SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

DATA TRANSFER AND REPORTING

Documentation and execution

T E K I J Ä / T : Petri Ahnger

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma Tietotekniikan koulutusohjelma	
Työn tekijä(t) Petri Ahnger	
Työn nimi Tiedonsiirron ja raportoinnin dokumentointi ja toteuttaminen	
Päiväys 20.4.2015	Sivumäärä/Liitteet 34/1
Ohjaaja(t) Keijo Kuosmanen	
Toimeksiantaja/Yhteistyökumppani(t) Honeywell Oy, Kuopio	

Tiivistelmä

Honeywell International Inc. on monikansallinen yhtiö, joka toimii monilla eri toimialoilla ja työllistää noin 132 000 ihmistä. Tämän opinnäytetyön toimeksiantajana toimii Honeywell Oy, Kuopio ja tämän opinnäytetyön kannalta oleellisia Honeywellin tuotteita ovat OptiVISION ja siihen liittyvät työkalut. OptiVISION on niin kutsuttu tuotannonohjausjärjestelmä (MES, Manufacturing Execution System), joka on erikoistunut sellu- ja paperiteollisuuteen.

Tämän opinnäytetyön tarkoituksena on selvittää ja kuvata mitä kaikkia vaiheita tarvitaan, jotta data saadaan tuotantojärjestelmästä raportille asti ja toteuttaa tämä samalla käytännössä sekä analysoida raportointiprosessin haasteet ja onnistuneen raportoinnin vaatimat edellytykset.

Tiedonsiirto ja raporttien tuottaminen toteutettiin Microsoft SQL-Serverin tarjoamilla työkaluilla. Työn ensimmäisessä vaiheessa suunniteltiin tarvittava tiedonsiirto ja tiedon jalostaminen sekä tehtiin suunnitelmat raporttien ulkoasuista. Tämän jälkeen toteutettiin itse tiedonsiirto ja raporttien luominen, jonka jälkeen varmistettiin datan luotettavuus ja tehtiin tarvittavat tarkistuspisteet virheiden varalle. Viimeisenä tapahtui järjestelmän ja raporttien käyttöönotto ja siinä ilmenneiden virheiden korjaus sekä tarvittavien parannuksien tekeminen.

Tavoitteisiin päästiin, koska käyttöönottopäivämäärään mennessä luotiin toimiva järjestelmä, joka tuotti myös halutut raportit. Raporttien osalta korjauksia tarvittiin käyttöönoton jälkeen, lähinnä vain niiden ulkoasuihin.

Avainsanat
Tiedonsiirto, Raportointi

SAVONIA UNIVERSITY OF APPLIED SCIENCES | THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Information Technology			
Author(s) Petri Ahnger			
Title of Thesis Documentation and Execution of Data Transfer and Reporting			
Date	20 April 2015	Pages/Appendices	34/1
Supervisor(s) Mr. Keijo Kuosmanen, Lecturer			
Client Organisation /Partners Honeywell Oy, Kuopio			

Abstract

Honeywell International Inc. is a multinational company, which works on multiple industries and employs about 132 000 employees. The client organization of this thesis was Honeywell Oy, Kuopio and the essential Honeywell's products for this thesis were OptiVision and tools used with it. OptiVision is a so called Manufacturing Execution System (MES), which is specialized in pulp, paper and flat sheet industries.

The purpose of this thesis was to define and describe all the different steps needed to get data from a production system to a report and at the same time execute this process in practice and analyze the challenges of the reporting process and the requirements of a successful reporting.

The data transfer and the production of the reports were done with the tools provided by Microsoft SQL Server. At the first phase of this thesis, the required data transfer and refining of the data were designed and also the designs of the report layouts were made. After this, the actual data transfer and creation of the reports was executed. When they were done, the data was validated and necessary checkpoints for errors were made. The last phase of this thesis was the deployment of the system and the reports. After this the errors noticed after the deployment were fixed and necessary improvements were made.

The goals of the thesis were achieved, because the functional system was created by the date of the deployment and it also produced the wanted reports. There were basically only layout modifications to the reports after the deployment of the system.

Keywords
Data transfer, Reporting

PROLOGUE

From Honeywell I want to thank Juha Marttinen, Ari Hyvönen and Ilpo Ruokonen for help and guidance. Additionaly I want to thank lecturers Keijo Kuosmanen and Ulla Huttunen-Finta for tutoring.

I also want to thank my family, girlfriend and Hyviä Reissuja Oy

Kuopio 20 April 2014
Petri Ahnger

ABBREVIATIONS:

ETL	Extract, Transform and Load
SQL	Structured Query Language
SSIS	SQL Server Integration Services
SSRS	SQL Server Reporting Services
PHD	Process History Database
ERP	Enterprise Resource Planning ERP system brings all different sections of an enterprise under one software. For example, status of an order from a customer can be seen by immediately by all departments. (WAILGUM, Thomas 2007.)
MES	Manufacturing Execution System MES systems extends the features of ERP system and these two are often joined together. The boundaries between these two have grown weaker lately. MES is like ERP but it is specialized to follow the manufacturing process. With MES manufacturing and quality data from any point of manufacturing process, can be accessed at any time when it is needed. (VINHAIS, Joseph A 1998.)
SMTP	Simple Mail Transfer Protocol SMTP server handles the sending of an email to the Internet, SMTP is used together POP3 and IMAP servers, which handle the incoming emails. (POP3-, SMTP- ja muut sähköpostipalvelimien tyypit 2015.)

CONTENTS

1 INTRODUCTION

Report generation from production data is usually considered as a simple task, a report is simply executed and then printed. One of the goals of this study is to bring out all the background work needed for successful reporting. Before the creation of a report or before anything else can be started in practice, specifications based on the customer requirements have to be done. Usually specification includes a detail design of the functionality and requested layout. When the specification with the design has been approved, the actual creation of reports and data processing starts. In theory, in case specification documents are detailed enough the actual creation of reports is done by following the designs.

Data validation in general is one of the most important sections of reporting, a report with false data is useless. Frequent data checkpoints for possible errors or false data improve the data validation. If data for some reason is not valid or it is not available, the checkpoint automatically sends an error message and the problem can be solved much more quickly than if the false data is noticed not until on the report. In case an error is found on the report, first the reason for false data has to be located and sometimes this takes a lot of time, because the whole data transfer chain may have to be checked for errors.

1.1 Honeywell

Honeywell International Inc. is a multinational company which has about 132 000 employees worldwide. Globally Honeywell business are divided into four businesses (Wood 2014.):

- Aerospace
- Automation and Control Solutions
- Performance Materials and Technologies
- Transportation Systems

Honeywell Process Solutions (HPS) is a part of Performance Materials and Technologies section and it is a leader in the process automation industry globally. HPS employs approximately 11 500 employees. For more than 35 years HPS has improved process automation control systems. HPS is divided into five lines of businesses (Wood 2014.):

- Projects & Automation Solutions
- Lifecycle Service Solutions
- Advanced Solutions & Consulting Services
- Engineered Fields Solutions
- Field Products

Honeywell is one of the biggest Advanced Solutions suppliers to process industries. For example chemical, metal, mineral and mining, oil and gas, power, pulp and paper and refining and petrochemical industries are served by Advanced Solutions. Honeywell's Business and Process Performance Solutions helps to manage business complexity better, improve production and optimize the supply chain. (Wood 2014.)

1.2 The goals of the thesis

The main goal of this thesis is to document the reporting process while considering different sections needed in reporting, such as:

- Data transmission from the OptiVision server to the data warehouse server
- Execution method used for reporting
- Analyze challenges and requirements of successful reporting

Checkpoints for errors are created at the same time with report development. When a reports is developed, it is tested and the data is validated. When an error is found, the root causes are investigated to prevent it from happening again.

2 TOOLS AND TECHNIQUES USED FOR DATA TRANSFER AND REPORTING

A database is a data storage which is a collection of organized information that is somehow connected to each other. There are many different database management software to choose from, the main differences among the software are the price, query language, features and the tools provided by the software. Microsoft SQL Server was used as the database environment in this project.

2.1 Microsoft SQL Server

Microsoft SQL Server is a database environment with many useful tools to manage and analyze the database and the data it contains. SQL Server Management Studio is a graphical user interface for managing a database. SQL Server uses Transact-SQL (T-SQL) as its primary query language. (Microsoft SQL Server 2015.)

Transact-SQL is an extension to basic SQL and allows programming and the use of functions and variables. There are lots of predefined functions in SQL Server like SUM, which sums the selected values. There are predefined functions for mathematics, date formatting and text processing but the user can also create his own custom functions using Transact-SQL. All applications communicating with SQL Server use Transact-SQL to communicate. (Transact-SQL Reference(Transact-SQL) 2015.)

2.2 SQL Server Integration Services

To keep data in a data warehousing database valid and up to date, the data needs to be loaded regularly. The extraction of data from the source database or databases and transferring it to a target database is known as ETL (Extract, Transform and Load) and the target database is commonly called a data warehouse. In this project ETL processes are done with SQL Server Integration Services. (Overview of Extraction, Transformation, and Loading 2014.)

SQL Server Integration Services (SSIS) is SQL Server’s tool to create data integration projects. SSIS is a tool for creating ETL packages, data can be simply extracted from the source and transferred to a target location or data can be refined to a completely different form. Source and target locations can be anything from a plain text file to a massive database. The refining of data can be done with tools provided by SSIS or the user can create a task with custom code. (SSIS Tutorial: Creating a Simple ETL Package 2015.)

2.3 SQL Server Reporting Services

SQL Server Reporting Services (SSRS) is SQL Server’s server based feature for creating and managing reports and it provides a lot of different tools to create and manage these reports. Tools of SSRS are used in Microsoft Visual Studio. Reporting Services allows the creation of reports from relational, XML-based or multidimensional data sources. Reports can be created from scratch or a with report wizard which creates the foundation of the report. Reports can be exported for example to pdf or

Microsoft Excel format or reports can be viewed for instance as part of a Microsoft Windows application or over a web based connection. (Reporting Services (SSRS) 2015.)

2.4 Microsoft SQL Server Data Tools

Microsoft SQL Server Data Tools provides an environment where all the SSIS packages and reports are created in Microsoft Visual Studio. Microsoft SQL Server Data Tools provides an alternative for SQL Server Management studio, because with SQL Server Object Explorer it is possible to:

- Execute queries
- Create and edit database objects
- Create and edit database data

Microsoft SQL Server Data Tools also provides a graphical user interface for table creation and editing. (Microsoft SQL Server Data Tools… 2015)

2.5 OptiVision

OptiVision is a product name for Manufacturing Execution System (MES) developed by Honeywell. OptiVision is specialized to pulp, paper and flat sheet industries. The development and support of OptiVision is sourced and delivered from The Honeywell Process Solutions Division located in Cincinnati USA, Kuopio Finland, Vitrolles France and Bangalore India. OptiVision consists of:

- Customer and Order Management
- Production Planning and Scheduling
- Production and Quality Tracking
- Inventory and Warehouse Management
- Load and Shipment Management
- Invoice and Credit Management
- Shop Floor Interfaces and ERP Integration.

In other words OptiVision is so-called order to cash solution, meaning possibility to have all functionality from customer order to invoicing. (Wood 2014.) When customer’s order is received, OptiVision handles following sections:

- Production planning and scheduling
- Raw materials management
- Warehouse management
- Logistics

With OptiVision it is possible to diminish the costs and losses and increase the profits. (Honeywell Process Solutions 2013.)

2.6 Process History Database (PHD)

Honeywell's Uniformance Process History Database (PHD) collects and stores continuous process data. The PHD shadow server combines the data gathered by PHD collector servers and stores this data to a database. There can be one or many collectors and these collectors can be far away from each other geographically. (Honeywell Process Solutions 2014a.)

