discovery field. The aggr() function should be used to show the aggregated value of the measure field for any dimension field. For example, the list box expression "aggr(sum(SubTotal),CustomerID))" will list the aggregated SubTotal values for CustomerId names in a list box.

### 3.3.2 How to create QlikView charts with direct discovery fields

The traditional way of creating charts is used to create a chart using direct discovery fields. It is possible to use the direct discovery fields as dimensions and/or expressions on the charts. However, please note that only measure direct discovery fields can be used in expressions and dimension direct discovery fields can be used as dimensions. In the previous example, only CustomerID, SalesPersonID,SalesOrderID,TerritoryID,OrderDate, OrderMonth and OrderYear fields can be used as chart dimensions and SubTotal,TaxAmt and TotalDue fields can be used as chart expressions.

For the QlikView charts that use only direct discovery fields, all of the aggregations are done on the database. If the QlikView chart has fields both from the in-memory and direct discovery tables, a second level aggregation is done on the chart level for the in memory fields once the database level aggregations are done.
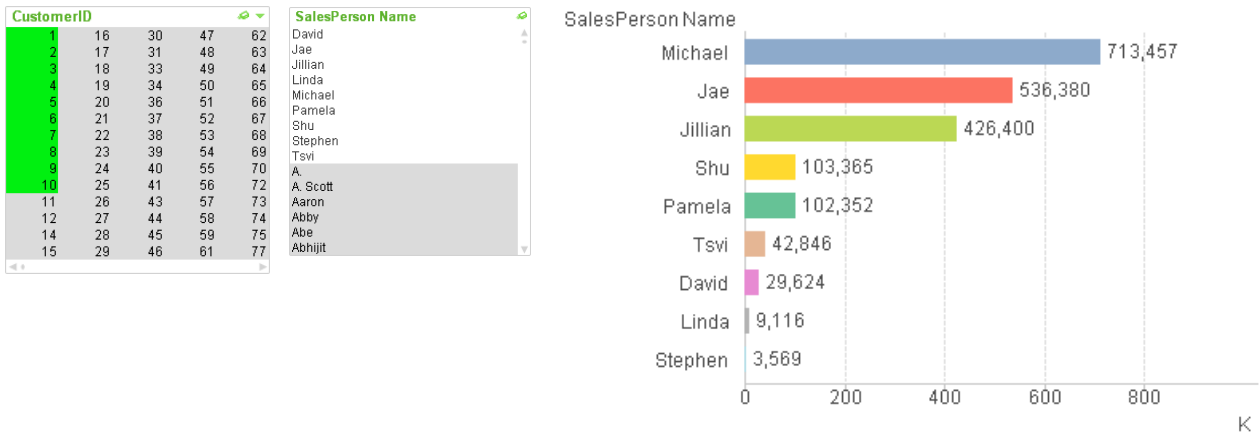
Figure 10. QlikView chart/selections using direct discovery fields as dimension and expression

The charts in Figure 10 would produce the following SQL pushed down to the database:

Associations are NOT stored in memory and are built from the source:

*SELECT DISTINCT "SalesPersonID" FROM "AdventureWorks"."Sales"."SalesOrderHeader" WHERE "CustomerID" IN ( 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 )*

And for the chart:

*SELECT "SalesPersonID", SUM("SubTotal") FROM "AdventureWorks"."Sales"."SalesOrderHeader" WHERE "CustomerID" IN ( 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ) GROUP BY "SalesPersonID"*

*SELECT  SUM("SubTotal") FROM "AdventureWorks"."Sales"."SalesOrderHeader"    WHERE "CustomerID" IN ( 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 )*

As QlikView Direct Discovery generates SQL as its base code, not all QlikView expression functionality will be compatible with the feature. The expression functions that are supported with this initial release are; Sum, Avg, Count, Min, Max. Please note that it would be important to consider the type of aggregations that the source database supports when using direct discovery. For example, most SQL database support DISTINCT in any aggregation, but Google BigQuery only supports COUNT(DISTINCT …).

Most of the QlikView chart functionalities (interactive sorting, formatting, visual clues, dimension limits etc...) are still available to be used on the charts that use direct discovery

fields. This provides the capability to leverage the same ease of use and rapid app development QlikView experience on the big data analysis.

It is also possible to use the in memory fields and direct discovery fields on the same chart. In these cases, QlikView associative technology automatically handles the associations between the in memory fields and direct discovery fields. The business user still has the same easy to develop QlikView experience and does not need to worry about how the data sets are joined together.

