

Declarix™ Information Management System

**The Next Evolution in Relational Database
Management Systems**

*A Declarix White Paper
December 2007*

The Next Evolution in Relational Database Management Systems

Abstract

This paper describes Declarix, a new declarative meta language programming technology for rapidly developing rich, robust web applications. Declarix 1.0 is discussed here to illustrate the power of the underlying metadata technology that will power Declarix 2.0, which will be commercially available in early 2008 as a complete web hosted development and deployment environment. Declarix 1.0 has been in limited use for over a year and has been successfully used to develop several industrial strength commercial web applications.

Introduction

Relational Database Management Systems (RDBMS) have been in commercial use now for over 30 years. They have been widely accepted as the definitive way of managing persistence storage of practically every type of corporate data. In fact, RDBMSs are such a fundamental core technology of enterprise applications that they have become synonymous with the use of the word database.

As such, programming of RDBMSs is fundamentally declarative in nature and, until the advent of triggered stored procedures and vendor specific tools like PL/SQL, they lacked any notion of procedural specification. RDBMSs have standardized on the Structured Query Language (SQL), which is a set-based, declarative programming language, not an imperative language such as Java, C, or Basic. SQL is used to declare the structure and layout of data as a set of tables, and to declare the form of creating, retrieving, updating and deleting that data. Along with declaring the type of data in table columns, all commercial implementations of SQL also allow some form of declaration for keys, constraints and indexes on those table columns, even though these are not standardized across vendors.

A RDBMS is essentially a software engine that interprets and executes the declarative SQL statements. The evolution of RDBMS technology has been geared to the configurability of performance and scalability characteristics, i.e., tuning for faster access to increasingly larger data sets. This evolution has been extremely valuable and fundamental to the widespread acceptance and ubiquity of RDBMSs. However, RDBMSs haven't gotten smarter about the data they manage. Any intelligence about the content of the data, in particular the relationships between the data entities, is left for software developers to understand and represent in explicit and tedious detail within SQL. The RDBMS has very little idea about how tables are related let alone how to exploit that knowledge to provide performant access, drill down, rollups and, in general, any high level data management constructs.

Furthermore, in today's web application domain, knowledge of how data is related and how it's turned into information is sprayed across multiple files; across multiple disparate technologies such as Java EJBs, JSPs, C#, .net, ASPs, SQL scripts, and javascript; and across disparate domain experts such as business analysts, DBAs, programmers, logical modelers, and so on.

Now, given the ever increasing complexity of enterprise applications and, particularly, the increasing demand for accessing data via rich web clients, something new is needed to reduce complexity and move the productivity curve forward. To that end a new product has emerged called **Declarix**, which takes RDBMS technology to the next level of intelligent declarative programming with an order of magnitude increase in productivity. **Declarix** defines a new query language called IQL™ (Information Query Language) that lets data relationships be described with logical information content along with how to intelligently view and interact with the information.

Declarix brings information centric application development much closer to the business problem with a productivity and cost profile that makes any consideration of off shore development totally uncompetitive.

What is Declarix?

Declarix is an Information Management System (IMS) and not a data management system. What does this mean exactly? It means that **Declarix**, through a rich meta language, combines business intelligence within structural data descriptions to produce an information model along with descriptions of how that model is to be viewed and interacted with. In other words, the **Declarix** meta language succinctly describes a complete business application with a very rich and dynamic web client.

There are two parts to **Declarix**:

1. An Information Query Language called IQL – a high level, declarative meta language used to capture in one place all the diverse pieces of a business application.
2. An Information Management System - an engine that works with a RDBMS system and a Web Server to execute IQL directives. In the same way that a RDBMS engine interprets and executes SQL, the **Declarix** IMS engine interprets and executes the IQL.

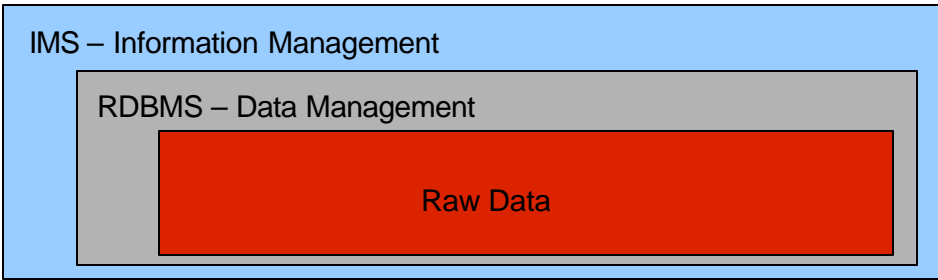


Figure 1: The **Declarix** IMS engine works in conjunction with an existing RDBMS to add value and insight to data, i.e., to turn data into information, which lets the IMS make decisions on how to most effectively manage that information.

The power of **Declarix** starts with its fully expressive IQL, which is able to capture all aspect and dimensions of an application in a single place. That is, IQL can express all DDL, business rules, reports, charts, user interface components such as forms, tables, trees, and workflows. Unlike other systems that store metadata in a mixture of tables and configuration files, Declarix represents all the aspects of describing an application in a single metadata language file written in IQL. This doesn't mean that Declarix requires an application be maintained in a single file. Declarix provides the notion of a name spaces which let applications be constructed from multiple, logically related IQL files.

