QUERY BUILDING USING SCHEMA

Applicant: Microsoft Corporation, Redmond, WA (US)

Inventors: Brad Olenick, Redmond, WA (US); Leon Ezequiel Welicki, Issaquah, WA (US); Timothy Michael McBride, Snohomish, WA (US); Federico Silva Armas, Seattle, WA (US); Jonathan Lucero, Bellevue, WA (US); Natisha Bhojajwala, Seattle, WA (US); David Anson, Snoqualmie, WA (US); Kristofer John Owens, Seattle, WA (US); Andrew Birch, Seattle, WA (US); Vishal R. Joshi, Redmond, WA (US); Jon Harris, Sammamish, WA (US); Stephen Michael Danton, Seattle, WA (US); Karandeep Singh Anand, Redmond, WA (US); Bradley D. Millington, Bellevue, WA (US); Adam Mohamed Abdelhamed, Bellevue, WA (US); Justin Beckwith, Bellevue, WA (US); Eric Hwa-Wei Wong, Bellevue, WA (US)

Appl. No.: 14/231,880
Filed: Apr. 1, 2014

Related U.S. Application Data

Provisional application No. 61/905,128, filed on Nov. 15, 2013, provisional application No. 61/884,743, filed on Sep. 30, 2013, provisional application No. 61/905,111, filed on Nov. 15, 2013, provisional application No. 61/905,243, filed on Nov. 17, 2013, provisional application No. 61/905,114, filed on Nov. 15, 2013, provisional application No. 61/905,116, filed on Nov. 15, 2013, provisional application No. 61/905,129, filed on Nov. 15, 2013, provisional application No. 61/905,105, filed on Nov. 15, 2013, provisional application No. 61/905,247, filed on Nov. 17, 2013, provisional application No. 61/905,101, filed on Nov. 15, 2013, provisional application No. 61/905,119, filed on Nov. 15, 2013.

Publication Classification

Int. Cl. G06F 17/30 (2006.01)
U.S. Cl.
CPC G06F 17/30654 (2013.01)
USPC 707/766

ABSTRACT

A query building mechanism in which a query builder component assists a user in generating queries to be used to populate user interface elements. A control provides a query schema to the query builder component. The query schema identifies available query parameters that the control is capable of using to populate a corresponding user interface element. The query builder component presents at least some of the available query parameters the user, such that the user may edit these parameters. The query builder component gathers those edits and generates a query therefrom. The query parameters may be provided to the query builder component in consistent manner across a wide variety of possible user interface controls. Furthermore, the resulting query is provided in a uniform query format.
Figure 4

Extension A
Transform AST To OData Using Built-In OData Helper

Extension B
Transform AST To A Proprietary URI Using Ad-Hoc Logic

Extension C
Transform AST To A SQL Sentence To Retrieve The Data

Figure 5
Figure 6

Figure 7
Figure 8
Figure 9
QUERY BUILDING USING SCHEMA

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application claims the benefit of each of the following provisional patent applications, and each of the following provisional patent applications are incorporated herein by reference in their entirety:

[0006] 5. U.S. Provisional Application Ser. No. 61/905, 114, filed Nov. 15, 2013;
[0008] 7. U.S. Provisional Application Ser. No. 61/905, 105, filed Nov. 15, 2013;
[0011] 10. U.S. Provisional Application Ser. No. 61/905, 128, filed Nov. 15, 2013; and

BACKGROUND

[0013] Computing systems and networks have transformed the way we work, play, communicate. Information can now be accessed from remote locations, often with the click of a button. Data stores are often now quite larger, and may be queried for desired information. For instance, a user interface control may generate a query, issue the query to the data store, receive results from the data store, and display some or all of the results in a corresponding user interface control. Nevertheless, conventional queries can require some knowledge to build, and often such knowledge is not possessed by the typical user of the computing system. Furthermore, even for those familiar with a query language, generating queries may be quite time consuming and mentally taxing.

SUMMARY

[0014] At least some embodiments described herein relate to a query building mechanism in which a query builder component assists a user in generating queries to be used to populate user interface elements. A user interface control itself issues the query with appropriate query parameters and populates the corresponding user interface element with the appropriate query results. However, the query builder component helps select the appropriate query parameters for the user interface control.

