Design and Implementation of Flexible-Specified Retrieval in a Unified Framework for Management System of Diverse Information

Xiaodong Wang*, Yi Miao, Xiaohui Liu, Zhen Zhang, Ti Zhou, Yan Liu

Beijing Aerospace Control Center, Beijing, China
**Corresponding author*

Abstract: *We have designed and implemented a diversed-information management system for both staff and department information based on .NET framework, B/S structure, Sql Server database and Visual Studio Develop Environment of Microsoft Corporation are adopted. Many kinds of personal and department-related information tables are created in the Sql Server database which results to difficulties in uniform retrieve and analysis. In traditional information retrieval applications, designers generate sql sentence from several fixed inputs and show the results in a fixed view. To enable flexible-specified advanced query of all that sorts of information in a unified framework, we import extra tables for describing information table columns called 'AllTableInfo' and extra views for personal information integration. Besides, query strings can be generated and transferred to the background processing program when the user selects the info-fields to output in specified form and sequence and sets the where condition. Sub query strings can be added to fulfill all-around queries using all kinds of where condition permutation and combination. Once specifying the query instance, the query string can be chosen as memorized so that the user can choose it easily and only set a few where conditions instead of specifying the query instance allover again. Through our design, the query input view and query results view can be generated dynamically and flexibly in a unified framework according to user's choice of query need. What's more, our work make it possible for user to set the query need accurately covering all conditions. In this paper we will explain how all these are designed and implemented.*

Keywords: *information management system, database, flexible-specified query, advanced query, information retrieve*

1. Introduction

The information management system contains various information, including both employee- related and department-related which are described more detailed in the database structure chapter. All these information are expressed as one or several tables in the Sql Server database [1], which compose of different fields and values.

Since the system manages too many sorts of information tables which contain different fields, it is a problem for advanced, detailed and flexible-specified information retrieval including cross-table search. This is quite unlike the book, magazine and paper retrieving conditions in library which consist of same or similar fields [2]. In these traditional information retrieval applications, designers generate sql sentence from several fixed inputs and show the results in a fixed view. Besides, the user can't set the query conditions accurately as they need because designers only supply few input conditions and can't cover all conditions. In our work, it is explained how program and database are designed all together to satisfy the retrieval need. Through our design, the query input view and query results view can be generated dynamically and flexibly in a unified framework according to user's choice of query need. What's more, our work make it possible for user to set the query need accurately covering all conditions.

2. Underground Framework

The system is based on the .NET core developed and maintained by the Microsoft Corporation, which is a free, cross-platform, open source developer platform for building many different types of applications such as web, mobile, desktop, games, and IoT using multiple languages, editors, and libraries [3]. The core defines a base set of APIs that are common to all .NET implementations called .NET Standard.

ASP.NET is an open source web framework for building modern web apps and services with .NET which extends the .NET platform with tools and libraries specifically for building web apps. Our system is developed and deployed under the ASP.NET Framework in Windows operating system but can be easily migrated to other platforms such as Linux, macOS, and Docker by using .NET standard[4]. Dynamic pages are built using C#, HTML, CSS, JavaScript and MasterPage feature for consistent page view. C# code is executed on the server and the resulting HTML content is sent to the user with Razor. Code that executes client-side is written in JavaScript and ASP.NET integrates with JavaScript frameworks and pre-configured templates for single page app (SPA) frameworks like React and Angular. C# programming language is chosen to build our Browser-based web application. The code handling database is fulfilled together in a uniform scheme that opens database connection, makes SqlParameter, executes SqlCommand, disposes and close database connection using namespace 'System.Data. SqlClient' [5]. The Microsoft Sql Server database offers ability for database recovery using complete backup or increment backup [6].

Among .Net APIs for browsers, GridView class of namespace 'System.Web.UI.WebControls' is frequently used to display the values of a data source in a table where each column represents a field and each row represents a record. The GridView control enables us to select, sort, and edit these items. The GridView control supports the following features: Binding to data source controls, such as SqlDataSource; Built-in sort capabilities; Built-in update and delete capabilities; Built-in paging capabilities; Built-in row selection capabilities; Programmatic access to the GridView object model to dynamically set properties, handle events, and so on; Multiple key fields; Multiple data fields for the hyperlink columns; Customizable appearance through themes and styles. When the GridView control is bound to a data source control, the GridView control can take advantage of the data source control's capabilities and provide automatic sort, update, and delete functionality. The GridView control provides several events that we can program against. This enables us to run a custom routine whenever an event occurs. Figure 1 gives a brief overview of the running framework of the implemented system.

