An Overview of SQL Support in Workflow Products

Marko Vrhovnik, Holger Schwarz, Sylvia Radeschütz, Bernhard Mitschang

University of Stuttgart
Universitätsstrasse 38
70569 Stuttgart, Germany
firstname.lastname@ipvs.uni-stuttgart.de

***Abstract*— Over the last years, data management products as well as workflow products have established themselves as indispensable building blocks for advanced IT systems in almost all application areas. Recently, many vendors have created innovative product extensions that combine service-oriented frameworks with powerful workflow and data management capabilities.**

In this paper, we discuss several workflow products from different vendors with a specific focus on their SQL support. We provide a comparison based on a set of important data management patterns and illustrate the characteristics of various approaches by means of a running example.

I. Introduction

Driven by the globalized economy, an increasing number of enterprises constantly automate their business tasks and streamline their IT infrastructure. Service-orientation, workflow management, and data management are perceived as core enabling technologies to achieve this. Hence, the solutions provided by the vendors comprise a service-oriented framework equipped with powerful workflow and data management capabilities typically made available by means of the vendors' own workflow and database products. Based on these products, enterprises organize their businesses in terms of business processes that automate certain business tasks by orchestrating a multitude of heterogeneous applications and data stores. In [1] and [2], we introduced an approach to optimize data processing in business processes based on a tight integration of SQL functionality into process logic. Such a tight integration allows for modeling data management tasks expressed via SQL explicitly within the process logic instead of encapsulating this functionality via Web services outside the process logic.

In this paper, we focus on approaches to tightly integrate SQL-specific data management capabilities into workflow languages and components for designing and processing such processes. Starting with three of the most prominent products, we reveal and contrast the main characteristics of the integration approaches they pursue.

This paper is organized as follows: In the following section, we introduce BPEL (business process execution language) as a well-known representative of workflow languages, prominent products that add SQL capabilities to such languages as well as the main criteria for our comparison. In Sections III, IV and V, we discuss three products in more detail and highlight the main conclusions in Section VI. We conclude this overview in Section VII.

II. SQL Support in Workflow Products

In order to enhance independence, substitutability and migration, the most important vendors of workflow technology started a standardization process. As a first result, the business process execution language BPEL was published in 2003, and in 2005 a next version was released [3]. Similar to other workflow languages, BPEL fosters a two-level programming model. The function layer consists of executable software components in form of Web services that realize the basic activities. The choreography layer specifies a process model defining the potential order of the business process execution. As BPEL does not provide full support for covering all business scenarios ranging from system workflows to human workflows, all vendors provide additional proprietary functionality for bridging this gap.

All products in our focus support BPEL in some way. There is either direct support by means of a BPEL engine, like in IBM's Business Integration Suite [4], Oracle's SOA Suite [5] as well as BEA's AquaLogic BPM Suite [6]. Alternatively, there is indirect support by means of import and export functions, as in Microsoft's BizTalk Server [7]. The latter tool is based on XLANG [8], Microsoft's proprietary workflow language. Microsoft's upcoming Extensible Object Markup Language (XOML) [9] is a declarative XML-based language for workflow specification used in Windows application programming. As part of the .NET Framework 3.0 [10], it is the language used to implement Microsoft's Workflow Foundation [9]. On top of XOML, one may implement other standards, such as BPEL. The current version of the Workflow Foundation provides import and export tools for BPEL and includes an activity library representing BPEL. Beside the listed products there are numerous other vendors of BPEL engines that we do not address in this paper.

The main approaches for adding SQL support to workflow languages are shown in Figure 1. The *adapter* technology is based on the idea of service integration. An adapter realizes a service that encapsulates SQL-specific functionality and that can be called by other processes. Adapters typically mask data management operations as Web services. They represent a proven and well-established technology and are provided in a similar way by different vendors. One important characteristic of this approach is that data management issues are separated from the process logic. Our focus is on *SQL inline support*, an uprising technology, which allows for a tight SQL inte-

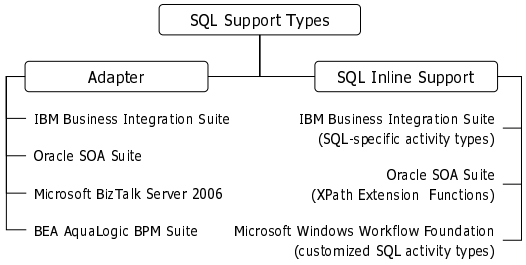

Fig. 1. SQL Support in selected Workflow Products

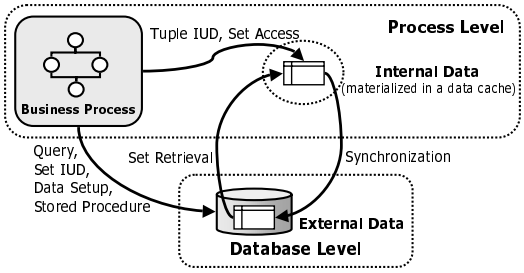

Fig. 2. Data Management Patterns

gration into the process logic. Unlike the adapter technology, SQL inline support uncovers data management issues at the process level. This is achieved by augmenting the set of activity types of a workflow language through SQL-specific functionality. IBM's Business Integration Suite provides a language extension to BPEL by adding SQL-specific activities. Microsoft's Workflow Foundation introduces an extensible set of activity types that can be augmented to customized activities supporting SQL. Oracle provides proprietary XPath Extension Functions for executing SQL on a database system.

A. Comparison Criteria

Although the different approaches seem to be quite similar at the abstract level, a more detailed view reveals major differences. A first group of comparison criteria is based on *general information* concerning each of the approaches. Compliance with the BPEL specification is one issue. Other aspects are the complexity of process design and the tool support that is provided. Approaches differ in the level of process modeling. It is quite common to model abstract business processes based on a graphical notation. This allows to focus on important issues of business process modeling and it hides minor language-specific issues. Nevertheless, one could also model processes directly in a programming language or a text-based workflow language like BPEL. The output of modeling tools may be twofold: A tool may provide a description of the process in some workflow language or it could directly produce code in some programming language.