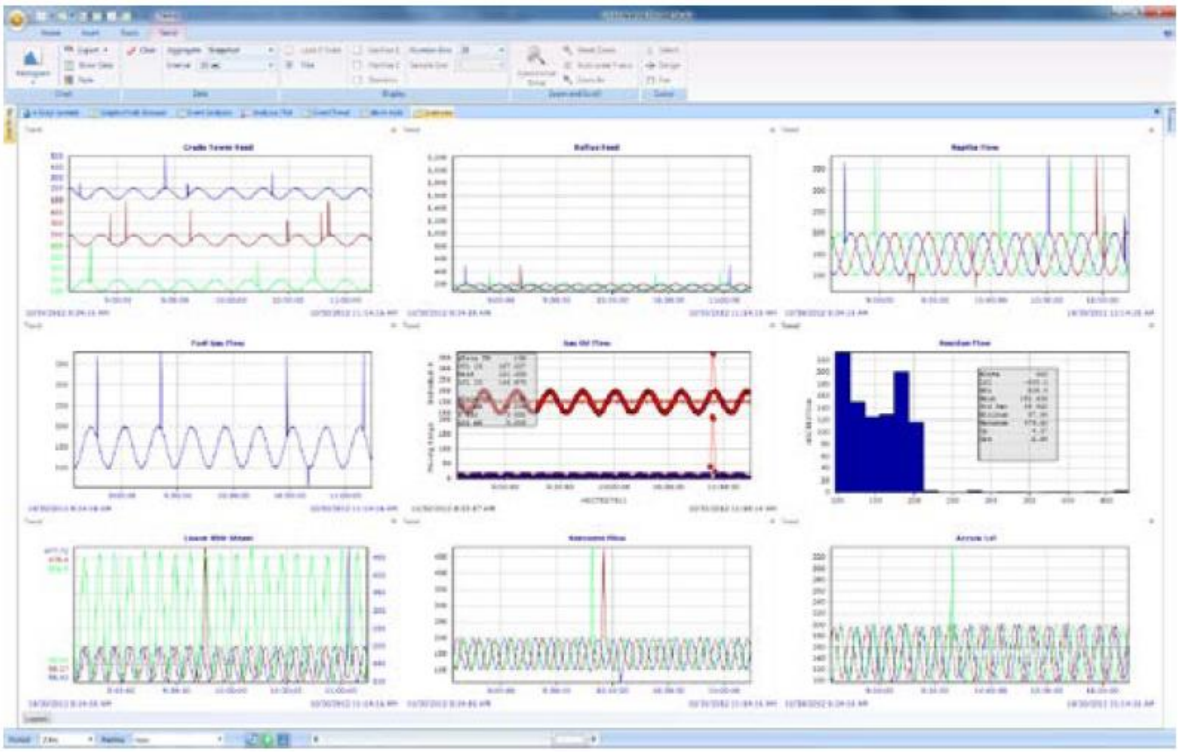


FIGURE 1. Uniformance Process Studio with different figures (Honeywell.)

With tools provided by the PHD the history data can transformed into information. As shown in Figure 1 the process data can be viewed with many different graphical displays. (Honeywell Process Solutions 2014a.)

2.7 Quality OptiMISER

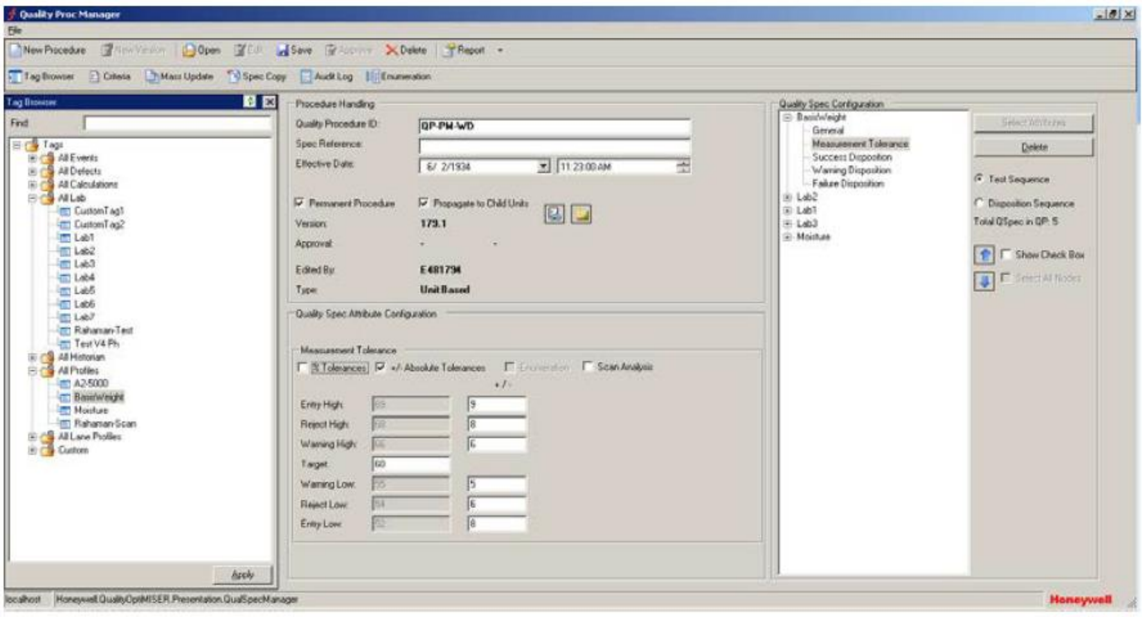
Quality OptiMISER is a data historian and analysis tool developed by Honeywell, which allows data from several sources to be combined and viewed simultaneously. Quality OptiMISER consists of a database and application suite which allows for example quality data from multiple sources to be stored for a long time and all quality data available from one software. (Honeywell Process Solutions 2014b.)

Quality OptiMISER is a tool for analyzing quality data effectively. The base package of Quality OptiMISER includes three application:

- Quality Procedure Manager

- Contour Map
- Trends

In Quality Procedure Manager (shown in Figure 2) the quality testing procedures are created and configured and these are used by all the other Quality OptiMISER applications. (Honeywell Process Solutions 2014b.)



Procedure Handling

Quality Procedure ID: QP-PM-WD
Spec Reference:
Effective Date: 6/ 2/1934 11:23:00 AM

[✓] Permanent Procedure [✓] Propagate to Child Units
Version: 179.1
Approval: - -
Edited By: E481794
Type: Unit Based

Quality Spec Attribute Configuration

Measurement Tolerance

[ ] % Tolerances [✓] +/- Absolute Tolerances [ ] Enumeration [ ] Scan Analysis

		+/-
Entry High:	69	9
Reject High:	68	8
Warning High:	66	6
Target:	60	
Warning Low:	55	5
Reject Low:	54	6
Entry Low:	52	8

[✓ Test Sequence] Disposition Sequence
Total QSpec in QP: 5

localhost Honeywell.QualityOptiMISER.Presentation.QualSpecManager

FIGURE 2. Quality Procedure Manager's user interface (Honeywell.)

With Contour Map the quality data can be shown as a two- or three-dimensional graphical representation. The data shown can be historical or current and the current data is updated in real time, so the production problems can identified as fast as possible. (Honeywell Process Solutions 2014b.)

Trends allow quality parameters to be trended on a selected time period. (Honeywell Process Solutions 2014b.) Optional applications for Quality OptiMISER are:

- As Cut Roll/Set
- Quality Diary and Lab Entry
- Multivariable Profile
- Web Inspection System Fault Map Overlay

Quality Diary and Lab Entry was the only essential optional application in this project. Lab test results can be manually entered and stored with it. With quality diary interesting data can be shown while the uninteresting data is hided. (Honeywell Process Solutions 2014b

2.8 NetEye

NetEye is a third party software that monitors for example:

- Workstations
- Servers
- Processes
- Network devices.

NetEye sends alerts as a text message or an email or both if some monitored device, process or service does not work as intended. Also resources can be monitored with NetEye, for example if the CPU or memory usage crosses a set limit, NetEye sends an alert notification. The user can produce reports of monitored servers, processes and devices. Figure 3 shows how the NetEye works. (Honeywell.)

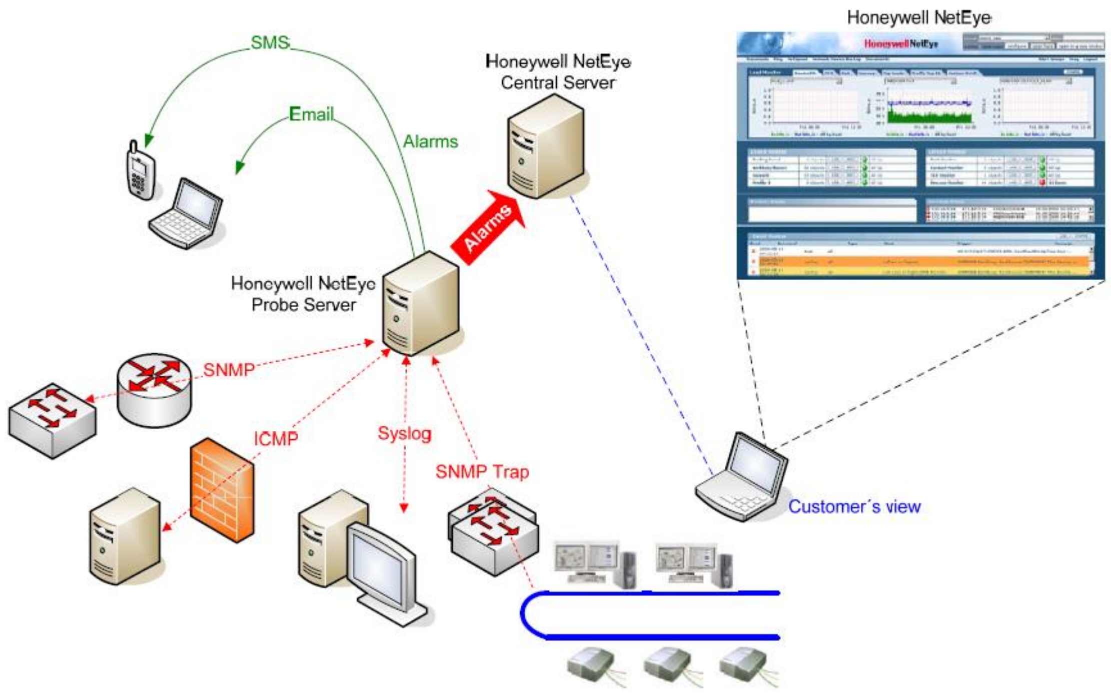


FIGURE 3. Graph of how the Honeywell's NetEye works (Honeywell.)

3 IDENTIFYING THE ESSENTIAL RAW DATA FROM A PRODUCTION DATABASE

The main part of this thesis was to load the data needed on the reports from the production database and create the reports. The essential data was loaded to the data warehouse with SQL Server Integration Services (SSIS) and the reports were done with SQL Server Reporting Services (SSRS). Most of the data refining was done in the stored procedures and some of the reports needed multiple procedures. There were dozens of SSIS packages and reports done during this project.

3.1 Gathering the data from the automation system

PHD Collector servers collect the data from the automation system at the mill based on its tag configuration. Every sample place has its own PHD tag or PHD virtual tag. This data is then combined and saved into a database at the PHD Shadow server. In the next step Quality OptiMiser server loads the data from the PHD Shadow server based on PHD tags, and if needed makes additional calculations and finally loads data into the OptiVision database. This data flow is shown in Figure 4.