[0015] The user interface control provides a query schema to the query builder component. For instance, this might be performed at the request of a user that is viewing a user interface element populated by the user interface control. The query schema identifies available query parameters that the user interface control is capable of using to populate a corresponding user interface element. The query builder component presents at least some of the available query parameters to the user, such that the user may edit these parameters. As examples, the query parameters may include query fields, and query rules. The query builder component gathers those edits and generates a query therefrom.

[0016] In some embodiments, the query schemas are provided to the query builder component in consistent manner across a wide variety of possible user interface controls. Furthermore, the resulting query is provided in a uniform query format. Accordingly, the query builder component may be used for any user interface control, regardless of whether user interface control is provided by a system, or by an application that runs in the system. Additionally, the user interface elements may be presented from a single class. Accordingly, the user interface element may appear consistent regardless of the identity of the user interface control, with customization occurring in the query parameters themselves.

[0017] In some embodiments, the user might be presented with presentation options for presenting the user interface element, but the presentation options may also be limited to options that would allow the user interface to retain a certain look and feel. Accordingly, the principles described herein may permit convenient generation of queries, and repopulation of user interface elements with different data, regardless of these being many different sources for the user interface controls.

[0018] This summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0019] In order to describe the manner in which the above-recited and other advantages and features of the invention can be obtained, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments thereof which are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings in which:

[0020] FIG. 1 abstractly illustrates an example computing system in which the principles described herein may be employed;
[0021] FIG. 2 illustrates an environment that shows collaborating elements that facilitate a query building capability;
[0022] FIG. 3 illustrates another example illustrating more specific example operations of the user interface control, the query builder component, and the query builder interface;
[0023] FIG. 4 illustrates a process whereby an abstract representation of a query may be transformed by different extensions into different query formats for submission to the underlying data source;
[0024] FIG. 5 illustrates an example of how the query builder component may be created via a user gesture;
[0025] FIG. 6 illustrates an example query builder user interface in the case of a locked query;
[0026] FIG. 7 illustrates an example query builder user interface in the case of an unlocked query;
FIG. 8 illustrates a query builder user interface that may be provided for purposes of changing presentation options for the user interface element; and

FIG. 9 illustrates a user interface in which multiple data views (one for each query) are offered for a user interface element, and in which the user is given the option of generating further views using the query builder component.

DETAILED DESCRIPTION

At least some embodiments described herein relate to a query building mechanism in which a query builder component assists a user in generating queries to be used to populate user interface elements. A user interface control itself issues the query with appropriate query parameters and populates the corresponding user interface element with the appropriate query results. However, the query builder component helps select the appropriate query parameters for the user interface control.

The user interface control provides a query schema to the query builder component. For instance, this might be performed at the request of a user that is viewing a user interface element populated by the user interface control. The query schema identifies available query parameters that the user interface control is capable of using to populate a corresponding user interface element. The query builder component presents at least some of the available query parameters to the user, such that the user may edit these parameters. As examples, the query parameters may include query fields, and query rules. The query builder component gathers those edits and generates a query therefrom.

In some embodiments, the query schemas are provided to the query builder component in consistent manner across a wide variety of possible user interface controls. Furthermore, the resulting query is provided in a uniform query format. Accordingly, the query builder component may be used for any user interface control, regardless of whether user interface control is provided by a system, or by an application that runs in the system—so long as the user interface control communicates the schema in the right manner, and is able to interpret the unified query format. Additionally, the user interface elements may be presented from a single class. Accordingly, the user interface element may appear consistent regardless of the identity of the user interface control, with customization occurring in the query parameters themselves.

In some embodiments, the user might be presented with presentation options for presenting the user interface element, but the presentation options may also be limited to options that would allow the user interface to retain a certain look and feel. Accordingly, the principles described herein may permit convenient generation of queries, and repopulation of user interface elements with different data, regardless of there being many different sources for the user interface controls.

Thus, the principles described herein may enable a distributed and composite application in which the user may control information that they see, and otherwise customize the user interface to fit their needs. For instance, for data sets that are projected through different visual experiences (like grids, charts, summaries, and so forth), the user can change the parameters that result in getting data from the data source. Users do this through a user experience that is standardized over a system.