Data management capabilities comprise a second group of comparison criteria. We have to consider the SQL functionality that is supported. Is it possible to exploit the complete functionality provided by the SQL standard or do any restrictions exist? This directly influences the options provided to the process designer in defining the data management aspects of business processes. Another issue is the processing of external data. One option is to always transfer and materialize external data into the processing engine. This approach has severe limitations with respect to the data volume to be transferred. Using references to external data in data processing activities helps to overcome these limitations. The processing itself is performed by the external data source. This reduces the volume of data transfer as well as the need for storage capacity in the processing engine. Furthermore, we analyze which of the *data management patterns* introduced in the following section can be realized in a certain approach. As we consider all of them to be essential for realistic business processes, we expect a complete coverage from all approaches.

A comparison of the approaches based on a performance evaluation is not beneficial, because the language extensions and the execution platforms are too different. Hence, a sheer comparison of performance numbers is meaningless. Instead, a qualitative comparison, as given here, is more significant.

B. Data Management Patterns

As shown in Figure 2, we distinguish between internal and external data. Internal data is directly managed in the process space whereas external data is managed external to it, e.g., by means of a database management system. We identified the following data management patterns for accessing and processing data in business processes.

- The *Query Pattern* expresses the need for querying external data by means of SQL queries. Query results are either stored in the external data source or materialized in the process space.
- The option to use SQL statements to perform set-oriented insert, update and delete operations on external data is reflected by the *Set IUD Pattern*.
- The *Data Setup Pattern* describes the need for executing Data Definition Language (DDL) statements on a relational database system for configuration and setup purposes during process execution.
- As complex data processing is frequently expressed by stored procedures, it is important that the processing of external data allows for calling stored procedures as well (*Stored Procedure Pattern*).
- Sometimes, it is important to process data in the process space. Hence, it is necessary to retrieve data from an external data source and to materialize it in a set-oriented data structure within the process space (*Set Retrieval Pattern*). This data structure acts like a data cache in the process space holding no connection to the original data source.
- The *Set Access Pattern* reflects the need for sequential and random access to this data cache.
- The *Tuple IUD Pattern* covers insert, update and delete on the data cache.
- The *Synchronization Pattern* represents the synchronization of a local data cache with the original data source.

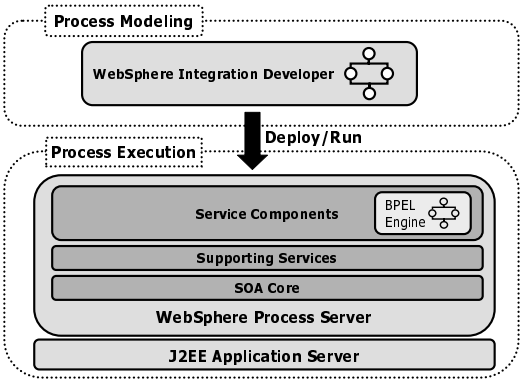

Fig. 3. Process Modeling and Execution in IBM BIS

In the following sections, we discuss how these data management patterns can be realized in IBM's Business Integration Suite, Microsoft's Workflow Foundation, and Oracle's SOA Suite, all providing means for directly embedding SQL functionality into the process logic. Furthermore, we provide general information and discuss data management capabilities of these products in detail.

III. IBM Business Integration Suite

IBM offers a complete Business Integration Suite (BIS) supporting all phases of the Business Process Management lifecycle, from modeling through development, deployment and monitoring of business processes. It is based on BPEL and the service-oriented architecture (SOA). We focus on two core products of BIS, i.e., the WebSphere Integration Developer (WID) for creating process models and the WebSphere Process Server (WPS) for process execution. IBM provides an Information Server Plugin for WID that adds SQL functionality to BPEL via so-called information service activities and allows to treat data manipulation and transformation issues at the choreography level. Thereby, simplified and flexible access to relational databases and to other data sources via federation technology is provided. Additionally, WPS supports life cycle management on the process level.

A. General Information

IBM's BIS version 6.1 provides a common service-oriented framework for all aspects of business process management solutions. Figure 3 shows a simplified view of those parts of BIS that are responsible for process modeling and execution. BPEL process models are created in *WebSphere Integration Developer* [11]. Targeted to non-programmers, it provides a process editor for creating workflows in a graphical manner. As a result of this design step, we get a description of the process in BPEL. From this description the tool generates code that is deployed and executed on the *WebSphere Process Server* [12]. WPS builds the foundation of IBM's service oriented architecture. Based on a *J2EE* runtime and SOA infrastructure, it provides a process engine that executes BPEL flows. This *BPEL Process Engine* is part of a set of *Service Components* that automate business processes spanning people, workflows, and all kinds of application systems and platforms. These Service Components are based on *Supporting Services* that simplify the development of integration solutions. The *SOA Core* is responsible for the invocation of the Service Components and for the interaction with all kinds of external services and systems. BIS provides additional tools for modeling, designing, and monitoring workflows, as well as runtime services for supporting a complete process lifecycle management.

B. Data Management Capabilities

In this section, we show how BIS allows to treat data management issues at the choreography level.

SQL Inline Support: The following types of *information service activities* augment the set of BPEL activities and provide access to external data:

1) *SQL activities* are essential for providing full SQL functionality directly in the process logic. An SQL activity embeds an SQL statement that is sent to a database system where it is processed. Queries, Data Manipulation Language (DML) statements, Data Definition Language (DDL) statements as well as stored procedure calls are supported. A specific feature of an SQL activity is that a resulting data set is not directly passed to the process space. Instead, it remains in the data source and is only referenced in the process. We explain further details on referencing external data sets below.
2) A *retrieve set* activity bridges the gap between external and internal data processing by loading external data into the process space. It returns a table's materialization, thereby, preserving the relational structure of the table in an appropriate XML structure.
3) WID provides an *atomic SQL sequence* activity type in order to add flexibility for defining transaction boundaries. It embeds a sequence of SQL and retrieve set activities. In long-running processes this sequence is processed as a single transaction. In short-running processes all SQL and retrieve set activities of a process are executed in a single transaction. Thus, the definition of transaction boundaries via atomic SQL sequence activities is not necessary in such scenarios.

