XXL and XXS tables rules enablement

Introduction

XXL

XXL table rules are performance related rules that help detect incorrect or poorly performing SQL queries running on **XXL tables**. XXL tables can be defined as **extremely large tables**, containing a large amount of data. The goal is to use table size from production systems because development / integration systems may not feature really large tables and would not help detect real threat on application performance levels. If the information is not physically accessible (e.g.: first release of an application with no production environment), it is worth simulating the information by identifying tables that are expected to be large and by inputting a fake size, yet greater than the threshold in use in the "XXL tables" diagnostics.

Each XXL rule has two versions: one for XXL and another for non-XXL:

- Violations and grades are **always calculated for the non XXL rule** whatever configuration is provided
- If a configuration for XXL is provided for one or several schemas, the extra XXL rule is activated on those.

XXS

The rule **Avoid SQL queries that no index can support for artifacts with high fan-in** is configured to ignore tables that are too small (i.e. XXS), simply because adding an index on a very small table is generally considered as useless.

Enabling XXL/XXS rules

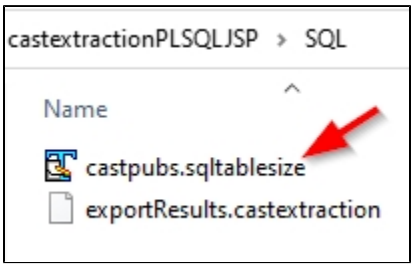
To enable these XXL/XXS rules it is necessary to provide the analyzer with table row size information in XML based ***.sqltablesize files**. This can be done as follows:

Using the "out of the box" analyzers

If you are using the "out of the box" analyzers in AIP Core for **Microsoft SQL Server**, **Sybase ASE**, **Oracle Server**, **SAP ABAP** and **Oracle Forms /Reports** then:

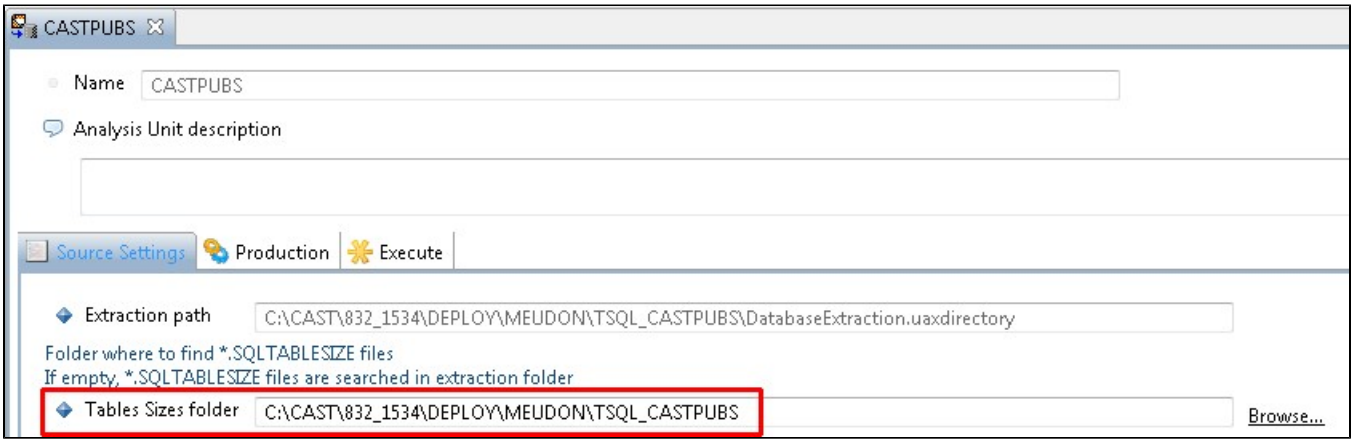
Using AIP Console

Deliver the ***.sqltablesize files** that define your table row size information along with your extraction files in the ZIP file or the source code upload folder:



Using CAST Management Studio

You can configure the **Table Size Folder** option in the relevant **Analysis Unit editor** in the **CAST Management Studio**. This folder must be populated with the ***.sqltablesize files** that define your table row size information. For example:



Using the SQL Analyzer extension

If you are using the SQL Analyzer extension then you can deliver the ***.sqltablesize** files that define your table row size information in the same folder alongside the source code.

> The **Table Sizes Folder** option in the relevant **Analysis Unit editor** in the **CAST Management Studio does not exist for Universal Analyzer Analysis Units** which are used for the **SQL Analyzer extension**.

