Improved Reporting and Document Generation via Object-Oriented Data Access and Enterprise Information Integration

Objects do not only change the way in which we develop software, they also allow a new breed of reporting solutions which makes better use of the information available in an enterprise.

This results in optimized business performance and lower development costs.

Qint Software GmbH
Fichtenstr. 19
82110 Germering
Germany
Tel.: +49 89 89026749
http://www.qintsoft.com

Introduction

Challenges for reporting and printing solutions have become greater in recent times. New multi-tier software architectures and object-oriented application development demand more intelligence from reporting tools. ReportWeaver rises to that challenge with the *object advantage*.

This paper describes a new approach to data access for reporting applications called *Object API-Based Access through the Business-Model Layer.* It provides considerable benefits namely better integration, increased ease of use and improved performance at a lower total cost of ownership.

Traditionally, reporting tools were connected to databases directly from where they loaded their data for analysis and output. For modern object-oriented and 3-tier architectures this poses a number of problems including difficulty of integration, security and loss of business logic. These are described in detail below. Inadequate solutions to these problems cause high implementation costs and even higher maintenance costs, slow performance and potentially unauthorized data access.

Qint Software has implemented a different method of data access in ReportWeaver. ReportWeaver accesses the interfaces of the objects that implement the business logic. This allows data access in real time. This way, ReportWeaver reuses existing business logic, security functions and data optimisations without any additional effort. The data modelling capabilities of ReportWeaver are fully object oriented, which allows ReportWeaver to integrate seamlessly into modern software architectures. This gives you the object advantage: it leverages your modern software technology to provide better service at reduced cost.

Technically this is a fundamentally different approach because objects do not provide SQL-based query interfaces (as databases do), but instead they offer application programming interfaces (API's).

Scenario for Data Access through the Business Logic with ReportWeaver

In what sort of scenario would you need ReportWeaver’s object-oriented access?

Imagine you are developing an application that runs on the server, is object oriented and has extensive business logic. Most likely, your application will have a database for storage. However, it is also very likely to have a data layer that presents the database contents as objects.

The data layer implements services such as encapsulation of queries, security mechanisms and performance optimisations such as caches etc. These mapped objects provide the basis for implementing the business logic. They are the business objects. They model the reality as closely as possible from a user’s point of view.

In addition, the separation of the business objects from the database model through the data access layer allows you to optimize the database model independently. As a result, the business-object model and the physical-database model may differ significantly. This difference does not pose a problem for application development. Therefore, this architecture is chosen quite frequently.

Problems Caused by Accessing Reporting Data via the Database Tier

At a first glance it seems natural to connect the reporting tool directly to the database. But what happens if we do that?

All the functionality of the data layer becomes unavailable and is bypassed or must be implemented a second time. Reimplementing the queries and maintaining them in both the application and the reporting tool is a costly process.

Worse still, all the access optimisations implemented in the data layer are lost, and reporting will run slower than necessary and create higher server loads, which, in turn, requires more powerful and expensive hardware.

However, this approach involves yet more problems. Security functions that are implemented in the middle tier are ignored, and users might obtain access to data that is not theirs to see. To avoid this, the whole security model must be implemented in the database, which can be complicated (because of the different physical model) and makes the database more difficult to manage.

Beyond the data access layer are the business objects. They provide many functions that are also useful for reporting. These may be specific aggregations or algorithms. Again, these would have to be reimplemented in the reporting tool. This means that you will have to use a vendor-specific scripting language, and the code will have to be maintained twice. The required training for the scripting language used creates additional costs.

The table below summarizes the problems of flat data access in traditional reporting tools:

Problem	***Consequence***
Queries in data layer must be reimplemented.	Increased development costs. Difficult maintenance of duplicated code.
Caches and optimized access patterns are not available.	Slow performance, high server load and more expensive hardware.
Physical database model must be used.	Hard to understand for end users. End users cannot design reports themselves.
Security model must be moved into the database.	Difficult to manage.
Aggregations in business logic are unavailable and must be reimplemented.	Requires use of non-standard scripting language, frequently with unproductive tools and extra costs for training. Code duplication.

