World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:3, No:1, 2009

DD Models for Reports Building

Ljerka Hrženjak-Šego, Željko Polić, and Zdravka Aljinović

***Abstract*—**In general, reports are a form of representing data in such way that user gets the information he needs. They can be built in various ways, from the simplest ("select from") to the most complex ones (results derived from different sources/tables with complex formulas applied). Furthermore, rules of calculations could be written as a program hard code or built in the database to be used by dynamic code. This paper will introduce two types of reports, defined in the DB structure. The main goal is to manage calculations in optimal way, keeping maintenance of reports as simple and smooth as possible.

***Keywords*—**Data Definition diagram, Server Model Diagram, system modelling, reports.

I. Introduction

THERE is a constant demand for various kinds of reports in every business application. They can be built in real time from up-to-date data, or stored as ready-to-use in a warehouse database in case of static and large data. Defining of such products can vary from very simple (e.g. aggregated as grouped by) to really complex compounds with different rules for various rows and/or columns of report. Such rules can be built in the database structure, instead of writing them into the hard code of program objects.

II. Basics of Terminology and Symbols

Using the appropriate modeler/diagrammer (The Server Modeler in this case), a Data Diagram (DD) can be created. The Server Modeler (SM) is a graphical tool for modeling logical database designs. The database objects within a schema and how they relate to one another are represented graphically on Server Model Diagrams (SMD), and the conventions used to model a database schema using it are shown in Fig. 1.

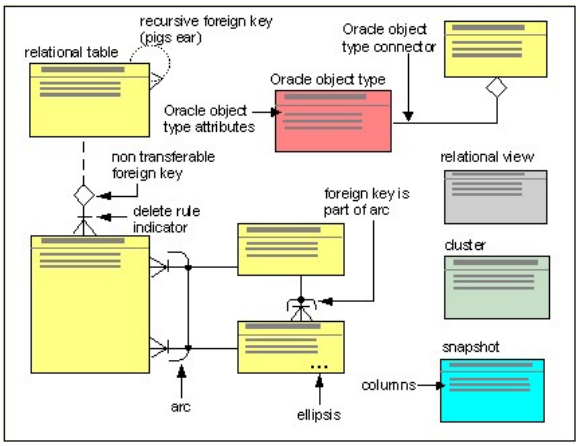

Fig. 1 Conventions used in SMD

On a Server Model Diagram, a relational table definition is represented by a yellow rectangle. A foreign key constraint is a type of referential integrity constraint for checking the integrity of data entered in a specified column or set of columns, thus a foreign key constraint ensures that no erroneous data can be entered in related tables. Foreign key constraints are represented using crowsfoot connectors and can be defined between two different tables or on a single table (recursive foreign key):

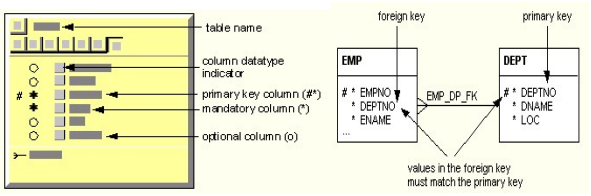

Fig. 2 Representation of Relational Tables and Foreign Keys in SMD

On a Server Model Diagram, secondary element types can be used to make model more transparent, and they are shown below.

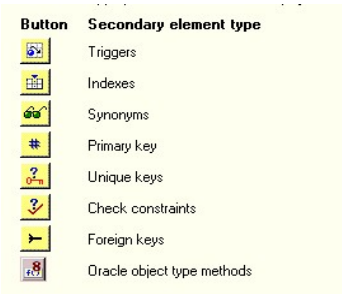

Fig. 3 Secondary element types used in SMD

Ljerka Hrzenjak-Sego is with the IT Department, Croatian National Bank, Zagreb, Croatia (e-mail: ljhrzenj@hnb.hr).
Zeljko Polic is with the IT Department, Croatian National Bank, Zagreb, Croatia (e-mail: zpolic@hnb.hr).
Zdravka Aljinović is with the Faculty of Economics, University of Split, Split, Croatia (e-mail: zdravka.aljinovic@efst.hr).