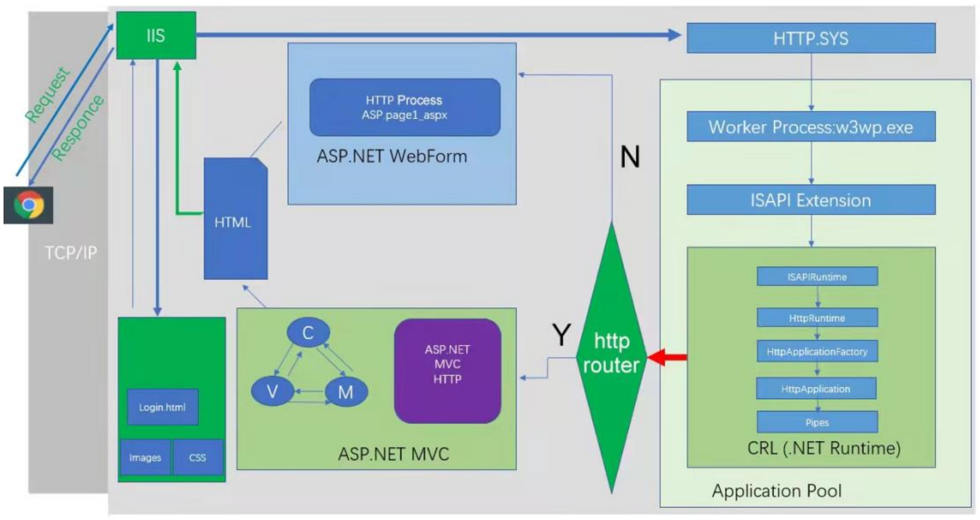


Figure 1: Underground framework of implemented B/S structured system

3. Class Design for Query Storage

In this chapter data and functions designed in several classes are explained. These classes are used for query storage, transformation and inverse transformation. Class 'selectedInfo', 'whereInfo' are designed to store selected 'select' part of sql, 'where' part of sql sentence. Class 'selectQuery' can contain multiple 'selectedInfo's and 'whereInfo's to store a complete sql query sentence. Besides, several 'selectQuery' can combine to form an all-around query of various condition, but 'selectedInfo's in the 'selectQuery' must be the same when 'whereInfo's are different. As in Table I, II and III, the left column of 'Data and Functions' describes what data and functions are included in the class. The right column of 'Explanations' describes the meaning and usage of the 'Data and Functions'.

Class 'selectedInfo' includes the select column and its output sequence and format if it's 'datetime' type. String formula tells whether a formula-computing will be carried out on this column. Bool 'setGroupBy' tells whether this column is included in the 'group by' part of sql sentence. A column can either has the 'formula' or the 'setGroupBy' set, because if an formula is used on some columns then other columns will be in the ‘group by’ part. Int ‘rowIndex’ can be used to locate the row number of selected column of table in the user’s GridView directly so that we can reshow the user-specified query in the GridView. All data can be transformed in order to form an easily-split string using function ‘string ToString()’. Inversely, All data can be set form a string of preset ‘infoStr’ using function ‘void setFromString(string infoStr)’;

Class ‘whereInfo’ includes the columns that are part of ‘where’ query conditions. Searching conditions of ‘equal’, ‘like’, ’min’ and ‘max’ are stored as ‘condtColumn’, in which ‘equal’ is used in accurate matching, ‘like’ is used in fuzzy matching, min is used in ’>=’ comparison, max is used in ‘<’ comparison. Int ‘rowIndex’, function ‘string ToString()’ and ‘void setFromString(string infoStr)’ has the same usage as in class ‘selectedInfo’.

Class ‘selectQuery’ contains the list of ‘selectedInfo’, ‘whereInfo’ and so on, through which the selected columns, output format and searching conditions can be known. A query can be expressed as one instance of class ‘selectQuery’, which is corresponding to a row storage of table SubQuery in database design chapter. When the query is complex-specified of condition, combination of several instances of class ‘selectQuery’ can be used, of which the responding SubQuery rows is added to form FullQuery in database design chapter.

Table 1: Class SelectedInfo.