.sqltablesize file samples

There are two formats used for sqltablesize files:

- **legacy** - this format is accepted by **both** the "out of the box" CAST AIP Analyzers and the SQL Analyzer extension, however, this format can only be used for XXL rules.

- **new** - this format is accepted ONLY by the SQL Analyzer extension and therefore cannot be used with "out of the box" CAST AIP Analyzers. This format can be used for XXL and XXS rules.

Both formats are explained below.

There are no examples for SAP ABAP since the CAST SAP Extractor NG generates the file automatically.

Legacy format

The following "legacy" format is accepted by **both** the "out of the box" CAST AIP Analyzers and the SQL Analyzer extension. This format can only be used for XXL rules:

Oracle Server

```
<?xml version="1.0" encoding="UTF-8" ?>
<config name="SQL Table Size" version="1.0.0.0" extraction-date="2009/02/10" >
<oracle castformat="7.0">
<!-- "server name" is equal to the value defined for the CAST_Oracle_Instance item in the .UAXdirectory file
resulting from the extraction process -->
<server name="ORA10G" >
<schema name="CASTPUBS">
...
<table name="ORDER_LINE" rows="2000000000"/>
...
</schema>
</server>
</oracle>
</config>
```

Microsoft SQL Server

```
<?xml version="1.0" encoding="UTF-8" ?>
<config name="SQL Table Size" version="1.0.0.0" extraction-date="2009/02/10" >
<microsoft>
<!-- server name can be written as HOST\INSTANCE_NAME but INSTANCE_NAME alone will also function -->
<server name="PDOXPLAP2\SQLS2K5" >
<!-- schema name is optional - if it is not applied, then the table row value is applied to all tables of that
name in the database -->
<schema name="schema or user name">
<database name="CASTPUBS">
...
<table name="Order_line" rows="20000000000"/>
...
</database>
</schema>
</server>
</microsoft>
</config>
```

Sybase ASE Server

```
<?xml version="1.0" encoding="UTF-8" ?>
<config name="SQL Table Size" version="1.0.0.0" extraction-date="2009/02/10" >
<sybase>
<!-- server name can be written as HOST\INSTANCE_NAME but INSTANCE_NAME alone will also function -->
<server name="Bordeaux" >
<!-- schema name is optional - if it is not applied, then the table row value is applied to all tables of that
name in the database -->
<schema name="schema or user name">
<database name="cwmm">
...
<table name="ACC" rows="20000000000"/>
...
</database>
</schema>
</server>
</sybase>
</config>
```

IBM DB2-UDB Server

```
<?xml version="1.0" encoding="UTF-8" ?>
<config name="SQL Table Size" version="1.0.0.0" extraction-date="2009/02/10" >
<ibm-udb>
<server name="Bordeaux" >
<schema name="cwmm">
...
<table name="ACC" rows="1000000"/>
...
</schema>
</server>
</ibm-udb>
</config>
```

IBM DB2 z/OS Server

```
<?xml version="1.0" encoding="UTF-8" ?>
<config name="SQL Table Size" version="1.0.0.0" extraction-date="2009/02/10" >
<ibm-zos>
<server name="Bordeaux" >
<database name="cwmm">
<schema name="cwmm">
...
<table name="ACC" rows="1000000"/>
...
</schema>
</database>
</server>
</ibm-zos>
</config>
```

SAP

```
<?xml version="1.0" encoding="utf-8"?>
<config name="SQL Table Size" version="v8.0.3" extraction-date="2017/12/04/15/33/19">
<!--Following data are extracted from MANDANT:100-->
<sap>
...
<table name="/.../..." client-dependant="X" rows="9"/>
<table name="/.../..." client-dependant="X" rows="2"/>
...
</sap>
</config>
```

New format

The following format is accepted ONLY by the SQL Analyzer extension and therefore cannot be used with the "out of the box" CAST AIP Analyzers. This format can be used for XXL and XXS rules. The new format uses the pattern: **schema_name.table_name=<table row count>** i.e. one line per XXL table. "**xxl_threshold**" and "**xxs_threshold**" options allow the default thresholds which will trigger the rules (100,000 for XXL and 10 for XXS) to be overridden:

```
xxl_threshold=100000
xxs_threshold=10

CASTPUBS.ORDER_LINE=2000000000
cwmm.ACC=2000000000
```

Starting with **3.4.1-funcrel**, the pattern **database_name.schema_name.table_name** is also accepted.