### 3.3.3 How to display drill to detail records

Detail records can ONLY be displayed in a Table box, the Table box MUST contain a DIMENSION field (CustomerID) and can also contain both DETAIL (CreditApprovalCode, AccountNumber, DueDate) and MEASURE (SubTotal, TaxAmt) fields.

| CustomerID △ | AccountNumber | DueDate | CreditCardApprovalCode | SubTotal | TaxAmt |
|---|---|---|---|---|---|
| 1 | 10-4020-000001 | 13/02/2002 12:00:00 AM | 105493Vi99678 | £34,066.19 | £2,725.30 |
| 1 | 10-4020-000001 | 13/08/2001 12:00:00 AM | 105571Vi99678 | £13,216.05 | £1,057.28 |
| 1 | 10-4020-000001 | 13/11/2001 12:00:00 AM | 105516Vi99678 | £23,646.03 | £1,891.68 |
| 1 | 10-4020-000001 | 13/05/2002 12:00:00 AM | 105533Vi99678 | £31,423.52 | £2,513.88 |
| 2 | 10-4020-000002 | 13/02/2004 12:00:00 AM | 17172Vi51504 | £3,668.73 | £293.50 |
| 2 | 10-4020-000002 | 13/11/2002 12:00:00 AM | 15790Vi51504 | £4,949.86 | £395.99 |
| 2 | 10-4020-000002 | 13/08/2002 12:00:00 AM | 15832Vi51504 | £9,216.36 | £737.31 |
| 2 | 10-4020-000002 | 13/05/2004 12:00:00 AM | 17000Vi51504 | £822.01 | £65.76 |
| 2 | 10-4020-000002 | 13/11/2003 12:00:00 AM | 15802Vi51504 | £4,106.65 | £328.53 |
| 2 | 10-4020-000002 | 13/08/2003 12:00:00 AM | 15862Vi51504 | £3,534.17 | £282.73 |
| 2 | 10-4020-000002 | 13/02/2003 12:00:00 AM | 15786Vi51504 | £1,574.12 | £125.93 |
| 2 | 10-4020-000002 | 13/05/2003 12:00:00 AM | 15795Vi51504 | £1,751.60 | £140.13 |
| 3 | 10-4020-000003 | 13/03/2004 12:00:00 AM | 38415Vi26986 | £15,431.52 | £1,234.52 |
| 3 | 10-4020-000003 | 13/12/2001 12:00:00 AM | 38408Vi26986 | £17,419.61 | £1,393.57 |
| 3 | 10-4020-000003 | 13/03/2003 12:00:00 AM | 38398Vi26986 | £27,899.96 | £2,232.00 |
| 3 | 10-4020-000003 | 13/12/2002 12:00:00 AM | 38392Vi26986 | £51,198.54 | £4,095.88 |
| 3 | 10-4020-000003 | 13/06/2003 12:00:00 AM | 38428Vi26986 | £47,138.00 | £3,771.04 |
| 3 | 10-4020-000003 | 13/06/2002 12:00:00 AM | 38384Vi26986 | £11,796.18 | £943.69 |
| 3 | 10-4020-000003 | 13/09/2003 12:00:00 AM | 38442Vi26986 | £83,436.18 | £6,674.89 |
| 3 | 10-4020-000003 | 13/09/2002 12:00:00 AM | 38431Vi26986 | £71,112.13 | £5,688.97 |
| 3 | 10-4020-000003 | 13/06/2004 12:00:00 AM | 37038Vi26986 | £31,559.75 | £2,524.78 |
| 3 | 10-4020-000003 | 13/03/2002 12:00:00 AM | 38389Vi26986 | £18,285.64 | £1,462.85 |
| 3 | 10-4020-000003 | 13/12/2003 12:00:00 AM | 37443Vi26986 | £40,109.06 | £3,208.72 |
| 3 | 10-4020-000003 | 13/09/2001 12:00:00 AM | 38409Vi26986 | £18,555.81 | £1,484.47 |
| 4 | 10-4020-000004 | 13/07/2003 12:00:00 AM | 96519Vi75503 | £73,368.09 | £5,869.45 |

Figure 11. Drill to details table box

Potentially a large amount of data could be retrieved from this process as such the table box will not display any data until the following SQL is executed:

*SELECT COUNT(*) FROM "AdventureWorks"."Sales"."SalesOrderHeader*

If the result of the count(*) is under the threshold of the following set variable

```
SET DirectTableBoxListThreshold  = 100000;
```

From the script then the data will be retrieved from the source database with the following SQL:

*SELECT "CustomerID", "CreditCardApprovalCode", "AccountNumber","DueDate", "SubTotal", "TaxAmt" FROM "AdventureWorks"."Sales"."SalesOrderHeader"*

Within Qlik Sense the Table box and chart have been combined, this results in a limitation of the type of fields allowed to be used, specifically Detail and measure fields cannot be shown in the same table.  The following error is shown if attempted:
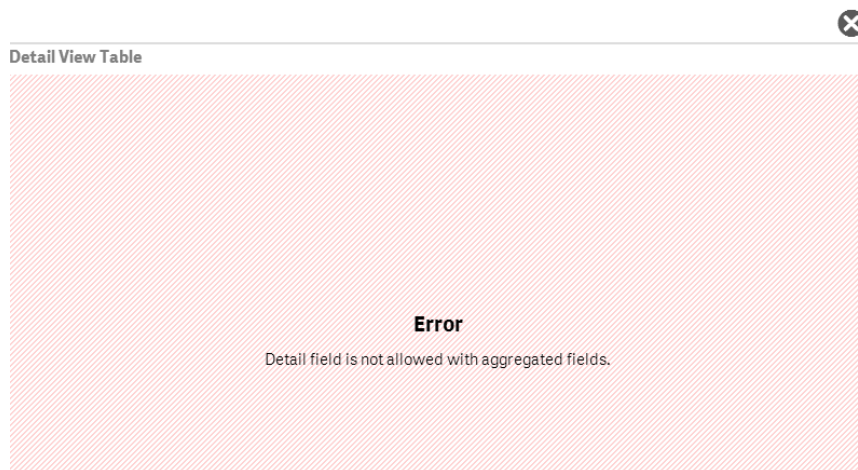


Figure 12. Drill to details table box

## 3.4 Supported data types and set statements

All data types are supported, however there maybe cases where specific source data formats need to be defined especially working with the date fields. This can be done on the load script by using the "SET Direct…" syntax.

| Set Statement | Description |
|---|---|
| DirectDateFormat | Controls the format of the date literals sent to the database |
| DirectTimestampFormat | Controls the format of the datetime literals sent to the database |
| DirectTimeFormat | Controls the format of the time literals sent to the database |
| DirectUnicodeStrings | Controls whether string literals are prepended with the ANSI standard wide character marker N. Not all databases support this. Set to **TRUE** to enable; any other setting disables. |
| DirectMoneyFormat | Controls the format of the money literals sent to the database, This is not a display format, so it should not include currency symbols or thousands separators, default '#.0000' |
| DirectMoneyDecimalSep | Controls the decimal separator used in money literals sent to the database. Note that this character must match the character used in **DirectMoneyFormat.** Default is '.' |
| DirectStringQuoteChar | Controls the character to be used to quote the strings in a generated query. The default is a single quote. |
| DirectIdentifierQuoteChar | Controls the quoting of identifiers in a generated query. This can be set to either one character (such as a double quote) or two (such as the pair "[]"). The default is a double quote. |
| DirectCacheSeconds | Controls the amount of time a query result set from an application is stored in memory cache, default setting is 3600 seconds. |
| DirectTableBoxListThreshold | Controls the amount of drill to detail data available to be shown in a table box, the default is 1000 rows |
| DirectConnectionMax | Controls it the number of parallel calls to the database by using the connection pooling capability, the default is 1 |
| DirectDistinctSupport | Changes the default syntax of SELECT DISTINCT to use GROUP BY for sources |

| | |
|---|---|
| | which do not support DISTINCT |
| **DirectIdentifierQuoteStyle** | For non-standard ANSI quoting anything other than 'ANSI' in the statement will change the syntax |
| **SQLSessionPrefix** *(Direct Discovery statements only are logged)* | For setting Teradata Querybanding parameters for the session e.g. `'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & ' FOR SESSION;';` |
| **SQLQueryPrefix** *(Direct Discovery statements only are logged)* | For setting Teradata Querybanding parameters for the query e.g. `'SET QUERY_BAND = ' & Chr(39) & 'Who=' & OSuser() & ';' & Chr(39) & ' FOR TRANSACTION;';` |
| **DirectEnableSubquery** | For use with high cardinality multi-table scenarios, setting this to 'true' will generate sub queries in the SQL vs large IN clauses. NOTE when set to 'true' it's not permitted to load non Direct Discovery Tables. Default is false |

## 3.5  Data Sources

Direct Discovery can only be used against SQL compliant data sources. The following data sources are supported;

- ODBC/OLEDB data sources - All ODBC/OLEDB sources are supported, including SQL Server, Teradata and Oracle.
- Custom connectors which support SQL – SAP SQL Connector, Custom QVX connectors for SQL compliant data stores.