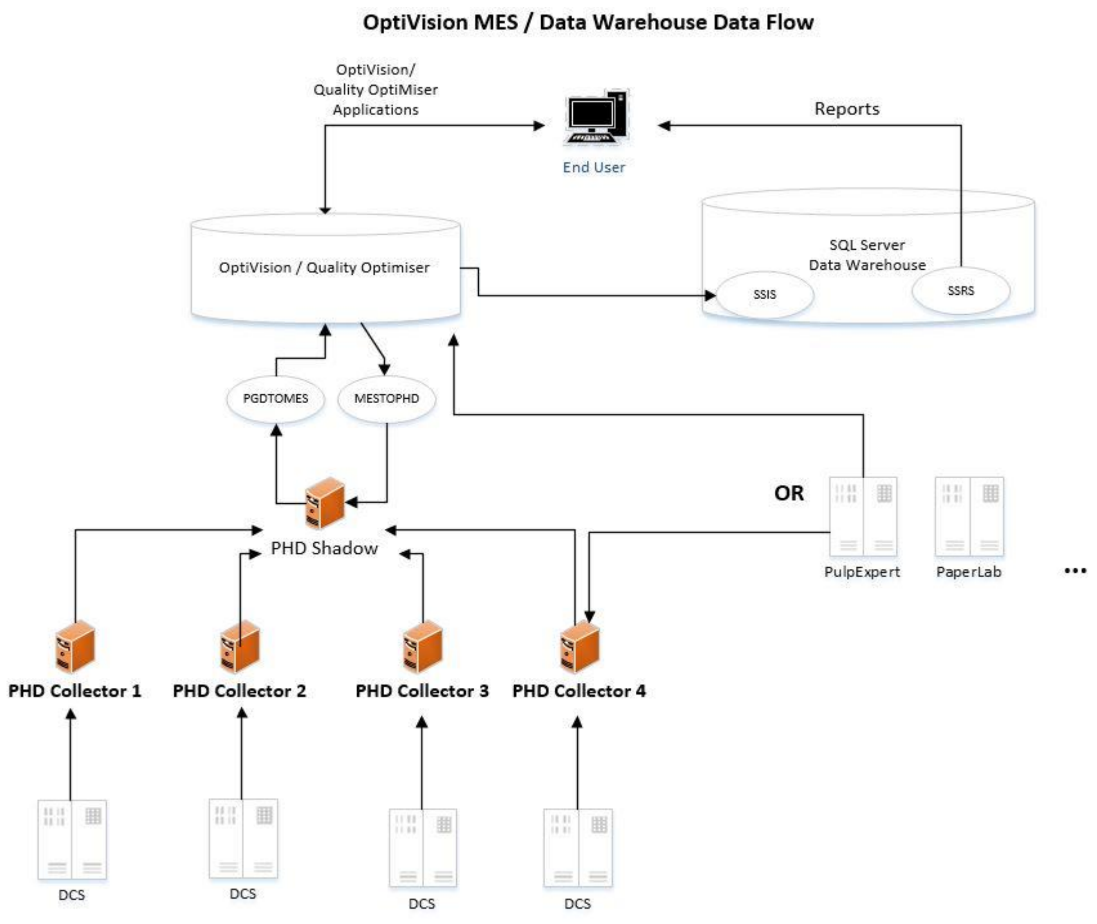
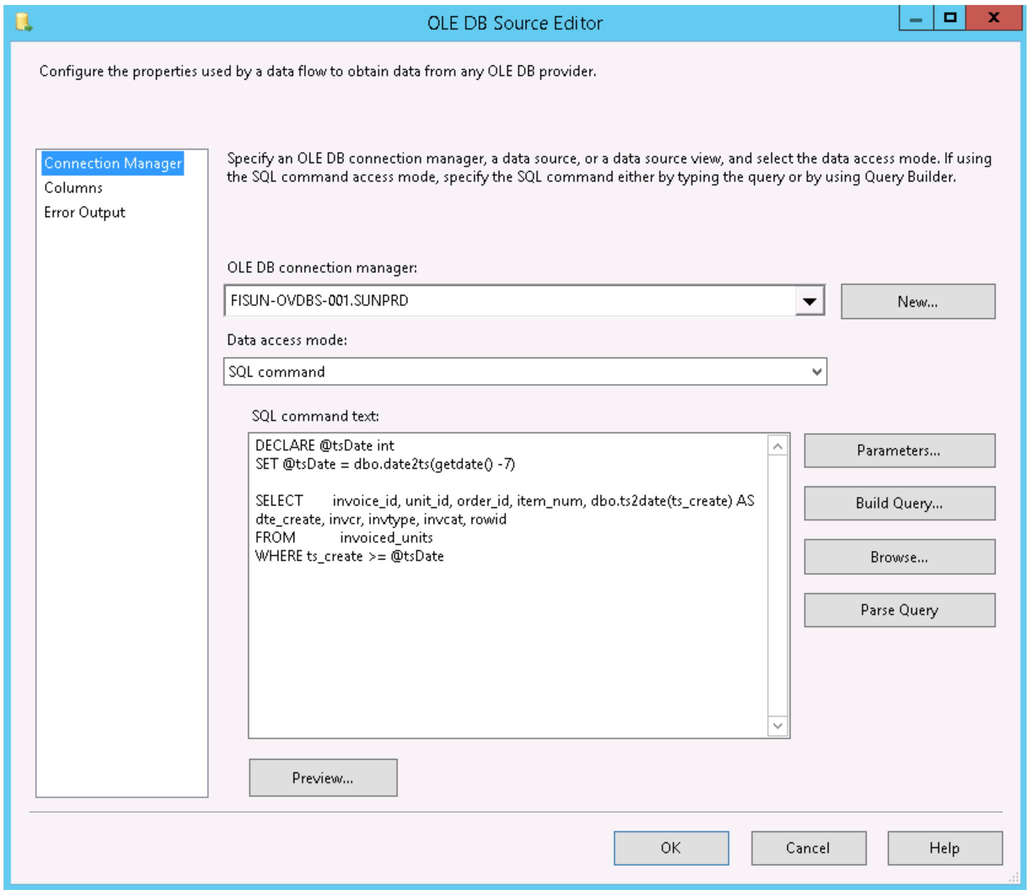


FIGURE 4. Graph of the data flow

3.2 Loading the production data to the data warehouse

Before the transfer of data from the source database is started, the essential data for reporting should be identified. Unnecessary data should be avoided, since it only causes extra load to the system. After the essential data for reporting in the OptiVision database has been specified, this data is then transferred to the data warehouse, in this project this is done with SSIS packages which are created by Microsoft Visual Studio.

The data needed on the reports is selected with SQL query (Figure 5) or the whole table can be selected and then the creation of actual SQL query by user is not needed. The OLE DB connection manager is set to the source database and Data access in this project is selected between SQL command and Table or view selection.


OLE DB Source Editor

Configure the properties used by a data flow to obtain data from any OLE DB provider.

Specify an OLE DB connection manager, a data source, or a data source view, and select the data access mode. If using the SQL command access mode, specify the SQL command either by typing the query or by using Query Builder.

OLE DB connection manager: FISUN-OVDBS-001.SUNPRD

Data access mode: SQL command

SQL command text:

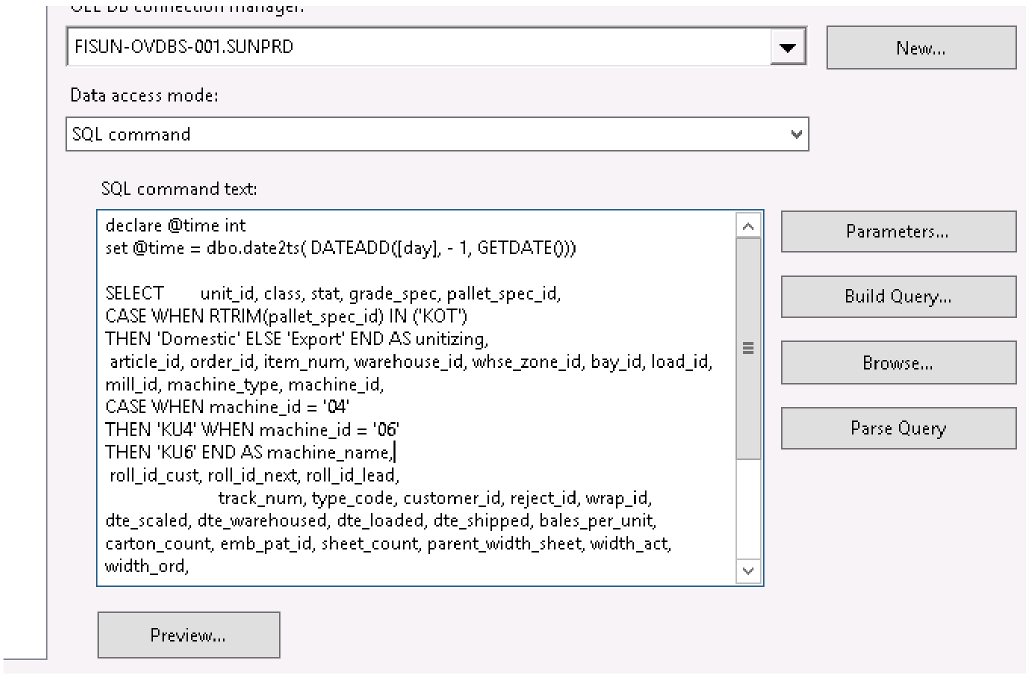
```
DECLARE @tsDate int
SET @tsDate = dbo.date2ts(getdate() -7)

SELECT      invoice_id, unit_id, order_id, item_num, dbo.ts2date(ts_create) AS
dte_create, invcr, invtype, invcat, rowid
FROM          invoiced_units
WHERE ts_create >= @tsDate
```

FIGURE 5. Selecting the data from source database with SQL query in SSIS

On the first data loading there are no limitations on how far from the history the data is taken because all the data is needed. During data loadings later on, the data from last couple of days is loaded. Some of the laboratory data is added in the OptiVision database some days later based on the actual lab measurements and there may be also corrections to other data. This is why the data is loaded from last couple of days rather than only from the last day.

The date condition is converted to Unix Timestamp and saved in variable because OptiVision uses this time format and if the date was converted to the Unix Timestamp format in WHERE clause, the conversion would be done to every row of data and the query would be many times slower and heavier.

Most of the data refining in this project is done after the data has been loaded to the data warehouse but this data can be also modified in the SSIS packages. If all of the data refining is done in the SSIS packages, almost every report needs its own package. In this project only data needed on every report was modified in the SSIS packages. For example Figure 6 shows how machine name is generated from the machine id and how unitizing is generated from the pallet spec id.



FISUN-OVDBS-001.SUNPRD

SQL command

```
declare @time int
set @time = dbo.date2ts( DATEADD([day], - 1, GETDATE()))

SELECT      unit_id, class, stat, grade_spec, pallet_spec_id,
CASE WHEN RTRIM(pallet_spec_id) IN ('KOT')
THEN 'Domestic' ELSE 'Export' END AS unitizing,
 article_id, order_id, item_num, warehouse_id, whse_zone_id, bay_id, load_id,
mill_id, machine_type, machine_id,
CASE WHEN machine_id = '04'
THEN 'KU4' WHEN machine_id = '06'
THEN 'KU6' END AS machine_name,
 roll_id_cust, roll_id_next, roll_id_lead,
                    track_num, type_code, customer_id, reject_id, wrap_id,
dte_scaled, dte_warehoused, dte_loaded, dte_shipped, bales_per_unit,
carton_count, emb_pat_id, sheet_count, parent_width_sheet, width_act,
width_ord,
```

FIGURE 6. SELECT-clause with data refining using CASE expression

When SSIS package is executed, it first truncates the temporary table to make sure the table. After this the package loads the data from source database to temporary database and merges it with da-ta warehouse table (Figure 7).

