An approach to information export from database for complex master-detail table hierarchies into a single flat table form

Alexei Hmelnov[1], **Gennadiy Ruzhnikov**[1], **Tianjiao Li**[2], **Huan Xu**[3]

[1]Matrosov Institute for System Dynamics and Control Theory of Siberian Branch of Russian Academy of Sciences, 134 Lermontov st. Irkutsk, Russia

[2]“the Belt and Road” Institute for Information Technology, Hangzhou Dianzi University, No.115, Wenyi Road, Xihu District, Hangzhou, China

[3]School of Automation (Artificial Intelligence), Hangzhou Dianzi University, No.1158, Number Two Street, Jianggan District, Hangzhou, China

E-mail: `hmelnov@icc.ru`

Abstract. When developing a database client application it is usually required to implement a capability to export information from database tables into some simple data-exchange representations of tables like the CSV (Comma-Separated Values) format. It is very straightforward to implement the export for a single database table. But some information about the records of a master table may be represented by the records from, sometimes, several detail tables, so the resulting CSV table would be incomplete without this information. We develop AIS (automated information systems) using declarative specifications of database applications (SDA). The AIS'es are implemented using general algorithms, which are directed by the specifications. In this article we'll consider the approach to generation of flat tables from the master-detail groups of tables, which allows users to represent compactly the data from a hierarchy of tables related by the master-detail relationships and to select conveniently which kind of information to include into the resulting table.

1. Introduction

We consider the approach to development of AIS (automated information system) using declarative specifications of database applications (SDA). Automated information systems are designed to accomplish specific information-handling operations [1]. Considerable part of AIS‘es uses relational database management systems (DBMS) for storing and processing the information they collect. Usually the database interaction is the central functionality of AIS.

In our work we consider database client applications (or, shorter, *database applications*) — the AIS'es that implement DBMS user interface. The database client application should allow their users to perform CRUD (create, read, update, delete), search and some other operations, for example, report generation.

The traditional way of database application development is based on the usage of some libraries like VCL (Visual Component Library) [2], MFC (Microsoft Foundation Classes) [3], FCL (Framework Class Library) [4] and so on. The choice of a particular library depends on the programming language used. Anyway, the process of application development will require a lot of similar steps, which should be performed for each table or entity type represented in the database. As a result the process becomes very time-consuming and tedious.

Some libraries like LINQ (Language Integrated Query) [5] are intended to simplify somehow the code we should write to process single table. But it doesn't removes the need to write the repetitive code for a lot of tables. The same effect gives the use of Object-Relational Mapping (ORM) libraries [6] — it will still be required to write a lot of code but this time in a more object-oriented way.

The approach, which potentially allows programmers to substantially decrease the development time, is the Model-Driven Architecture (MDA) [7]. It helps programmers to generate a baseline code of an application from the application model. The main disadvantage of generation unfinished code is the necessity to finish its development manually, because after manual editing it may be hard to regenerate the code again while saving the manual edits following the changes in the application specification.

The need for more effective methods of application development becomes widely accepted nowadays. To present an argument for the need in [8] they cite the estimate of Microsoft specialist, that *"Over 500 million new apps will be built during the next five years, which is more than all the apps built in the last 40 years."* To meet the need the low-code [8] and the no-code [9] approaches emerged. The low-code platforms work by automating generation of some repetitive code and require for developer to have some programming knowledge. The no-code platforms are advertised to allow non IT-professionals to develop applications using some kind of visual programming. The descriptions of the low-code and no-code platforms don't specify a particular check list of the information they need to be specified by the developers to build an application.

Our approach is based on the specifications of database applications (SDA) and can be considered as a low-code one when using the modern terms. The specifications of database applications contain all the information about database structure, which is required to build a typical AIS. The information is represented in its pure form, so the specifications are rather concise. The AIS'es are implemented using general algorithms, which are directed by the specifications. We have developed the algorithms for such tasks as: user interface generation, query building, report generation, GIS (Geographic Information Systems) interaction. Using the specifications of database applications and the algorithms the software system MetaDBApp/GeoARM was implemented. We have described the approach in more details in [10].

In this article we consider the sub-task of generation of a text or a spreadsheet file from the results of a user-defined query. This task shouldn't be mixed with that of report generation, because here we are not interested in a particular data formatting according to a report template, but in the export of all the selected data for further analysis outside the application, primarily in the form of a spreadsheet. To be able to formulate this kind of problems it is required to be in possession of the meta-information about the tables and their relations, like that we have in SDA. That's why no developers try to solve this problem when using the traditional approaches and it is hard to find any articles directly considering this kind of tasks.

The main contributions of the paper are:

(i) We suggest a human-readable representation of hierarchic master-detail data in the form of single grid.

(ii) We consider the capabilities of the SDA-controlled algorithms as exemplified by the algorithms of master-detail data export and interactive query building.

2. Research problem

While the task of export into CSV-like file format of a single table is very simple, it becomes non-trivial for a group of tables interconnected by the master-detail relationships. For example, if we have a table of `Persons` and suppose, that each person may have several contacts, then the `Contacts` should be stored in a separate detail table. The same may happen to the other attributes of the `Persons` entity, which may have several values, e.g. `Addresses`, `Documents`. If we'll export just the attributes of the `Persons` table itself from the database, we will miss all the multi-valued attributes of the entity.