III. Reports

Defining the report means to give the format or mask of its appearance, and to determine the rules/formulas according to which the certain values in report's rows will be calculated. Each report in database must have at least code and title, and its items/rows must have some ID, name and format (and, if needed, sort order of rows) in which they will be printed:

CODE	ITEM	NAME	FORMATT	ORD_NO
SKB	A01	1. Pričuve banaka kod središnje banke	BI	1
SKB	A0101	1.1. Kunske pričuve kod središnje banke	R	2
SKB	A0102	1.2. Devizne pričuve kod središnje banke	R	3
SKB	A02	2. Inozemna aktiva	BI	4
SKB	A03	3. Potraživanja od središnje države	BI	5
SKB	A0301	3.1. Obveznice za blokiranu deviznu štednju građana	R	6
SKB	A0302	3.2. Velike obveznice	R	7
SKB	A0303	3.3. Ostala potraživanja	R	8
SKB	A0401	4.1. Potraživanja od lokalne države	R	9
SKB	A0402	4.2. Potraživanja od poduzeća	R	10
SKB	A0403	4.3. Potraživanja od stanovništva	R	11
SKB	A99	Ukupno (1+2+3+4)	BU	12

Fig. 4 Report definition example

Fig. 4 shows an example of such report definition stored in database, where format codes R, B, U and I mean regular, bold, underline and italic (including combinations) font style of printed row. In that way maintaining becomes very simple because style of rows (that means look of reports) can be changed only by changing data in definition table, and no line of reports code. Beside that, the amount(s) claimed have to be included in such definition. It can be done in various ways, depending on complexity of particular report.

A. Simple Reports

When each row of report consists only of name and one value/amount, having only basic functions (addition/ subtraction) in calculations, it is as simple as it can be. DD model for such case is shown on Fig. 5: the report consists of rows, each having only one column of amounts. Title of report and names of its rows, font style and order are defined in relational tables REP_DERIVATS and REP_DERIVAT_ITEMS. Basic structure for source amounts (e.g. transactions) is defined in REPS_LISTS with its columns, rows and fields/tags belonging (REP_COLS, REP_ROWS, REP_FIELDS). Source values for different dates are stored in REPS_SAVED and REP_VALUES. In REP_SHEMAS, only signs and tags for desired amounts are given. Once the derivated report is done, it can be saved for future in REP_DERS_DONE and REP_VALUES_DI. For example, it could be Profit&Loss report or Balance Sheet.

Nota bene: Source for reports in this paper are other already stored reports, since this work is continuation of previous one (References [3] and [4]), dealing with optimal storing of incoming reports. However, the same model for report definitions is applicable on any similar problem having transactions with amounts of any type or meaning as whole source for calculations.

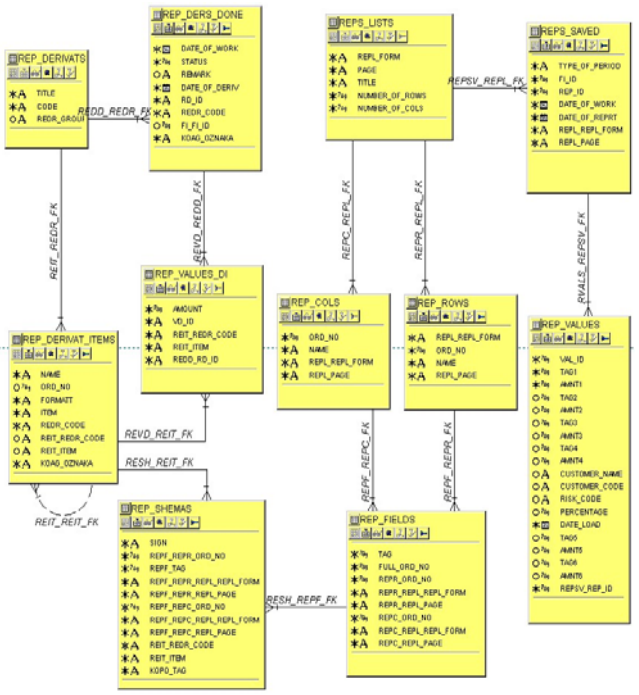

Fig. 5 Simple report example

Table REP_SHEMAS in this model includes much more columns than there were attributes of entity pertaining (Reference [4]). That is so because generation from entity to real table in relational database generates columns from each attribute and each relation-ship of an entity, receiving some of them by inheritance such are primary keys of subordinate objects (UID relationships).