IQL has significant ease-of-use benefits compared to SQL, even if it's only used as a data query language. With such features as auto-joining and support for collection attributes, IQL queries are often half the length of the equivalent SQL queries. Where SQL is used to retrieve and update raw data, IQL is used to not only retrieve and update data, but also to process data and present it in forms, reports, charts and workflows.

With an application described in IQL, the Declarix IMS automatically manages all dimensions of a web application. Some of the most significant areas are:

Database Administration - Declarix automatically manages many aspects of database administration, saving significant developer and DBA time by providing:

- Generation of the database schema
- Creation of table indices.
- Generation of dynamic, optimized SQL.
- Execution of precise SQL queries
- Management of column, table, and database constraints.
- Out-of-the-box management of multi-tenant data models.
- Creation and management OLAP Data Warehouse schemas.

Full Featured Forms - Declarix provides a rich client user interface for applications by:

- Automatic generation of sophisticated data entry forms (IQL allows full access to detailed form layout if needed).
- Automatic, on-the-fly, update of calculated fields requiring no web page refreshes.
- Access to a pallet of rich data entry controls, including pop-up calendar controls, auto-complete select boxes, radio button groups, drop down lists, too-from lists, and many more.
- Generation of Query-by-Example (QBE) search forms.
- Attachment of fully featured forums at the record or table level.

Analysis & Reporting - Declarix provides complete reporting capabilities, including:

- Dynamic report generation.
- Table, crosstab, and tree views.
- Math expression and formulas.
- Intelligent auto-aggregation.
- Drilling and pivoting.
- Paging and sorting.
- Intelligent charting.
- Exports to XML and Microsoft Excel™ files (complete with formulas).
- Hot-links.

Content Management - Declarix provides complete content management capabilities with broad support for:

- Storage and retrieval of any type of content
- Full content search of most popular file formats, including PDF, Microsoft Word™, and so on
- Business Rules and workflow, including:
 - Constraint rules and validation rules (domain, column, table, and database constraints).
 - Trigger rules
 - Inference rules
 - Navigation rules

Robust Security - Declarix provides both authentication as well as both data and functional authorization capabilities at the user, role, organization, and group level. In IQL *actors* are given *permissions* to structures and data by:

- Assignments to *resources* (data or user interface elements),
- rules specified by *grants*, or
- the collaborative *sharing* system (sharing access to other users).

Where an actor is one of the following: a person, an organization, a group, or a role. When a user logs into a Declarix application, they receive a superset of the permissions assigned to them, specifically, to any groups they belong to, to the organization they belong to, or to any role that they have.

Versioning and Audit Trails - Declarix optionally provides full version and audit tracking of all database changes -- including what changed, what it changed from, when it changed, and who changed it.

Help & Documentation - Declarix provides automatic context-sensitive help on forms, reports, and user interface controls based on descriptive elements contained within IQL. In addition, it automatically generates documentation that describes the basis of any computed values displayed in the user interface.

Declarix is an extensible platform based on open industry standard development technologies and standard development processes. In fact, IQL is easily extended through registration of Java classes and methods or by directly inserting JavaScript or HTML/DHTML. Corporate users are also able to easily adapt their application look and feel to meet their own standards by using IQL support for CSS style sheets.

Furthermore, Declarix applications are easily integrated with legacy systems and more traditional development technologies such as popular Java J2EE application servers. For example, Declarix provides APIs for directly accessing lower level IMS functionality from Java, JSPs, Servlets, EJBs or SOA architectures. Since Declarix itself is written in Java, it can be deployed on any operating system that supports Java - Microsoft Windows, Linux, Solaris, and OS X. In addition, it supports any SQL92-compliant RDBMS, including Oracle, DB2, Microsoft SQL Server, or MySQL. For rich user interfaces, Declarix generates to the Adobe Flex framework or it optionally generates strict standards-based HTML and XHTML.

Web 2.0

Declarix applications are inherently Web 2.0 compliant because it leverages the Adobe Flash client to support Rich Internet Applications (RIA). This provides a much more dynamic user experience. For example, blogs and forums are easily constructed or added to an application. Support for drag and drop, right click context menus, dynamic tree controls, auto-complete text boxes, and pop-up calendar controls are standard and automatic features of Declarix. With Declarix 2.0 a rich styling, layout and presentation technology is provided with an online Designer/View editor and Stylist/Theme editor.

Why do we need Declarix?

Declarix is able to dynamically create a customized user experience based upon the user's application rights, the organization they belong to, and their own user-based privileges, groups, and preferences. Declarix has a comprehensive style and security architecture that allows multiple applications, organizations and users to exist within the same instance of Declarix. For example, different applications can share different fields from the same table. This eliminates redundancy and ensures that different information systems are automatically synchronized.

Support for multi-tenancy is an out-of-the-box feature of Declarix. Portions of an application are able to be bound on an organization basis. For example, a generic medical application that caters to a large number of organizations but also has certain tables, fields and business rules that only apply to one or a few organizations. This allows easy organization-specific customization of an application while still leaving the larger portion of it generic and reusable.