There exist structured text file formats, which are well-suited for representation of hierarchical data, such as `XML` (eXtensible Markup Language), `JSON` (JavaScript Object Notation), `YAML` (Yet Another Markup Language). After implementation of the export into flat tables we can now easily add to our software the capability to export into the structured formats if it will be required. But the primary goal of the structured text formats is the information exchange between applications. And the formats are not intended for data visualization for humans. It doesn't matter that user can easily read any line of a text file, when the file has, say, 100000 lines. In the `YAML` representation the attributes of a single record will hardly fit into one screen, so it will be difficult to compare even two consecutive records.

3. The proposed approach

Another area, where we can see tabular representations of rather complex relations between objects is OLAP (On-Line Analytical Processing). The OLAP grids represent the contents of multidimensional cubes in 2D [11]. The user of the OLAP application may select various representations of the hyper-cube data by choosing the dimensions, corresponding to the top and left table headers and their order. The selected hierarchy of dimensions and hierarchies of their values (for the multilevel dimensions) are represented by the hierarchical table headers (Figure 1). The position in the OLAP hypercube, corresponding to an internal grid cell, can be unambiguously determined by the corresponding to its row and column left and top headers.

		Irkutsk				**Angarsk**			
		2019		**2020**		**2019**		**2020**	
		I	**II**	**I**	**II**	**I**	**II**	**I**	**II**
Summer	**In**	1.1	2.1	3.2	2.5	1.4	5.1	4.5	3.7
	Out	2.3	1.2	5.1	3.4	4.0	4.7	7.8	5.0
Winter	**In**	6.3	8.2	2.1	4.6	3.9	3.4	1.8	7.0
	Out	2.2	9.6	1.7	7.4	8.6	3.5	3.0	2.3

Figure 1. An example of OLAP grid.

The OLAP data representation can be very dense from the point of view of the number of values per screen. And the approach considered in this article is inspired by the OLAP representation. Indeed, we also have a tree — the tree of tables connected by the master-detail relationships. The leafs of the tree are the fields of the tables. To reflect this hierarchy we can use the OLAP-like top table headers.

But in contrast to the OLAP grids we will not use the left headers. Instead, we will represent the master-detail hierarchy of the table records by placing the 1st detail record after its master in the same row, and the next detail record in the next row with the master fields in the next row being empty.

Let us consider our approach in more details. Figure 2 presents an example of the master-detail tree for some illustrative tables. The names of the tables correspond to their levels in the hierarchy. The selected fields in the tables to be exported are shown in the round brackets here.

And the Figure 3 illustrates the suggested idea. It shows the grid representation of a fragment of the tables from the Figure 2, corresponding to couple of records of the main table. The fields have synthetic values here, which consist of the field name, the master record key and the detail

```
Main(A,B)
 |__Detail1(C,D,E)
 |   |__Subdetail1(F)
 |   |__Subdetail2(G,H)
 |__Detail2(I)
 |   |__Subdetail3(J,K)
 |__Detail3(L)
```

Figure 2. A three of detail tables

record ordinal number. The key feature of the data representation is that the cells in the columns, corresponding to different detail tables, even from the same resulting grid row, are mutually independent and are governed by their master records only.

Main											
		Detail1						**Detail2**			**Det.3**
					Subdet.1	**Subdetail2**			**Subdetail3**		
A	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**J**	**K**	**L**
A1	B1	C1.1	D1.1	E1.1	F1.1.1	G1.1.1	H1.1.1	I1.1	J1.1.1	K1.1.1	L1.1
					F1.1.2	G1.1.2	H1.1.2		J1.1.2	K1.1.2	L1.2
					F1.1.3			I1.2	J1.2.1	K1.2.1	L1.3
		C1.2	D1.2	E1.2	F1.2.1	G1.2.1	H1.2.1		J1.2.2	K1.2.2	
						G1.2.2	H1.2.2	I1.3	J1.3.1	K1.3.1	
A2	B2	C2.1	D2.1	E2.1	F2.1.1	G2.1.1	H2.1.1	I2.1	J2.1.1	K2.1.1	L2.1
						G2.1.2	H2.1.2		J2.1.2	K2.1.2	L2.2
		C2.2	D2.2	E2.2	F2.2.1	G2.2.1	H2.2.1		J2.1.3	K2.1.3	
					F2.2.2			I2.2			

Figure 3. The suggested grid representation of the data from Figure 2.