Data and Functions	Explanations
string columnE	Name of selected column
string columnC	Chinese name of selected column
string datetimeFormat	Output format of datetime
string formula	Formula performed on the column
bool setGroupBy	Whether it’s a group by column
int seq	Output sequence in sheet
int rowIndex	Number of row in the showing GridView for quick indexing
string ToString()	Function: transform all data in the class instance to a easily-split string
void setFromString(string infoStr)	Function: set all data from string input of infoStr

Table 2: Class WhereInfo.

Data and Functions	Explanations
string columnE	Name of column that is in sql sentence’s where part
string condtColumn	Which of equal, like ,min or max is used
string condition	The query condition’s input value
int rowIndex	Number of row in the showing GridView for quick indexing
string ToString()	Function: transform all data in the class instance to a easily-split string
void setFromString(string infoStr)	Function: set all data from string input of infoStr

Table 3: Class SelectQuery.

Data and Functions	Explanations
string queryStr	Sql query sentence
string fromTb	Table that query is executed on
List<selectedInfo> selectedInfoList	List of selected column’s selectedInfo
string whereStrEng	Where part of sql sentence that can be added to include other conditions
string whereStrChi	Chinese explanation for what query condition is set
List<whereInfo> whereInfoList	List of selected column’s whereInfo
string ToString()	Function: transform all data in the class instance to a easily-split string
void setFromString(string infoStr)	Function: set all data from string input of infoStr
string selectedInfoListToString()	Function: transform selectedInfoList in the class instance to a easily-split string
void setSelectedInfoListFromString	
(string infoStr)	Function: set data selectedInfoList from string input of infoStr
string whereInfoListToString()	Function: transform whereInfoList in the class instance to a easily-split string

4. Database Design

The information management system adopts classical relational database using Sql Server. The information contained can generally classified as login account information, basic personal information, overtime work information, ask for leave information, fixed assets information, car information, cellphone information, duty information, equipment information, family members information, on holiday information, level update information, training plan information, documents transferring information, schedule information and uploaded-files information .

The information tables all have a primary key for retrieval. For those information table connected to a certain person, the person's unique ID and name will be included in the table. Except for the common tables describing personal or department-related information, three tables called SubQuery, FullQuery and AllTableInfo are designed for query and retrieval building whose columns are clearly shown in Table IV, Table V and Table VI. In addition, A dynamic view called 'AllPersonalInfoJoined' is introduced to join all the person-related tables using the person's unique ID so that different information types in different tables of a certain person can be chosen and retrieved all together at the same time. By using 'AllPersonalInfoJoined', cross-table search is fulfilled.

Table AllTableInfo is a describing table for all information tables in which all columns in different tables are described in a uniform structure including column name, corresponding Chinese name, data type and so on. When the user tries to perform a retrieval on a table or the 'AllPersonalInfoJoined' view, the columns of the table or view will be shown using 'tableDescribe', 'columnCName' and 'queryFormat' fields in the showing GridView. Besides, query need and conditions can be set in the showing Gridview as in Fig. 2. Column 'tableDescribe' and 'columnCName' is the Chinese representation of 'tableName' and 'columnEName'. Column 'columnType' is the data type. Column 'queryFormat' gives tips for the input value pattern of query. Column 'regularExp' is the regular expression of query input value that restrict it to be proper.

Table 'Subuery' and 'FullQuery' are used to store and restore user-specified query in various condition. 'FullQuery' is a supplement to 'SubQuery' and consists of one or several 'SubQuery's because it is not capable of covering all kinds of query condition settings in a single 'SubQuery'. Using 'FullQuery', we are able to combine '&& and ||' arbitrarily when setting 'where' conditions. In 'FullQuery', 'queryTitle' is an explanation for user to easily understand what the query has done; 'queryStr' is the sql language expression for the query; 'fromTb' tells us from which information table or view the results are select. In SubQuery, 'FullQueryID' tells us to which FullQuery the SubQuery is related; 'queryStr' and 'fromTb' has the same usage as in FullQuery; 'selectedInfoList' is a string transformed from the list of class 'selectedInfo' objects defined in the program that can be inverse transformed to list of class 'selectedInfo' objects by the program; 'whereStrEng' is the 'where part' in the sql language expression for recomposing in FullQuery; 'whereStrChi' is Chinese expression shown to user for easily understanding; 'whereInfoList' is a string transformed from the list of class 'whereInfo' objects defined in the program that can be inverse transformed to list of class 'whereInfo' objects by the program.

Table 4: Table FullQuery.

S/N	Column	Data Type	Constraint
1.	ID	int	not null primary key identity
2.	queryTitle	varchar	not null
3.	queryStr	varchar	not null
4.	fromTb	varchar	not null

Table 5: Table SubQuery.