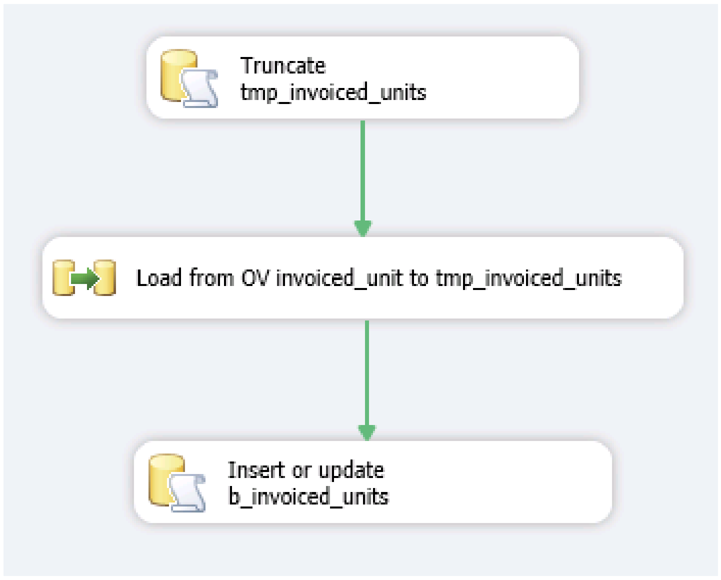
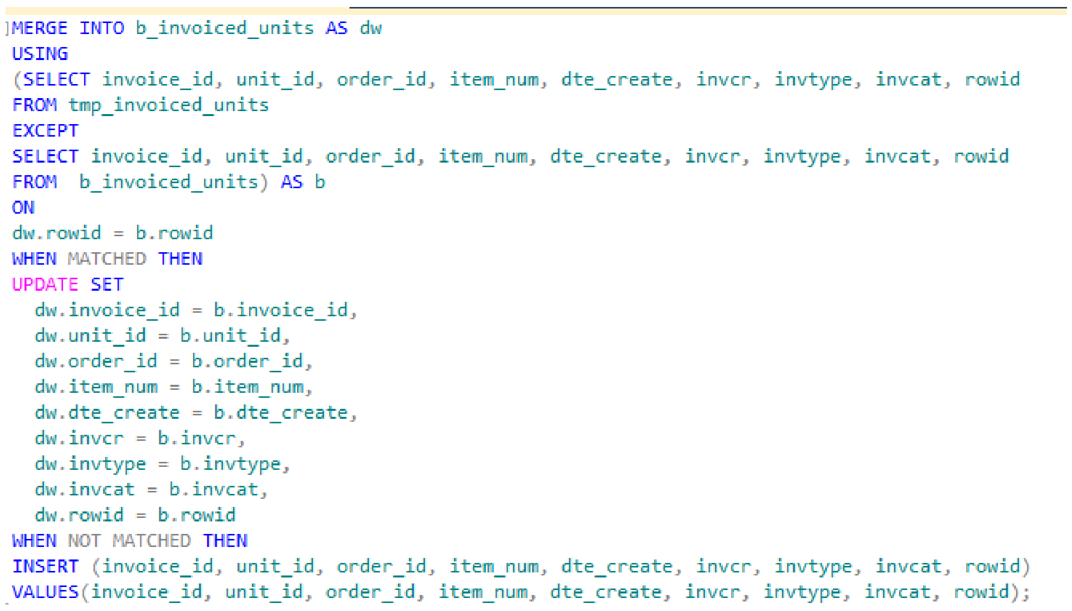

Truncate tmp_invoiced_units
Load from OV invoiced_unit to tmp_invoiced_units
Insert or update b_invoiced_units

FIGURE 7. Example structure of a SSIS package

Insert or update b_invoiced_units step in Figure 7 contains merge query which merges the data in temporary table and the data in data warehouse table. Query in Figure 8 first selects all the data from temporary table except the data found from destination table with exactly same data. After this the temporary table's rowid column is compared to destination table's rowid column and if they match, the query updates the destination table and if not, it inserts the data to the destination table.



```
MERGE INTO b_invoiced_units AS dw
USING
(SELECT invoice_id, unit_id, order_id, item_num, dte_create, invcr, invtype, invcat, rowid
FROM tmp_invoiced_units
EXCEPT
SELECT invoice_id, unit_id, order_id, item_num, dte_create, invcr, invtype, invcat, rowid
FROM  b_invoiced_units) AS b
ON
dw.rowid = b.rowid
WHEN MATCHED THEN
UPDATE SET
  dw.invoice_id = b.invoice_id,
  dw.unit_id = b.unit_id,
  dw.order_id = b.order_id,
  dw.item_num = b.item_num,
  dw.dte_create = b.dte_create,
  dw.invcr = b.invcr,
  dw.invtype = b.invtype,
  dw.invcat = b.invcat,
  dw.rowid = b.rowid
WHEN NOT MATCHED THEN
INSERT (invoice_id, unit_id, order_id, item_num, dte_create, invcr, invtype, invcat, rowid)
VALUES(invoice_id, unit_id, order_id, item_num, dte_create, invcr, invtype, invcat, rowid);
```

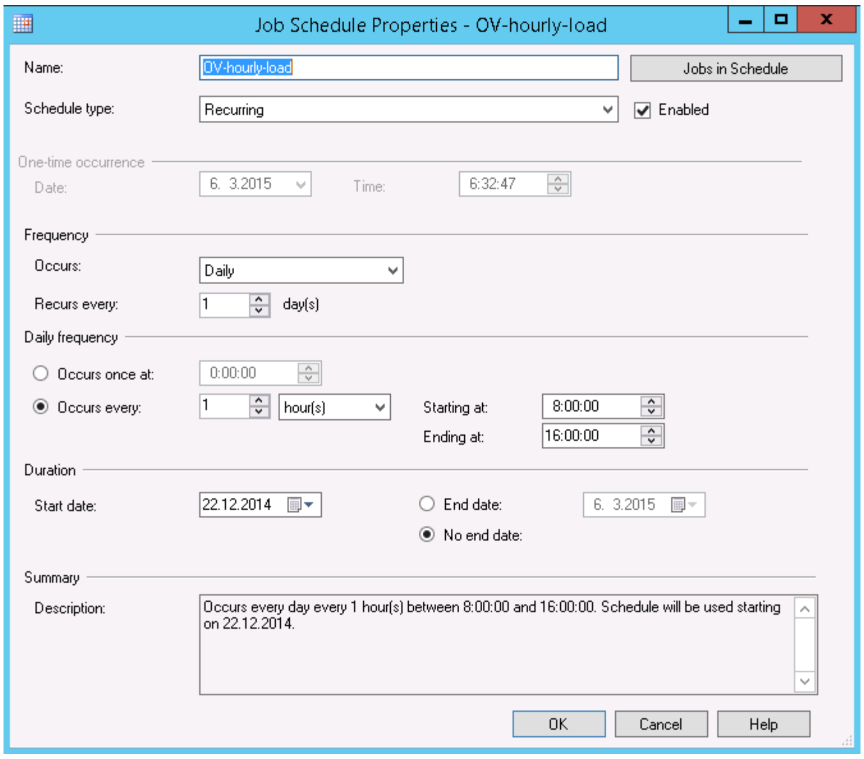
FIGURE 8. Example of a merge query between a temporary table and a data warehouse table

There were nearly 20 different SSIS packages in this project. Most of the packages were for transferring the production data from production database to the data warehouse. Some packages were for refining the data in the data warehouse. This data refining could have been done in database views or in stored procedures but it would have loaded the system on every use. When this was done in the procedures, it only occurred once in the morning. After this the information can be acquired from a single table and this barely loads the system.

3.3 Scheduling the execution of SSIS packages

A SSIS solution can contain many packages and in this project all tables from one database are loaded in one solution and every table has its own package. After the solution and all its packages are complete and tested, it can be deployed. The whole project needs to be deployed since SSIS packages in one project cannot be deployed separately.

The execution schedule for SSIS packages is done with SQL Server Agent, after the package is deployed, otherwise the SQL Server agent will not find it. SQL Server Agent executes scheduled jobs, a job may contain one or many steps. (SQL Server Agent 2015.) Figure 9 shows the settings of OV-hourly-load job, which is executed every hour between 08:00 and 16:00 every day.


Job Schedule Properties - OV-hourly-load

Name: OV-hourly-load
Schedule type: Recurring
[✓ Enabled]

One-time occurrence
Date: 6. 3.2015 Time: 6:32:47

Frequency
Occurs: Daily
Recurs every: 1 day(s)

Daily frequency
○ Occurs once at: 0:00:00
● Occurs every: 1 hour(s) Starting at: 8:00:00 Ending at: 16:00:00

Duration
Start date: 22.12.2014
○ End date: 6. 3.2015
● No end date:

Summary
Description: Occurs every day every 1 hour(s) between 8:00:00 and 16:00:00. Schedule will be used starting on 22.12.2014.

FIGURE 9. Example settings of job execution schedule in SQL Server Agent.

Some of the SSIS packages has to be executed every hour during office hours in order to get the essential data to reports almost in real time. OptiVision data loading solution has two jobs in SQL Server Agent, one which is executed once a day at 06 in the morning and it loads the data from last seven days. Other one is in picture 5 and it is executed every hour but it only loads data from current day.

There are 2 versions of every package that has to be executed every hour during the office hours. The package which loads data from the last seven days would be quite a heavy process to be executed every hour and the data is already loaded once in the morning, so only the data from current day is essential for hourly executed packages.

3.4 Report development

The production data from different tables is joined together in database views and report gets its data from these views with a stored procedure or a direct SELECT-clause. In most cases in this project it was more practical to create a stored procedure because handling of the data is easier that way. The columns selected and calculations made in a SELECT-clause or in a stored procedure should be specified in designs as precisely as possible, because it makes the creation of a SELECT-clause or a stored procedure much faster and easier.

In the first appendix is an example of a stored procedure. At the beginning of a stored procedure the report parameters are introduced and then the temporary tables are created. These temporary tables are used to store a data before taking it to the report. In this example the data is stored in multiple temporary tables because some of the data is aggregated and grouped differently from the other, so they are calculated separately and then joined together before the data is delivered to the report. There are temporary tables for:

- All the data matching report parameters
- Total weight per invoice
- Total weight of selected dataset
- Unit count

First all the data that matches with the report parameters is loaded to the temporary table called #temp and the data for all the other temporary tables is loaded from this #temp table.

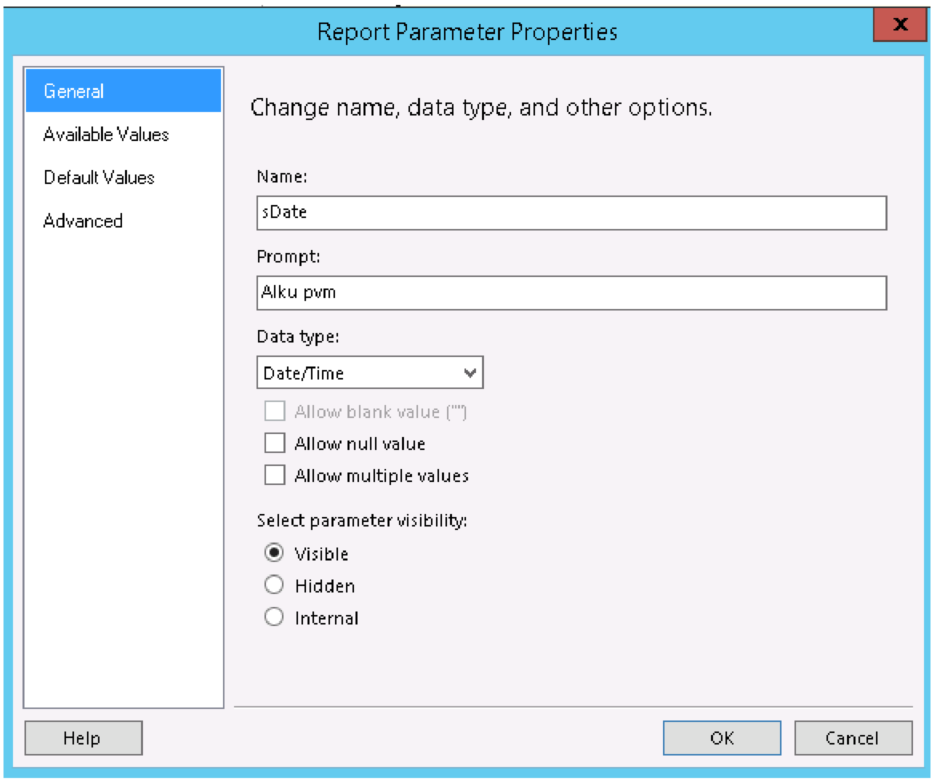
In the IF-statement of the first appendix, the content of the column headers is set based on the language selected in the report parameters. In some reports the parameter selected may change the whole query and this is handled with the IF-statements. At the end of the stored procedure all the temporary tables and possibly other tables also if needed, are joined together and delivered to the report.

There were dozens of stored procedures in this project, some reports needed even seven different procedures. Some of the parameter lists were also made with stored procedures, because in that way the information that was not in the database, was rather easy to be added. The length of the procedures varied a lot, some were ten lines of code and some were hundreds of lines.