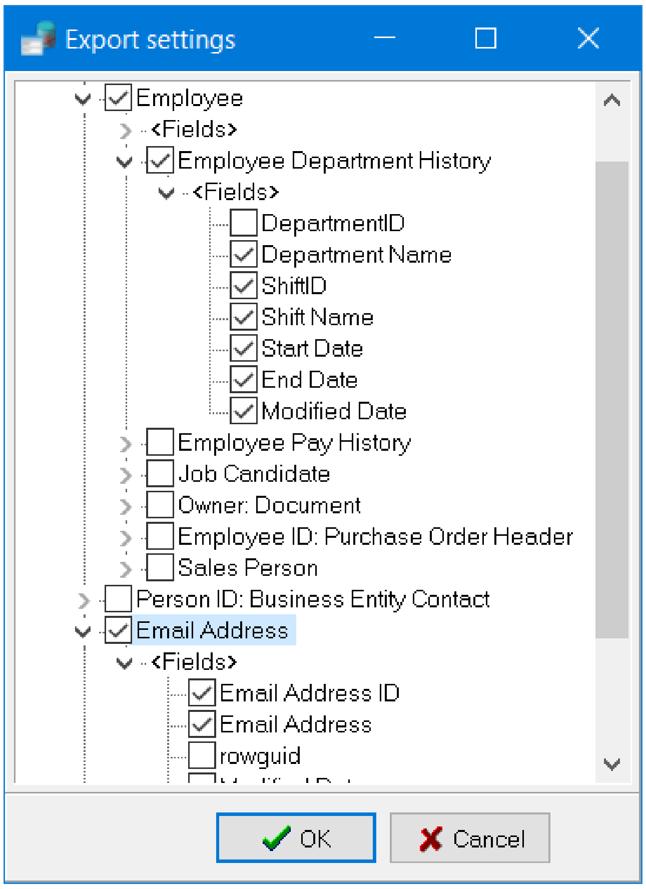
4. Applied solution

The specification of database application contains information about the database table fields and the master-detail links between the tables. The availability of this information makes it possible to create universal export setup dialogs, which allow users to control the data export. Let us consider the dialogs that we use here in more details.

4.1. The export setup dialog

The export setup dialog allows users to select the tables to be exported, their records and fields (Figure 4). The dialog represents the hierarchy of details of the current table using the TreeView nodes. We use the information about the master-detail links from SDA to create the detail nodes. We build the tree dynamically, so the dialog will work correctly even in the presence of loops in the graph of master-detail relations (for example, when a table has a field named, say `id_parent`, with the foreign key pointing to the parent record in the same table).

The first sub-node of each table node represents the list of the table fields. Using the check-marks of the tree nodes user can select the tables and fields to be exported. And to select the records to be exported we use the query builder dialogs, which can be called for any detail table node.



```
v [✓] Employee
    > <Fields>
    v [✓] Employee Department History
        v <Fields>
            [ ] DepartmentID
            [✓] Department Name
            [✓] ShiftID
            [✓] Shift Name
            [✓] Start Date
            [✓] End Date
            [✓] Modified Date
    > [ ] Employee Pay History
    > [ ] Job Candidate
    > [ ] Owner: Document
    > [ ] Employee ID: Purchase Order Header
    > [ ] Sales Person
> [ ] Person ID: Business Entity Contact
v [✓] Email Address
    v <Fields>
        [✓] Email Address ID
        [✓] Email Address
        [ ] rowguid
```

Figure 4. The hierarchic data export setup dialog

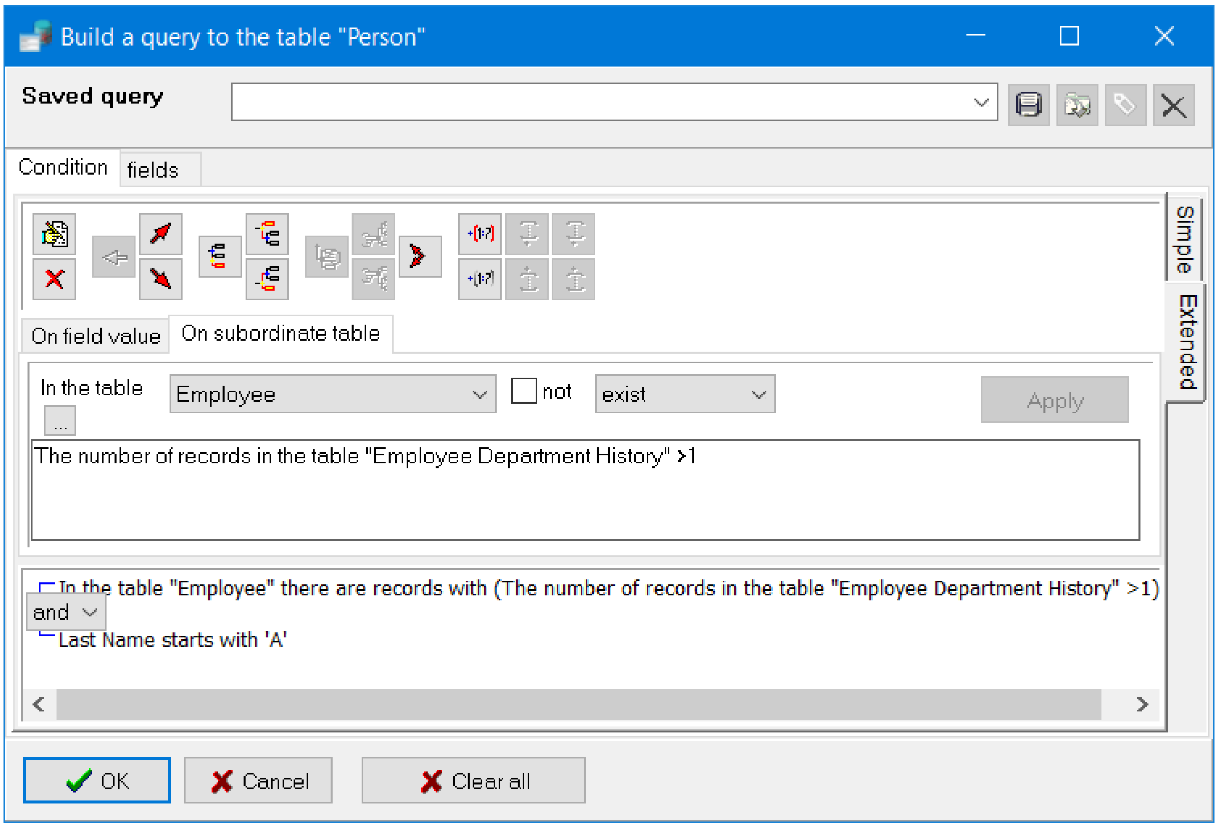
4.2. The query builder
To select the records of the tables to be exported we use another universal dialog controlled by specification — the query builder dialog (Figure 5).