Both the 32-bit and 64-bit connections are supported.

### 3.5.1 SAP as a Data Source

Direct discovery can be used in conjunction with the SAP SQL Connector only and requires the following parameters in the set variables:

```
SET DirectFieldColumnDelimiter=' ';
SET DirectIdentifierQuoteChar=' ';
```

SAP uses OpenSQL which delimits columns with a space rather than a comma so the above set statements will cater for this difference to ANSI SQL.

### 3.5.2 Google Big Query as a Data Source

Direct discovery can be used in conjunction with Google Big Query and requires the following parameters in the set variables:

```
SET DirectDistinctSupport=false;
SET DirectIdentifierQuoteChar='[]';
SET DirectIdentifierQuoteStyle='big query';
```

Google Big Query does  not support Select distinct or quoted column/table names and has non ANSI quoting configuration using '[]'.

### 3.5.3 MySQL and MS Access as a Data Source

Direct discovery can be used in conjunction with MySQL and MS Access but may require the following parameters in the set variables due to the quoting characters used in these sources:

```
SET DirectIdentifierQuoteChar='``';
```

## 3.6  Cancelling a Direct Discovery Query

The cancel icon on QlikView charts can be used to abort the direct discovery query. The direct discovery query running for a chart will be automatically aborted when the QlikView application is closed or a new tab is selected.



Figure 13. Cancelling direct discovery query

The equivalent icon is available on all Qlik Sense charts and can be used to abort the direct discovery query.

## 3.7 Unsupported QlikView and Qlik Sense Functionality

Due to the interactive and SQL syntax specific nature of the Direct Discovery approaches, the following QlikView features are not supported;

- Set Analysis
- More than one calculated dimension in a direct discovery chart
- Direct Discovery Measure and Detail fields are not supported on Global Search
- Client side section access and data reduction
- Loop and Reduce
- Synthetic keys on the direct discovery table

## 3.8 Multi-Table Support

Direct Discovery can be used to load more than one table/view and supports ANSI SQL join functionality.  A limitation exists in the fact that in a single chart all measures must be derived from the same logical table, this could in fact be a combination of tables from source linked via join statements.

- Direct Discovery can be deployed in a single fact/multi-dimension in memory scenario with large datasets.
- Direct Discovery can be used with more than one table which match any of the following criteria
  - The cardinality of the key field in the join is low.
  - The cardinality of the key field in the join is high AND DirectEnableSubquery is set to true AND the joined tables are both Direct Discovery.
- Direct Discovery is not suitable for deployment in a third normal form scenario with all tables in direct discovery form.

### 3.8.1 Supported Scenario I

In the following model the Product and Sales Order tables are joined via ProductID
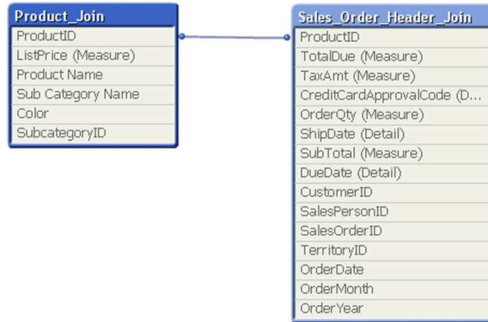
As the key field has a low distinct value count in this case 266 small SQL statements would be passed to the underlying source.