The actual report is created with Microsoft Visual Studio. A SSRS project contains one or many reports and a SSRS solution also contains one or many SSRS projects. A SSRS project can be created without it being in any solution. Reports can be created with a report wizard or from scratch. Report Wizard can be a handy tool when creating reports for the first time but creating and configuring everything manually is in fact faster and even easier when being familiar with report creation. The result is exactly as wanted when everything is done manually.

There were two shared data sources in this project, which means that each one of them has to be created and setup only once, after that it is possible to choose them from the dropdown list. After the report is created and data source or sources are selected, report parameters should be created first so that the stored procedure created for the report can communicate with the report.

If the dataset is setup before the report parameters, an error message is raised even if the dataset and stored procedure is setup correctly, because the parameters do not match between the stored procedure and the report.


Report Parameter Properties

Change name, data type, and other options.

Name: sDate

Prompt: Alku pvm

Data type: Date/Time

- [ ] Allow blank value ("")
- [ ] Allow null value
- [ ] Allow multiple values

Select parameter visibility: [✓ Visible] Hidden Internal

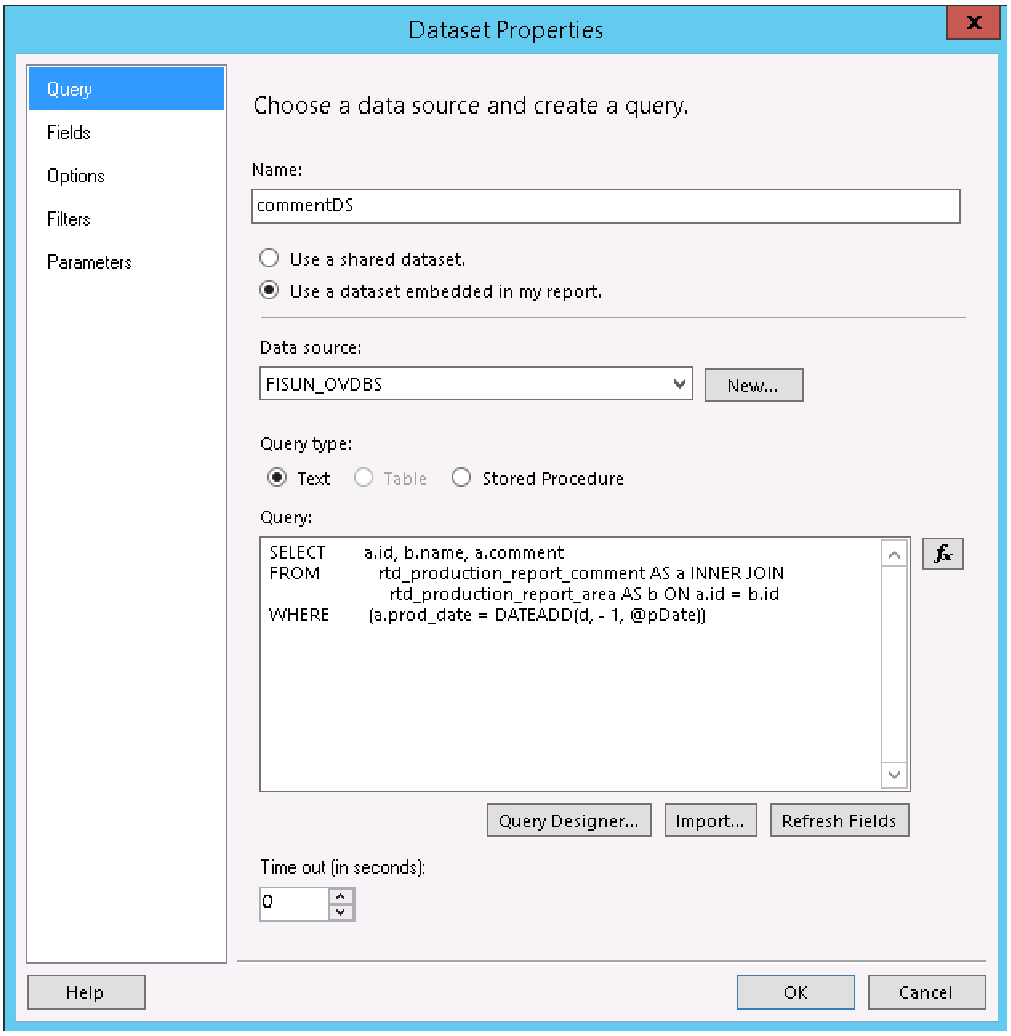
FIGURE 10. Properties of a report parameter.

In Figure 10 is an example of a report parameter, in general tab are:

- Name of the parameter
- Prompt, the text shown to user of a report
- Data type of a parameter
- Allow blank, null or multiple values
- Visibility of a parameter

In the Available Values and the Default Values tabs, the values can be set to come from a query or they can be set manually. In the Advanced tab, refreshing of the data can be set to happen always, automatically or never, when parameter changes.

In Figure 11 are properties of a dataset, first dataset needs a name and data source, then data is got by text query or a stored procedure. Dataset in picture 7 gets its data by text query because it is a simple dataset that only gets a couple of columns from two tables and the data does not need to be modified or calculated.



Choose a data source and create a query.

Name: commentDS

( ) Use a shared dataset.
(●) Use a dataset embedded in my report.

Data source: FISUN_OVDBS

Query type: (●) Text ( ) Table ( ) Stored Procedure

Query:

```
SELECT     a.id, b.name, a.comment
FROM         rtd_production_report_comment AS a INNER JOIN
                 rtd_production_report_area AS b ON a.id = b.id
WHERE      (a.prod_date = DATEADD(d, - 1, @pDate))
```

Time out (in seconds): 0

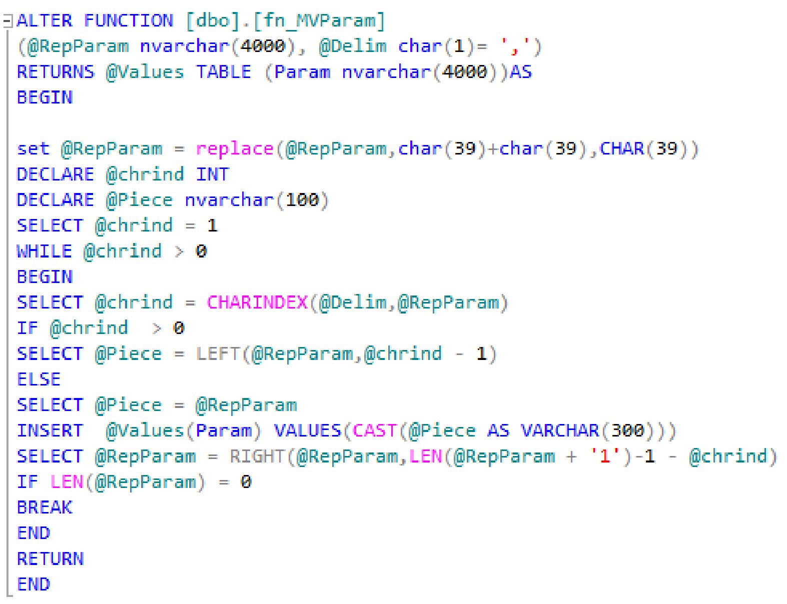
FIGURE 11. Properties of a dataset

With more complex datasets the stored procedure is usually better choice. When the Stored Procedure radio button is selected, the stored procedure wanted can be selected from the drop down list but the stored procedure has to be created before it can be selected.

In the Options tab of Dataset Properties, the following properties can be set:

- Collation
- Case sensitivity
- Accent sensitivity
- Kanatype sensitivity
- Width sensitivity
- Interpret subtotals as detail rows

In the Filters tab, filtering for rows with the specific data, can be added, removed and managed. The report parameters are managed in the Parameters tab and the fields are managed in the Fields tab, many times when using a stored procedure and a new parameter is created after the dataset, it has to be manually added in the Parameters tab.

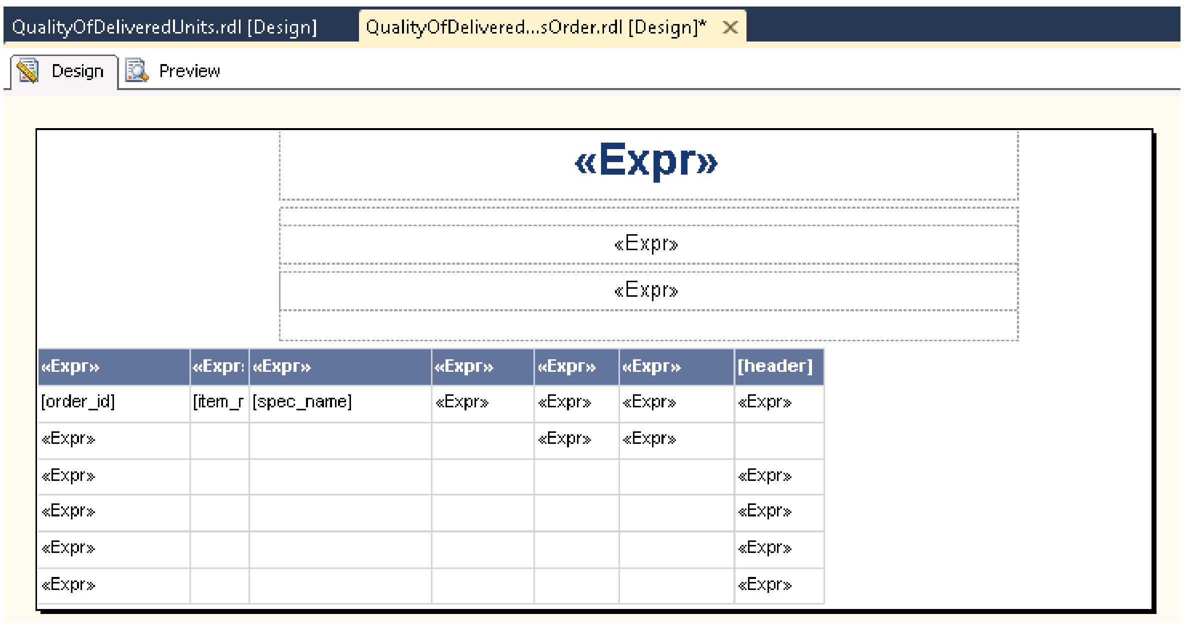


```
ALTER FUNCTION [dbo].[fn_MVParam]
(@RepParam nvarchar(4000), @Delim char(1)= ',')
RETURNS @Values TABLE (Param nvarchar(4000))AS
BEGIN

set @RepParam = replace(@RepParam,char(39)+char(39),CHAR(39))
DECLARE @chrind INT
DECLARE @Piece nvarchar(100)
SELECT @chrind = 1
WHILE @chrind > 0
BEGIN
SELECT @chrind = CHARINDEX(@Delim,@RepParam)
IF @chrind  > 0
SELECT @Piece = LEFT(@RepParam,@chrind - 1)
ELSE
SELECT @Piece = @RepParam
INSERT  @Values(Param) VALUES(CAST(@Piece AS VARCHAR(300)))
SELECT @RepParam = RIGHT(@RepParam,LEN(@RepParam + '1')-1 - @chrind)
IF LEN(@RepParam) = 0
BREAK
END
RETURN
END
```

FIGURE 12. Function for multiple value parameters handled in stored procedure.

When a parameter has multiple values, the stored procedure can not directly handle it, for example in WHERE-condition. In appendix 1 parameters with multiple values are called with: "WHERE machine_id IN (select * from fn_MVparam(@pMachine))", fn_MVparam is a custom function which modifies the parameter string so that a stored procedure can handle it, the function is shown in Figure 12. Dataset with text query does not need a custom function because it can handle the parameter with IN – condition, for example "WHERE machine_id IN (@pMachine)".