First of all we may use the query builder to filter the records of the main table. Then we may filter the records of some detail tables by calling the query builder dialog from the export setup dialog.

The query builder dialog has two modes: simple and advanced. In the simple mode it allows user to fill some positions in the list of conditions on the values of the table fields and the resulting condition will be the conjunction of the filled conditions from the list. And the advanced mode allows user to construct an arbitrary logical formula by combining any number of conditions on the values of the fields and the conditions on the detail data-sets. The conditions on the detail data-sets are the requirements on the existence or on the number of their records satisfying some criteria. And the criteria on the records of the detail data-sets are constructed using the recursive calls of the same query builder dialog.

The query-builder dialogs get from SDA the information about the table fields, their types and roles in the application, and about the links to the detail tables. Using the information entered in the dialog we generate SQL queries to the tables (Figure 6 for an example).

4.3. Application implementation
To perform the export we use array of buffers of cell values for each resulting grid column. The main loop processes the records of the master data-set. For each record it clears the buffers, fills them using the recursive function `EnumDetails`, and flushes the resulting values to the resulting grid.



The number of records in the table "Employee Department History" >1

In the table "Employee" there are records with (The number of records in the table "Employee Department History" >1)
and
Last Name starts with 'A'

Figure 5. The Query builder dialog in the advanced mode, which sets a condition on a detail table `Employee` using a condition on the number of its subdetail records in the table `[Employee Department History]`

```
select T.BusinessEntityID,T0.ModifiedDate F_1,T.PersonType,T.NameStyle,T.Title,
  T.FirstName,T.MiddleName,T.LastName,T.Suffix,T.EmailPromotion,
  T.AdditionalContactInfo,T.Demographics,T.rowguid,T.ModifiedDate
from Person.[Person] T
left outer join Person.[BusinessEntity] T0 on (T.BusinessEntityID=T0.BusinessEntityID)
where (exists(select * from HumanResources.[Employee] S
  where (S.BusinessEntityID=T.BusinessEntityID)and
    ((select count(*) from HumanResources.[EmployeeDepartmentHistory] R
       where (R.BusinessEntityID=S.BusinessEntityID))>1)) and
    T.LastName like 'A%')
order by T.BusinessEntityID
```

Figure 6. The SQL query generated using the data entry in the query builder dialog from the Figure 5

The function `EnumDetails` has two parameters: `TableInfo`, which represents the information about the table to be exported, and l_0 — the buffer row, starting from which the table data will be placed (Figure 7). The `TableInfo` object has the following attributes:

- `Fields` — the list of fields selected for export;
- `Children` — the list of detail tables selected for export;

```
function $\textsc{EnumDetails}$( TableInfo, $l_0$ )
    for $i \in 0..$TableInfo.Fields.Count do        ▷ Show the field values of the current record
        Buffer[TableInfo.StartCol $+ i$][$l_0$] $\leftarrow$ TableInfo.Fields[$i$].AsString
    end for
    $l_{\max} \leftarrow l_0 + 1$                    ▷ The next free buffer line by now
    for Child $\in$ TableInfo.Children do           ▷ Process detail tables
        $l \leftarrow l_0$
        for each record of Child.Dataset do         ▷ Process the detail table records
            $l \leftarrow \textsc{EnumDetails}$(Child,$l$)    ▷ Show current record and its subdetails
        end for
        $l_{\max} \leftarrow \max(l_{\max}, l)$          ▷ Update the next free buffer line
    end for
    return $l_{\max}$                               ▷ return the number of the next free buffer line
end function
```

Figure 7. The output buffer filling function

- `StartCol` — the starting column of the table in the resulting grid;
- `Dataset` — already opened table or query, which contains the selected records.

The `TableInfo` data structures are constructed using the information entered in the export setup dialog. And all the detail data-sets are linked to their master data-sets by the master-detail relations, so that their records are automatically filtered after moving to the next master record to show only the detail records of the current master record. The SDA engine already had the operation for generation of the detail data-sets, which is required for construction of the data view/edit forms, so here we just reuse this capability.

5. Empirical research

In the Figure 8 we can see an output of the algorithm considered. The output file was generated in the Microsoft Excel `XLS` file format with some additional styling and headers in comparison with the `CSV` file format.

To demonstrate the algorithm we use the well-known `AdventureWorks` `MS SQL` sample database. Here we select the records from the `Person` table and from its detail tables `Employee` and `Email Address`. And the table `Employee` also has its own detail table `EmployeeDepartmentHistory`. The hierarchy of the tables is reflected by the 3-level header (the rows $3-5$) above the field names (the row 6).