Building a chart of OrderMonth vs sum(Subtotal) and selecting a Color from the Product table will traverse the table and produce SQL as follows:
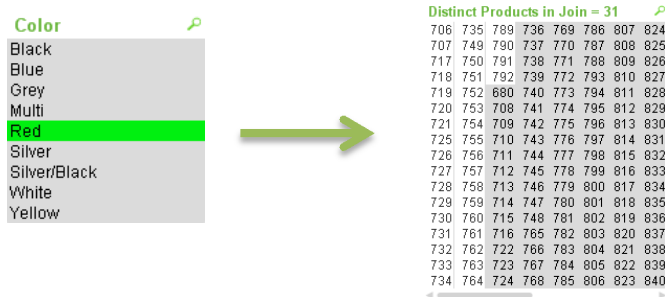


*SELECT ProductID, month([OrderDate]), SUM(OrderQty), SUM(SubTotal)*
*FROM  SalesTable*
*WHERE ProductID IN ( 739, 741, 742, 748, 771, 772, 773, 774, 779, 780, 781, 880, 894, 904, 905, 906, 907, 917, 918, 919, 920, 942, 944, 945, 948, 952, 980, 981, 982, 983, 984, 985, 986, 987, 988, 740 )*
*GROUP BY ProductID, month([OrderDate])*

### 3.8.2 Supported Scenario II

The following set statement is included in the script:

```
SET DirectEnableSubquery='true';
```

**AND**

**All tables in the application are in direct discovery mode**

**NOTE inclusion of an in memory table in the same archipelago as the Direct Discovery tables is not supported**

In the following model the Fact and Part tables are joined via l_partkey, the join field has high cardinality with over 20,000 distinct rows.  Selecting any dimension filter from the Part table will produce a sub query being sent to the underlying source



For example the p_discount_flag has a distribution of 19,996 l_part_key's with a flag of "Y". Building a chart of l_part_key vs sum(Quantity) and selecting  Y in this dimension field from the Product table will generate a sub query to find the corresponding l_part_key's and produce SQL as follows:

*SELECT "l_partkey", SUM("l_quantity") FROM "lineitem"*
*WHERE "l_partkey" IN (SELECT DISTINCT "TPCH"."dbo"."PART"."P_PARTKEY" FROM "TPCH"."dbo"."PART" WHERE "TPCH"."dbo"."PART"."P_DISCOUNT_FLAG" IN ('Y') )*
*GROUP BY "L_PARTKEY"*

### 3.8.3 Where Clause

Direct discovery tables can be linked via a where clause, an example script which joins the Product/Product Sub Category table via ProductSubCategoryID is shown below:

```
Product_Join:
DIRECT QUERY
DIMENSION
        [ProductID],
        [AdventureWorks2012].[Production].[Product].[Name] as [Product Name],
        [AdventureWorks2012].[Production].[ProductSubCategory].[Name] as [Sub
Category Name],
        Color,
        [AdventureWorks2012].[Production].[Product].ProductSubcategoryID as
[SubcategoryID]
        MEASURE
        [ListPrice]
FROM
[AdventureWorks2012].[Production].[Product],[AdventureWorks2012].[Production].[Pr
oductSubcategory]
WHERE [AdventureWorks2012].[Production].[Product].ProductSubcategoryID =
[AdventureWorks2012].[Production].[ProductSubcategory].ProductSubcategoryID   ;
```

Figure 14. Where clause join

### 3.8.4 From Clause

Multiple tables can be added into the Direct Discovery load statement using standard ANSI SQL, an example with a FROM/JOIN/ON statement is shown in Figure 15, this methodology vs the WHERE clause is more widely employed where you need to introduce OUTER joins. This statement joins the SalesOrderHeader to the SalesOrderDetail table via SalesOrderID and also joins the Customer table to the SalesOrderHeader table via the CustomerID.

```
Sales_Order_Header_Join:
DIRECT QUERY
        DIMENSION
                AdventureWorks2012.Sales.Customer.CustomerID as CustomerID,
                AdventureWorks2012.Sales.SalesOrderHeader.SalesPersonID as
        SalesPersonID,
                AdventureWorks2012.Sales.SalesOrderHeader.SalesOrderID as
        SalesOrderID,
                ProductID,
                AdventureWorks2012.Sales.Customer.TerritoryID as TerritoryID,
                OrderDate,
```

```
        NATIVE('month([OrderDate])') as OrderMonth,
        NATIVE('year([OrderDate])') as OrderYear
MEASURE
    SubTotal,
    TaxAmt,
    TotalDue,
    OrderQty
DETAIL
    DueDate,
    ShipDate,
    CreditCardApprovalCode,
    PersonID,
    StoreID,
    AccountNumber,
    rowguid,
    ModifiedDate
FROM AdventureWorks2012.Sales.SalesOrderDetail
        JOIN  AdventureWorks2012.Sales.SalesOrderHeader
        ON (AdventureWorks2012.Sales.SalesOrderDetail.SalesOrderID =
        AdventureWorks2012.Sales.SalesOrderHeader.SalesOrderID)
        JOIN AdventureWorks2012.Sales.Customer
        ON(AdventureWorks2012.Sales.Customer.CustomerID =
        AdventureWorks2012.Sales.SalesOrderHeader.CustomerID);
```

Figure 15. FROM clause join

### 3.8.5 Chart Limitations

As previously mentioned a single chart can only contain measures from the same logical table. Figure 15 contains an example which joins multiple tables and has measures which can be used in the same chart, for example a chart could be built showing both OrderQty and SubTotal.