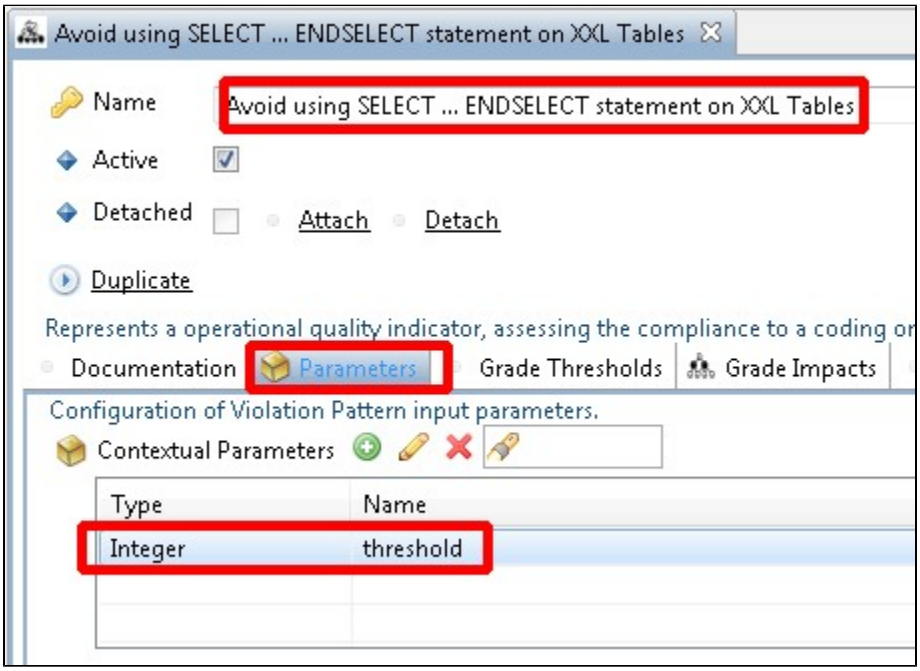
Managing thresholds

Each XXL/XXS rule has a threshold which determines when a table should be considered as "XXL" or "XXS". This threshold is, by default set to 100,000 for XXL (i.e. anything over 100,000 will trigger the XXL rules) and 10 for XXS (i.e. anything under 10 will trigger the XXS rules). You can override the default threshold in two ways:

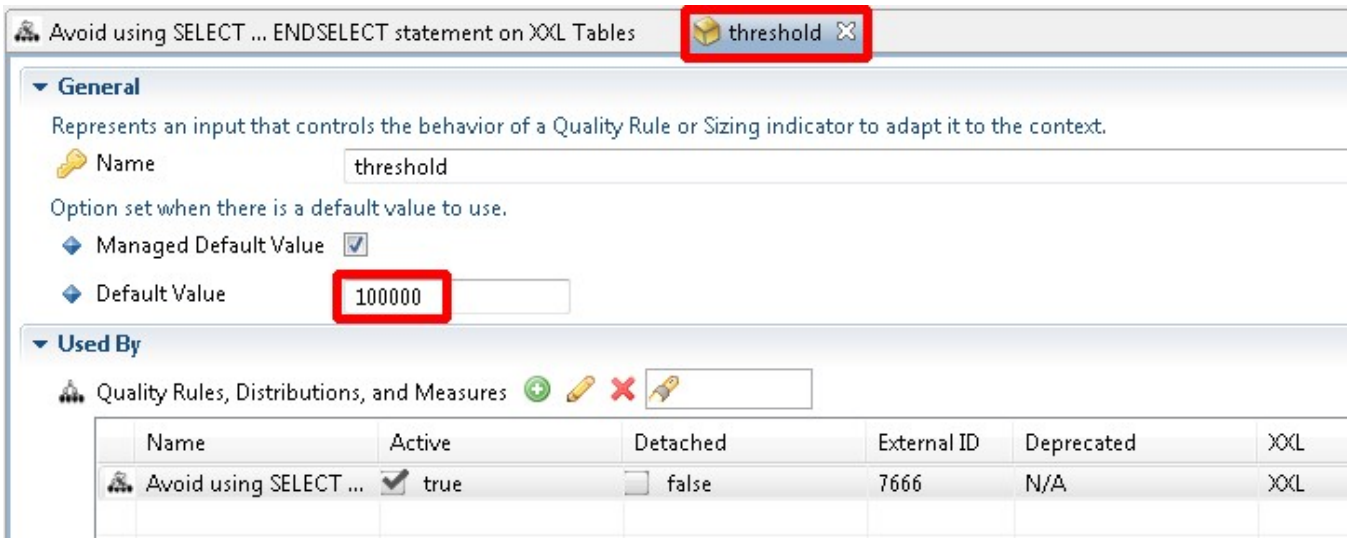
Using the CAST Management Studio

This method is valid for **both new and legacy** .sqltablesize formats.