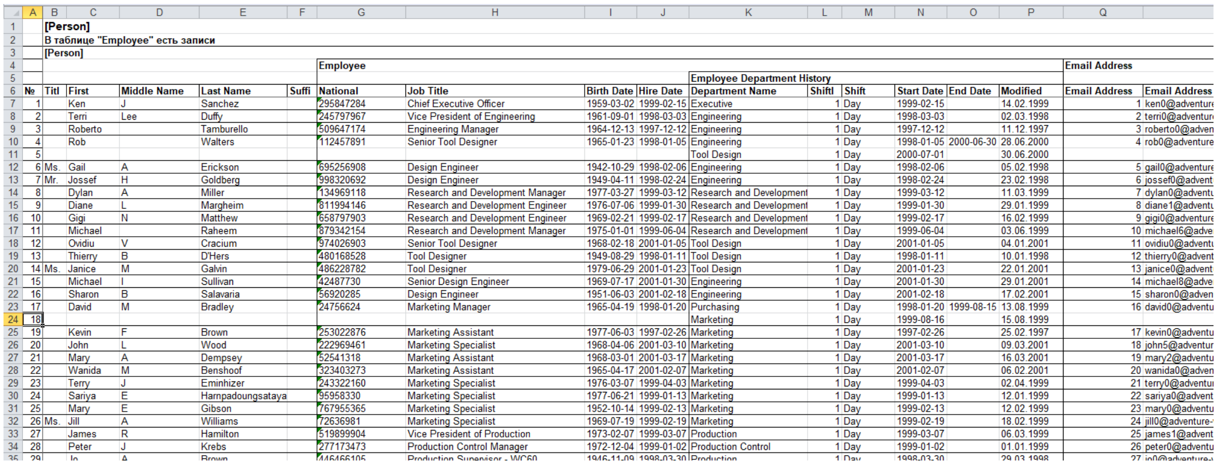
Using the query builder we have selected from the main table only the persons, which are employees, this condition is shown in human-readable form before the table header (in the row 2).

In the figure we can see two persons, who have two records in their job history (see the rows $10-11$ and $23-24$). So they have two sub-detail records in the `EmployeeDepartmentHistory` table and all the other cells in the 2nd row of the resulting table are empty.

And the Figure 9 shows another export example from the other master table `SalesPerson`, where the master records have much more detail records, than in the previous case.

Note, that when a master record has several details, the records of the two distinct details from the same resulting table row are not related to each other, they just have the same ordinal number in the corresponding lists of the detail records.

In fact the specification of database application for the `AdventureWorks` database, that we use here, was generated automatically by the database structure analysis algorithm. The algorithm allows developer to quickly create a starting database application specification, that can be



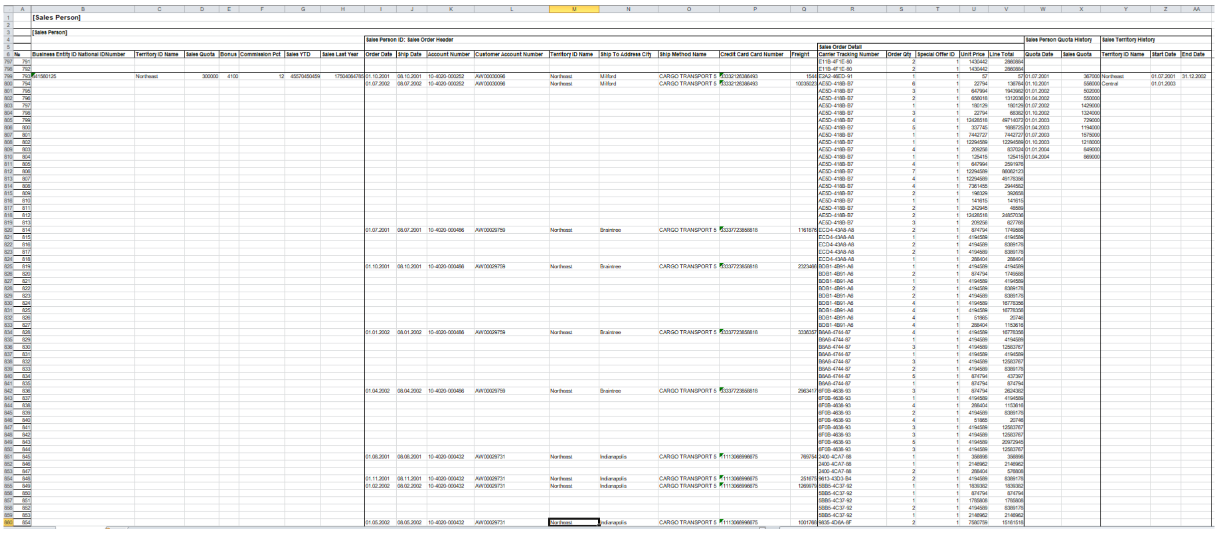
[Person]

В таблице "Employee" есть записи

[Person]