CODE	ITEM	TAG	FORM	PAGE	SIGN	COL_ORD_NO
KBB	A010101	201	BS/GOD	4	-	2
KBB	A010101	101	BS/GOD	4	+	1
KBB	A01010201	106	BS/GOD	4	+	1
KBB	A01010201	206	BS/GOD	4	-	2
KBB	A01010202	107	BS/GOD	4	+	1
KBB	A01010202	207	BS/GOD	4	-	2
KBB	A01010203	108	BS/GOD	4	+	1
KBB	A01010203	208	BS/GOD	4	-	2
KBB	A010201	206	BS/GOD	4	+	2
KBB	P0101	103	BS/DEP	7A	+	1
KBB	P010201	208	BS/DEP	7A	-	2
KBB	P010201	108	BS/DEP	7A	+	1
KBB	P010202	209	BS/DEP	7A	-	2
KBB	P010202	109	BS/DEP	7A	+	1
KBB	P010301	110	BS/DEP	7A	+	1
KBB	P010302	111	BS/DEP	7A	+	1
NPD	1	001	NP/SD	1	+	1
NPD	1	132	NP/SD	4	+	1
NPD	11	002	NP/SD	1	+	1
NPD	111	003	NP/SD	1	+	1
NPD	1111	004	NP/SD	1	+	1
NPD	11111	005	NP/SD	1	+	1
NPD	111111	006	NP/SD	1	+	1
NPD	111112	007	NP/SD	1	+	1
NPD	11112	008	NP/SD	1	+	1
NPD	111121	009	NP/SD	1	+	1
NPD	111122	010	NP/SD	1	+	1
NPO	1	193	NP/OD	1	+	1
NPO	2	002	NP/OD	1	+	1
NPO	3	003	NP/OD	1	+	1
NPO	4	004	NP/OD	1	+	1
NPO	5	005	NP/OD	1	+	1

Fig. 6 Defining formulas for simple report

That is why table REP_SHEMAS in this example (Fig. 6) has columns *CODE* and *ITEM* from already mentioned so-

called definition tables REP_DERIVATS and REP_DERIVAT_ITEMS, as same as columns *FORM*, *PAGE*, *ORD_NO* and *TAG* from tables REPS_LISTS, REP_COLS and REP_FIELDS which keep basic structure of source items. Adding such constraints on the tables ensures that the referential integrity of the database is maintained. Now it is obvious why table REP_SHEMAS has so many columns beside *SIGN*. With structure like this it is easy to find source amounts claimed and get the final results adding or subtracting values according to formula, i.e. sign. Just entering the right codes and sign for source value in this table can easily do any change of calculations. There is no need to generate the new code in such case, because these are dynamically builded calculations.

B. Complex Reports

Basic structure and source values are the same as before, but in this report (Fig. 7) up to five columns shoud be calculated, and formulas for each of them are much more complex, including higher matemathical operations. Rules for each row and column are stored in REP_FORMULAS in such way (Fig. 8) that program can execute all calculations (defined for each row in COLn, n=1, 2, 3, 4, 5) and get the results, forming them in a desired way to produce the report. For example, it could be Effective Interest Rate report.

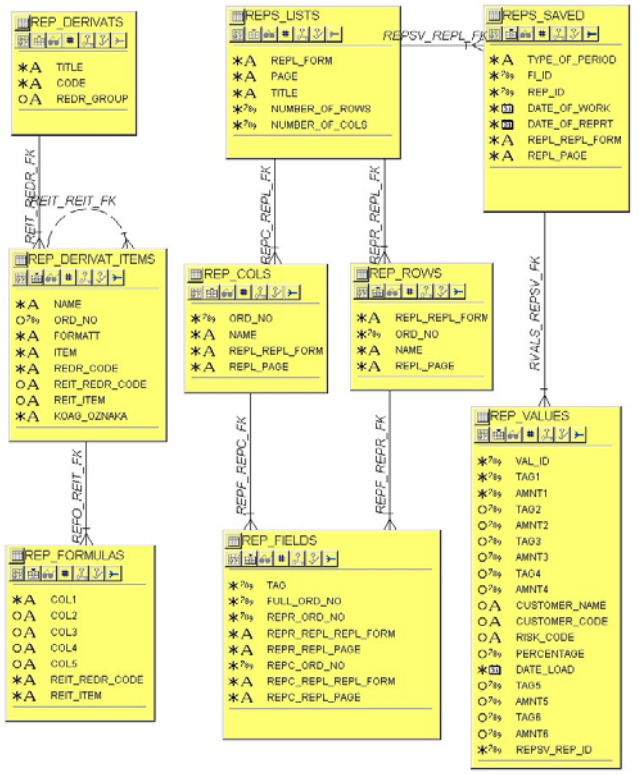

Fig. 7 Complex report example

Dealing with more demands in getting results, we had to change not only the way to choose the source values (reaching them by tags was quite clumsy and complicated). We had to involve more columns and to define new, complex formulas for these calculations. For example, one of the expressions was: divide sum of two multiplications (amount * percent) with another sum, by definition, with variations depending on report's row. In our database it was written like this:
(KS(11).PR11+KS(12).PR12)/abs(sign(KS(10).IZ1)-1+KS(10).IZ1)
Nota bene: An expression
"*abs(sign*(KS(10).IZ1)-*1*+KS(10).IZ1)" was involved to avoid problems with dividing by zero.