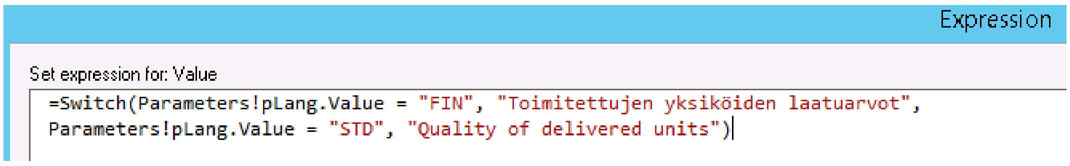
After the dataset is setup the data needs to be displayed, this can be done for example with graphs or tables. Table can be done so it creates fields dynamically based on the data or the fields can be done manually beforehand. In most cases, dynamic fields are the best way to go at least in this project. Many reports contain multiple tables and usually every table needs its own dataset and stored procedure.


QualityOfDeliveredUnits.rdl [Design]
QualityOfDelivered...sOrder.rdl [Design]*
Design
Preview
«Expr»
«Expr»
«Expr»

«Expr»	«Expr:	«Expr»	«Expr»	«Expr»	«Expr»	[header]
[order_id]	[item_r	[spec_name]	«Expr»	«Expr»	«Expr»	«Expr»
«Expr»				«Expr»	«Expr»	
«Expr»						«Expr»
«Expr»						«Expr»
«Expr»						«Expr»
«Expr»						«Expr»

FIGURE 13. Structure of a report in design mode.

Figure 13 shows an example of a report in the design mode. In most of the fields are only text "<<Expr>>", this is because this report has two language options and it is handled with an expression, example in Figure 14. Order id is the grouping parameter for rows in Figure 13, so for every order id a row will be created. Below the order id row there are summary rows and above a header row. Every column is static except the one on the right with the header "[header]", this column has qparam as its column grouping parameter. For every quality parameter selected a column will be created.



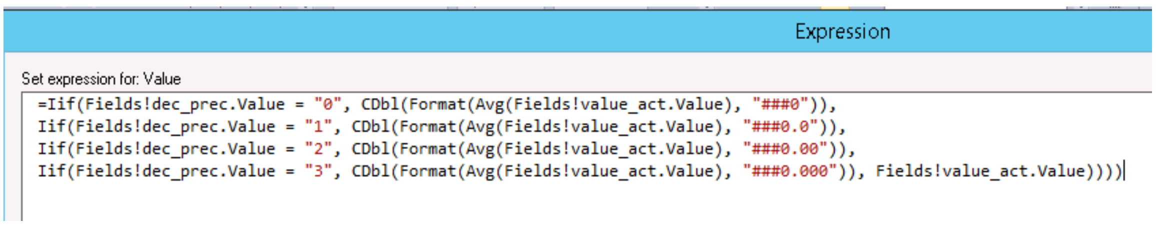
```
=Switch(Parameters!pLang.Value = "FIN", "Toimitettujen yksiköiden laatuarvot",
Parameters!pLang.Value = "STD", "Quality of delivered units")
```

FIGURE 14. Expression to change text of a field based on language selected.

In SSRS the content of a field can be altered with an expression, it can execute custom code or premade functions for:

- Math
- Text
- Date and time
- Inspection
- Program Flow
- Aggregate
- Financial
- Conversion

The Switch function is used to set a text of a textbox in Figure 14. If the pLang parameter value is "FIN" it sets the text to "Toimitettujen yksiköiden laatuarvot" and if the parameter's value is STD it sets the text to "Quality of delivered units".



```
=Iif(Fields!dec_prec.Value = "0", CDbl(Format(Avg(Fields!value_act.Value), "###0")),
Iif(Fields!dec_prec.Value = "1", CDbl(Format(Avg(Fields!value_act.Value), "###0.0")),
Iif(Fields!dec_prec.Value = "2", CDbl(Format(Avg(Fields!value_act.Value), "###0.00")),
Iif(Fields!dec_prec.Value = "3", CDbl(Format(Avg(Fields!value_act.Value), "###0.000")), Fields!value_act.Value))))
```

```
CASE    WHEN c.dec_prec = 0 THEN ROUND(a.value_act, 0)
        WHEN c.dec_prec = 1 THEN ROUND(a.value_act, 1)
        WHEN c.dec_prec = 2 THEN ROUND(a.value_act, 2)
        WHEN c.dec_prec = 3 THEN ROUND(a.value_act, 3) END AS rounded_value
```

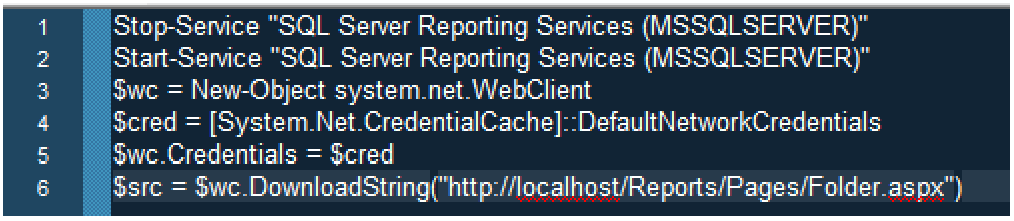
FIGURE 15. Rounding of numeric value in report expression and in SQL query.

Most of the functions used in expressions can also be used in the stored procedure and vice versa, Figure 15 shows an example of rounding a numeric value in the expression and in the stored procedure. In practice they both do the same thing but in this case the stored procedure version is simpler and better. When rounding is done in the expression with the Format-function it converts the numeric value to text and it has to be converted back to integer. There are many more ways to do this kind of conversion and the best way is found with testing the different possibilities.

Some functions like standard deviation will not work as intended if there is only one value and text to integer conversions, in this case it would only give error message: “NaN”. But when the rounding is done in stored procedure the standard deviation function gives value zero as it should. Nevertheless in some cases expression is better place to use functions than stored procedure, because it uses functions to every row and column it dynamically creates.

At some point it was noticed that sometimes the report manager starts very slowly. After some research, the reason was revealed to that by default SSRS Services recycles the data after 720 minutes and goes into some kind of a sleep mode. To work around this sleep mode, the recycle time has to be changed in rsreportserver.config file. In this project the recycle time was changed to 1440 minutes which is 24 hours.

The change of recycle time itself will not fix the problem, because the SSRS Services still goes to sleep mode. The problem with the sleep mode was fixed with PowerShell script (Figure 16). The script was scheduled to happen every day at 04 AM with Windows’s Task Scheduler.



```
Stop-Service "SQL Server Reporting Services (MSSQLSERVER)"
Start-Service "SQL Server Reporting Services (MSSQLSERVER)"
$wc = New-Object system.net.WebClient
$cred = [System.Net.CredentialCache]::DefaultNetworkCredentials
$wc.Credentials = $cred
$src = $wc.DownloadString("http://localhost/Reports/Pages/Folder.aspx")
```

FIGURE 16. PowerShell script which prevents the SSRS Services from going to sleep

The PowerShell script preventing the SSRS service from going to sleep is shown in Figure 16. First the script stops the SSRS Services and then starts it again. After this the script forces the report manager to load for the first time, by downloading the report manager page as a string.

3.5 Testing and deployment of reports

After the report is done it has to be tested with every possible parameter combinations. When the report is tested with every possible ways, the data should be validated. If raw data does not change a lot between PHD and report, the numbers on report should be compared to the data on PHD.

Data that comes from Quality OptiMISER should be compared to it with quality diary. When there is differences between quality OptiMISER and the report, the difference is most likely caused by programming mistake. Sometimes the data between Quality OptiMISER and the report matches but it is known that the data is not valid. In this case the first thing to check is the calculations made in the Quality OptiMISER. If the calculations are correct or there are none, the next step is to check from

Quality Procedure Manager what PHD tag is assigned to the quality property. If the tag is correct, then the reason for false data is before PHD.

When the report is tested it needs to be deployed to report server. Otherwise than with SSIS, with SSRS you can deploy each report in the project separately or the whole project or solution. Before the reports can be deployed, the reporting server has to be defined in the project properties. Reports can also be deployed manually to the server. In this case the reports are uploaded to the server with browser. The address of Report Manager is usually: reportserver/Reports/Pages/Folder.aspx.

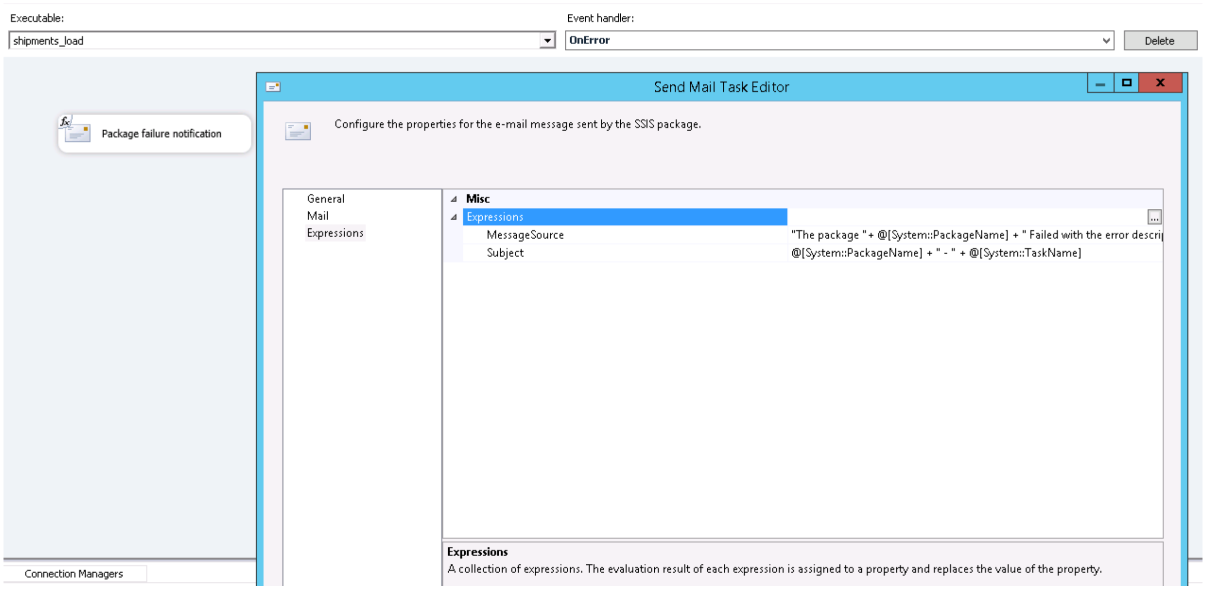
4 ERROR HANDLING

It is of course important to try to prevent errors at all-time but no matter what is done, errors are inevitable. It is important to specify how to notice an error as soon as possible when it happens and specify how to fix it as efficiently as possible.

4.1 Checkpoints for errors

Most of the error monitoring in this project is done with Honeywell's NetEye. It monitors that the servers and services are up and running.

When loading the data from the production database to the data warehouse database, the check-points are handled by creating an event handler to every package (Figure 17). The event handler sends an email if the package fails, the email contains information of what caused the error. If the SSIS package has worked in the first place, the error is probably caused by a duplicate row or some-thing has changed in the source database.



Configure the properties for the e-mail message sent by the SSIS package.

Misc	
Expressions	
MessageSource	"The package "+ @[System::PackageName] + " Failed with the error descri
Subject	@[System::PackageName] + " - " + @[System::TaskName]

Expressions
A collection of expressions. The evaluation result of each expression is assigned to a property and replaces the value of the property.

FIGURE 17. SSIS event handler with Send Mail Task

As shown in Figure 17 after the event handler and a send mail task is created, the send mail task has to be configured. In General tab of Send Mail Task Editor, the name and the description of the task are set. In Mail tab, the mail settings are set, including:

- SmtpConnection
- From
- To
- CC
- BCc

- Subject
- MessageSourceType
- MessageSource
- Priority
- Attachments

First the SMTP connection has to be set up with SMTP Connection Manager. Basically SMTP Connection Manager only needs the name of the SMTP server and the connection is then configured. When the connection is created, it can be selected from the SmtpConnection drop down list.

Other content in the Mail tab are the contents of the mail; the subject and the actual message are generated with expressions in this case. In Expressions tab, for example the content of the mail can be set with expressions. In Figure 17, the subject and the message of the error email are generated with expressions.

After the report is done and tested, there should not be any major errors with it as long as the SSIS loads the data to the data warehouse. Errors in the stored procedures or in the actual report files are fixed during testing, if the testing is done correctly.