The principles described herein may be implemented using a computing system. For instance, the users may be engaging with a portal that displayed multiple user interface elements using a client computing system. The executable logic supporting the portal and providing visualizations thereon may also be performed using a computing system. The computing system may even be distributed. Accordingly, a brief description of a computing system will now be provided.

Computing systems are now increasingly taking a wide variety of forms. Computing systems may, for example, be hand held devices, appliances, laptop computers, desktop computers, mainframes, distributed computing systems, or even devices that have not conventionally been considered a computing system. In this description and in the claims, the term “computing system” is defined broadly as including any device or system (or combination thereof) that includes at least one physical and tangible processor, and a physical and tangible memory capable of having thereon computer-executable instructions that may be executed by the processor. The memory may take any form and may depend on the nature and form of the computing system. A computing system may be distributed over a network environment and may include multiple constituent computing systems.

As illustrated in FIG. 1, in its most basic configuration, a computing system 100 typically includes at least one processing unit 102 and memory 104. The memory 104 may be physical system memory, which may be volatile, non-volatile, or some combination of the two. The term “memory” may also be used herein to refer to non-volatile mass storage such as physical storage media. If the computing system is distributed, the processing, memory and/or storage capability may be distributed as well. As used herein, the term “executable module” or “executable component” can refer to software objects, routines, or methods that may be executed on the computing system. The different components, modules, engines, and services described herein may be implemented as objects or processes that execute on the computing system (e.g., as separate threads).

In the description that follows, embodiments are described with reference to acts that are performed by one or more computing systems. If such acts are implemented in software, one or more processors of the associated computing system that performs the act direct the operation of the computing system in response to having executed computer-executable instructions. For example, such computer-executable instructions may be embodied on one or more computer-readable media that form a computer program product. An example of such an operation involves the manipulation of data. The data-executable instructions (and the manipulated data) may be stored in the memory 104 of the computing system 100. Computing system 100 may also contain communication channels 108 that allow the computing system 100 to communicate with other message processors over, for example, network 110.

The computing system 100 also includes a display 112 on which a user interface, such as the user interfaces and portal described herein, may be rendered. Such user interfaces may be generated in computer hardware or other computer-represented form prior to rendering. The presentation and/or rendering of such user interfaces may be performed by the computing system 100 by having the processing unit(s) 102 execute one or more computer-executable instructions
that are embodied on one or more computer-readable media. Such computer-readable media may form all or a part of a computer program product.

[0039] Embodiments described herein may comprise or utilize a special purpose or general-purpose computer including computing hardware, such as, for example, one or more processors and system memory, as discussed in greater detail below. Embodiments described herein also include physical and other computer-readable media for carrying or storing computer-executable instructions and/or data structures. Such computer-readable media can be any available media that can be accessed by a general purpose or special purpose computer system. Computer-readable media that store computer-executable instructions are physical storage media. Computer-readable media that carry computer-executable instructions are transmission media. Thus, by way of example, and not limitation, embodiments of the invention can comprise at least two distinctly different kinds of computer-readable media: computer storage media and transmission media.

[0040] Computer storage media includes RAM, ROM, EEPROM, CD-ROM or other optical disk storage, magnetic disk storage or other magnetic storage devices, or any other tangible medium which can be used to store desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose computer.

[0041] A “network” is defined as one or more data links that enable the transport of electronic data between computer systems and/or modules and/or other electronic devices. When information is transferred or provided over a network or another communications connection (either wired, wireless, or a combination of hardwired or wireless) to a computer, the computer properly views the connection as a transmission medium. Transmissions media can include a network and/or data links which can be used to carry or desired program code means in the form of computer-executable instructions or data structures and which can be accessed by a general purpose or special purpose computer. Combinations of the above should also be included within the scope of computer-readable media.

[0042] Further, upon reaching various computer system components, program code means in the form of computer-executable instructions or data structures can be transferred automatically from transmission media to computer storage media (or vice versa). For example, computer-executable instructions or data structures received over a network or data link can be buffered in RAM within a network interface module (e.g., a “NIC”), and then eventually transferred to computer system RAM and/or to less volatile computer storage media at a computer system. Thus, it should be understood that computer storage media can be included in computer system components that also (or even primarily) utilize transmission media.