Each XXL/XXS rule has its own specific **Contextual Parameter** that defines the threshold. For example, the Quality Rule 7666 "Avoid using SELECT ... ENDSELECT statement on XXL Tables" uses a Contextual Parameter as follows:



Editing the parameter shows the threshold value:



It is therefore possible to change this value and therefore the threshold at which a table is deemed to be "XXL".

Using a parameter in the .sqltablesize file

> This method is valid for the **new** .sqltablesize format only.

If you are using the new .sqltablesize format accepted ONLY by the SQL Analyzer extension, then you can **override all threshold values for all XXL / XXS rules** by adding the following to the top of your .sqltablesize file:

```
xxl_threshold=200000
XXS_threshold=10
```

.sqltablesize file generation methods

You can generate .sqltablesize files by executing the following scripts:

- If you are collecting table size from a production system while analyzing a development system, please remember to change the server name value in the generated file.
- There are no examples for SAP ABAP since the CAST SAP Extractor NG generates the file automatically.
- When running any scripts for generating **legacy format** .sqltablesize files, please make sure that the **very first line of the output file is not blank**, and if it is, **remove the blank line** as this will cause issues during the analysis processS.

Oracle Server

Legacy format

Execute the following script using the owner of the schema you are interested in. This will generate output in the "**legacy format**".

```
set pagesize 0;
set echo off;
set heading off;
set feedback off;
spool ORACLE.SQLTABLESIZE;
select '<?xml version="1.0" encoding="UTF-8" ?>' from dual;
select '<config name="SQL Table Size" version="1.0.0.0" extraction-date="' ||to_char(sysdate, 'YYYY' )||'
/'||to_char(sysdate, 'MM' ) ||'/'||to_char(sysdate, 'dd' )||'" >' from dual;
select '<oracle castformat="7.0">' from dual;
select '<server name="'|| sys_context('USERENV', 'INSTANCE_NAME')||'" >' from dual;
select '<schema name="'||sys_context('USERENV', 'CURRENT_SCHEMA')||'">' from dual;
select '<table name="'||table_name||'" rows="'||nvl(num_rows,0) ||'"/>' from user_tables ;
select '</schema>' from dual;
select '</server>' from dual;
select '</oracle>' from dual;
select '</config>' from dual;
spool off;
exit;
```

You can also automate it as follows:

```
sqlplus <owner>/<password>@<connect identifier> @<script name>
```

New format for the SQL Analyzer extension

This will generate output in the "**new format**" for the SQL Analyzer extension (note that this is provided as an example with no official support):

```
set echo off;
set heading off;
set feedback off;
spool ORACLE.SQLTABLESIZE;
select 'xxl_threshold=200000' from dual
union all
select 'xxs_threshold=10' from dual
union all
select '   ' from dual
union all
select sys_context('USERENV', 'CURRENT_SCHEMA')||'.'||table_name||'='||to_char(nvl(num_rows,0)) from
user_tables;
spool off;
exit;
```

Microsoft SQL Server

Legacy format

Execute the following script against the database you are interested in. This will generate output in the "**legacy format**".

```
set nocount on
select '<?xml version="1.0" encoding="UTF-8" ?>'
select '<config name="SQL Table Size" version="1.0.0.0" extraction-date="' + convert(varchar(10),GETDATE()) +
'" >'
select char(9) + '<microsoft>'
select char(9) + char(9) + '<server name="'+ @@servername +'">'
select char(9) + char(9) + char(9) + '<database name="'+ DB_NAME() +'">'
select char(9) + char(9) + char(9) + char(9) +'<table name="' + so.name + '" rows="' + convert(varchar(10),
convert(int,MAX(si.rows))) +'"/>'
FROM
sysobjects so,
sysindexes si
WHERE
so.xtype = 'U'
AND
si.id = so.id
GROUP BY
so.name
select char(9) + char(9) + char(9) + '</database>'
select char(9) + char(9) + '</server>'
select char(9) + '</microsoft>'
select '</config>'
go
```

You can automate it as follows:

```
sqlcmd -U sa -P <password> -S <server> -H <host> -d <target database> -i <script> -o SQLSERVER.SQLTABLESIZE -h
-1
```