Referencing External Data Sets: WID provides two types of *set reference* variables as handles to external tables. In an information service activity, they can be used in place of static table names. This allows to dynamically bind a set reference to a table at runtime. An *input set reference* refers to a table, e.g., it may denote a table that is changed by an SQL UPDATE statement. A *result set reference* refers to a table (called *result set table*) holding the result of an SQL query or a stored procedure call. It may be redefined as input set reference of a consecutive information service activity. This concept allows to pass external data sets across activities or processes by reference instead of by value. WID also allows to include scalar values as input parameters of an SQL statement.

Materialized Set Representation: WID provides a *set* data type that is an XML-based representation of a set-oriented

data structure in the process space. It is used to define a set variable that usually holds the materialization of a table denoted by a set reference. Retrieve set activities provide this materialization, which makes data from databases available for further processing in the process space.

Referencing External Data Sources: An important characteristic of IBM's approach is that it supports the dynamic binding of data sources. In WF and Oracle SOA Suite a static connection is established. WID provides data source variables that hold the connection string to refer to a database system. When defining an information service activity, the target database system is specified by referring to the corresponding data source variable. This allows to dynamically switch between different databases without re-deploying the process. Such a dynamic binding can take place either at deployment time when a process is installed on WPS or at runtime.

Additional Features: WID and WPS support the configuration and setup of database systems before, during and after workflow execution. This task is not supported by the approaches of Microsoft and Oracle, hence, it has to be modeled explicitly as part of the process logic or outside the process execution context.

WID allows to define preparation and cleanup statements for data sources that may run before or after process execution. Preparation and cleanup statements are DDL statements for managing database entities like database tables, table configuration data, buffer pools, and stored procedures. Additionally, preparation and cleanup statements can directly be bound to a set reference variable. For example, a result set reference typically points to a temporary table in a workflow. To avoid having to create such tables before a process starts and dropping them when they are no longer needed, WID and WPS allow to control the lifecycle of result set tables at the activity level by specifying preparation and cleanup statements.

C. Data Management Patterns

It is straightforward to realize the patterns for processing external data (*Query Pattern*, *Set IUD Pattern*, *Stored Procedure Pattern*) using an SQL activity as it allows to specify arbitrary SQL statements. This simple realization is characteristic for SQL inline support. Furthermore, an SQL activity allows to execute DDL statements at process runtime, thus, covering the *Data Setup Pattern*.

There is a significant difference in the realization of the *Set Retrieval Pattern* in BIS and the two other products. In WID, a result set is treated as external data managed by a database system and referred to by a result set reference within the process space. This results in an explicit materialization step in the process logic realized by a retrieve set activity where relational data is loaded into set variables in order to allow local processing. In the other approaches the execution of a query or stored procedure call is always aligned with a consecutive materialization step that automatically imports the resulting data set into the process space.

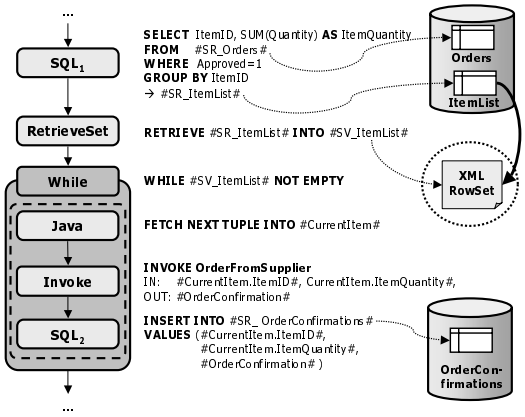

Fig. 4. Sample Workflow using IBM BIS Technology

Assign activities are essential for the local processing of set variables (*Tuple IUD* and *Set Access Pattern*). They allow to copy data from one variable to another. The BPEL specification predetermines XPath [13] as language for expressions over the source and target variable of an assign activity. This way, it is possible to determine specific elements in BPEL variables as well as to select and update certain tuples in a set variable. In contrast, insertion and deletion of tuples is not possible by means of an assign activity. For this purpose, one may use a workaround based on Java-Snippets [14]. A Java-Snippet is an IBM specific extension allowing to embed Java code directly into the process logic. Within a Java-Snippet one may directly access a set variable as Java object and update, insert and delete tuples in this variable.

Since assign activities only support random access to set variables, we need other means to provide sequential access to set variables. Such a cursor functionality is based on a while activity and on a Java-Snippet. A Java-Snippet accesses the set variable as Java object and retrieves the next tuple in each iteration. This tuple is stored in an appropriate process variable. The body of the while activity is repeated as long as the set variable contains further tuples.

WID provides no activity type for synchronizing a set variable and the original data source. As a simple workaround, one may specify appropriate UPDATE statements in an SQL activity in order to realize the *Synchronization Pattern*.

Example: The sample workflow shown in Figure 4 demonstrates the realization of several data management patterns in BIS. Each activity within the flow is represented by a box that shows its name. We omit this for sequence activities, but represent them by a dashed box that surrounds all the activities contained in the sequence and by a set of arrows that determine the order in which these activities are executed. SQL statements and additional information that is necessary to understand the semantics of an activity is shown in a pseudo notation next to the activities. For clarity, we mark BPEL variables by surrounding hash marks (#). Furthermore, we omit preparation and cleanup statements. The sample workflow implements a fraction of a business process that aggregates

approved orders and determines the required quantity of each item type.