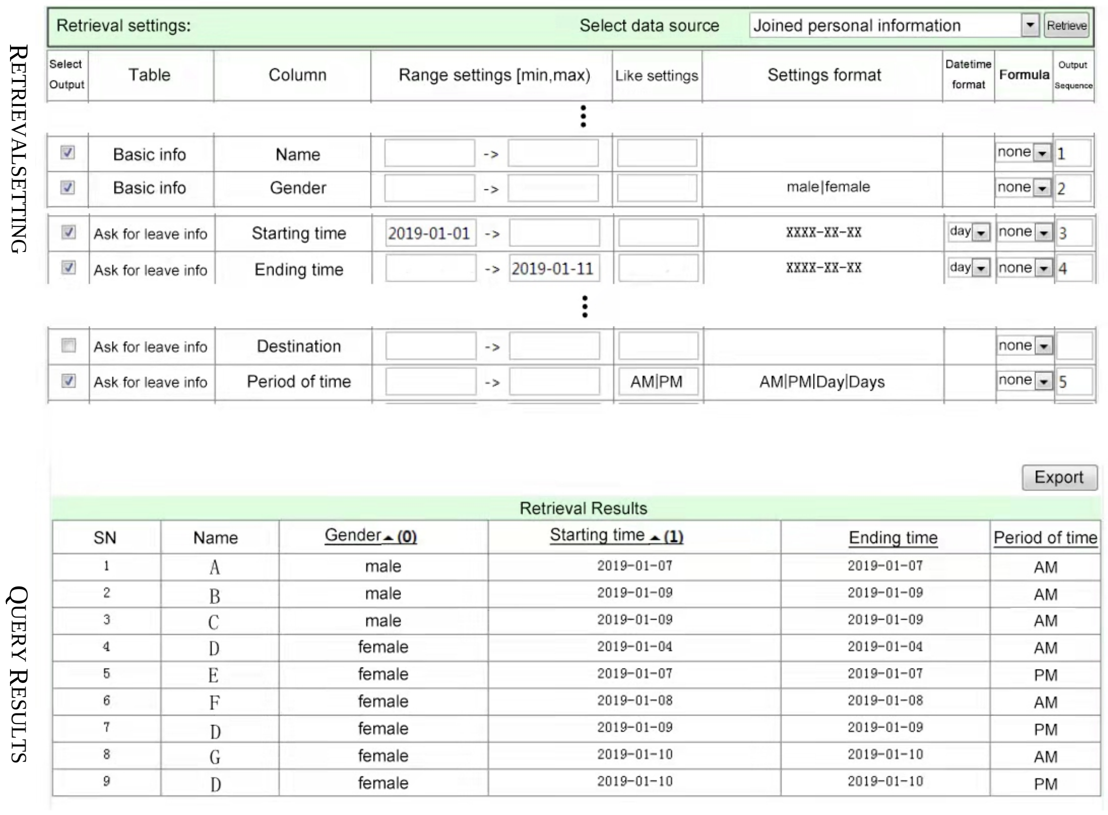
S/N	Column	Data Type	Constraint
1.	ID	int	not null primary key identity
2.	FullQueryID	int	not null
3.	queryStr	varchar	not null
4.	fromTb	varchar	not null
5.	selectedInfoList	varchar	not null
6.	whereStrEng	varchar	not null
7.	whereStrChi	varchar	not null
8.	whereInfoList	varchar	not null

Table 6: Table AllTableInfo.

S/N	Column	Data Type	Constraint
1.	ID	int	not null primary key identity
2.	tableName	varchar	not null
3.	tableDescribe	varchar	not null
4.	columnEName	varchar	not null
5.	columnCName	varchar	not null
6.	columnType	varchar	not null
10.	queryFormat	varchar	-
11.	regularExp	varchar	-

5. UI Design and Retrieval Implementation

As is discussed in Chapter II, GridView class is frequently used for information table fields display. As far as user-specified advanced retrieval is concerned, two GridViews are adopted among which one for retrieval setting and the other for result display.



