Leveraging Your Physical Data Model

March 2011

Intelligent Solutions, Inc.
P. O. Box 4587
Boulder, CO 80306
Voice: (303) 444-2411
Fax: (720) 293-0060
www.IntelSols.com

Introduction

Database design is a critical step in any development process. The developers are under continuous pressure to deliver quick results, and it is no wonder that they focus their attention on the tasks they feel directly help the programming effort. Data modeling is often perceived to be a superfluous development step, which means that, barring mandates dictated by the methodology, developers avoid building data models in the interest of delivering their products as quickly as possible.

Ideally, a company has a team of people charged with developing and maintaining a logical data model (a.k.a., business data model or enterprise data model), and then supporting developers who need to create data structures for individual projects. Such a team doesn't always exist in large companies, and is often absent in smaller organizations. When this group doesn't exist, there is no unifying enterprise view, but that doesn't mean that data modeling isn't needed at all.

The physical data model, which is a representation of the data structure that is being built, is very beneficial even if a logical view does not exist. Further, development of such a model can actually shorten the data structure design process, while also enforcing some database design best practices and providing a framework that can be replicated on subsequent projects.

Creating the Physical Data Model

The physical data model can be developed from scratch based on a set of requirements, or it can be built on the foundation of an existing structure. To build the model from scratch, the designer, who is typically either a programmer or DBA, reviews the requirements and creates a data structure that (1) contains the data needed to satisfy the requirements, (2) meets performance constraints for data updates as well as data usage, and (3) conforms to the shop's standards. To use an existing data structure as the starting point, the designer reviews that data structure, decides what can be reused, and replicates it as the starting point for the new structure.

In both instances, a data modeling tool can greatly facilitate the process. Following are some physical data modeling tool features that are useful:

- **Diagramming:** The data modeling tool provides a way to develop the structure diagrammatically, thereby making the structure visible. This helps the modeler visualize the data structure while it's being developed and facilitates its review for completeness and technical accuracy. The user interface is a key factor in helping develop the needed structure as quickly and accurately as possible.

- **Replication:** Models often contain similar data elements in multiple tables and data elements frequently share characteristics. The modeling tool facilitates replicating various objects within the model. These objects may be whole tables or columns, or they can be characteristics about the tables or columns. The richness and flexibility of the replication capabilities can both reduce the developer's effort and automatically enforce certain shop standards.
- **Reverse Engineering:** The data modeling tool has the ability of creating an initial data model by examining an existing data structure and surmising the design upon which it was based. The tool can create a diagram that includes tables, relationships, data characteristics, etc. and present these pictorially to facilitate an understanding of the existing environment and leveraging it to meet the new needs. The reverse engineering feature facilitates database migrations and may also help in selecting the best data source structure to use by examining and comparing alternatives.
- **Documentation:** The data modeling tool provides a place to capture metadata about the data structure being developed, including definitions, notes, and comments. In addition to the modeling tool's ability to capture the information, flexibility in providing access to this information through queries and reports helps the developer review and explain the model and ensure that it meets the requirements.
- **Model Subsetting:** A full data model is often very complex and contains many tables and relationships. During the development process, it's often useful to focus on a particular part of the model, and the tool supports displaying the desired view by enabling the user to specify the objects to include in that view either manually or by defining criteria for inclusion. In addition to helping the developer, these partial views assist during the review process by isolating components of the model.
- **DDL Generation:** In addition to providing a diagramming aid for development of the physical structure, the data modeling tool enables the DBA to generate the data definition language for the physical schema almost effortlessly. In addition to reducing the time to create the DDL, this feature also ensures that the physical structure and the physical model are always in sync.

These are but a few of the ways in which a data modeling tool helps during the initial development of the physical structure. Impressive as this list may be, an even greater benefit is provided during the maintenance cycle of the data structures.

Maintaining the Physical Data Model

Data structures evolve over time. Some of the evolution may happen while development is still underway, while other changes take place after implementation. Since the data structure may last for years, it is highly likely that the people responsible for maintaining the structure will change, and hence it becomes very

important to have good support for making the needed changes, providing an audit trail of the changes, and updating the documentation to reflect the changes to the structure.

Several of the data modeling tool features previously delineated support this activity as well. In addition, useful modeling tool capabilities include:

- **Version Control:** The data modeling tool can retain images of the data structure each time it achieves a development milestone or is released to production. These images include all of the relevant documentation and hence provide an "as is" image of the data structure at all times.
- **Change Detection:** When developers make changes, they can begin with the existing model and then apply the modifications. The collision management features of the data modeling tool enable the developer to review all changes to the structure. During the review process, the developer may confirm that all the changes should be incorporated or may deselect some of the changes, thereby adjusting the revised model and resultant data structure.
- **Change Script Generation:** When changes are made, it is unnecessary to fully regenerate the data structure. A more efficient approach is to generate an ALTER script to reflect the modifications to the schema. Once invoked to create the physical structure, that structure and the model are automatically in sync.
- **Audit Trail:** In addition to version control, which is done at the model level, an audit trail of changes to individual objects within the model can be automatically generated by the data modeling tool. This enables the developer to keep track of each of the changes and to document, at the individual change level, the reason for the changes.

Summary

Development and maintenance of a physical data model can significantly improve developer productivity even when a driving logical model is absent. When a physical model is developed using a data modeling tool, the process is more rigorous without placing an under burden on the designer. The modeling tool:

- provides the developer with a diagramming capability that can be tailored to each individual's preferences;
- facilitates capture of relevant documentation while the design activity is underway rather than tackling documentation as a follow-up activity (which is often skipped);
- supports reuse of existing structures through reverse engineering and of components of the new structure through replication capabilities;

- generates the DDL needed to create the physical schema, thereby ensuring that the physical structure is always accurately depicted in the data model; and
- supports on-going maintenance of the physical structure through version control, change detection, documentation, and generation of the ALTER scripts to keep the model and the physical structure in sync.

A physical data model is always created by a person developing a physical structure, but unfortunately, without support of a data modeling tool, that model is often virtual (in the person's head) or not documented. Use of a physical data modeling tool not only helps in the development process of the individual data structure, it provides a model which can be reviewed by subsequent developers and leveraged for additional structures. As such it offers a great vehicle to help even small organizations develop and maintain their databases.