Declarix dynamically modifies forms and reports based on users, roles, workgroups, and organizations. For example, if an HR support representative doesn't have access to a salary field or salary history table, these fields and tables are automatically removed from any forms and reports that this user can see. Also, if a particular user or organization doesn't have access or use for a field, then the data is not even queried. In other words, Declarix only queries a column or table if it is absolutely required to handle the particular scenario. This is in sharp contrast to typical business applications today, where business objects are created at various levels of coarseness and then particular business requirements are mapped to the business object that comes closest to meeting the business need. This approach almost always leads to querying more data than is required, which leads to slower and less responsive systems.

Declarix has a number of key technical advantages over other modern development platforms that are claiming major productivity enhancements, e.g., Ruby on Rails. Some of the features that are unique to Declarix are:

- Join Attributes - simple yet profoundly powerful expressions.
- Integrated OLAP - facilitates innovations like Rational Logic & Automatic Aggregation.
- Round-trip IQL - for complete computer-assisted RAD, in addition to Visual RAD.
- Powerful information model and database refactoring.
- Metadata Language and not just metadata Attributes.
- Dynamic Views - dynamic forms, reports, tables, ... based on security, roles, organizations.
- Rosetta Stone” Computations - IQL computations automatically translated into Java, JavaScript, Excel, and SQL on demand.
- Expressive Charting - powerful multi-plot charts created simply and with aggregation.
- Features Required of Any Modern Web Information Systems Environment
 - Web 2.0 Features - for more richly interactive web applications.
 - Forum functionality.
 - Social networking features.
 - Integration with other hosted web services (RSS, flickr and so on)
 - Database Interoperability
 - Operating System Interoperability

Developing a Web Applications with Declarix 1.0

A simple To-Do list web application will be built to illustrate the power of Declarix and the IQL language. This application was chosen as a comparison to Ruby on Rails, which claims a factor of ten in development productivity over traditional technologies such as Java or .Net (see the To-Do List demo described in “Four Days on Rails” compiled by John McCreesh at <http://www.rails4days.pwp.blueyonder.co.uk/Rails4Days.pdf>). Using Declarix for this simple web application results in another factor of ten reduction over Ruby on Rails in the amount and complexity of the coding effort.

Development Environment

Because Declarix is a hosted environment, there are no database or development tool downloads or setups to be done. With Declarix 1.0 all development is done on a server, which in this case is a Linux system (see Declarix 2.0 below). Each user of Declarix 1.0 is given a development sandbox area to process and maintain their IQL files. A command line tool, *dx*, is used to perform a number of administrative and development tasks in the user's Declarix environment. The commands used for the To-Do List application are:

```
dx readiql - to process an IQL file.  
dx recreate - to recreate the database from the current contents of the IQL file.  
dx writeiql - to the data stored in the database.  
dx stop - to stop the Declarix server.  
dx start - to start the Declarix server.
```

Declarix 2.0

Declarix 2.0 will provide a completely automated web based development environment where users can both develop & deploy directly on the web. This environment will include:

- Declarix Designer - used to define new applications, reports, forms, workflow, etc. in a visual WYSIWYG and interactive manner, requiring no programming skills. This includes a view editor along with a style and theme editor.
- Declarix Modeler - used to define, create, modify and maintain the underlying data model for an information system; it can evolve throughout an information system's life-span, from conception to deployment. This also includes a function/method code editor.

The To-Do List application will simply keep a list of typed items along with a due date, a priority and an optional note to augment the item. A simple data model might look like:

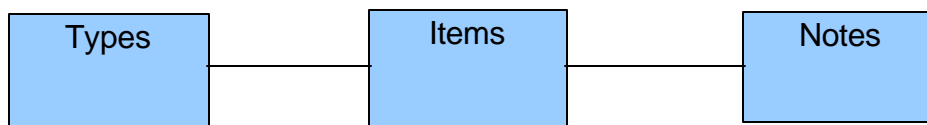


Figure 2: The ToDo List data model.

Here an Item belongs to one Type and, optionally has one Note. In the file named *todo.iql* the IQL meta language description of this model looks like:

```
CATEGORY todo ABBR td FILE

ENTITY type
  HAS name IS name NAME

ENTITY item AS "Items"
  HAS done IS boolean,
  priority IS int,
  description IS string,
  dueDate IS date NOTIME,
  private IS boolean,
  createdOn IS date DEFAULT TYPE DATE,
  updatedOn IS date,
  category IS LINK tdType.items,
  note IS LINK tdNote.items

ENTITY note AS "Notes"
  HAS content IS largeString,
  createdOn IS date,
  updatedOn IS date
```

Each Declarix application is developed within the context of one or more categories or name spaces, which isolate its data from other applications running on the same server. In this case the category is named *todo* and is simply abbreviated as *td*. Each entity in an IQL data model has one or more attributes defined, e.g., the Type entity has a single attribute called name. Each attribute of an entity belongs to a Domain, where Domains are types that make up the primitive building blocks for storing data. There are a large number of Domains defined in Declarix from simple strings and Dates to complex types like money and quantities with units of measure.