[0043] Computer-executable instructions comprise, for example, instructions and data which, when executed at a processor, cause a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions. The computer executable instructions may be, for example, binaries, intermediate format instructions such as assembly language, or even source code. Although the subject matter has been described in language specific to structural features and/or methodological acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the described features or acts described above. Rather, the described features and acts are disclosed as example forms of implementing the claims.

[0044] Those skilled in the art will appreciate that the invention may be practiced in network computing environments with many types of computer system configurations, including, personal computers, desktop computers, laptop computers, message processors, hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, mobile telephones, PDAs, pagers, routers, switches, and the like. The invention may also be practiced in distributed system environments where local and remote computer systems, which are linked (either by hardwired data links, wireless data links, or by a combination of hardwired and wireless data links) through a network, both perform tasks. In a distributed system environment, program modules may be located in both local and remote memory storage devices.

[0045] The system offers a user interface (or a “portal”) that displays multiple user interface elements that may also be called herein “parts”. The part acts as a container for the results of execution of one or more underlying user interface controls (also called herein simply “controls”). The manner of display of the part may be customized, as may be the ordering and composition of the part with respect to other user interface elements.

[0046] The execution of the controls often involves one or more underlying queries to retrieve information that is displayed in the part, or otherwise processed. In accordance with the principles described herein, the user may modify the underlying query that retrieves the information. This allows the users to further control the user experience in fundamental ways, such as defining the actual data that is being displayed. In accordance with the principles described herein, this capability is offered in a consistent way, which honors the constraints of the portal to provide a portal in which many contributors may offer parts without sacrificing a consistent look and feel of the portal. Accordingly, the query customizations described herein may be applied to any part, regardless of the contributor of the part.

[0047] The principles described herein allow the user to filter different sets of displayed data, and do further queries on their data sources. Even though the data sources are different, depending on the part, the representation of the query may share a common visual representation. For instance the common visual representation may be a set of conditions connected by logical connecters and, eventually, groupings.

[0048] The query building capability includes a query builder user interface element that is displayed to a user and represents a user interface for authoring queries over data sources. The query building capability also includes a query building component that receives query schemas from the user interface control, manifests at least some of the query schemas via the query builder user interface element, and then builds a formal representation of a query based on user edits conducted through user interaction with the query builder user interface element.

[0049] FIG. 2 illustrates an environment 200 that shows collaborating elements that facilitate a query building capability. The environment 200 includes a user interface control 210, a query builder component 220, a query builder user interface 230, and a user interface element 240. The user
interface control 210 is queriable in that it may generate a submit queries, and populate the corresponding user interface element 240 with at least a portion of the query results.

[0050] A query builder component 220 exposes an input interface whereby the user interface control 210 may provide a query schema 201 to the query builder component (reference arrow 211). The query schema 201 identifies available parameters for composing queries that the corresponding user interface control 210 is capable of using to populate the corresponding user interface element 240. The query builder component 220 then proceeds to build a query corresponding to the schema. Although only one user interface control 210 is illustrated in FIG. 2, the query builder component 220 may proceed to build a distinct query in response to each of multiple schemas received various different times at the query builder component 220. Furthermore, even for a single user interface control, the query builder component may operate as described herein to build multiple queries, each selectable to provide a different view of the user interface element.

[0051] The query builder component 220 builds queries by generating a query builder user interface 230 (reference arrow 212). The query builder user interface 230 is built from a particular query builder user interface class and thus there is some similarity in appearance each time the query builder component 230 generates a query builder user interface. However, the query builder user interface 230 may be customized to the corresponding user interface control 210 by presenting at least some of the available query parameters to the user. In the case of query builder user interface 230, a name field, an age field, and a newness status field are presented to the user. At this point, the user may interact with the query builder user interface 230 to edit the query parameters.

[0052] At some point, the query builder component 220 determines that a query is to be built based on the edited data in the query builder user interface 230 (reference arrow 213). The query builder component 220 then generates a query using the then current values of the available query parameters. The query is generated in a pre-existing format recognized by the user interface control 210. For instance, in FIG. 2, the user may select the Save button in the query builder user interface 230, causing the query builder component 220 to generate an Abstract Syntax Tree (AST) representation 202 of the query. The Abstract Syntax Tree representation is an example of a uniform query format that the user interface element 210 may then access (reference arrow 214) from an output interface of the query builder component 220. The user interface control 210 then performs the query and provides the resulting data for presentation in the corresponding user interface element 240 (reference arrow 215).