If you run the above query in **Microsoft SQL Server Management Studio**, the default output mode cannot easily be copied out of the results window. Therefore CAST recommends that you change the query output to either **Text** or **File** as follows:

- Click the **Query** menu (available when working in the Query window)
- Select **Results To** and then choose either:
  - **Results To Text**
  - **Results To File**

New format for the SQL Analyzer extension

This will generate output in the "**new format**" for the SQL Analyzer extension (note that this is provided as an example with no official support):

```
set nocount on
SELECT su.name + '.' + so.name + '=' + convert(varchar(10),convert(int,MAX(si.rows))) TableSize
FROM
sysobjects so,
sysusers su,
sysindexes si
WHERE
so.uid = su.uid
AND
so.xtype = 'U'
AND
si.id = so.id
GROUP BY su.name, so.name
ORDER BY su.name, so.name
go
```

Sybase ASE

Execute the following script against the database you are interested in. This will generate output in the "**legacy format**".

```
select "<?xml version="1.0" encoding="UTF-8" ?>"
select "<configSQL Table Size"" version=""1.0.0.0"" extraction-date="""|| convert(varchar(12),getdate()) ||
"""">"
select char(9) ||"<sybase>"
select char(9) || char(9) || "<server name="'+ @@servername +'">"
select char(9) || char(9) || char(9) || "<database>"
begin SELECT "<table" || o.name || """ rows="""" || convert(varchar(50),convert(int,s.rowcnt)) || """ />"
FROM sysobjects o, systabstats s
WHERE o.id = s.id AND s.indid IN (0,1)
AND o.type = 'U'
ORDER BY o.type, o.name end
select char(9) || char(9) || char(9) || "</database>"
select char(9) || char(9) || "</server>"
select char(9) || "</sybase>"
select "</config>"
go
```

IBM DB2 UDB

Execute the following script against the schema you are interested in.

Legacy format

This will generate output in the "**legacy format**".

```
connect to <server> user <user> using <role>;
select '<?xml version="1.0" encoding="UTF-8" ?>' from SYSIBM.SYSDUMMY1;
select '<config name="SQL Table Size" version="1.0.0.0" extraction-date="' || varchar(current date) || '" >'
from SYSIBM.SYSDUMMY1;
select CAST(char(' ') as char(4)) || '<ibm-udb>' from SYSIBM.SYSDUMMY1;
select CAST(char(' ') as char(8)) || '<server name="' || HOST_NAME || '">' FROM TABLE(SYSPROC.
ENV_GET_SYS_INFO()) AS SYSTEMINFO;
select CAST(char(' ') as char(12)) || '<database name="'DB2UDB'">' from SYSIBM.SYSDUMMY1;
select CAST(char(' ') as char(16)) || '<schema name="CASTPUBS">' from SYSIBM.SYSDUMMY1;
select CAST(char(' ') as char(20)) || '<table name="' || tabname || '" rows="' || TRIM(TRAILING FROM CAST(card
as char(128))) || '" />' FROM syscat.tables where tabschema = 'CASTPUBS' and type = 'T' order by tabname;
select CAST(char(' ') as char(16)) || '</schema>' from SYSIBM.SYSDUMMY1;
select CAST(char(' ') as char(12)) || '</database>' from SYSIBM.SYSDUMMY1;
select CAST(char(' ') as char(8)) || '</server>' from SYSIBM.SYSDUMMY1;
select CAST(char(' ') as char(4)) || '</ibm-udb>' from SYSIBM.SYSDUMMY1;
select '</config>' from SYSIBM.SYSDUMMY1;
```

Please ensure that you:

- Modify the first line to use your own login credentials.
- You will need to run this script against each schema you are interested in and for each schema you will need to modify the lines containing:
  - <schema name="CASTPUBS">
  - where tabschema = 'CASTPUBS'
- The resulting DB2UDB.SQLTABLESIZE file is not valid XML, please ensure that you remove the first few lines that are not part of the valid generated XML.