SQL activity SQL_1 realizes the *Query Pattern*. It selects orders being marked as approved, groups them by their item type, and aggregates the quantities for each item type. The table accessed by this query is referenced via the input set reference variable *SR_Orders*. This table contains a set of orders where each order consists of an identifier (OrderID), the type of item to order (ItemId), the quantity to order (Quantity) and an attribute that indicates whether the order is approved or not. The result of the query is temporarily stored in a table referenced by the result set reference variable *SR_ItemList*. For each workflow instance such a table is created with a generated unique name before the activity SQL_1 is executed and it is dropped at the end of the workflow. Therefore, the lifecycle of the result table has to be defined appropriately. The retrieve set activity represents the *Set Retrieval Pattern* by loading the content of the table referenced by *SR_ItemList* into the set variable *SV_ItemList* organized as XML RowSet.

The while activity and the Java-Snippet are necessary for representing the cursor functionality (*Set Access Pattern*). As part of each iteration one tuple is bound to the variable *CurrentItem*, and an invoke activity and activity SQL_2 representing the *Set IUD Pattern* are executed. The former calls the Web service OrderFromSupplier to order required items from a supplier. This Web service has two input parameters, the item type and the required quantity. They are provided by the attributes *ItemID* and *Quantity* of the tuple bound to *CurrentItem*. The Web service returns an order confirmation string that indicates whether the order has been processed successfully or not. This confirmation string is stored in the scalar variable *OrderConfirmation*. In the subsequent SQL activity it is stored in a table referenced by *SR_OrderConfirmations* together with the type and the quantity of the ordered item. This persistent table stores the confirmations of all workflow instances.

Conclusion: The SQL integration approach pursued by IBM's Business Integration Suite provides a powerful concept for accessing and processing relational data directly on the process level. The provided SQL activity types cover a large number of data management patterns. However, for the *Sequential Access Pattern*, parts of the *Tuple IUD Pattern* and the *Synchronization Pattern* one has to use appropriate workarounds for realizing the missing data management capabilities.

IV. Windows Workflow Foundation

Windows Workflow Foundation (WF) is Microsoft's technology for defining, executing, and managing workflow-based Windows applications. It provides capabilities for defining customized SQL activity types. This allows for a tight integration of SQL into WF workflows that are defined based on .NET programming languages, such as C# and Visual Basic (VB). As part of the *.NET Framework 3.0*, WF has an impact on other Microsoft technologies providing workflow functionality. For example, the release following BizTalk Server 2006 will include the ability to create WF workflows alongside with BizTalk's current orchestration capabilities. In the following, we introduce this technology focusing on its SQL inline support.

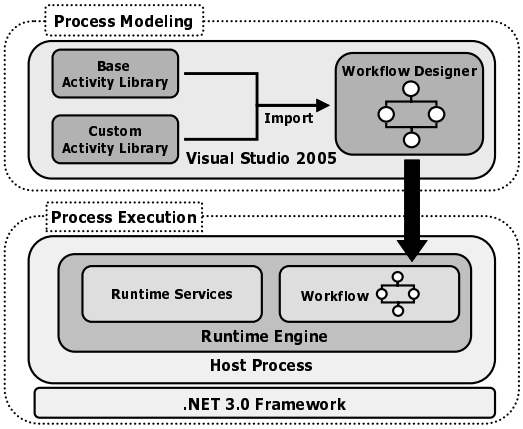

Fig. 5. Process Modeling and Execution in Microsoft WF

A. General Information

WF is a code library for software developers that provides the basic infrastructure for developing and executing workflow-based Windows applications. Figure 5 shows the main components of WF.

Workflows are built from a group of activities of which each executes some tasks in the workflow. For constructing such a workflow, WF provides a *Workflow Designer*, a graphical tool hosted within Visual Studio that supports the graphical construction of workflow-based applications for simplifying the code development process. Since WF will be integrated in other Microsoft technologies, it is foreseeable that there will be appropriate design tools for constructing WF-based workflows at a higher abstraction level as it is already known, e.g., from Microsoft's BizTalk Server.

WF includes a *Base Activity Library* (BAL) that provides proprietary functionality for control flow, conditions, event handling, state management, and communication with applications and services. When designing workflows, one may use the activities provided by the base activity library or one may create a *Custom Activity Library* (CAL) for a particular problem space. One may implement customized activity types from scratch or assemble them from existing activity types. Additionally, import and export tools for BPEL as well as an activity library representing BPEL are available. This way, one may also model workflows conforming to the BPEL specification.

Executing a workflow is the job of the *Runtime Engine*. It relies on a group of *Runtime Services* for, e.g., persisting a workflow's state or tracking its execution and communication with external applications. WF includes standard implementations for these services, but developers may replace them as needed.

The runtime services, the runtime engine and the workflow

itself are contained in a *host process*, that may be any Windows .NET process, ranging from a simple console application to a workflow-based server process that is executed by the .NET runtime.

Besides modeling workflows graphically, it is also possible to implement a workflow in a .NET programming language, to describe a workflow in a markup language, and to combine these workflow authoring modes. In particular, the following different workflow authoring modes are supported by WF: (i) In the *code-only* authoring mode, the developer implements workflows directly in a .NET programming language such as C# or VB by using the WF API. (ii) The *markup-only* mode enables developers to author a workflow using Microsoft's *Extensible Object Markup Language* (XOML). A XOML description can either be compiled by WF's *workflow compiler* that maps the markup to corresponding C# and VB code respectively, or it can directly be loaded into the workflow runtime engine. (iii) The *code-separation* authoring style supports developers in defining the workflow structure using XOML and combining it with C# or VB code implementations. When using a code-only or code-separation mode, a workflow's structure can graphically be modeled using the Workflow Designer. As a result of the design step, the tool generates an appropriate code file or an XOML description. The workflow structure may also be created directly without using the Workflow Designer. As a result of the workflow design and compilation, we get a .NET assembly that is executed by the workflow runtime engine.