Declarix derives much of its intelligence from the way relationships are expressed in IQL through the LINK attribute. In the above, the Item entity has simple links to both the Type and Note entities. For example, the category attribute of Item is related to the Type entity in the *td* name space. Specifically, the target of the LINK is *tdType.items*, which is of the general form:

```
<target entity>.<attribute in target entity>
```

If it doesn't already exist, Declarix will automatically create a logical attribute in the target entity, named *items*, which links back to the originating Item entity. From the Item entity perspective, the multiplicity of the LINK to the Type entity is "0..1". However, the multiplicity of the logical attribute *items* attribute of the Type entity is automatically created to be "0..*", which implies a one-to-many relationship between the two attributes. In IQL relationship multiplicities can be explicitly declared as "0..1", "0..*", or "m..n". In addition, rules or constraints of arbitrary complexity can also be attached to an attribute relationship.

All that is left to finish the initial cut of the ToDo List application is to add at least one IQL VIEW declaration. The simplest IQL VIEW looks like:

```
VIEW item NAME root AS "Todos"
```

By default this declaration creates a tabular view of the Item entity along with “Add”, “Edit” and “Delete” forms and functions. However, before the application can be viewed, the *todo.iql* file must be processed by executing the following commands:

```
dx stop
dx readiql
dx recreate
dx start
```

Once the IQL is processed, the ToDo application is fully functional with a fully operational data schema and a web based user interface including complete add and edit forms.

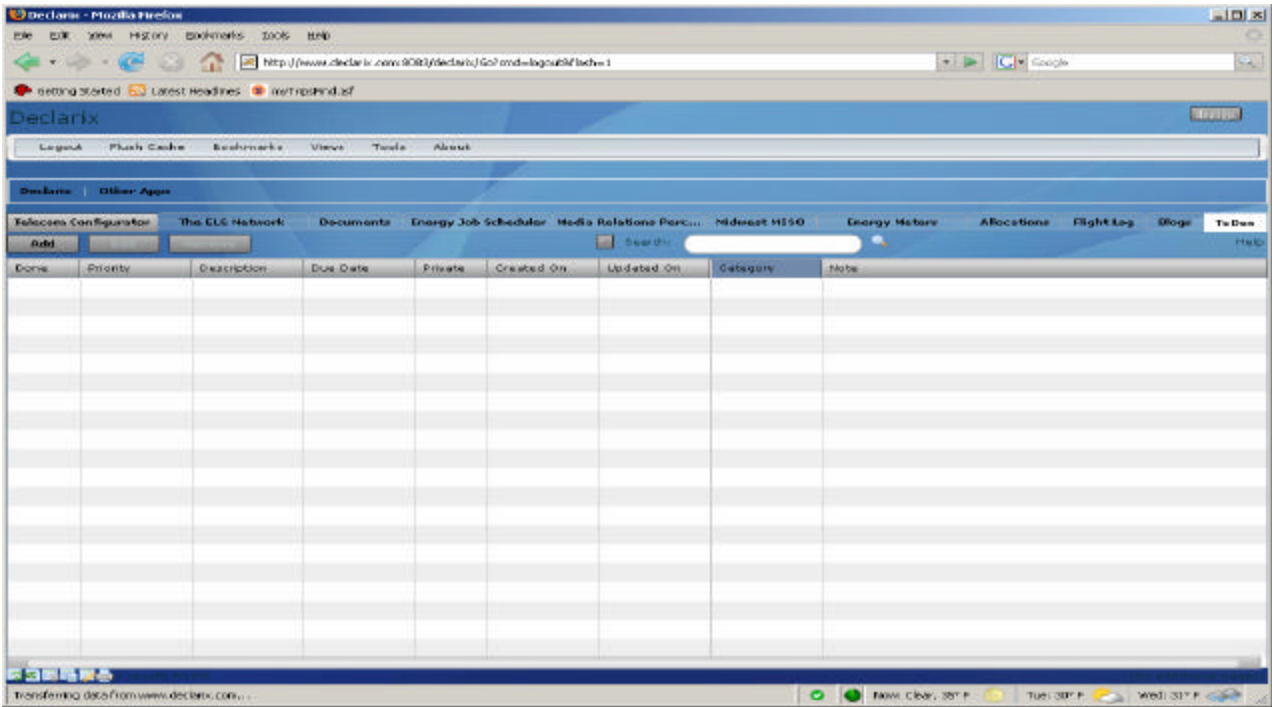


Figure 3: The ToDo List main screen complete with Add, Edit and Remove functions.

Selecting the “Add” button brings up the data entry form in figure 4. Declarix automatically puts calendar widgets next to dates and drop lists for the one-to-many relationships. Data can be entered for the Type and Note entities by simply selecting one of the icons next to a drop down list (see figure 5).