In this scenario it would be ideal if the reporting tool could talk directly to the business objects and gain automatic access to business logic, caches, security features and package queries.

Avoiding Pitfalls of Half-Hearted Object Access

Some vendors provide partial solutions to the problems mentioned above. They talk about objects or beans serving as data sources. However, care must be taken to ensure that the data model of the tool is actually object oriented. If not, then only part of the problem is solved and you must decide whether the considerable price for these solutions is justified.

- Web-based reporting without application integration

While it is obvious that reports must be accessible from a web browser, this alone does not solve any of the problems mentioned above. The only bonus you get is that the reporting engine can run on the server and that reports can be distributed to thin clients. What is missing is application integration and the support of object models.

- Application integration without object model

  Some tools can read data from an API if the API implements some kind of table model. As a result, you have to map the object model back to a table model. This is another pseudo-solution to be avoided. This at least gives you the benefits of reusing the data access layer of your application, but a whole set of methods must be implemented and maintained. Since this forces your application to act as a relational database (which it probably isn't), the operations to connect data elements (joins) will occur in the reporting tool, and the performance benefit of the data access layer is lost again.

Requirements for Reporting Modules in Modern Applications

To allow API-based reporting, the reporting tool must have a number of properties to make this approach work.

Object-oriented data model

Since the business-object model of the application is object oriented, the reporting tool must understand objects and all their consequences such as inheritance, methods, references and collections. This is what sets ReportWeaver apart. Its internal logic is capable of directly representing any Java or C# object. With an object database it can even access C++ objects or it can access objects managed by a Corba server. Access to objects can be via methods, which preserves all business logic, optimisations and security checks.

Data navigation

Most reporting tools are designed to retrieve their objects from tables. However, in the application, business objects are not stored in tables: they are organized in collections and connected via references. ReportWeaver supports this kind of representation and allows navigation via references and the processing of collections. Navigational access is fast and simple. Data modelled with navigational references is more comprehensive for end users.

Needs for integration

To enable this kind of operation, the reporting tool must run inside the application. Therefore it must be packaged in the applications format, which is as a bean in a jar for Java or as an assembly for .net. In addition the tool must provide API's that integrate with the initialisation procedures of the envirnment and it must connect to the error handling and logging of the environment. Integration may be such that the end user does not even know that ReportWeaver is involved.

Reporting API

Inside the application, the reporting is now under the complete control of the application. ReportWeaver offers an API to query the report definitions and to start them. This way, they may be assigned to buttons or menus, and users can run them from there. Alternatively, the API may be used from a scheduler, which triggers reports at specific times and controls their distribution.

Benefits of Object-Oriented Access Features

Information integration

Object-oriented models are accepted as the most powerful way to describe the properties of an application. Therefore, they are also the preferred models used for enterprise information integration. Through these powerful models you can integrate any information source (databases, XML or application data). Object orientation also gives you polymorphism, which enables you to combine different object types (e.g. customers and prospects) in one list. This provides a more complete view of your data.

Ease of use

ReportWeaver encapsulates the complex technical details of relational joins in its internal layer, thus removing this technical burden from the users. Less technical knowledge is required for report design. Report design can be delegated to the end users, thus speeding up the development process and providing faster results.

Robustness to changes

Accessing data through the business-logic interface makes reporting more robust to changes in the internal structures. This reduces maintenance costs. Any additional functions implemented in the business-logic layer automatically become available to ReportWeaver.

Security functions

Using the methods of the business-logic layer ensures that the data is accessed in the right way. Any security measure implemented in this layer is automatically enforced for report access.

Performance

Faster access means more data to be processed in less time. This makes real-time reporting possible. Your enterprise benefits from faster reaction to market changes.