B. Data Management Capabilities

Currently, BAL does not provide any activity type considering SQL issues. As a workaround, one may use a code activity for executing arbitrary .NET code in a workflow. With such a code activity, it is possible to use Microsoft's ADO.NET API [15] for accessing and processing relational data and for managing relational data in main memory. But, embedding ADO.NET code directly in a workflow reduces the reusability of a data management pattern. Hence, we do not discuss this option any further. Nevertheless, in WF another more desirable integration of SQL into the process logic is achievable by augmenting CAL through customized SQL activity types that provide the needed functionality on a higher level of abstraction. A first step in this direction is provided by a customized SQL database activity type (available at [9]) whose current functionality we discuss next.

SQL Inline Support: Based on ADO.NET, an *SQL Database Activity* provides the ability to execute an SQL statement including queries, Data Manipulation Language (DML) statements, Data Definition Language (DDL) statements, and stored procedure calls. Furthermore, one has the option to specify event-handlers for this activity type containing arbitrary code that can be executed either before or after an SQL statement. This way, one may, e.g., initialize parameter values of the SQL statement or directly process result data.

Referencing External Data Sets: An SQL statement may include several input parameters as host variables that are valid at any position where a scalar value is allowed. Table names, however, are considered as *static* part of an SQL statement.

Materialized Set Representation: The execution of a query or stored procedure call is always aligned with a consecutive materialization step that automatically imports the result data set into the process space. Such a materialization is represented as a *DataSet object* provided by the ADO.NET API. This DataSet object provides a materialization of the resulting table that can further be processed within the process space. The DataSet class is part of the ADO.NET API that underlies the implementation of the SQL database activity. It can be regarded as a cache for relational data on the client side that holds no connection to the original data. We discuss comprehensive facilities for managing and processing such caches in Section IV-C.

Referencing External Data Sources: For an SQL Database Activity one has also to specify a *static* connection string that is used at runtime to connect to the corresponding database system. After having executed the statement, the connection is closed again.

C. Data Management Patterns

By means of an SQL database activity, all patterns for querying, inserting, updating, deleting and setting up data as well as for calling stored procedures on a relational database system can be realized in a straightforward way. As the SQL database activity materializes a resulting set into a DataSet object it also covers the functionality of the *Set Retrieval Pattern*.

For the implementation of the remaining patterns, we have to go back to a code activity using the ADO.NET API that provides appropriate methods (i) for inserting, updating, and deleting tuples of a result set table that is encapsulated within a DataSet object, (ii) for sequentially iterating over a result set, (iii) for querying a specific tuple within a result set, and (iv) for synchronizing a cached result set table with its original data source. A drawback is that we have to use code activities in the process logic that increase the complexity of process design and reduce its reusability. However, such a code activity is currently the only way to realize the *Random Set Access*, *TupleIUD* and *Synchronization Pattern* in WF. For a simpler and more intuitive realization of the *Sequential Set Access Pattern* we can use a while activity that provides another option to iterate over the tuples of a DataSet object. In addition, we have to specify an appropriate condition based on the ADO.NET API either in C# or in VB for the while activity.

Example: In Figure 6, we show how to realize the example flow by means of WF. We have chosen the same notation and terminology as before, in order to better compare the SQL integration approaches.

We use SQL database activities ($SQLDatabase_1$ and $SQLDatabase_2$) for executing the query and the INSERT statement. Unlike in IBM's WebSphere Integration Developer, a database table that is specified as part of an SQL statement is not represented by a variable but by its name. As query result, we get a DataSet object containing a materialization of

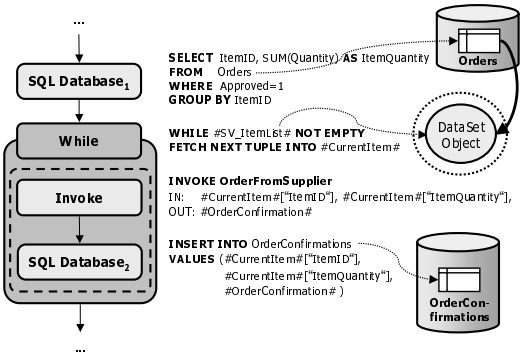

Fig. 6. Sample Workflow using Microsoft WF Technology

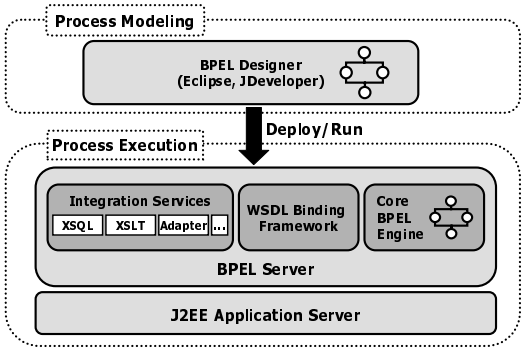

Fig. 7. Process Modeling and Execution in Oracle SOA Suite

the result table. It is stored in host variable *SV_ItemList*. Since the lifecycle of this variable is associated with the lifecycle of the process instance, the table resulting from $SQLDatabase_1$ is only temporarily available during process execution.

By using the ADO.NET API we have direct access to all tuples of the resulting table. We use a while activity for iterating over the result of $SQLDatabase_1$. In each iteration, the next tuple is stored in host variable *CurrentItem*. The tuple values in *CurrentItem* are used as input parameters by the consecutive two activities executed within a sequence activity (dashed line in Figure 6). The invoke activity calls Web service OrderFromSupplier and $SQLDatabase_2$ executes the INSERT statement. Since a tuple is represented by an array data structure in ADO.NET, we can use the attribute names to access the required tuple values (*CurrentItem*["*ItemID*"], *CurrentItem*["*ItemQuantity*"]).

Conclusion: As conclusion we can state, that in WF there is currently no proprietary SQL inline support, since BAL does not provide any SQL functionality. User-specific code that is based on the powerful ADO.NET API closes this gap, but is not preferable, since it decreases reusability and at the same time increases the complexity of process modeling. Nevertheless, a more intuitive and reusable SQL integration in WF is achievable by augmenting CAL through SQL-specific activity types. This way, it is possible to realize all data management patterns on a more abstract level. Currently, there is only one such SQL-specific activity type in CAL abstracting from the underlying ADO.NET implementation whose functionality covers a subset of the data management patterns.