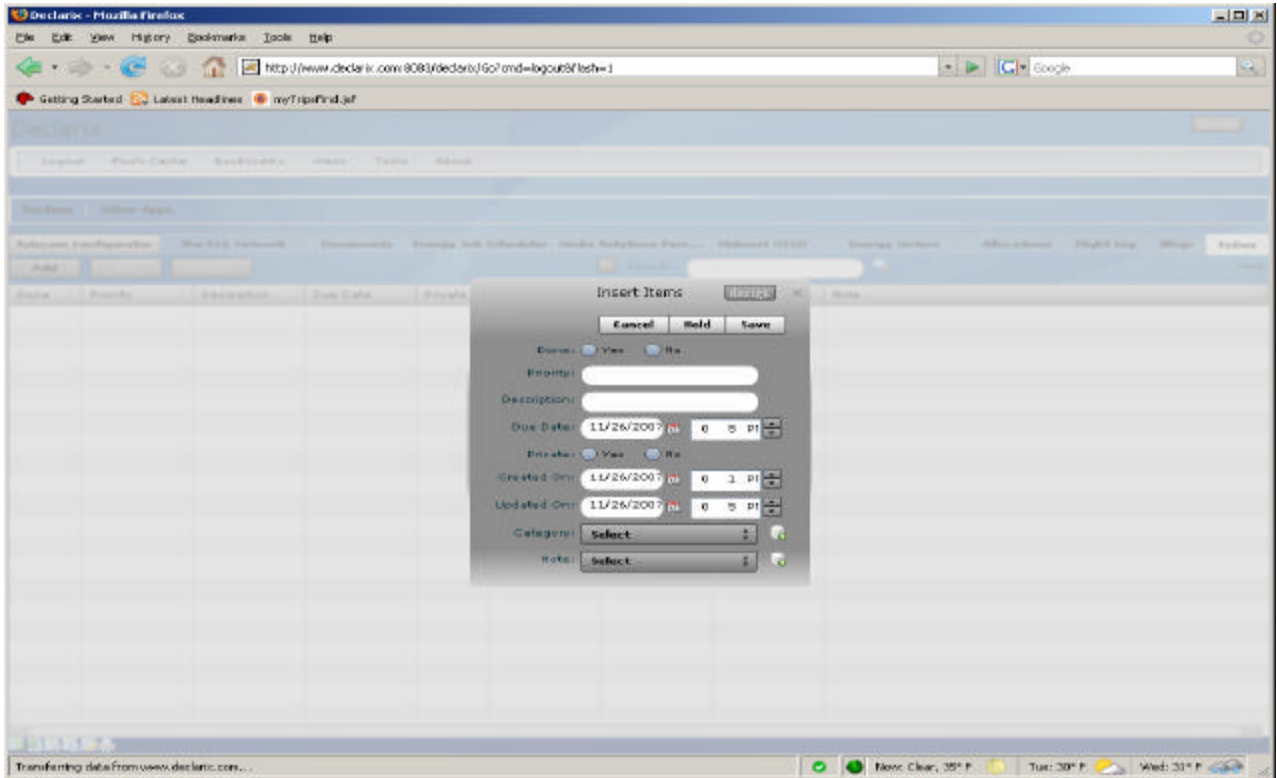


Figure 4: The ToDo Add Item form.