						Employee				**Employee Department History**						**Email Address**	
№	**Titl**	**First**	**Middle Name**	**Last Name**	**Suffi**	**National**	**Job Title**	**Birth Date**	**Hire Date**	**Department Name**	**Shiftl**	**Shift**	**Start Date**	**End Date**	**Modified**	**Email Address**	**Email Address**
1		Ken	J	Sanchez		295847284	Chief Executive Officer	1959-03-02	1999-02-15	Executive	1	Day	1999-02-15		14.02.1999	1	ken0@adventure
2		Terri	Lee	Duffy		245797967	Vice President of Engineering	1961-09-01	1998-03-03	Engineering	1	Day	1998-03-03		02.03.1998	2	terri0@adventur
3		Roberto		Tamburello		509647174	Engineering Manager	1964-12-13	1997-12-12	Engineering	1	Day	1997-12-12		11.12.1997	3	roberto0@adven
4		Rob		Walters		112457891	Senior Tool Designer	1965-01-23	1998-01-05	Engineering	1	Day	1998-01-05	2000-06-30	28.06.2000	4	rob0@adventure
5										Tool Design	1	Day	2000-07-01		30.06.2000		
6	Ms.	Gail	A	Erickson		695256908	Design Engineer	1942-10-29	1998-02-06	Engineering	1	Day	1998-02-06		05.02.1998	5	gail0@adventure
7	Mr.	Jossef	H	Goldberg		998320692	Design Engineer	1949-04-11	1998-02-24	Engineering	1	Day	1998-02-24		23.02.1998	6	jossef0@advent
8		Dylan	A	Miller		134969118	Research and Development Manager	1977-03-27	1999-03-12	Research and Development	1	Day	1999-03-12		11.03.1999	7	dylan0@adventu
9		Diane	L	Margheim		811994146	Research and Development Engineer	1976-07-06	1999-01-30	Research and Development	1	Day	1999-01-30		29.01.1999	8	diane1@adventu
10		Gigi	N	Matthew		658797903	Research and Development Engineer	1969-02-21	1999-02-17	Research and Development	1	Day	1999-02-17		16.02.1999	9	gigi0@adventure
11		Michael		Raheem		879342154	Research and Development Manager	1975-01-01	1999-06-04	Research and Development	1	Day	1999-06-04		03.06.1999	10	michael6@adve
12		Ovidiu	V	Cracium		974026903	Senior Tool Designer	1968-02-18	2001-01-05	Tool Design	1	Day	2001-01-05		04.01.2001	11	ovidiu0@adventu
13		Thierry	B	D'Hers		480168528	Tool Designer	1949-08-29	1998-01-11	Tool Design	1	Day	1998-01-11		10.01.1998	12	thierry0@advent
14	Ms.	Janice	M	Galvin		486228782	Tool Designer	1979-06-29	2001-01-23	Tool Design	1	Day	2001-01-23		22.01.2001	13	janice0@advent
15		Michael	I	Sullivan		42487730	Senior Design Engineer	1969-07-17	2001-01-30	Engineering	1	Day	2001-01-30		29.01.2001	14	michael8@adve
16		Sharon	B	Salavaria		56920285	Design Engineer	1951-06-03	2001-02-18	Engineering	1	Day	2001-02-18		17.02.2001	15	sharon0@adven
17		David	M	Bradley		24756624	Marketing Manager	1965-04-19	1998-01-20	Purchasing	1	Day	1998-01-20	1999-08-15	13.08.1999	16	david0@adventu
18										Marketing	1	Day	1999-08-16		15.08.1999		
19		Kevin	F	Brown		253022876	Marketing Assistant	1977-06-03	1997-02-26	Marketing	1	Day	1997-02-26		25.02.1997	17	kevin0@adventu
20		John	L	Wood		222969461	Marketing Specialist	1968-04-06	2001-03-10	Marketing	1	Day	2001-03-10		09.03.2001	18	john5@adventur
21		Mary	A	Dempsey		52541318	Marketing Assistant	1968-03-01	2001-03-17	Marketing	1	Day	2001-03-17		16.03.2001	19	mary2@adventu
22		Wanida	M	Benshoof		323403273	Marketing Assistant	1965-04-17	2001-02-07	Marketing	1	Day	2001-02-07		06.02.2001	20	wanida0@adven
23		Terry	J	Eminhizer		243322160	Marketing Specialist	1976-03-07	1999-04-03	Marketing	1	Day	1999-04-03		02.04.1999	21	terry0@adventu
24		Sariya	E	Harnpadoungsataya		95958330	Marketing Specialist	1977-06-21	1999-01-13	Marketing	1	Day	1999-01-13		12.01.1999	22	sariya0@advent
25		Mary	E	Gibson		767955365	Marketing Specialist	1952-10-14	1999-02-13	Marketing	1	Day	1999-02-13		12.02.1999	23	mary0@adventu
26	Ms.	Jill	A	Williams		72636981	Marketing Specialist	1969-07-19	1999-02-19	Marketing	1	Day	1999-02-19		18.02.1999	24	jill0@adventure-
27		James	R	Hamilton		519899904	Vice President of Production	1973-02-07	1999-03-07	Production	1	Day	1999-03-07		06.03.1999	25	james1@advent
28		Peter	J	Krebs		277173473	Production Control Manager	1972-12-04	1999-01-02	Production Control	1	Day	1999-01-02		01.01.1999	26	peter0@adventu
29		Jo	A	Brown		446466105	Production Supervisor - WC60	1946-11-09	1998-03-30	Production	1	Day	1998-03-30		29.03.1998	27	jo0@adventure-

Figure 8. An example of exported table, generated using the suggested approach for the main table `Person` from the `AdventureWorks2008` `MS SQL` sample database.



