Physically Implementing Universal Data Models to INTEGRATE DATA

By Len Silverston

There have been several recent articles and books published concerning universal data models which are reusable models for common data constructs and industry applications (see *DM Review*, January, March, May and July 2002, and *The Data Model Resource Books, Volumes 1 and 2*, Wiley 2002), but how can one use these templates or universal data models to really make a difference?

These models can be used as a tool to quickly develop quality database designs by reusing commonly available data models which are applicable to the enterprise's requirements and customizing the details for the application at hand. However, there is another purpose for many of these models – they provide practical designs and insights for integrating data, thus providing enterprises with powerful enterprise-wide information.

This article provides suggestions on how to use and implement universal data models to help provide more integrated and better quality information. For instance, how can these models be used to identify and resolve data inconsistencies between various source systems or application packages? How can these models be used to physically consolidate and integrate data, providing powerful integrated views of information?

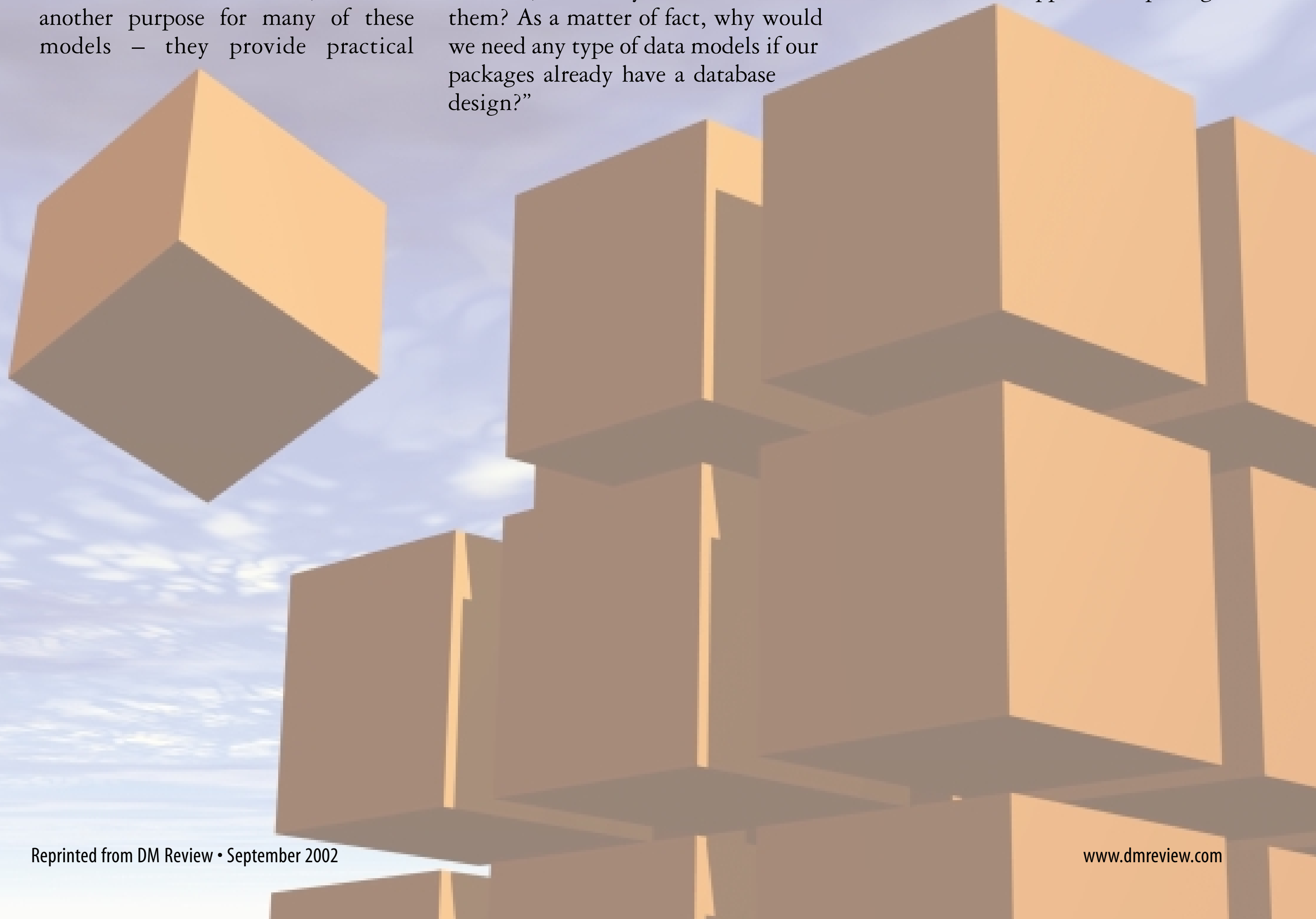
Integrating Data

I have heard the statement, "Our systems are mostly based upon application packages. Universal data models seem valuable for building a new database, but why would we need them? As a matter of fact, why would we need any type of data models if our packages already have a database design?"

Aside from the usage of data models to define the data requirements of possible application packages, universal data models can be used to synchronize and integrate information across various applications within an enterprise. Without an enterprise-wide integrated picture of how data relates across applications, it is very difficult to build integrated architectures that provide accurate, consistent and integrated information.

Flexible Structure

Enterprise resource planning (ERP) application packages claim to offer complete enterprise-wide, integrated, customizable solutions. However, organizations that select an ERP package generally also have numerous other application packages