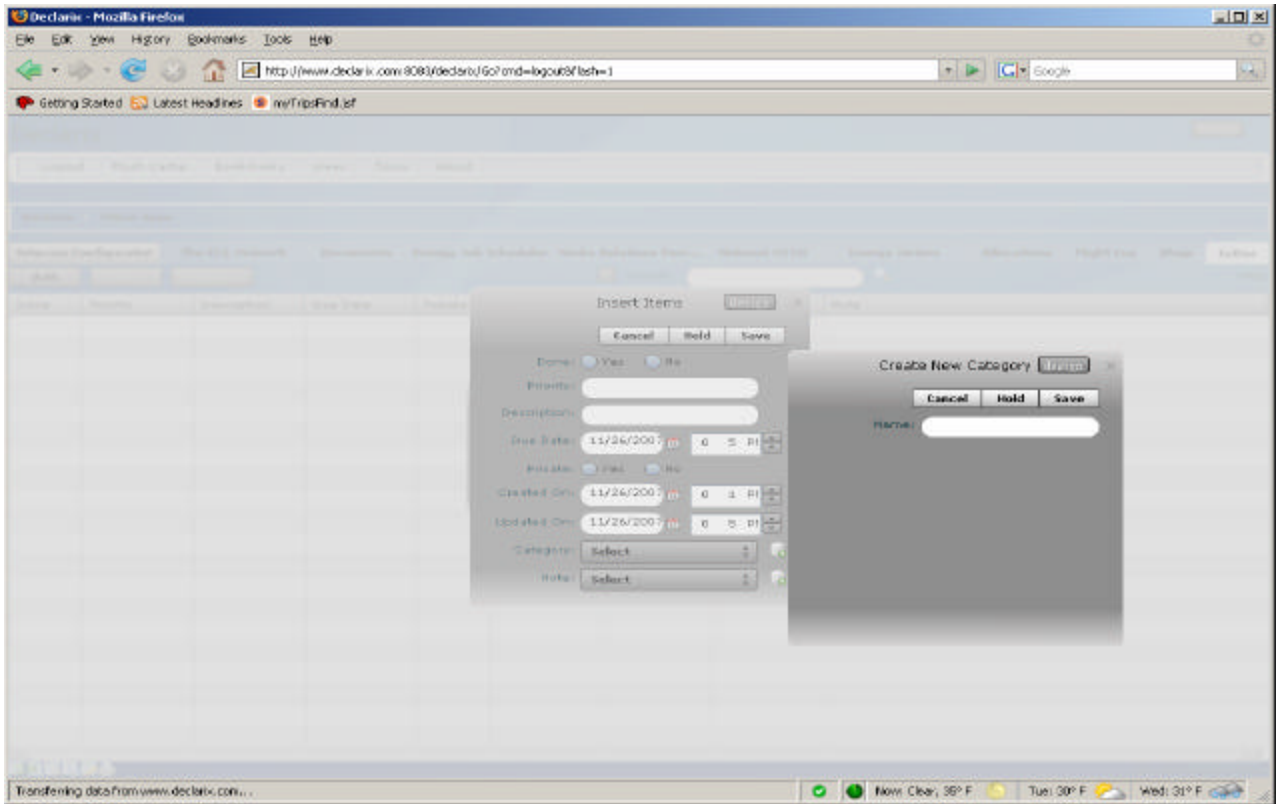


Figure 5: The Create New Category form accessed by selecting the icon next to the Category drop down.

A first enhancement to the ToDo List, will be to constrain the Item's table view to show only a subset of its attributes. This is done by augmenting the root Item view with a `SELECT` clause. For example,

```
VIEW item NAME root AS "Todos"
SELECT description, category.name AS "Category", done AS "Complete",
dueDate, note.content AS "Note"
```

Notice here that there are two implicit joins: `category.name` specifies a join to the name attribute of the type entity; and `note.content` specifies a join to the content attribute of the note entity. Declarix supports traversing to arbitrary levels with this join notation. This VIEW not only provides visibility to these fields in the table view but also allows data to be directly entered into those fields from the Add Item form (Figure 6).

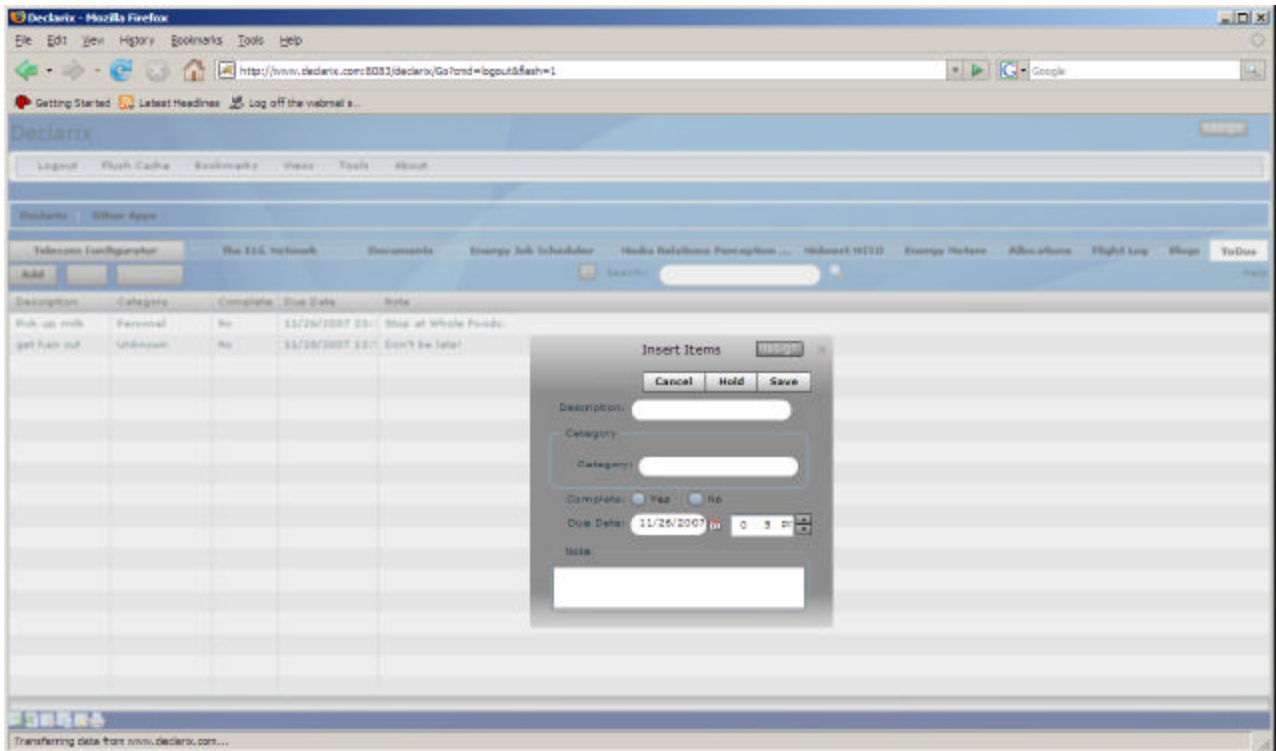


Figure 6. Direct data entry into Type and Note entities.