V. Oracle SOA Suite

The Oracle SOA Suite offers an environment for building, deploying, and managing service-oriented applications including process orchestration of services. It contains the BPEL Process Manager (BPEL PM) [16] to compose services as BPEL processes. In contrast to IBM and Microsoft, Oracle's SQL inline support is not based on SQL-specific activity types. Instead, BPEL PM provides proprietary XPath Extension Functions that can be called by assign activities in a BPEL process.

A. General Information

Oracle BPEL Process Manager is one part of the Oracle SOA Suite 10g Release 3. Based on a *J2EE Application Server* it offers an environment for designing, deploying and managing BPEL processes. Figure 7 shows a simplified view of its core components. For building BPEL processes visually, BPEL PM provides a *BPEL Designer*, which is either available as a plug-in to the Eclipse environment [17] or to the JDeveloper [18], an integrated development environment from Oracle. JDeveloper, e.g., supports process designers in creating and deploying BPEL processes. After deployment, the processes are executed by the *Core BPEL Engine*. The *WSDL Binding Framework* is responsible for connecting a process to other Web services by using various protocols and message formats. *Integration Services* support transformations on XML documents that are exchanged between a BPEL process and other Web services. To integrate BPEL processes with access to other resources, e.g., files, FTP servers, and database tables, BPEL Server provides support for the adapter technology.

Besides BPEL PM, the SOA Suite offers further technologies for monitoring, managing and developing business processes to support a complete Business Process Management lifecycle.

B. Data Management Capabilities

Oracle provides an inline SQL support by means of proprietary *XPath Extension Functions* [16] as part of the BPEL assign activity. These functions use built-in BPEL capabilities and XPath standards and are defined in namespaces indicated by the prefixes *ora* and *orcl*.

SQL Inline Support: The following XPath Extension Functions are available:

1) The function *query-database* executes any valid SQL query that is provided to the function as string parameter and returns a result set.
2) The function *sequence-next-val* returns the next value of a predefined sequence of integers. It is useful, e.g., when creating a unique number as a primary key in a table.
3) The function *lookup-table* executes the SQL query

```
SELECT outputColumn FROM table
WHERE inputColumn = key
```

that is generated from the function's input parameters *outputColumn*, *table*, *inputColumn*, and *key*. It returns exactly one column value of the output tuple that is specified by its primary key.

4) The function *processXSQL* accesses an XML file, which includes an SQL statement, executes it in the XSQL Framework [19] that is part of the BPEL Server, and returns its result in XML. The XSQL Framework combines XML, XSLT (Extensible Stylesheet Language for Transformations) [20], and SQL. It generates XML results from parameterized SQL queries and supports DML and DDL operations as well as stored procedures.

Referencing External Data Sets: Similar to Microsoft's Workflow Foundation, all XPath Extension Functions mentioned above consider external data sets as a *static* part of an SQL statement, i.e., a table name is simply defined as text that is either provided as input of a function or directly embedded within an SQL statement.

Materialized Set Representation: When calling the functions query-database or processXSQL for executing an SQL query or stored procedure, a result set is returned that is materialized into a proprietary XML RowSet representation that can be further processed within the process space.

Referencing External Data Sources: Like in Microsoft's Workflow Foundation, one has to provide a *static* connection string for each XPath Extension Function.

C. Data Management Patterns

The concept of XPath Extension Functions covers the data management patterns as follows:

All SQL patterns for data setup, for querying, inserting, updating and deleting data as well as for calling stored procedures can be implemented using the XPath Extension Function processXSQL. The remaining XPath Extension Functions are only able to handle the *Query Pattern*.

By materializing the resulting tuples of a query via an XML RowSet representation, all functions realize the *Set Retrieval Pattern*. Each output tuple of an XML RowSet becomes a numbered XML element with a text node for every attribute value. Since the XPath Extension Functions are embedded into an assign activity, their output can simply be stored into the associated variable.

The BPEL-specific XPath Function *getVariableData* allows to extract entire row sets or only a single node value from an XML RowSet. Because it is available either as XPath Extension Function or as Java built-in method, an assign activity as well as an Oracle-specific Java-Snippet activity may cover the *Set Access Pattern* for accessing a result set in a random manner.

Like in the IBM approach, one can use as workaround a while activity and an Oracle-specific Java-Snippet activity for providing sequential access to rows of an XML RowSet.

The *Tuple IUD Pattern* can be realized by including Java parts into the process. Another option for this pattern is to use Oracle-specific XPath operations denoted by the namespace *bpelx* that allow to update, insert and delete local XML data.

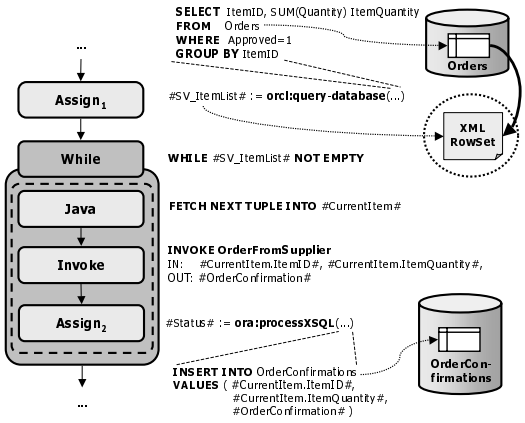

Fig. 8. Sample Workflow using Oracle SOA Suite Technology

For the *Synchronization Pattern* one has to use another workaround, e.g., to manually add the function processXSQL to the process that ensures that updates in the process space are also reflected in external data.