[Sales Person]

[Sales Person]

								Sales Person ID: Sales Order Header									**Sales Order Detail**					**Sales Person Quota History**		**Sales Territory History**		
№	**Business Entity ID National IDNumber**	**Territory ID Name**	**Sales Quota**	**Bonus**	**Commission Pct**	**Sales YTD**	**Sales Last Year**	**Order Date**	**Ship Date**	**Account Number**	**Customer Account Number**	**Territory ID Name**	**Ship To Address City**	**Ship Method Name**	**Credit Card Card Number**	**Freight**	**Carrier Tracking Number**	**Order Qty**	**Special Offer ID**	**Unit Price**	**Line Total**	**Quota Date**	**Sales Quota**	**Territory ID Name**	**Start Date**	**End Date**
791																	E11B-4F1E-80	2	1	1430442	2860884					
792																	E11B-4F1E-80	2	1	1430442	2860884					
793	841560125	Northeast	300000	4100	12	45570450459	17504064785	01.10.2001	08.10.2001	10-4020-000252	AW00030096	Northeast	Milford	CARGO TRANSPORT 5	33332126386493	1544	E2A2-46ED-91	1	1	57	57	01.07.2001	367000	Northeast	01.07.2001	31.12.2002
794								01.07.2002	08.07.2002	10-4020-000252	AW00030096	Northeast	Milford	CARGO TRANSPORT 5	33332126386493	10035023	AE5D-418B-B7	6	1	22794	136764	01.10.2001	556000	Central	01.01.2003	
795																	AE5D-418B-B7	3	1	647994	1943982	01.01.2002	502000			
796																	AE5D-418B-B7	2	1	656018	1312036	01.04.2002	550000			
797																	AE5D-418B-B7	1	1	180129	180129	01.07.2002	1429000			
798																	AE5D-418B-B7	3	1	22794	68382	01.10.2002	1324000			
799																	AE5D-418B-B7	4	1	12428518	49714072	01.01.2003	729000			
800																	AE5D-418B-B7	5	1	337745	1688725	01.04.2003	1194000			
801																	AE5D-418B-B7	1	1	7442727	7442727	01.07.2003	1575000			
802																	AE5D-418B-B7	1	1	12294589	12294589	01.10.2003	1218000			
803																	AE5D-418B-B7	4	1	209256	837024	01.01.2004	849000			
804																	AE5D-418B-B7	1	1	125415	125415	01.04.2004	869000			
805																	AE5D-418B-B7	4	1	647994	2591976					
806																	AE5D-418B-B7	7	1	12294589	86062123					
807																	AE5D-418B-B7	4	1	12294589	49178356					
808																	AE5D-418B-B7	4	1	7361455	2944582					
809																	AE5D-418B-B7	2	1	196329	392658					
810																	AE5D-418B-B7	1	1	141615	141615					
811																	AE5D-418B-B7	2	1	242945	48589					
812																	AE5D-418B-B7	2	1	12428518	24857036					
813																	AE5D-418B-B7	3	1	209256	627768					
814								01.07.2001	08.07.2001	10-4020-000486	AW00029759	Northeast	Braintree	CARGO TRANSPORT 5	33337723858818	1161876	ECD4-43A8-A8	2	1	874794	1749588					
815																	ECD4-43A8-A8	1	1	4194589	4194589					
816																	ECD4-43A8-A8	2	1	4194589	8389178					
817																	ECD4-43A8-A8	2	1	4194589	8389178					
818																	ECD4-43A8-A8	1	1	288404	288404					
819								01.10.2001	08.10.2001	10-4020-000486	AW00029759	Northeast	Braintree	CARGO TRANSPORT 5	33337723858818	2323466	BDB1-4B91-A6	1	1	4194589	4194589					
820																	BDB1-4B91-A6	2	1	874794	1749588					
821																	BDB1-4B91-A6	1	1	4194589	4194589					
822																	BDB1-4B91-A6	2	1	4194589	8389178					
823																	BDB1-4B91-A6	2	1	4194589	8389178					
824																	BDB1-4B91-A6	4	1	4194589	16778356					
825																	BDB1-4B91-A6	4	1	4194589	16778356					
826																	BDB1-4B91-A6	4	1	51865	20746					
827																	BDB1-4B91-A6	4	1	288404	1153616					
828								01.01.2002	08.01.2002	10-4020-000486	AW00029759	Northeast	Braintree	CARGO TRANSPORT 5	33337723858818	3336357	B8A8-4744-87	4	1	4194589	16778356					
829																	B8A8-4744-87	1	1	4194589	4194589					
830																	B8A8-4744-87	3	1	4194589	12583767					
831																	B8A8-4744-87	1	1	4194589	4194589					
832																	B8A8-4744-87	3	1	4194589	12583767					
833																	B8A8-4744-87	2	1	4194589	8389178					
834																	B8A8-4744-87	5	1	874794	437397					
835																	B8A8-4744-87	1	1	874794	874794					
836								01.04.2002	08.04.2002	10-4020-000486	AW00029759	Northeast	Braintree	CARGO TRANSPORT 5	33337723858818	2963417	6F0B-4638-93	3	1	874794	2624382					
837																	6F0B-4638-93	1	1	4194589	4194589					
838																	6F0B-4638-93	4	1	288404	1153616					
839																	6F0B-4638-93	2	1	4194589	8389178					
840																	6F0B-4638-93	4	1	51865	20746					
841																	6F0B-4638-93	3	1	4194589	12583767					
842																	6F0B-4638-93	3	1	4194589	12583767					
843																	6F0B-4638-93	5	1	4194589	20972945					
844																	6F0B-4638-93	3	1	4194589	12583767					
845								01.08.2001	08.08.2001	10-4020-000432	AW00029731	Northeast	Indianapolis	CARGO TRANSPORT 5	11113066996675	769754	2400-4CA7-88	1	1	356898	356898					
846																	2400-4CA7-88	1	1	2146962	2146962					
847																	2400-4CA7-88	2	1	288404	576808					
848								01.11.2001	08.11.2001	10-4020-000432	AW00029731	Northeast	Indianapolis	CARGO TRANSPORT 5	11113066996675	251675	9613-43D3-B4	2	1	4194589	8389178					
849								01.02.2002	08.02.2002	10-4020-000432	AW00029731	Northeast	Indianapolis	CARGO TRANSPORT 5	11113066996675	1269979	5BB5-4C37-92	1	1	1839382	1839382					
850																	5BB5-4C37-92	1	1	874794	874794					
851																	5BB5-4C37-92	1	1	1785808	1785808					
852																	5BB5-4C37-92	2	1	4194589	8389178					
853																	5BB5-4C37-92	1	1	2146962	2146962					
854								01.05.2002	08.05.2002	10-4020-000432	AW00029731	Northeast	Indianapolis	CARGO TRANSPORT 5	11113066996675	1001768	9835-4D6A-8F	2	1	7580759	15161518					