[0053] FIG. 3 illustrates more specific example operations of the user interface control 310, the query builder component 320, and the query builder user interface 330, which may be examples of the respective elements 210, 220 and 230 of FIG. 2. The “Control UX” represents an example query builder user interface. The query builder user interface 330 presents a user experience for authoring queries, the user experience being data driven from the schema provided by the part. The query builder user interface 330 presents at least some of the available query parameters to a user.

[0054] For instance, in this case, the query builder user interface 330 shows date range fields, an “assigned to” field, a state field, and tags fields. Those fields may be defined by the query schema provided by the user interface control 310. Likewise, those queries are grouped in a certain way using logical operators. Such groupings may also be defined by the query schema provided by the user interface control 310. Thus, the schema defines available fields and rules for composing a query, and the query building user interface presents at least some of those fields and rules to a user for editing.

[0055] A visual query created by a user results in an abstract query representation based on a portal-defined Abstract Syntax Tree (AST). This Abstract Syntax Tree is created by the query builder component 320, and represents the predicates of the query. The query is then passed to the user interface control 310 (i.e., the data layer or the owner of the data). The AST format may be uniform across the system so that all data sources and schemas will project the same user experience (e.g., display the same query builder control albeit driven by different data and schemas) and the same predicated definition (e.g., via the AST).

[0056] The user interface control (or some other portion of the data layer) receives the AST and translates to their query format of choice. For instance, intrinsic parts may be offered by the system itself and have a standard component for execution abstract representations of a predicate in a certain format (e.g., OData). On the other hand, an application provider may have provided a set of parts for execution within the system. The application provider also includes a component for executing abstract representations of a predicate, but perhaps in a different format than the system. The data layer may thus translate the query format to an abstract representation of choice. The results are returned to the user interface element (or part) that projects the resulting data set on the screen.

[0057] There is a high level of independence between each of the three layers illustrated in FIG. 3. A primary connecting piece is the abstract predicate representation (AST with the query). Multiple user experiences can consume an instance of a predicate following this schema. Predicates can be later translated into different types of queries, depending on the target data source. For instance, the tree can be translated to OData predicates, SQL queries, and so forth.

[0058] Once the general tree representation is established, the available components (or fields) of the query are metadata-driven. By doing this, different configurations of predicates can be produced using the same stack (e.g. queries for bugs, web sites, team members, etc.).

[0059] The query builder component is data-driven and results in a consistent experience regardless of which entity generated the part. Any application can provide a schema associated with any of their parts and as a result of getting a query builder component, the application will get an updated query based on user input.

[0060] In some embodiments, the query building may not be made available to all parts. To be queriable, a part implements a certain contract that indicates that it can be queried. The contract exposes a query schema (that governs the user experience of the query builder) and receives back an AST with the query entered by the user. For instance, FIG. 2 shows the implementation of the contract with the queriable user interface control 210 on the left offering (211) a schema (201). That schema drives the query builder user interface 230 shown on the right, in the manner previously described. The query builder control generates a corresponding AST, as previously described, and provides the AST with the query back to the user via the queriable part.

[0061] Query building is a data-driven task where the user experience is controlled by the query builder component.
Users provide their schema and query instance, and in return receive an updated query (described using a common predicate structure).

As mentioned above, the AST that represents the query is a uniform abstraction in the system. Once an application (often referred to as an "extension" herein) receives a query, it is the application’s responsibility to get the AST (the query representation) and transform it to a query to their underlying data repository. By default the system provides some converters to popular formats, but applications parse the AST and issue the query to the backend. The query builder itself does not run the query.