in order to obtain best-of-breed solutions. Therefore, enterprises will often have overlapping, redundant and incomplete data sources.

Universal data models offer flexible structures that can accommodate almost any data format from any application package, allowing the data from multiple packages to be synchronized and integrated within a single data construct.

For example, many enterprises select a specialized contact management application package; an ERP package to handle their mainline order, shipment and invoicing processing; a product configuration and/or quoting software package; and an accounting software package. Customer, supplier and employee information may, therefore, be recorded redundantly in these systems.

How can information from these packages be synchronized and integrated? Consider the standard party, party role and party relationship universal data model that is shown in Figure 1. Note that this is just one example of a universal data model. There are many other universal data models for other common constructs.

This model illustrates that a PARTY may be either a PERSON or an ORGANIZATION, that each PARTY may be acting in one or more PARTY ROLES over time and that each combination of PARTY ROLES forms various types of PARTY RELATIONSHIPS. The PARTY entity facilitates a consistent place to store data such as contact data or demographics, regardless of their role, thus avoiding redundant, inconsistent data. The PARTY ROLE entity maintains information that is relevant to a specific role that a party may play such as payroll data related to the role of EMPLOYEE or credit-check data related to the role CUSTOMER. The PARTY RELATIONSHIP entity provides a place to maintain information relevant to the relationship between parties such as the relationship status, the relationship priority or meetings, phone conversations and other communication events that occurred within the context of each relationship.

This universal data model provides a single place to maintain PERSON and ORGANIZATION information, namely in the PARTY entity. However, because there are usually multiple applications and databases within an enterprise, the same party may exist in each of the enterprise's systems and may be inconsistent and disjointed.