New format for the SQL Analyzer extension

This will generate output in the "**new format**" for the SQL Analyzer extension.

```
SELECT concat(concat(trim(TABSCHEMA), '.') , trim(TABNAME)) concat '='  concat trim(cast(CARD as char(50)))
TableSize
FROM SYSCAT.TABLES
WHERE type = 'T'
AND TABSCHEMA = 'YOUR SCHEMA NAME'
ORDER BY TableSize
```

For example:

```
SELECT concat(concat(trim(TABSCHEMA), '.') , trim(TABNAME)) concat '='  concat trim(cast(CARD as char(50)))
TableSize
FROM SYSCAT.TABLES
WHERE type = 'T'
AND TABSCHEMA = 'TT3'
ORDER BY TableSize
```

Will generate the following ouput :

```
TT3.BAN_IM_1=7
TT3.BAN_IM_2=7
TT3.BATCH_REF=4
TT3.BUS_LIST_1=18
....
```

IBM DB2 z/OS

Legacy format

There is no script available to generate XXL table size information in the **legacy format** for a schema hosted on an IBM DB2 z/OS server. In previous releases of CAST AIP the DB2 z/OS Extractor would automatically generate the raw XXL table size data and therefore no manual file creatiion was necessary, however, this extractor is no longer used for IBM DB2 z/OS analyses in **CAST AIP 8.3.x**. Please use the "new format" explained below instead.

New format for the SQL Analyzer extension

This JCL statement can be used as is to generate output in the "**new format**" for the SQL Analyzer extension:

```
/*
//*-----------------------------------------------------
//*
//* 26 - EXTRACTING NUMBER OF ROWS FOR TABLES
//*
//* COLUMN        TYPE     COL-SIZE EXTRACT-SIZE
//* ============= ======== ======== ============
//* CREATOR       VARCHAR  128      128
//* NAME          VARCHAR  128      128
//* CARDF         FLOAT    11       11
//*-----------------------------------------------------
//* Following filters can be inserted in the WHERE clause:
//*   - CREATOR IN ('xxx','yyy', ...)
//*   - CHAR(DBNAME) IN ('xxx','yyy', ...)
//*-----------------------------------------------------
//STEP26   EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//*
//SYSREC00 DD DSN=CAST.DB2.TABROWS,DISP=(NEW,CATLG,),
//            SPACE=(TRK,(100,100),RLSE)
//SYSPUNCH DD SYSOUT=*
//SYSPUNCH DD DUMMY
//SYSTSIN  DD *
  DSN SYSTEM(DBx)
  RUN  PROGRAM(DSNTIAUL) PLAN(DSNTIBxx) -
      LIB('DSNxxx.RUNLIB.LOAD') PARMS('SQL')
//SYSIN    DD *

  SELECT
      CONCAT(CREATOR, CONCAT('.', CONCAT(NAME, CONCAT('=', CARDF))))
  FROM SYSIBM.SYSTABLES
  WHERE TYPE = 'T'
  ORDER BY DBNAME, CREATOR, NAME;
```

MySQL/MariaDB

This will generate output in the "**new format**" for the SQL Analyzer extension (note that this is provided as an example with no official support):

```
select concat(TABLE_SCHEMA, '.', TABLE_NAME, '=', TABLE_ROWS) as TABLESIZE
from INFORMATION_SCHEMA.TABLES
where TABLE_TYPE = 'BASE TABLE'
and TABLE_SCHEMA not in ('INFORMATION_SCHEMA', 'MYSQL', 'PERFORMANCE_SCHEMA', 'SYS')
and TABLE_SCHEMA = 'XXX' --  to be specified
```

PostgreSQL

This will generate output in the "**new format**" for the SQL Analyzer extension (note that this is provided as an example with no official support):

```
SELECT concat(n.nspname, '.',relname, '=', reltuples) as TABLESIZE
        -- n.nspname = schema_name,
        -- relname = Table_Name,
        -- reltuples = number of rows in the table
FROM pg_class, pg_namespace n
WHERE pg_class.relnamespace = n.oid
        AND relkind = 'r' -- r = ordinary table
        AND n.nspname NOT LIKE 'pg_%' -- we don't want the system tables
        AND n.nspname = 'XXX' -- to be specified
```

Teradata

This will generate output in the "**new format**" for the SQL Analyzer extension (note that this is provided as an example with no official support). Replace MY_DB with your database name:

```
select 'show ' || case
  when tablekind = 'T' then 'table'
  when tablekind = 'V' then 'view'
  when tablekind = 'P' then 'procedure'
end || ' ' || trim(databasename) || '.' || trim(tablename) || ';'
from dbc.tables
where tablekind in ( 'T', 'V', 'P')
  and databasename = 'MY_DB'
order by 1;
```

Informix

This will generate output in the "**new format**" for the SQL Analyzer extension (note that this is provided as an example with no official support). Replace MY_DB with your database name:

```
database sysmaster;
set isolation to dirty read;
select trim(n.owner) || "." || trim(n.tabname) || "=" || cast(h.nrows as varchar(50)) as TableSizeFile
from sysptnhdr h, MY_DB:systabnames n
where h.dbsname = 'MY_DB'
and h.partnum = n.partnum;
```