After in-depth testing of the reports, basically only errors that may occur in the report files are SSRS service or reporting server being down. As mentioned before, servers and services are monitored with NetEye, so in problem situations NetEye handles the error notifications. Of course if there are malfunctions before the actual report, the results of the malfunction also show on the report. Also if there have been modifications to the data, it may require actions with the stored procedures or report files.

4.2 Error types and sections

Errors are mostly caused by operators' mistakes, hardware malfunctions, or a bug or malfunction in a software. Bugs caused by programming mistakes are kind of easier errors, because they can be noticed and fixed during testing when it does not affect the production. When hardware breaks or malfunctions, it usually has to be replaced or fixed and with high probability it breaks when the production has already started.

Errors usually occur in this system on one of the following sections:

- PHD
- Quality OptiMISER
- SSIS packages
- Database procedures
- Calculations made on the report

NetEye monitors every section above but it only sends the alerts if the whole server or the service goes down or some resource reaches limit value. Errors that are not caught on the checkpoints, are usually noticed not until on the report as false data.

If the false data is noticed not until on the report, the easiest way to locate the cause of the error, is to compare the data on the report, first to the data on Quality OptiMISER. If the data between Quality OptiMISER and the report does not match, the error is located in the SSIS package, database procedure or on the calculations made on the report.

If the data matches between Quality OptiMISER and the report, the error is located in one of the earlier steps and it can be found by comparing every step to each other. If there are no differences among steps, the problem is probably in automation system.

5 SUMMARY

The goal of this thesis was to create working reporting and transfer the production data from production database to the data warehouse database. Additionaly this process and steps needed for successful reporting was documented. At the startup all the reports were working, all the data was successfully transferred to the data warehouse server and all SSIS packages were working and also this report works as a documentation of the process.

There was a little problem with the SSIS error email event handlers. When the error email event handlers were tested, there was no antivirus software installed on the data warehouse server and the error emails worked exactly as they were planned to work. After the antivirus software was installed to the data warehouse server, it blocked all the error email. Fortunately this was noticed with the first SSIS package failure and the exceptions for SSIS were made to the antivirus software.

Other than that there were basically only modifications to some of the report layouts and decimal precision. Most of the layout changes were made because a report had to fit on one page when exported to a pdf.

5.1 Further Development

The system created in this project needs to be maintained and monitored. There will be some new features added later, meaning new reports and possibly SSIS packages for them.

Even though the development of the SSIS packages and the reports went smoothly, improvements can be made in next project based on this project. For example some of the stored procedures needed a lot of adjusting because they were approached from the wrong direction in the beginning. Also some of the calculations on the reports or in the stored procedures were made too complicated in the beginning, although there were simpler ways to do it.

The SSIS packages in this project were designed and developed in the way they can be used also on other projects with little modifications. The packages did not refine the data so far, that they would have became usable only in this project. Most of the project specific data refining was aimed to happen in the stored procedures, so that the reports would also be as flexible as possible between projects.

REFERENCES

Honeywell. (Year not mentioned). NetEye Remote Monitoring

Honeywell Process Solutions. 2014a. Uniformance PHD. [accessed 7 March 2015] Available from: https://www.honeywellprocess.com/library/marketing/notes/Uniformance-PHD-PIN.pdf

Honeywell Process Solutions. 2014b. Quality OptiMISER – Manage Quality Data [accessed 11 March 2015] Available from: https://www.honeywellprocess.com/library/marketing/notes/quality-optimiser-pin.pdf

Honeywell Process Solutions. 2013. OptiVision – Unified Business & Manufacturing Solutions for Pulp, Paper & Flat Sheet Industries. [accessed 29 March 2015] Available from: https://www.honeywellprocess.com/library/marketing/notes/optivision_pin.pdf

Microsoft SQL Server Data Tools… 2015. [accessed 26 March 2015] Available from: https://msdn.microsoft.com/en-us/data/tools.aspx

Microsoft SQL Server. 2015. [accessed 10 February 2015] Available from: https://msdn.microsoft.com/en-us/library/bb545450.aspx

Overview of Extraction, Transformation, and Loading. 2014. [accessed 22 February 2015] Available from: http://docs.oracle.com/cd/B19306_01/server.102/b14223/ettover.htm

POP3-, SMTP- ja muut sähköpostipalvelimien tyypit. 2015. [accessed 5 April 2015] Available from: http://windows.microsoft.com/fi-fi/windows-vista/pop3-smtp-and-other-e-mail-server-types

Reporting Services (SSRS). 2015.) [accessed 20 February 2015] Available from: https://msdn.microsoft.com/en-us/library/ms159106.aspx

SQL Server Agent. 2015. [accessed 29 March 2015] Available from: https://msdn.microsoft.com/en-us/library/ms189237.aspx

SSIS Tutorial: Creating a Simple ETL Package. 2015. [accessed 15 February 2015] Available from: https://msdn.microsoft.com/en-us/library/ms169917.aspx

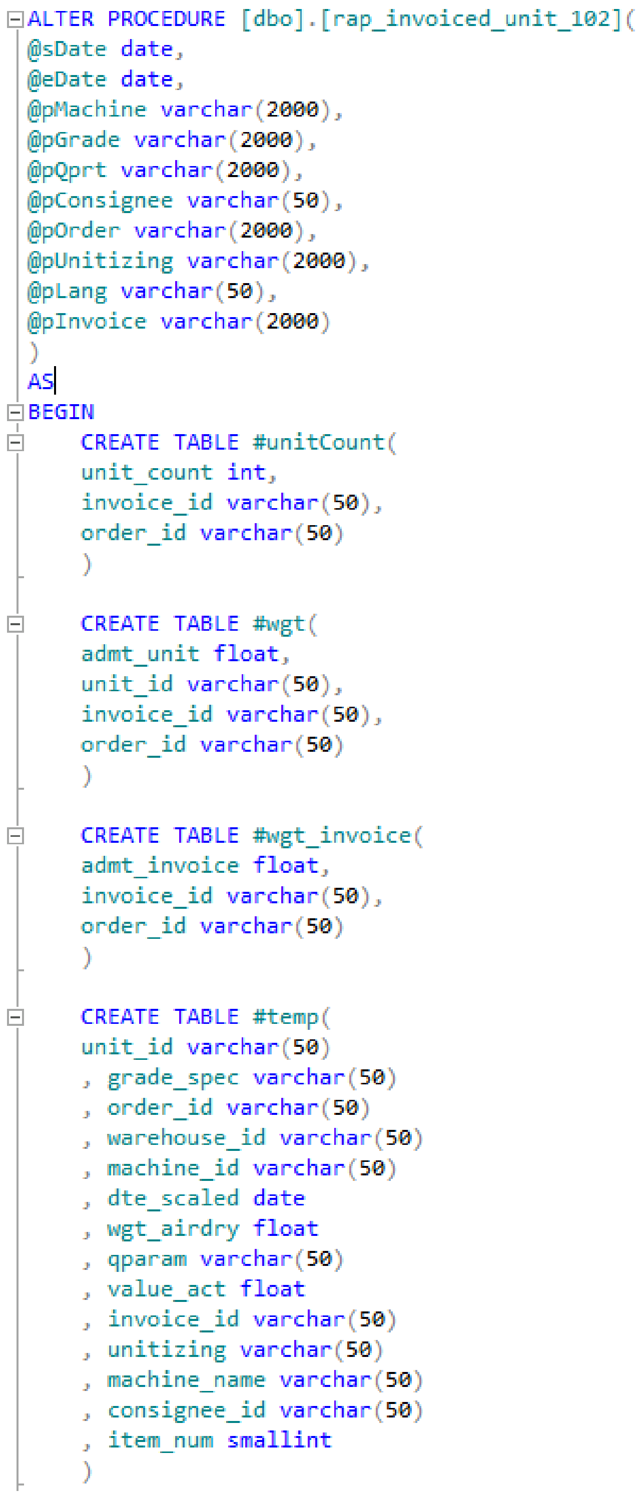
Transact-SQL Reference(Transact-SQL). 2015. [accessed 10 February 2015] Available from: https://technet.microsoft.com/en-us/library/ms189826(v=sql.90).aspx

VINHAIS, Joseph A. 1998. Manufacturing Execution Systems: The One-Stop Information Source [accessed 25 March 2015] Available from: http://www.qualitydigest.com/sept98/html/mes.html

WAILGUM, Thomas. 2007. ERP Definition and Solutions. [accessed 25 March 2015] Available fom: http://www.cio.com/article/2439502/enterprise-resource-planning/erp-definition-and-solutions.html#improve

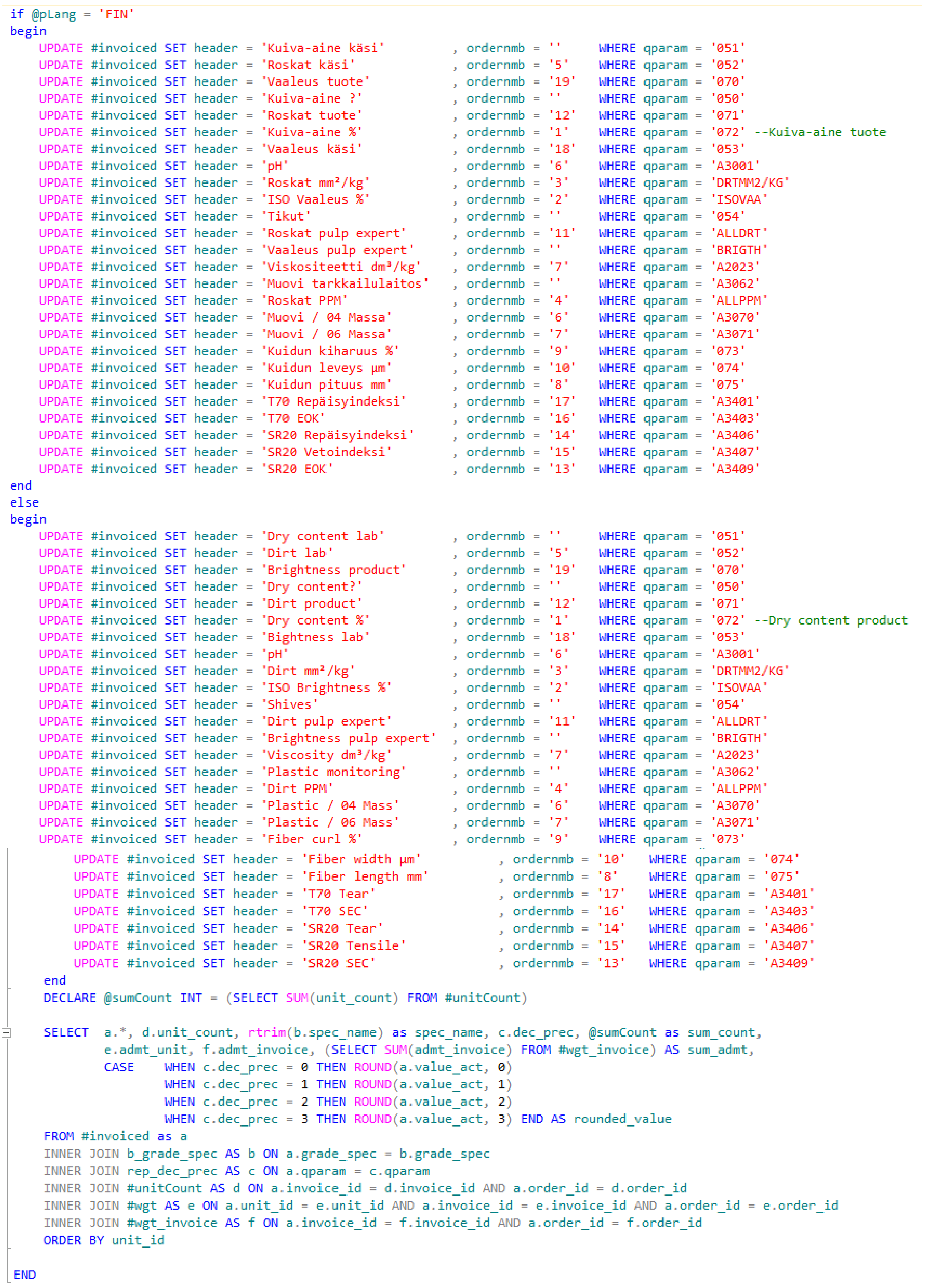
WOOD, Jeanne. 2014. OptiVision Proposal Template v1.0. Honeywell Oy.