CODE	ITEM	COL1	COL2	COL3
KSR	1.1	KS(2).IZ1	KS(2).PR11/abs(sign(KS(2).IZ1)-1+KS(2).IZ1)	(KS(4).PR11+KS(6).PR12+KS(7).PR12+KS(8).PR12+KS(9).PR11+KS(11).PR11+KS
KSR	1.1.1	KS(3).IZ1	KS(3).PR11/abs(sign(KS(3).IZ1)-1+KS(3).IZ1)	(KS(4).PR11+KS(6).PR12+KS(7).PR12+KS(8).PR12+KS(9).PR11+KS(11).PR11+KS
KSR	1.1.1.1	KS(4).IZ1	KS(4).PR11/abs(sign(KS(4).IZ1)-1+KS(4).IZ1)	
KSR	1.1.1.2	KS(5).IZ1	KS(5).PR11/abs(sign(KS(5).IZ1)-1+KS(5).IZ1)	(KS(6).PR12+KS(7).PR12+KS(8).PR12+KS(9).PR11)/abs(sign(KS(5).IZ1)-1+KS(5)
KSR	1.1.1.3	KS(10).IZ1	KS(10).PR11/abs(sign(KS(10).IZ1)-1+KS(10).IZ1)	(KS(11).PR11+KS(12).PR12)/abs(sign(KS(10).IZ1)-1+KS(10).IZ1)
KSR	1.1.2	KS(13).IZ1	KS(13).PR11/abs(sign(KS(13).IZ1)-1+KS(13).IZ1)	KS(13).PR12/abs(sign(KS(13).IZ1)-1+KS(13).IZ1)
KSR	1.1.2.1	KS(14).IZ1	KS(14).PR11/abs(sign(KS(14).IZ1)-1+KS(14).IZ1)	KS(14).PR12/abs(sign(KS(14).IZ1)-1+KS(14).IZ1)
KSR	1.1.2.2	KS(17).IZ1	KS(17).PR11/abs(sign(KS(17).IZ1)-1+KS(17).IZ1)	KS(17).PR12/abs(sign(KS(17).IZ1)-1+KS(17).IZ1)
KSR	2.1	KS(36).IZ1	KS(36).PR21/abs(sign(KS(36).IZ2)-1+KS(36).IZ2)	
KSR	2.2	KS(54).IZ1	KS(54).PR21/abs(sign(KS(54).IZ2)-1+KS(54).IZ2)	
KSR	3.1	KS(70).IZ1	KS(70).PR11/abs(sign(KS(70).IZ1)-1+KS(70).IZ1)	KS(70).PR12/abs(sign(KS(70).IZ1)-1+KS(70).IZ1)
KSR	3.1.1	KS(71).IZ1	KS(71).PR11/abs(sign(KS(71).IZ1)-1+KS(71).IZ1)	KS(71).PR12/abs(sign(KS(71).IZ1)-1+KS(71).IZ1)
KSR	3.1.2	KS(92).IZ1	KS(92).PR11/abs(sign(KS(92).IZ1)-1+KS(92).IZ1)	KS(92).PR12/abs(sign(KS(92).IZ1)-1+KS(92).IZ1)
KSR	4.1	KS(107).IZ1	KS(107).PR21/abs(sign(KS(107).IZ2)-1+KS(107).IZ2)	
KS	1.	KS(1).IZ1	(KS(2).PR11+KS(20).PR11)/abs(sign(KS(1).IZ1)-1+KS(1).IZ1)	(KS(4).PR11+KS(6).PR12+KS(7).PR12+KS(8).PR12+KS(9).PR11+KS(11).PR11+KS
KS	1.1.1	KS(3).IZ1	(KS(4).PR11+KS(5).PR11+KS(10).PR11)/abs(sign(KS(3).IZ1)-1+KS(3)	(KS(4).PR11+KS(6).PR12+KS(7).PR12+KS(8).PR12+KS(9).PR11+KS(11).PR11+KS
KS	1.1.1.1	KS(4).IZ1	KS(4).PR11/abs(sign(KS(4).IZ1)-1+KS(4).IZ1)	

Fig. 8 Defining formulas for complex report

Fig. 8 shows an excerpt from this new formula table. Our customers were very reluctant in defining the very last version of calculations, so we have had a lot of simulations with many tries and fails before rules for getting results were finally established. It would be more than painful to change program code every time, for each simulation. In this way, we only had to change the formula in table row and repeat the execution of report by running the program with dynamic code again.