Figure 9. An example of exported table, generated using the suggested approach for the main table `SalesPerson` from the `AdventureWorks2008` `MS SQL` sample database.

then corrected manually, if it will be required. And because the `AdventureWorks` database is a showcase `MS SQL` database of high quality, it was not required here to correct the generated specification, and we used it as is.

6. Conclusion

We have proposed and implemented a method for compact representation of information from the group of associated by the master-detail relationships tables. The algorithms get the information about the relations between tables from the declarative specifications of database applications. The export parameters dialogue allows user to select the tables and their fields to export. It is also possible to select the records of the detail tables of interest using the interactive query builder, which is also controlled by specifications of database applications. The practical usage of the suggested approach to data export has shown, that even nontechnical people easily understand this kind of data representation.

Acknowledgments
The results were obtained within the framework of the State Assignment of the Ministry of Education and Science of the Russian Federation for the project "Methods and technologies of cloud-based service-oriented platform for collecting, storing and processing large volumes of multi-format interdisciplinary data and knowledge based upon the use of artificial intelligence, model-guided approach and machine learning". Some results were obtained using the facilities of the Centre of collective usage "Integrated information network of Irkutsk scientific educational complex".

References

[1] ATIS Telecom Glossary - American National Standard, Automated information system (AIS), `https://glossary.atis.org/glossary/automated-information-system-ais/`. Last accessed 12 Feb 2021
[2] VCL Overview, `http://docwiki.embarcadero.com/RADStudio/Rio/en/VCL`. Last accessed 12 Feb 2021
[3] MFC Desktop Applications, `https://docs.microsoft.com/en-us/cpp/mfc/mfc-desktop-applications?view=msvc-160`. Last accessed 12 Feb 2021
[4] .NET Class Library Overview, `https://docs.microsoft.com/en-us/dotnet/standard/class-library-overview`. Last accessed 12 Feb 2021
[5] Calvert C and Kulkarni D 2009 *Essential LINQ (1st. ed.)* Addison-Wesley Professional
[6] Kouraklis J 2019 *Introducing Delphi ORM: Object Relational Mapping Using TMS Aurelius* doi:10.1007/978-1-4842-5013-6
[7] Pastor O, Ruiz M and España S 2013 From Requirements to Code: A Full Model-Driven Development Perspective *Communications in Computer and Information Science* **303** 56-70 doi:10.1007/978-3-642-36177-7_4
[8] *Everything You Need to Know about Using Low-Code Platforms* ZDNet `https://www.techrepublic.com/resource-library/whitepapers/everything-you-need-to-know-about-using-low-code-platforms-free-pdf`
[9] Woo M 2020 The Rise of No/Low Code Software Development-No Experience Needed? *Engineering* Beijing, China **6(9)** 960–961 `https://doi.org/10.1016/j.eng.2020.07.007.007`
[10] Bychkov I V, Hmelnov A E, Fereferov E S, Rugnikov G M and Gachenko A S 2018 Methods and Tools for Automation of Development of Information Systems Using Specifications of Database Applications *In 3rd Russian-Pacific Conference on Computer Technology and Applications (RPC)*IEEE, Vladivostok 1-6 doi:10.1109/RPC.2018.8482170
[11] Schrader M, Vlamis D, Nader M, Claterbos C, Collins D, Conrad F and Campbell M 2010 *Oracle Essbase & Oracle OLAP: The Guide to Oracle's Multidimensional Solution* McGraw-Hill Companies, Inc