Example: Figure 8 shows the realization of our sample workflow using XPath Extension Functions. Again, we use the terminology introduced in Section III-C. All tables are identified by their names via static text. For executing the query, we use the XPath Extension Function query-database, which is encapsulated in an assign activity (*Assign*$_1$). The result of the query is an XML RowSet obtaining the materialization of the resulting table. This RowSet is stored in variable *SV_ItemList*.

To access the resulting tuples and to continue working on them, a while activity iterates over the result set of the query. In each loop iteration, the Java-Snippet stores the next tuple into the variable *CurrentItem* whose values are used afterwards as input parameters for the two following activities (*Invoke* and *Assign*$_2$). *Invoke* calls the Web service OrderFromSupplier. Thereafter, *Assign*$_2$ executes function processXSQL. It defines an INSERT statement that contains *CurrentItem* and *OrderConfirmation* as input parameters. *Status* provides the return status of the function call.

Conclusion: Oracle provides an SQL integration approach that differs from the two other approaches. Instead of providing SQL-specific activity types, Oracle supports SQL functionality by means of proprietary XPath Extension Functions as part of the BPEL assign activity. As discussed above, most data management patterns can be realized by means of this approach. However, one has to use workarounds for implementing missing data management capabilities, namely the *Sequential Access Pattern* and the *Synchronization Pattern*.

VI. Discussion

In this section, we contrast the different approaches and show the main conclusions as summarized in Tables I and II.

A. General Information

BPEL is the workflow language that underlies the products offered by IBM and Oracle. In contrast, Microsoft's Workflow Foundation uses C#, VB or XOML. BPEL support is reflected

TABLE I
GENERAL INFORMATION AND DATA MANAGEMENT CAPABILITIES

	IBM Business Integration Suite (BIS)	**Microsoft Workflow Foundation (WF)**	**Oracle SOA Suite**
General Information			
Workflow Language	BPEL	C#, VB, XOML (BPEL)	BPEL
Level of Process Modeling	graphical, (markup)	graphical, code, markup	graphical, (markup)
Workflow Design Tool	WebSphere Integration Developer	Workflow Designer	Process Designer
Data Management Capabilities			
SQL Inline Support	SQL Activity, Retrieve Set Activity, Atomic SQL Sequence	customized SQL Activity	XPath Extension Functions
Reference to External Data Set	Set Reference, static text	static text	static text
Materialized Set Representation	proprietary XML RowSet	DataSet Object	proprietary XML RowSet
Reference to External Data Source	dynamic, static	static	static
Additional Features	Lifecycle Management for DB Entities	-	-

in a specific BPEL activity library. Using this library, it is possible to create workflows that can be exported to BPEL as well as to import a BPEL definition into WF.

All three vendors provide graphical solutions for constructing processes, thus, significantly reducing the complexity of process design. However, the solutions provided by IBM and Oracle differ from those provided by Microsoft. IBM and Oracle address developers that create business processes at a modeling level abstracting from BPEL and its underlying implementation. On the other hand, developers use Microsoft's Workflow Designer for implementing business processes via predefined activity libraries. Since WF will be integrated in other Microsoft technologies, it is foreseeable that there will be appropriate design tools for constructing WF-based workflows at a higher abstraction level as it is already known, e.g., from Microsoft's BizTalk Server.

B. Data Management Capabilities

All vendors provide different facilities for integrating SQL functionality into the process logic. BIS provides a predefined set of SQL-specific activity types. WF offers no proprietary SQL inline support, but allows the user to implement his own set of SQL-specific activity types. Oracle realizes SQL inline support by means of proprietary XPath Extension Functions as part of the BPEL assign activity. An SQL activity in BIS as well as the proprietary XPath Extension Functions in SOA Suite can be bound to any supported database system, whereas the implementation of the SQL database activity in WF we presented here is restricted to SQL Server and Oracle database systems. Specific features of BIS are the retrieve set activity that loads external data into the process space and the atomic SQL sequence activity that is useful in long-running processes as it allows to bundle several SQL operations into one transaction.

Another important aspect is how activities at the process level refer to external data. WF and SOA Suite consider a table name only as static part of an SQL statement. In contrast, set references in BIS act as pointers to tables. Hence, it is possible to dynamically choose at runtime which table to use and to pass relational data across different activities and processes by reference rather than by value.

Since IBM and Oracle use BPEL as workflow language, both vendors refer to the concept of XML RowSets, adapted to XML-based representations of materialized data sets. In WF, the implementation of the SQL database activity uses a DataSet object provided by the ADO.NET API for representing a materialization.

IBM's approach differs from the other approaches in the way external data sources are referenced. While in WF and SOA Suite a static connection is established, BIS supports besides a static connection also a dynamic binding of data sources at runtime. This allows, e.g., to switch between a test environment and a production environment without redeploying a process.

Additionally, IBM provides facilities for managing the lifecycle of result sets as well as for setting up data sources at process level. Such a management is not supported by the other two approaches. Hence, it has to be modeled explicitly as part of the process logic or outside the process execution context.

C. Data Management Patterns

In BPEL, an activity represents a discrete processing step that abstracts from its concrete implementation. This decreases complexity of process modeling and increases its reusability. Furthermore, a process designer is able to model complex process models without knowing about concrete implementation techniques of single processing steps. Such a two-level programing model is also preferable for SQL inline support. The more implementation details of a data management pattern are hidden from the process designer the easier it is for him to realize such a pattern. As shown in Table II such an abstract modeling of all data management patterns is currently not achieved by any of the three vendors, since there are patterns that can only be realized using workarounds.