C. Complex Reports with Special Formulas

Basic structure and source values are again the same as before, but using the same report model (Fig. 7) up to two columns should be calculated as the numerator and denominator of a composed fraction. Formulas for each of them are much more complex here, including higher mathematical operations and special functions or procedures.

CODE	ITEM	COL1	COL2
29	65	100*#RDG(1096)	#prosj('BS-2',28,to_number(NULL),'IZN1','BS',24)
49	37	100*#PIV(3004)	fn_Anual('1',#v_Date)*#PIV(3004)+#RS(7164)
55	34	100*fn_PuniDevPoz (1,add_months(#v_Date,-3)+1,#v_Date,#v_MbrBank)	agr_kovr ('IZN1',39,'JK2',2,#v_Date,#v_Oznakalzvj,'1','0')

Fig. 9 Defining special formulas for complex report

Expressions in Fig. 9 contain some business rules, such as special kind of average value and anualization in Profit and Loss Report, and some aggregate values. The goal value is type of percentage. Every calculated amount deals with one reporting institution and that is why the numerator and denominator are in separate columns of a table. When report for whole system (or for a group of institutions) has to be

done, it would be wrong to make simple sum of percentages. Calculations of averages and similar functions have to be done separately for both numerators and denominators, giving the correct percentage value (from fraction) for the group of institutions. The same method can be applied for compound calculations in BI (Business Intelligence, for Management) with some constraints and conditions, such as 'top ten institutions according to asset/liabilities in balance sheet' or 'the middle peer group according to credit/debt' or 'the one(s) that have capital adequacy lower/higher than...' etc.

IV. Practical Experiences

It was not easy to make all the inputs necessary for accurate database, but that was wise investment: huge demands and efforts in time and human resources in the beginning for such a great benefits afterwards. The first model has been in use for more than ten years and the second one for more than seven years. That is so because there have been no need for complex reports in the first years of applications (for supervision and statistical reports), till seven years ago, when we started with effective interest rate reports. Otherwise, we would use only second model because it covers all types of our reporting system. Nevertheless, we keep on using the simple model because it works excellent and there was no need to change it by adaptation to a new one. These models do not cover all possible definitions of reports, yet they could be applied to many types of a kind, simply by changing and adjusting the main parameters. Using this way of defining report rules, managing became simpler, allowing quick and accurate maintenance within the application. As far, these two models cover all our needs for building reports of any level of complexness. New reports in our applications do not ask for change program code – only appends in form of new rows in relational database tables. Benefits of both models were proven: the goals are achieved demanding as low efforts and time as possible to maintain all changes through optimal use of hardware, software and human resources.

Applications were developed on Oracle 8.1.6 and Oracle 9. database, using CASE tools Oracle Designer R6.0 and Oracle Developer 6.0.

References

[1] Billings, C. &M. Rapid Development with Oracle CASE. Addison-Wesley Publishing Company, England, 1993.
[2] Hrženjak-Šego, Lj. and Šego, B. Two ER Models for Optimal Storing Of Reports. Proceedings of the 5th. International Symposium on Operational Research SOR'99, 1999 Sep 30.- Oct 2., Preddvor, Slovenia, pp. 179-183.
[3] Hrženjak-Šego, Lj. and Polić, Ž. ER Models For Producing Reports, 25th International Conference Information Technology Interfaces ITI'03, 2003 June 16-19, Cavtat / Dubrovnik, Croatia, Poster Abstracts, pp. 17-18.
[4] Hrženjak-Šego, Lj., Polić, Ž. and Šego, B. ER Models For Optimal Building Of Reports, Proceedings of the 7th. International Symposium on Operational Research (SOR'03), Podčetrtek, Slovenia, September 24-26, 2003, pp. 223-228.
[5] ***. Help Topics of Oracle Design Editor V6.0.3.10.0 Oracle Corporation.