Declarix supports a variety of constructs for fine tuning the layout of forms, including the use of style sheets. One of the simplest layout controls is the use of a comma or semi colon between the attributes in a SELECT clause. Using a comma to separate attributes indicates that the next element should be placed to the right of the previous one; and using a semi colon to separate attributes indicates that the next element should be placed on the next row. For example, adding this additional VIEW of Item to our IQL file will result in the form in Figure 7.

```
VIEW item NAME root AS "Todos" ON FORM
SELECT done;
      priority;
      description;
      category AS "Category";
      dueDate;
      note.content AS "Note";
[ createdOn +R -U, updatedOn DEFAULT TYPE DATE ]
```

There are several interesting points with this VIEW statement. One is that this VIEW is explicitly declared to be on a FORM. Declarix will always first look for a VIEW declared on FORM when deciding which VIEW to display when the Add or Edit buttons are selected. If it doesn't find one, it will use the first VIEW it finds on that entity. The next interesting feature is the square bracket separation of the *createdOn* and *updatedOn* attributes. The attributes contained within the [] will be framed within a box and, in this case, on the same row. The +R and -U following the *createdOn* attribute are explicit access permissions which,

in this case, state that this attribute can be read (viewed) and can not be updated.

Finally, the `DEFAULT TYPE DATE` specification, following the `updatedOn` attribute, sets the current date as its initial default value in the form.

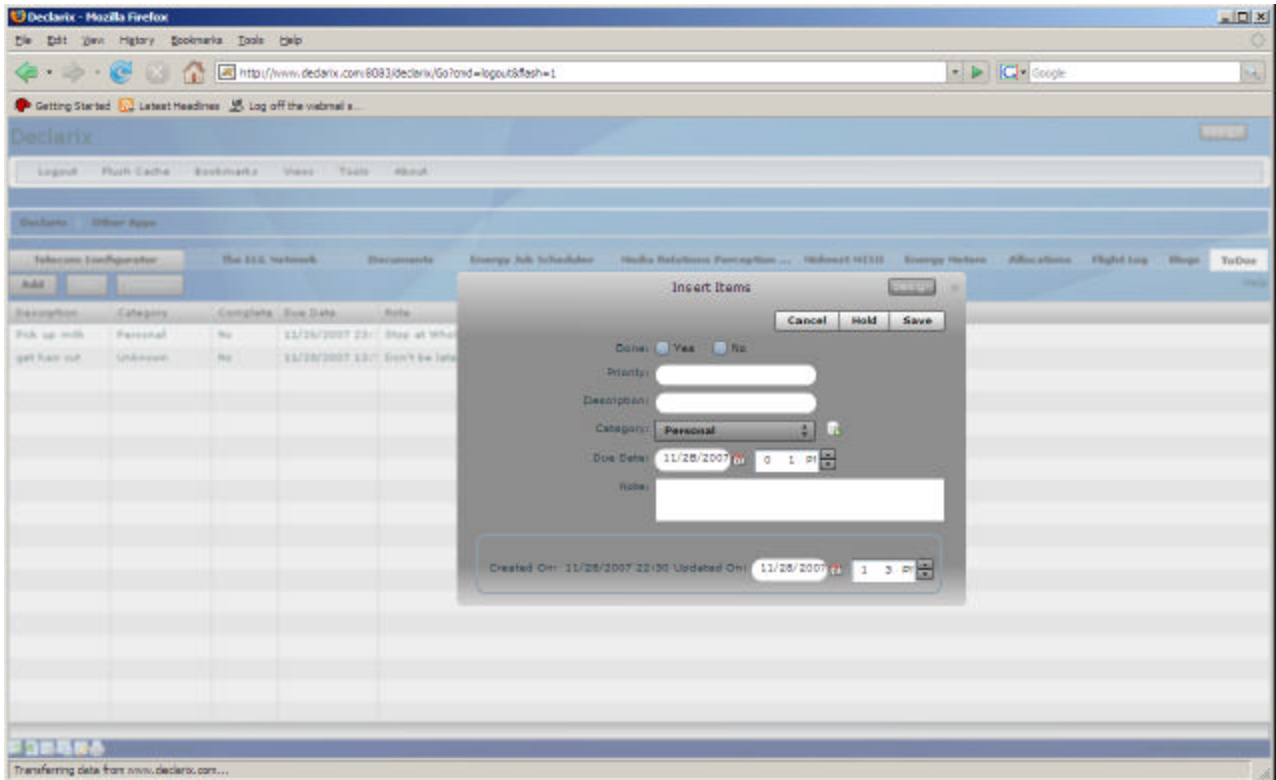


Figure 7. New Item form VIEW

Rules around data and around presentation of data are powerful concepts in the Declarix IQL meta language. To wrap up the ToDo List application, a simple rule on the Item VIEW will be used to highlight in green items that are complete and highlight in red items that are overdue. The specification of this rule is added to the main Item VIEW:

```
VIEW item NAME root AS "Todos" ON TABLE
( ROW-BACKGROUND-COLOR "<[ CASE WHEN done: '#AAFFAA'
  WHEN dueDate < .now: '#FF0000' END ]>" )
SELECT description, category.name AS "Category", done AS "Complete",
dueDate, note.content AS "Note"
```

This results in the display in Figure 8.