For instance, referring to FIG. 4, the uniform abstraction 108 of the AST query is represented on the left. The uniform abstraction may be provided to any number of different extensions (e.g., Extension A through C on the right in FIG. 3). The extension then transforms the AST to an appropriate query format for its backend. For instance, as illustrated in FIG. 4, extension A transforms the uniform AST query into an OD data query using an OD data helper provided by the portal, extension B transforms the uniform AST query into a proprietary Uniform Resource Identifier (URI) using their proprietary logic, and extension C transforms the uniform AST query into a SQL sentence. The extensions themselves then submit the appropriately transformed query to their back end data. Alternatively, given the abstract AST representation, the extension could also decide not to use the query builder control and generating their own UI (built out of our existing controls, so they still have user experience consistency). In this case, the same contracts apply (metadata+AST). This allows for extensibility and is an extensibility point.

The query schema identifies the fields (which includes name, type, operations, and so forth) available to construct the query. These query parameters are passed as input to the query builder component, which then drives the query builder user interface. The query instance is the actual query entered by the user. This parameter is passed as input to the query builder component to show the current query and, the query is passed back to the user interface element once the query is modified by the user.

The query builder user interface can be initiated by a gesture. FIG. 5 shows an example where that gesture is a right-click command, but that is just one of an infinitely variety of gestures that might be chosen. In the illustrated example of FIG. 5, by selecting the “edit data” option, the query builder is displayed. This option might only be available for user interface controls that honor the query building contract that allows the part to be queriable as previously described.

Building queries is a complex task and building them from scratch may not be an option for all users and all scenarios. Accordingly, in some embodiments, the queries generated by the query builder component can have two modes; namely, locked and unlocked.

In locked mode, the fields displayed in the query builder are the same. The user can enter the values for the fields but cannot alter the composition of the query. FIG. 6 illustrates an example query builder user interface in the case of a locked query.

In unlocked mode, the user can compose queries as desired. Users can add, remove, change filters, group filters, change operators, change logical connectors, and so forth.

FIG. 7 illustrates an example query builder user interface in the case of an unlocked query.

The query builder control is not limited to altering data, but can also help to drive how the data is presented. FIG. 8 illustrates a query builder user interface that may be provided for this purpose.

The implementation is similar to the query building already described. The query builder control receives a schema with all the available presentation options. To preserve a consistent look and feel of the portion, the presentation options provided by the schema may be constrained to what is supported by the system. After the user inputs their desired selection, the selection is returned back to the originating part to be applied. The query builder control does not have any control over how data is presented in the actual part, but just presents available options and collects preferences from the user.

Queries created with the query builder control can be saved and associated with the originator. By doing this, a user can have multiple data or presentation views that can be associated to a given part. FIG. 9 illustrates a user interface in which multiple data views are offered for a part, and in which the user is given the option of generating further views (using the query builder control).

Accordingly, the principles described herein provide a convenient manner for allowing users to control how data is being presented in a user interface element. Furthermore, such query building may be performed in a consistent manner regardless of whether the user interface element is an intrinsic or extrinsic user interface element. Furthermore, since any user interface element that is queriable may use the query building component, so long as the user interface element abides by the contract with the query builder component, this query building may be enabled in a composable system in which there are many contributors of user interface elements.

The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

What is claimed is:

1. A computer program product comprising one or more computer-readable storage media having thereon computer-executable instructions that are structured such that, when executed by one or more processors of a computing system, cause the computing system to instantiate and/or operate the following:

   a query builder component that is configured to read schemas provided by controls associated with user interface elements, the schemas including a given schema provided by a given control identifying available query parameters that the given control is capable of using to populate a corresponding user interface element,

   the query builder user interface configured to present at least some of the available query parameters to a user for editing by a user, and configured to generate a query based on the edits provided by the user.

2. The computer program product in accordance with claim 1, the schema also describing presentation options for results of the query, the query builder user interface configured to
present at least some of the presentation options to the user, and configured to provide user selection of the presentation options back to the given control.

3. The computer program product in accordance with claim 2, the given control using the selected presentation options in order to present the user interface element.

4. The computer program product in accordance with claim 1, the query builder user interface further configured to provide the generated query in a uniform format.

5. The computer program product in accordance with claim 4, wherein the computer-executable instructions are further structured such that, when executed by the one or more processors of the computing system, cause the computing system to instantiate and/or operate the following:
   a first query transforming component that transforms queries in the uniform form to a first specific format.