Adding a measure from an additional table is not currently supported and will result in the error message shown in Figure 16.
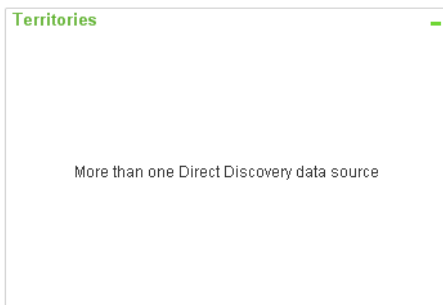


Figure 16. Multi-Table error message.

Dimensions sourced from other tables ARE supported in the same chart.

## 4   Performance Considerations

The performance of the Direct Discovery feature fundamentally reflects the performance of the underlying system as the feature queries an external system. It is possible to use standard database and query tuning best practices for this feature. All of the performance tuning should be done on the source database. Direct Discovery feature does not provide any support for query performance tuning from the application. However, it is possible to do asynchronous, parallel calls to the database by using the connection pooling capability. The load script syntax to setup the pooling capability is as follows:

```
SET DirectConnectionMax=10;
```

The default value is set to 1.

To improve the overall user experience, caching is used. Please see the section on caching for more information.

## 5   Security

Some security best practices should be taken into considerations when using Direct Discovery feature.

- All of the users using the same application with the Direct Discovery capability will be using the same connection. With this release, authentication pass through or credentials per user are not supported.
- Section Access on the desktop is not supported, server side section access IS supported.
  - DIMENSION values will be reduced on both the desktop and server implementations of section access, however only the server side implementation reduces the MEASURE calculations by inclusion of additional WHERE clauses in the SQL generation. For example a table chart with sum(SubTotal) by country with the section access to only show 4 of 8 countries would show a total sum for ALL countries in desktop but ONLY those limited by the section access reduction in server.
- With the new NATIVE() expression function, it would be possible to execute custom SQL statements in the database. It is advised that the database connection set up in the load script should use an account with only read access to the database.

- With this release of Direct Discovery, there is no logging capability. However it is possible to use the ODBC tracing capability. Please see the logging section for more details on this.
- It is possible to flood the database with requests from the client.
- It is possible to get detailed error messages from the log files.

# 6   QlikView Server Settings

Some settings on the QlikView Server and on the config.xml file should be reviewed if the Direct Discovery capability is used on a QlikView application. Please note that once these settings are changed, it will affect all of the QlikView applications that are on the same QlikView Server.

**Object Calculation Time Limit**

As Direct Discovery feature queries an external system from QlikView, the chart calculation time is dependent on the performance of the underlying system. It is advised to set the object calculation time limit setting on the QlikView Management Console high enough to allow enough time for the QlikView chart to get the direct discovery query results back from the data source. This setting is located under Performance tab of QlikView Server listed on the QlikView Management Console (Figure 17).