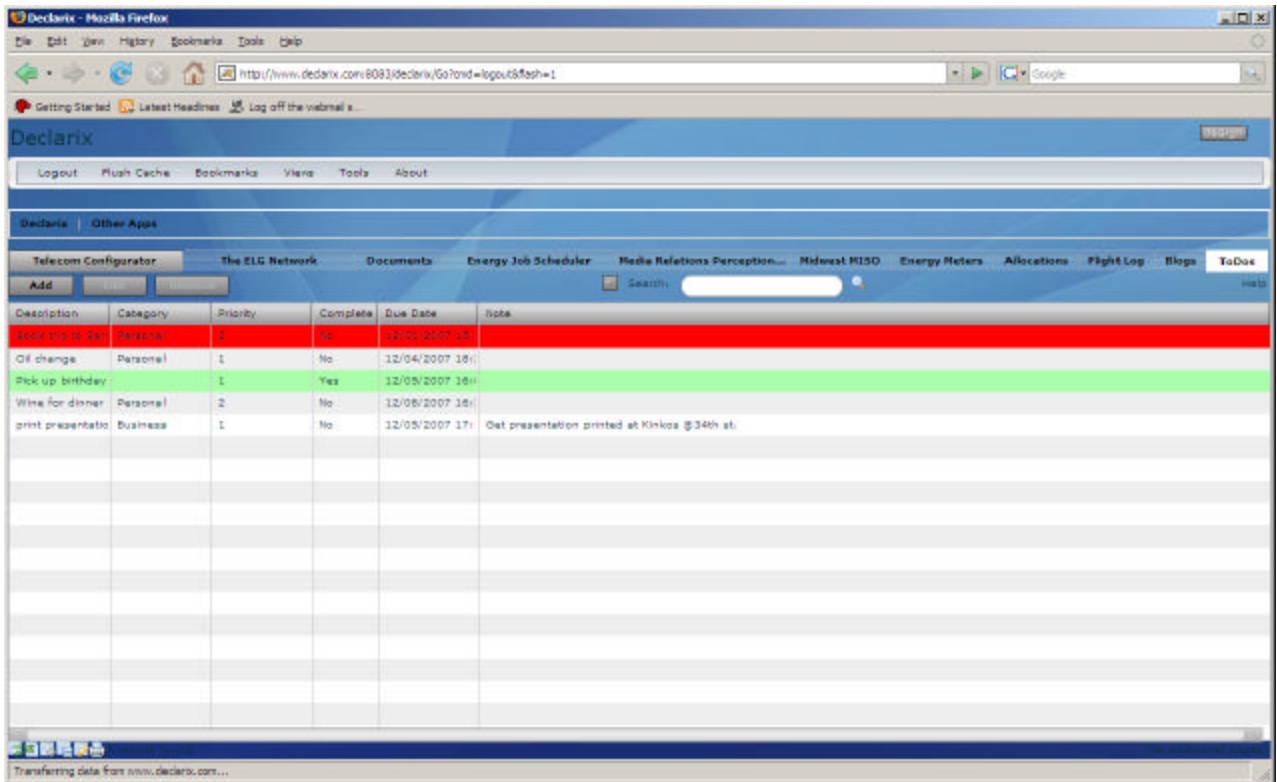


Figure 8. Display rule for completed and overdue items.

With this latest edition to the Item VIEW, the complete IQL for the ToDo List application looks like:

```
CATEGORY todo ABBR td FILE

ENTITY type
  HAS name IS name NAME
    [ "Personal"; "Business"; "None" ]

ENTITY item AS "Items"
  HAS done IS boolean,
    priority IS int,
    description IS string,
    dueDate IS date,
    private IS boolean,
    createdOn IS date DEFAULT TYPE DATE,
    updatedOn IS date,
    category IS LINK tdType.items,
    note IS LINK tdNote.items

ENTITY note AS "Notes"
  HAS content IS largeString,
    createdOn IS date,
    updatedOn IS date
```

```

VIEW item NAME root AS "Todos" ON TABLE
  ( ROW-BACKGROUND-COLOR "<[ CASE WHEN done: '#AAFFAA'
    WHEN dueDate < .now: '#FF0000' END ]>" )
  SELECT description, category.name AS "Category", done AS
"Complete", dueDate,
  note.content AS "Note"

VIEW item NAME root AS "Todos" ON FORM
  SELECT done;
  priority;
  description;
  category AS "Category";
  dueDate;
  note.content AS "Note";
  [ createdOn +R -U, updatedOn DEFAULT TYPE DATE ]

```

This simple ToDo List application doesn't do justice to the depth and richness of programming with the Declarix IQL meta language. Hopefully, it does convey a flavor of what a new generation of metadata programming looks like and, more importantly, what the future of web application development will look like.



**The Next Evolution in Relational Database
Management Systems**

**Declarix, Inc.
PO Box 390682
Minneapolis, MN 55439
U.S.A.**

www.declarix.com
info@declarix.com

Copyright © 2007-2008, Declarix. All rights reserved.