SQLite

This will generate output in the **"new format**" for the SQL Analyzer extension (note that this is provided as an example with no official support):

```
SELECT "xxl_threshold=100000" AS TableSizeFile
UNION ALL
SELECT "xxs_threshold=10" AS TableSizeFile
UNION ALL
SELECT "" AS TableSizeFile
UNION ALL
SELECT tbl || "=" || stat AS TableSizeFile FROM sqlite_stat1;
```

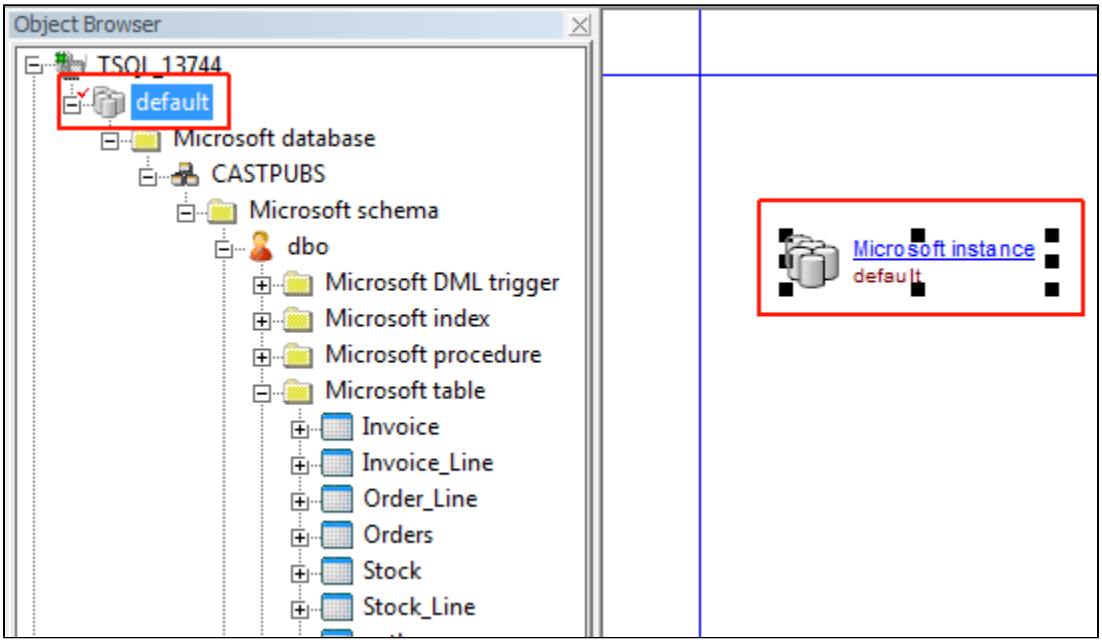
Tips - Server, database, schemas, table identification

The **server**, **database/schema** and **table** nodes in both new and legacy formats of the sqltablesize file must match the information stored in the CAST Analysis Service schema following an initial analysis. You can determine this information in several ways:

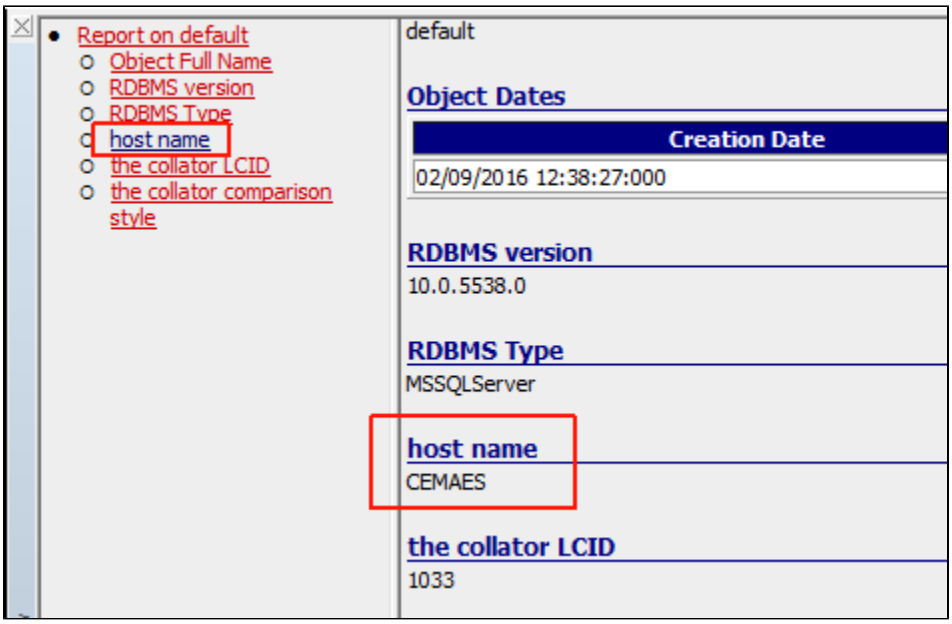
Find the server name

Using CAST Enlighten

- Right click the **Instance** either in the **Object Browser** or in the **Graphical View**:



- Select **Properties** to update the Property window. The "**host**" name highlighted below can be used to define the "**Server**" node in the . SQLTABLESIZE file:



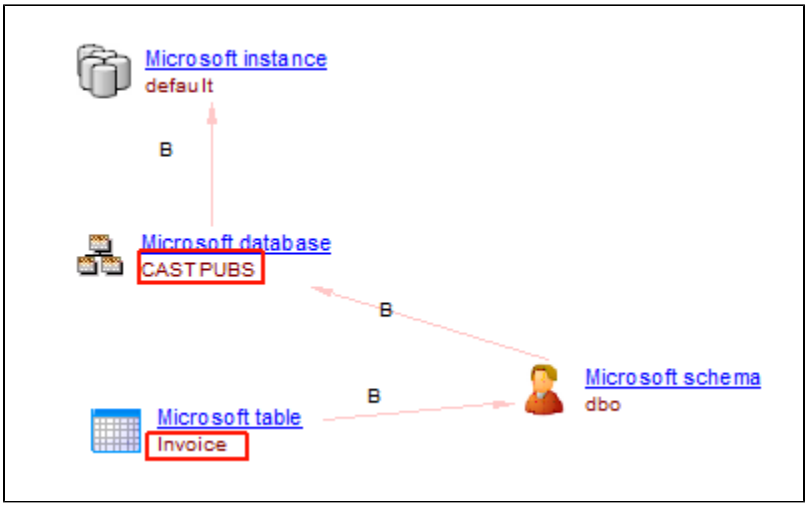
Using CAST System Views

You can query the CAST System Views in the Analysis Service to identify the "server" name. Run the following query (this is customised for CSS servers, but can be adapted to Microsoft SQL Server and Sybase ASE):

```
select DESCRIPTION
from <analysis_service>.CSV_OBJECT_DESCRIPTIONS
where DESC_TYPE = 'host name'
```

Find the Database and Table name

Databases/Schemas and **Tables** can be easily identified in CAST Enlighten using the **Object Browser** or the **Graphical View**:



Troubleshooting

Checking sqltablesize information upload

To identify which tables have been tagged with the sqltablesize information, you can run on the following query against your CAST Analysis Service schema, following an analysis:

Using the legacy "out of the box" analyzers	Using SQL Analyzer extension

`Select OBJECT_NAME, OBJECT_FULLNAME, OI.InfVal`
`From CDT_OBJECTS CO, ObjInf OI`
`Where OI.IdObj = CO.OBJECT_ID`
`And OI.InfTyp = 115`
`And OI.InfSubTyp = 1`
	`Select OBJECT_NAME, OBJECT_FULLNAME, OI.InfVal`
	`From CDT_OBJECTS CO, ObjInf OI`
	`Where OI.IdObj = CO.OBJECT_ID`
	`And OI.InfTyp = 1101000`
	`And OI.InfSubTyp = 2`

Finding XXL tables

To identify the XXL tables, you can run the following query against your CAST Analysis Service schema, following an analysis:

Using the legacy "out of the box" analyzers	Using SQL Analyzer extension

`Select OBJECT_NAME, OBJECT_FULLNAME, OI.InfVal`
`From CDT_OBJECTS CO, ObjInf OI`
`Where OI.IdObj = CO.OBJECT_ID`
`And OI.InfTyp = 115`
`And OI.InfSubTyp = 1`
`And OI.InfVal >= 100000 -- change the value if you changed the threshold`
	`Select OBJECT_NAME, OBJECT_FULLNAME, OI.InfVal`
	`From CDT_OBJECTS CO, ObjInf OI`
	`Where OI.IdObj = CO.OBJECT_ID`
	`And OI.InfTyp = 1101000`
	`And OI.InfSubTyp = 2`
	`And OI.InfVal >= 100000 -- change the value if you changed the threshold`