Appendix 1: Example of stored procedure



```
ALTER PROCEDURE [dbo].[rap_invoiced_unit_102](
@sDate date,
@eDate date,
@pMachine varchar(2000),
@pGrade varchar(2000),
@pQprt varchar(2000),
@pConsignee varchar(50),
@pOrder varchar(2000),
@pUnitizing varchar(2000),
@pLang varchar(50),
@pInvoice varchar(2000)
)
AS
BEGIN
    CREATE TABLE #unitCount(
    unit_count int,
    invoice_id varchar(50),
    order_id varchar(50)
    )

    CREATE TABLE #wgt(
    admt_unit float,
    unit_id varchar(50),
    invoice_id varchar(50),
    order_id varchar(50)
    )

    CREATE TABLE #wgt_invoice(
    admt_invoice float,
    invoice_id varchar(50),
    order_id varchar(50)
    )

    CREATE TABLE #temp(
    unit_id varchar(50)
    , grade_spec varchar(50)
    , order_id varchar(50)
    , warehouse_id varchar(50)
    , machine_id varchar(50)
    , dte_scaled date
    , wgt_airdry float
    , qparam varchar(50)
    , value_act float
    , invoice_id varchar(50)
    , unitizing varchar(50)
    , machine_name varchar(50)
    , consignee_id varchar(50)
    , item_num smallint
    )
```

```
CREATE TABLE #invoiced(
unit_id varchar(50)
, grade_spec varchar(50)
, order_id varchar(50)
, warehouse_id varchar(50)
, machine_id varchar(50)
, dte_scaled date
, wgt_airdry float
, qparam varchar(50)
, value_act float
, invoice_id varchar(50)
, unitizing varchar(50)
, machine_name varchar(50)
, header varchar(50)
, ordernmb int
)

INSERT INTO #temp
SELECT      a.unit_id, grade_spec, a.order_id, warehouse_id, machine_id, dte_invoiced, wgt_airdry, qparam,
            value_act, a.invoice_id, unitizing, machine_name, a.consignee_id, a.item_num
FROM        dw_ov_delivery_view as a
            INNER JOIN b_consignee as b on a.consignee_id = b.consignee_id
WHERE       (a.invoice_id IN (select * from fn_MVParam(@pInvoice, ','))) AND
            (a.order_id IN (select * from fn_MVParam(@pOrder, ',')))

INSERT INTO #unitCount
    SELECT DISTINCT COUNT(unit_id), invoice_id, order_id
    FROM            #temp
    WHERE qparam = 'ISOVAA'
    GROUP BY invoice_id, order_id

INSERT INTO #wgt
    SELECT  SUM(wgt_airdry), unit_id, invoice_id, order_id
    FROM    #temp
    WHERE   qparam = 'ISOVAA'
    GROUP BY unit_id, invoice_id, order_id

INSERT INTO #wgt_invoice
    SELECT  SUM(wgt_airdry), invoice_id, order_id
    FROM    #temp
    WHERE   qparam = 'ISOVAA'
    GROUP BY invoice_id, order_id

INSERT INTO #invoiced
SELECT      unit_id, grade_spec, order_id, warehouse_id, machine_id, dte_scaled, wgt_airdry, qparam, value_act, invoice_id, unitizing, machine_name, '', ''
FROM        #temp
WHERE       (machine_id IN (select * from fn_MVParam(@pMachine, ','))) AND
            (grade_spec IN (select * from fn_MVParam(@pGrade, ','))) AND
            (qparam IN (select * from fn_MVParam(@pQprt, ','))) AND
            (unitizing IN (select * from fn_MVParam(@pUnitizing, ',')))
```



```sql
if @pLang = 'FIN'
begin
    UPDATE #invoiced SET header = 'Kuiva-aine käsi'          , ordernmb = ''    WHERE qparam = '051'
    UPDATE #invoiced SET header = 'Roskat käsi'              , ordernmb = '5'   WHERE qparam = '052'
    UPDATE #invoiced SET header = 'Vaaleus tuote'            , ordernmb = '19'  WHERE qparam = '070'
    UPDATE #invoiced SET header = 'Kuiva-aine ?'             , ordernmb = ''    WHERE qparam = '050'
    UPDATE #invoiced SET header = 'Roskat tuote'             , ordernmb = '12'  WHERE qparam = '071'
    UPDATE #invoiced SET header = 'Kuiva-aine %'             , ordernmb = '1'   WHERE qparam = '072' --Kuiva-aine tuote
    UPDATE #invoiced SET header = 'Vaaleus käsi'             , ordernmb = '18'  WHERE qparam = '053'
    UPDATE #invoiced SET header = 'pH'                       , ordernmb = '6'   WHERE qparam = 'A3001'
    UPDATE #invoiced SET header = 'Roskat mm²/kg'            , ordernmb = '3'   WHERE qparam = 'DRTMM2/KG'
    UPDATE #invoiced SET header = 'ISO Vaaleus %'            , ordernmb = '2'   WHERE qparam = 'ISOVAA'
    UPDATE #invoiced SET header = 'Tikut'                    , ordernmb = ''    WHERE qparam = '054'
    UPDATE #invoiced SET header = 'Roskat pulp expert'       , ordernmb = '11'  WHERE qparam = 'ALLDRT'
    UPDATE #invoiced SET header = 'Vaaleus pulp expert'      , ordernmb = ''    WHERE qparam = 'BRIGTH'
    UPDATE #invoiced SET header = 'Viskositeetti dm³/kg'     , ordernmb = '7'   WHERE qparam = 'A2023'
    UPDATE #invoiced SET header = 'Muovi tarkkailulaitos'    , ordernmb = ''    WHERE qparam = 'A3062'
    UPDATE #invoiced SET header = 'Roskat PPM'               , ordernmb = '4'   WHERE qparam = 'ALLPPM'
    UPDATE #invoiced SET header = 'Muovi / 04 Massa'         , ordernmb = '6'   WHERE qparam = 'A3070'
    UPDATE #invoiced SET header = 'Muovi / 06 Massa'         , ordernmb = '7'   WHERE qparam = 'A3071'
    UPDATE #invoiced SET header = 'Kuidun kiharuus %'        , ordernmb = '9'   WHERE qparam = '073'
    UPDATE #invoiced SET header = 'Kuidun leveys µm'         , ordernmb = '10'  WHERE qparam = '074'
    UPDATE #invoiced SET header = 'Kuidun pituus mm'         , ordernmb = '8'   WHERE qparam = '075'
    UPDATE #invoiced SET header = 'T70 Repäisyindeksi'       , ordernmb = '17'  WHERE qparam = 'A3401'
    UPDATE #invoiced SET header = 'T70 EOK'                  , ordernmb = '16'  WHERE qparam = 'A3403'
    UPDATE #invoiced SET header = 'SR20 Repäisyindeksi'      , ordernmb = '14'  WHERE qparam = 'A3406'
    UPDATE #invoiced SET header = 'SR20 Vetoindeksi'         , ordernmb = '15'  WHERE qparam = 'A3407'
    UPDATE #invoiced SET header = 'SR20 EOK'                 , ordernmb = '13'  WHERE qparam = 'A3409'
end
else
begin
    UPDATE #invoiced SET header = 'Dry content lab'          , ordernmb = ''    WHERE qparam = '051'
    UPDATE #invoiced SET header = 'Dirt lab'                 , ordernmb = '5'   WHERE qparam = '052'
    UPDATE #invoiced SET header = 'Brightness product'       , ordernmb = '19'  WHERE qparam = '070'
    UPDATE #invoiced SET header = 'Dry content?'             , ordernmb = ''    WHERE qparam = '050'
    UPDATE #invoiced SET header = 'Dirt product'             , ordernmb = '12'  WHERE qparam = '071'
    UPDATE #invoiced SET header = 'Dry content %'            , ordernmb = '1'   WHERE qparam = '072' --Dry content product
    UPDATE #invoiced SET header = 'Bightness lab'            , ordernmb = '18'  WHERE qparam = '053'
    UPDATE #invoiced SET header = 'pH'                       , ordernmb = '6'   WHERE qparam = 'A3001'
    UPDATE #invoiced SET header = 'Dirt mm²/kg'              , ordernmb = '3'   WHERE qparam = 'DRTMM2/KG'
    UPDATE #invoiced SET header = 'ISO Brightness %'         , ordernmb = '2'   WHERE qparam = 'ISOVAA'
    UPDATE #invoiced SET header = 'Shives'                   , ordernmb = ''    WHERE qparam = '054'
    UPDATE #invoiced SET header = 'Dirt pulp expert'         , ordernmb = '11'  WHERE qparam = 'ALLDRT'
    UPDATE #invoiced SET header = 'Brightness pulp expert'   , ordernmb = ''    WHERE qparam = 'BRIGTH'
    UPDATE #invoiced SET header = 'Viscosity dm³/kg'         , ordernmb = '7'   WHERE qparam = 'A2023'
    UPDATE #invoiced SET header = 'Plastic monitoring'       , ordernmb = ''    WHERE qparam = 'A3062'
    UPDATE #invoiced SET header = 'Dirt PPM'                 , ordernmb = '4'   WHERE qparam = 'ALLPPM'
    UPDATE #invoiced SET header = 'Plastic / 04 Mass'        , ordernmb = '6'   WHERE qparam = 'A3070'
    UPDATE #invoiced SET header = 'Plastic / 06 Mass'        , ordernmb = '7'   WHERE qparam = 'A3071'
    UPDATE #invoiced SET header = 'Fiber curl %'             , ordernmb = '9'   WHERE qparam = '073'
        UPDATE #invoiced SET header = 'Fiber width µm'           , ordernmb = '10'  WHERE qparam = '074'
        UPDATE #invoiced SET header = 'Fiber length mm'          , ordernmb = '8'   WHERE qparam = '075'
        UPDATE #invoiced SET header = 'T70 Tear'                 , ordernmb = '17'  WHERE qparam = 'A3401'
        UPDATE #invoiced SET header = 'T70 SEC'                  , ordernmb = '16'  WHERE qparam = 'A3403'
        UPDATE #invoiced SET header = 'SR20 Tear'                , ordernmb = '14'  WHERE qparam = 'A3406'
        UPDATE #invoiced SET header = 'SR20 Tensile'             , ordernmb = '15'  WHERE qparam = 'A3407'
        UPDATE #invoiced SET header = 'SR20 SEC'                 , ordernmb = '13'  WHERE qparam = 'A3409'
    end
    DECLARE @sumCount INT = (SELECT SUM(unit_count) FROM #unitCount)

    SELECT  a.*, d.unit_count, rtrim(b.spec_name) as spec_name, c.dec_prec, @sumCount as sum_count,
            e.admt_unit, f.admt_invoice, (SELECT SUM(admt_invoice) FROM #wgt_invoice) AS sum_admt,
            CASE    WHEN c.dec_prec = 0 THEN ROUND(a.value_act, 0)
                    WHEN c.dec_prec = 1 THEN ROUND(a.value_act, 1)
                    WHEN c.dec_prec = 2 THEN ROUND(a.value_act, 2)
                    WHEN c.dec_prec = 3 THEN ROUND(a.value_act, 3) END AS rounded_value
    FROM #invoiced as a
    INNER JOIN b_grade_spec AS b ON a.grade_spec = b.grade_spec
    INNER JOIN rep_dec_prec AS c ON a.qparam = c.qparam
    INNER JOIN #unitCount AS d ON a.invoice_id = d.invoice_id AND a.order_id = d.order_id
    INNER JOIN #wgt AS e ON a.unit_id = e.unit_id AND a.invoice_id = e.invoice_id AND a.order_id = e.order_id
    INNER JOIN #wgt_invoice AS f ON a.invoice_id = f.invoice_id AND a.order_id = f.order_id
    ORDER BY unit_id

END
```