Retrieval settings: Select data source Joined personal information

Select Output	Table	Column	Range settings [min,max]	Like settings	Settings format	Datetime format	Formula	Output Sequence
✓	Basic info	Name	->				none	1
✓	Basic info	Gender	->		male|female		none	2
✓	Ask for leave info	Starting time	2019-01-01 ->		XXXX-XX-XX	day	none	3
✓	Ask for leave info	Ending time	-> 2019-01-11		XXXX-XX-XX	day	none	4
☐	Ask for leave info	Destination	->				none	
✓	Ask for leave info	Period of time	->	AM|PM	AM|PM|Day|Days		none	5

Retrieval Results

SN	Name	Gender▲(0)	Starting time ▲(1)	Ending time	Period of time
1	A	male	2019-01-07	2019-01-07	AM
2	B	male	2019-01-09	2019-01-09	AM
3	C	male	2019-01-09	2019-01-09	AM
4	D	female	2019-01-04	2019-01-04	AM
5	E	female	2019-01-07	2019-01-07	PM
6	F	female	2019-01-08	2019-01-08	AM
7	D	female	2019-01-09	2019-01-09	PM
8	G	female	2019-01-10	2019-01-10	AM
9	D	female	2019-01-10	2019-01-10	PM

Figure 2: Results of a user-specified cross-table retrieval instance.

Before using GridView for retrieval setting, the user must select the table or view to query from ahead. Once selected, some columns of the GridView are bound to records selected from table of 'AllTableInfo'. The 'tableName' value of these records' is equal to the user selected table. Other columns of the GridView are preset for user to choose or kept empty for user to input. For example, in fuzzy query setting or range condition setting of a column, 'like' parameter and '[min,max)' parameters are kept empty for user to input. When we want to perform formulas on a few columns, the formulas of these columns need user to choose and the other columns appear in the 'goup by' part of the sql sentence automatically when producing query sql according user's setting.

In the GridView for result display, all columns and rows are generated dynamically according to what is retrieved from database using the generated sql sentence. For each row of records, there is a generated hyperlink for looking up to the original detailed information of record. Users can sort the result according to a single column or a combination of columns.

When user finishes retrieval setting and presses the query button, an instance of class 'SelectQuery' including list of 'selectedInfo' and 'whereInfo' will be generated by a function. The query sql is stored as a data in the 'SelectQuery' instance. The 'SelectQuery' instance is transformed to an easily-split string using function 'string ToString()'. The transformed string will be passed to client-side and reload the user page as a 'PostBack'. When reloading the user page, the transformed string will be transformed inversely to a 'SelectQuery' instance using function 'void setFromString(string infoStr)'. By executing the query sql of 'SelectQuery' instance in the Sql Server database, the resulting DataSource is bound to the GridView for result display. Other elements such as format for display can be got from the'SelectQuery' instance at the same time. What's more, 'SelectQuery' queries of the same output columns and different retrieval conditions can be combined to cover all-around retrieval conditions. Using table 'SubQuery' and 'FullQuery', the user-specified retrieval can be stored in database and reuse again by user easily instead of resetting.

Figure 2 gives an example of performing user-specified cross-table retrieval on basic info and ask-for-leave info. Flexible specification is available, including formulas and conditions setting on any column.

6. Conclusion

Our work offers a framework and method for user-specified advanced retrieval of various information under all-around conditions. Through our design, the query input view and query results view can be generated dynamically and flexibly in a unified framework according to user’s choice of query need. What’s more, our work make it possible for user to set the query need accurately covering all conditions. To ensure the robustness and safety of database storage, a database backup strategy is set up that makes backup both on the local server machine and a server machine on the web. Since usability and flexibility of use is our main concern, some other factors such as concurrency and efficiency can be improved in the future optimization.

References

[1] https://docs.microsoft.com/zh-cn/sql/?view=sql-server-ver15.
[2] Zhong Zhishui and Yao Jun, "Implementation of library information management system based on Delphi," 2010 International Conference on Computer and Communication Technologies in Agriculture Engineering, 2010, pp. 170-173, doi: 10.1109/CCTAE.2010.5544394.
[3] https://docs.microsoft.com/en-us/dotnet/api/?view=netframework-4.8.
[4] Ying Bai, "Accessing Data in ASP.NET," in SQL Server Database Programming with Visual Basic. NET: Concepts, Designs and Implementations, IEEE, 2020, pp.429-504, doi: 10.1002/9781119608493. ch8.
[5] Zhu Li-juan, "Research and application of SQL Server in the trade management system," 2011 3rd International Conference on Computer Research and Development, 2011, pp. 209-213, doi: 10.1109/ ICCRD.2011.5764282.
[6] C. Qi, "On index-based query in SQL Server database,"2016 35th Chinese Control Conference (CCC), 2016, pp. 9519-9523, doi: 10.1109/ChiCC.2016.7554868.