6. The computer program product in accordance with claim 5, wherein the computer-executable instructions are further structured such that, when executed by the one or more processors of the computing system, cause the computing system to instantiate and/or operate the following:
   a second query transforming component that transforms queries in the uniform form to a second specific format.

7. The computer program product in accordance with claim 1, wherein the computer-executable instructions are further structured such that, when executed by the one or more processors of the computing system, cause the computing system to instantiate and/or operate the following:
   a first control capable of running queries to populate a first user interface element and that offers a first schema identifying available parameters for composing the query.

8. The computer program product in accordance with claim 7, wherein the computer-executable instructions are further structured such that, when executed by the one or more processors of the computing system, cause the computing system to instantiate and/or operate the following:
   a second control capable of running queries to populate a second user interface element and that offers a second schema identifying available query parameters for composing queries.

9. The computer program product in accordance with claim 8, the first control provided by system provider that provides a system, and the second control provided by an application developer of an application that runs in the system.

10. A method for constructing a query using a query builder component, the method comprising:
    an act of the query builder component receiving a schema from a user interface control, the schema identifying available parameters for composing queries that the user interface control is capable of using to populate a user interface element corresponding to the user interface control;
    an act of the query builder component generating a query builder user interface that is customized to the user interface control by presenting at least some of the available parameters to a user;
    an act of the query builder component determining that a query is to be constructed based on current values of the available parameters displayed in the query builder user interface; and
    an act of the query builder component providing the query on an output interface that is accessible by the user interface component.

11. The method in accordance with claim 10, the schema being a first schema, the user interface control being a first user interface control, the available parameters being first available parameters, the query being a first query, the user interface element being a first user interface element, the query builder user interface being a first query builder user interface, the method further comprising:
    an act of the query builder component receiving a second schema from a second user interface control, the second schema identifying second available parameters for composing queries that the second user interface control is capable of using to populate a second user interface element corresponding to the second user interface control;
    an act of the query builder component generating a second query builder user interface that is customized to the second user interface control by presenting at least some of the second available parameters to a user;
    an act of the query builder component determining that a second query is to be constructed based on current values of the second available parameters displayed in the second query builder user interface; and
    an act of the query builder component providing the second query on the output interface that is also accessible by the second user interface component.

12. A method in accordance with claim 11, the first and second query builder user interface being of the same class.

13. A method in accordance with claim 12, the first and second query being of the same uniform format.

14. A method in accordance with claim 11, further comprising:
    an act of transforming the first query from the uniform format to a first specific format; and
    an act of transforming the second query from the uniform format to a second specific format.

15. A method in accordance with claim 10, the first user interface control being provided by system provider that provides a system, and the second user interface control provided by an application developer of an application that runs in the system.

16. A method in accordance with claim 10, the first user interface control being provided by system an application developer of a first application that runs in the system, the second user interface control being provided by an application developer of a second application that runs in the system.

17. A method in accordance with claim 10, the query builder component allowing query fields to be edited, but not allowing the structure of the query to be altered.

18. A method in accordance with claim 10, the query builder component the structure of the query to be altered.

19. A method for constructing queries using a query builder component, the method comprising:
    an act of the query builder component exposing an input interface through which a plurality of user interface controls may each provide a corresponding schema identifying available parameters for composing queries that the corresponding user interface control is capable of using to populate a corresponding user interface element;
    for each of at least some of the user interface controls that provide schemas to the query builder component via the input interface, an act of generating a query corresponding to the schema by performing the following:
an act of the query builder component generating a query builder user interface that is of the same class as when generating any of the queries for any of the at least some of the user interface controls, the query builder user interface being customized to the corresponding user interface control by presenting at least some of the available parameters to a user;
an act of the query builder component determining that a query is to be constructed based on current values of the available parameters displayed in the query builder user interface;
an act of the query builder component generating a query using the current values of the available parameters, the query being in a predetermined format that is of the same format when generating any of the queries for any of the at least some of the user interface controls; and
an act of the query builder component providing the generated query on an output interface that is accessible by the user interface component.

20. A method in accordance with claim 19, further comprising the following for each of at least some of the user interface controls that provide schemas to the query builder component via the input interface:
an act of generating multiple query corresponding to the schema provided by a particular user interface control; and
an act of saving the multiple queries in a manner associated with the particular user interface control.

* * * * *