**Max Symbols in Charts**

Max symbols in charts setting is used to set the number of data points to be displayed on QlikView charts. Please note that as Direct Discovery query can return many distinct values, it is advised to review this setting to allow QlikView to display the desired number of data points on charts (Figure 17).
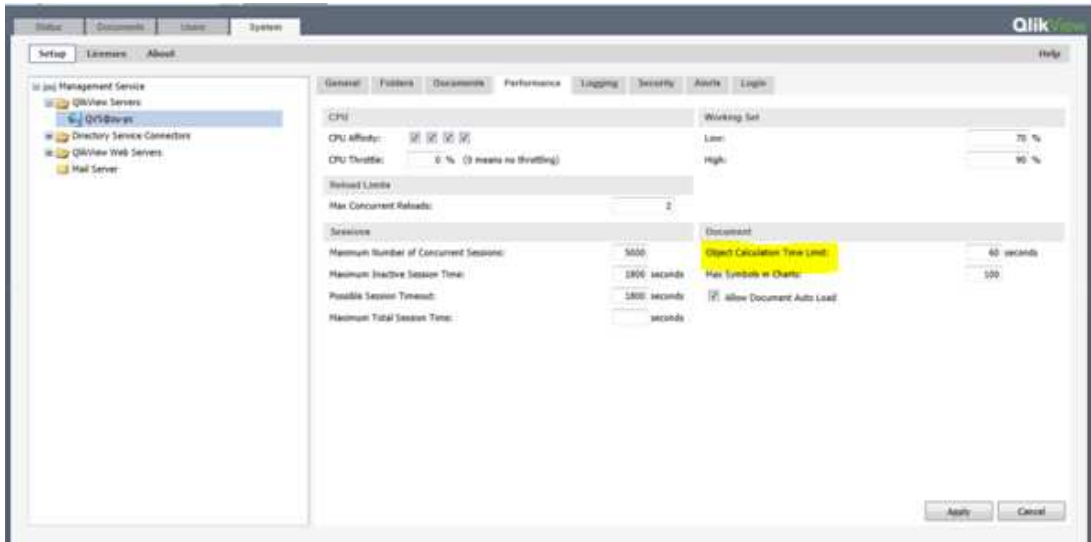
Figure 17. Object calculation time limit and Max symbols in charts settings on QlikView Management Console

## QVS Time Out Setting on the Config.xml File

As Direct Discovery feature queries an external system from QlikView, the QlikView Server time out setting on the config.xml file could be adjusted to allow enough time to QlikView to get the query results back. This change should be made if "Lost connection to server" error is seen when using the Ajax client.

The default setting on this option is 60 seconds. It is advised to increment this value to the possible maximum query time. Config.xml file is located under C:\ProgramData\QlikTech\WebServer folder. Please note that during upgrades this file will be overwritten with the default value:

<QvsTimeout>60</QvsTimeout>

## 6.1  QlikView Publisher

When QlikView applications with direct discovery are used with QlikView Publisher, please make sure that the service account running QlikView Publisher has read access on the source direct discovery table. This is required to allow QlikView Publisher access to the direct discovery table during scheduled data refreshes.

## 6.2 Caching

The performance of querying an external system from QlikView/Qlik Sense fundamentally reflects the performance of the underlying system. To improve the overall user experience, caching is used and stores selection states of queries in memory. As the same types of selections are made, QlikView/Qlik Sense will leverage the query from the cache (Figure 18). These cached result sets are shared across users. Please note that, when the same types of selections are done, QlikView/Qlik Sense will not query the source data and will leverage the cached result set. Also, when back and forward buttons are used, QlikView will leverage the query results from the cache.

It is possible to set a time limit on caching via a set variable:

```
SET DirectCacheSeconds= 10;
```

Once this time limit hits, the Server will clear the cache for the Direct Discovery query results that were generated for the previous selections. When this happens, QlikView/Qlik Sense will query the source data for the selections and will create the cache again for the designated time limit.
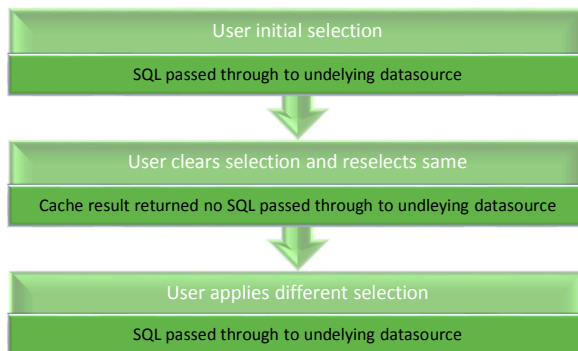


Figure 18. QlikView/Qlik Sense caching process for Direct Discovery data

The default caching time for direct discovery query results is 30 minutes unless the Set statement is used.

## 6.3 Error Messages

Because of the SQL syntax nature of Direct Discovery some errors may happen on the charts when direct discovery fields are used. Below is a brief description of these errors.

- Direct query attempted against missing or non-direct table
  - This error message is displayed on the chart when the load script has been changed so that a previously defined direct query is now against a table that is no longer there or against an in memory table.

- Direct query failed
  - This error message is displayed when the query that is executed for direct discovery is failed and could be caused by a number of factors, see section 6.4 to activate logging to investigate further.

- Table box row limit exceeded
  - This warning is specific to the drill to details feature for a table box where the amount of rows to be retrieved is over the SET DirectTableBoxListThreshold parameter (default is 1000).