It is obvious for SQL inline support that all patterns concerning the processing of external data can be realized at an abstract level. The *Query Pattern*, the *Data Setup Pattern*, the *Set IUD Pattern* and the *Stored Procedure Pattern* can be achieved by means of an SQL activity, an SQL database activity, or appropriate XPath Extension Functions. There is a significant difference in the realization of the *Set Retrieval*

TABLE II
DATA MANAGEMENT PATTERN SUPPORT

	Query	Set IUD	Data Setup	Stored Procedure	Set Retrieval	Seq. Set Access	Random Set Access	Tuple IUD	Synchronization
IBM Business Integration Suite									
SQL	x	x	x	x					
Retrieve Set					x				
Assign (BPEL-specific XPath)							x	x[1]	
Only workarounds possible						x		x[2]	x
Microsoft Workflow Foundation									
SQL Database	x	x	x	x	x				
Only workarounds possible						x	x	x	x
Oracle SOA Suite									
Assign (XPath Ext. Functions)	x	x	x	x	x			x	
Assign (BPEL-specific XPath)							x	x[1]	
Only workarounds possible						x			x

[1]only UPDATE, [2]only DELETE and INSERT

Pattern between BIS and the other approaches. In BIS a result set is treated as external data managed by a database system and referred to by a result set reference within the process space. This results in an explicit materialization step in the process logic realized by a retrieve set activity in order to allow local processing. This way, BIS provides the flexibility for treating a result set either as external data or as internal data. The former allows further processing steps in SQL, whereas the latter implies local XML processing of the result set. In the other two approaches the execution of a query or stored procedure call is always aligned with a consecutive materialization step that automatically imports a resulting data set into the process space and, thus, always treats result sets as internal data.

In contrast to the data management patterns concerning external data, the realization of local processing steps on internal data is not always possible at an abstract level as shown in Table II. Here, in some cases workarounds are needed including user-specific code. The solutions of IBM and Oracle cover some of these data management patterns at the abstract level. For example, consider the *Random Set Access Pattern*. Since BPEL is the common language used by IBM and Oracle, random access to materialized data set is possible in both solutions using an assign activity and a BPEL-specific XPath expression to select one or more tuples. The same way, we can use an assign activity in both approaches for selecting and updating one or more tuples within a materialized data set covering a part of the *Tuple IUD Pattern*. In contrast to BIS, Oracle SOA Suite provides an additional proprietary XPath function for covering the complete *Tuple IUD Pattern* at an abstract level. Both vendors provide no specific functionality for iterating over a materialized data set (*Sequential Set Access Pattern*) as well as for synchronizing internal and external data sets (*Synchronization Pattern*). In both cases, the missing functionality has to be realized using appropriate workarounds.

In WF the processing of internal data is currently only possible through user-specific code based on ADO.NET. But WF provides the possibility to augment its activity library through user-specific functionality. This way, SQL inline support is achievable that confirms to the two-level programming model that is essential for simplifying the modeling of data management tasks in processes.

VII. SUMMARY

Driven by market requests, many vendors have recently shipped products that at a technical level combine a service-oriented framework with workflow and data management capabilities. In this paper, we addressed the state-of-the-art with special emphasis on SQL support in workflow products. We further concentrated on a tight integration approach and defined a set of essential properties including data management patterns that are all needed to provide for a comprehensive integration of SQL functionality into workflows. We described the different techniques to integrate SQL functionality at the process level as well as to provide for a comprehensive data management, i.e., data transport between external data and workflow-internal data.

REFERENCES

[1] M. Vrhovnik, H. Schwarz, O. Suhre, B. Mitschang, V. Markl, A. Maier, and T. Kraft, "An Approach to Optimize Data Processing in Business Processes," in *Proc. VLDB Conference*, 2007.

[2] M. Vrhovnik, O. Suhre, S. Ewen, and H. Schwarz, "PGM/F: A Framework for the Optimization of Data Processing in Business Processes," in *Proc. ICDE Conference*, 2008.

[3] OASIS, "Web Services Business Process Execution Language Version 2.0. Committee Draft," 2005.

[4] IBM. WebSphere Business Process Management. [Online]. Available: http://www-306.ibm.com/software/info/bpmsoa, last accessed: 07-11-22

[5] Oracle. Oracle SOA Suite. [Online]. Available: http://www.oracle.com/technologies/soa, last accessed: 2007-11-22

[6] BEA. BEA Aqualogic BPM Suite. [Online]. Available: http://www.bea.com, last accessed: 2007-11-22

[7] Microsoft. Microsoft BizTalk Server. [Online]. Available: http://www.microsoft.com/biztalk, last accessed: 2007-11-22

[8] S. Thatte, "XLANG: Web Services For Business Process Design," 2001.

[9] Microsoft. Windows Workflow Foundation. [Online]. Available: http://wf.netfx3.com, last accessed: 2007-11-22

[10] ——. .NET 3.0 Framework. [Online]. Available: http://www.netfx3.com, last accessed: 2007-11-22

[11] IBM. WebSphere Integration Developer. [Online]. Available: http://www-306.ibm.com/software/integration/wid, last accessed: 2007-11-22

[12] ——. WebSphere Process Server. [Online]. Available: http://www-306.ibm.com/software/integration/wps, last accessed: 2007-11-22

[13] W3C. XML Path Language (XPath): Version 1.0. [Online]. Available: http://www.w3.org/TR/xpath, last accessed: 2007-11-22

[14] M. Blow, Y. Goland, M. Kloppmann, F. Leymann, G. Pfau, D. Roller, and M. Rowley, "BPELJ: BPEL for Java," March 2004.

[15] Microsoft. ADO.NET. [Online]. Available: http://msdn2.microsoft.com/en-us/data, last accessed: 2007-11-22

[16] Oracle. Oracle BPEL Process Manager. [Online]. Available: http://www.oracle.com/technology/products/ias/bpel, last accessed: 2007-11-22

[17] Eclipse Foundation. Eclipse Development Platform. [Online]. Available: http://www.eclipse.org, last accessed: 2007-11-22

[18] Oracle. Oracle JDeveloper. [Online]. Available: http://www.oracle.com/technology/software/products/jdev, last accessed: 2007-11-22

[19] L. Ashdown, "Oracle XML Developer's Kit," September 2007.

[20] W3C. XSLT Extensible Stylesheet Language for Transformations. [Online]. Available: http://www.w3.org/TR/xslt, last accessed: 07-11-22