- Database connection failed
  - This error message is displayed during chart execution (data is not cached) when a communication error between QlikView/Qlik Sense and the source database occurs.

- Detail field is not allowed with aggregated fields
  - This error message is displayed in Qlik Sense when adding detail and measure fields are added to the same table.

## 6.4 Logging

Behind the scenes a direct SQL statement is passed to the underlying data source as such the statement passed can be viewed through the trace files of the underlying connection. An example of this is shown below for a standard ODBC connection:
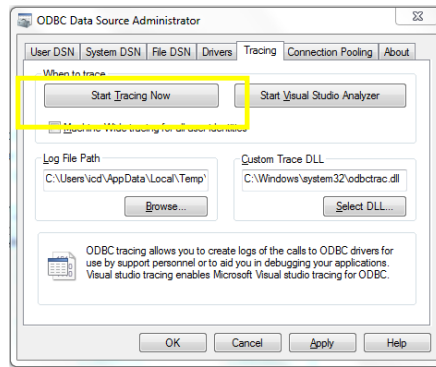
Figure 19. Standard ODBC connection tracing tab

The subsequent trace file details SQL statements generated through the user selections and interactions.

Logging can provide a more detailed explanation of the generic "Direct Query Failed" message, for example building a chart with the following expression

TextCount(CustomerID)

results in the chart failing to render and write the following to the log file:

*[Microsoft][ODBC SQL Server Driver][SQL Server]'TEXTCOUNT' is not a recognized built-in function name."*

# 7   Under The Hood

To understand the differences between the behaviour of an in memory versus a Direct Discovery load script its necessary to look at how QlikView/Qlik Sense populates its internal structures during both of these processes

## 7.1  In Memory

Taking the following script as an example:

```
LOAD CustomerID, SalesPersonID, SalesOrderID, SubTotal
```

```
SQL SELECT CustomerID, SalesPersonID, SalesOrderID, SubTotal
FROM AdventureWorks.Sales.SalesOrderHeader;
```

During the reload process the data is transformed into two table types one associative data table and multiple symbol tables (one per field). The symbol tables contain one row per distinct value of the field. Each row contains a pointer and the value of the field, both the numeric value and the textual component. Basically, the symbol tables are look-up tables for the field values as shown in figure 20.

### Unique values are loaded into the structures (symbol tables)

| CustomerID | | SalesPersonID | | SalesOrderID | | SubTotal | |
|---|---|---|---|---|---|---|---|
| Pointer | Value | Pointer | Value | Pointer | Value | Pointer | Value |
| 10 | Acme Ltd | 100 | Paul S. | 1101 | AB1234 | 10001 | 2345.57 |
| 11 | Smith Co. | 101 | Mary D. | 1110 | AC1234 | 10010 | 45.78 |

### The associative data table is simultaneously populated

| CustomerID | SalesPersonID | SalesOrderID | SubTotal |
|---|---|---|---|
| 10 | 100 | 1101 | 10001 |
| 11 | 100 | 1110 | 10010 |
| 10 | 101 | 1110 | 10001 |
| 10 | 100 | 1101 | 10010 |
| ... | ... | ... | ... |

Figure 20. In-memory data load

## 7.2 Direct Discovery

Modifying the script to enable Direct Discovery:

```
DIRECT QUERY DIMENSION CustomerID, SalesPersonID, SalesOrderID
MEASURE SubTotal
FROM AdventureWorks.Sales.SalesOrderHeader;
```

During the reload process the only the DIMENSION data is transformed into multiple symbol tables (DETAIL and MEASURE column data remains in the source) and the associative data table remains empty with only the metadata being loaded (column names) as shown in figure 21.

| CustomerID | | SalesPersonID | | SalesOrderID | | SubTotal |
|---|---|---|---|---|---|---|
| **Pointer** | **Value** | **Pointer** | **Value** | **Pointer** | **Value** | |
| 10 | Acme Ltd | 100 | Paul S. | 1101 | AB1234 | |
| 11 | Smith Co. | 101 | Mary D. | 1110 | AC1234 | |

**The associative data table is not populated on reload and created by selections (with caching)**

| CustomerID | SalesPersonID | SalesOrderID | SubTotal |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

Figure 21. Direct discovery data load

The association table is populated on the fly per user click with the result set being cached. The exception to this rule would be dimension fields flagged as "DETACH" which would not populate the association data table at all resulting in a faster query execution BUT without any associations.