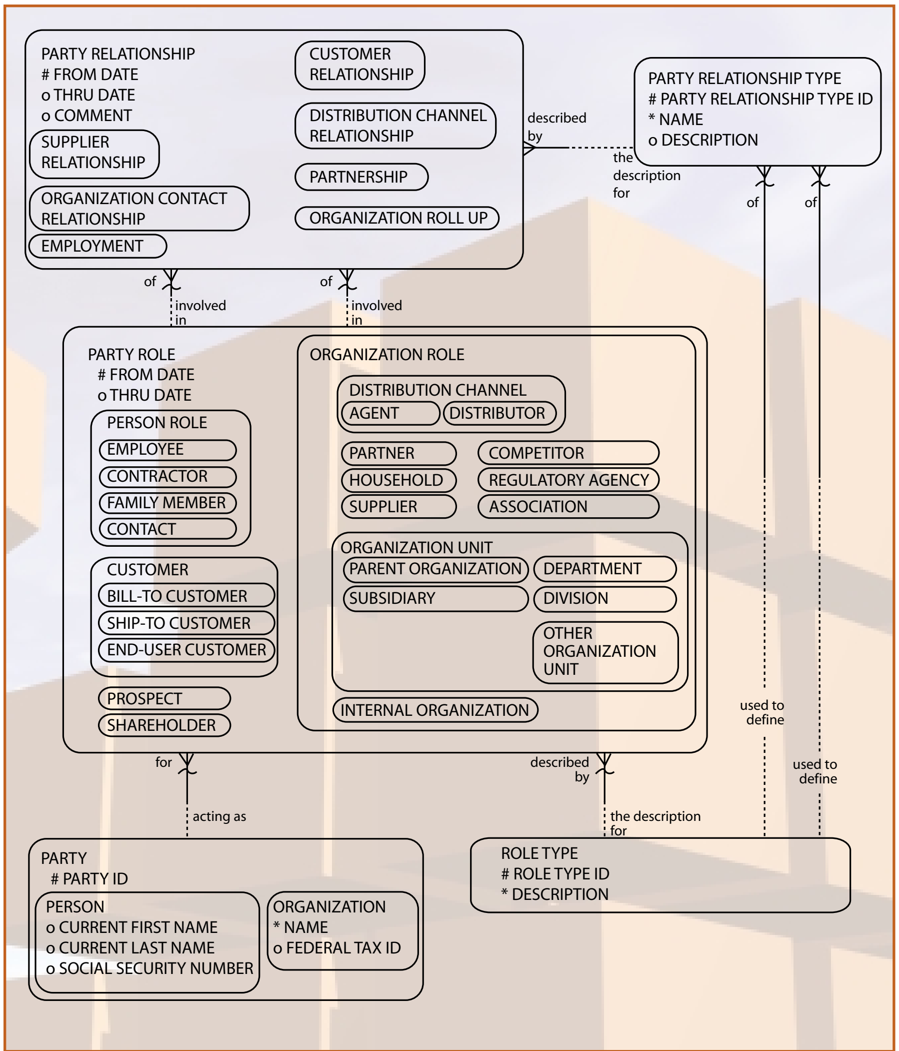


Figure 1: Universal Data Model for Parties, Party Roles and Relationships

The Whole Party

The "party" universal data model in Figure 1 can offer the capability to view consolidated, integrated information for each party only if there is a mechanism to cross-reference the party's enterprise key with each of the application keys.

For example, suppose there are several records for the same person, John Smith – one in the contact management system and another record in the ERP application. The information from both systems could be different; for instance, the name could be spelled differently or there could be a different postal address, inconsistent phone number or inconsistent value for the demographic data.

Cross-Referencing the Data From Multiple Applications

The key to providing an integrated profile is to set up an enterprise-wide key for each PARTY (in the universal data model) and to cross-reference it against each of the application systems – in this case, the contact management system and the ERP application.

There are three general strategies for cross-referencing this information:

- Create a foreign key, party_id, from each application that cross-references each application key with the enterprise key.
- Create a cross-reference table that

cross-references each application key with the enterprise key.
- Use a combination of these two strategies.

Cross-Reference Applications

If the system could relate and cross-reference all occurrences of any given party, it would be easy to obtain all the information about that party. This is significant because the ability to have complete information on each person or organization in an enterprise is a major advantage to most aspects of business, including selling and servicing.

Figure 2 illustrates how the physical tables would look using this cross-reference strategy. The party_id for John Smith is identified as "111257." Anywhere John Smith has a record, there is a foreign key of "111257," thus allowing his data to be consolidated.

Thus, the enterprise could create queries and reports that join the PARTY table from the enterprise-wide schema (possibly contained within an operational data store or a data warehouse) with each of the application tables that has a corresponding party_id foreign key.

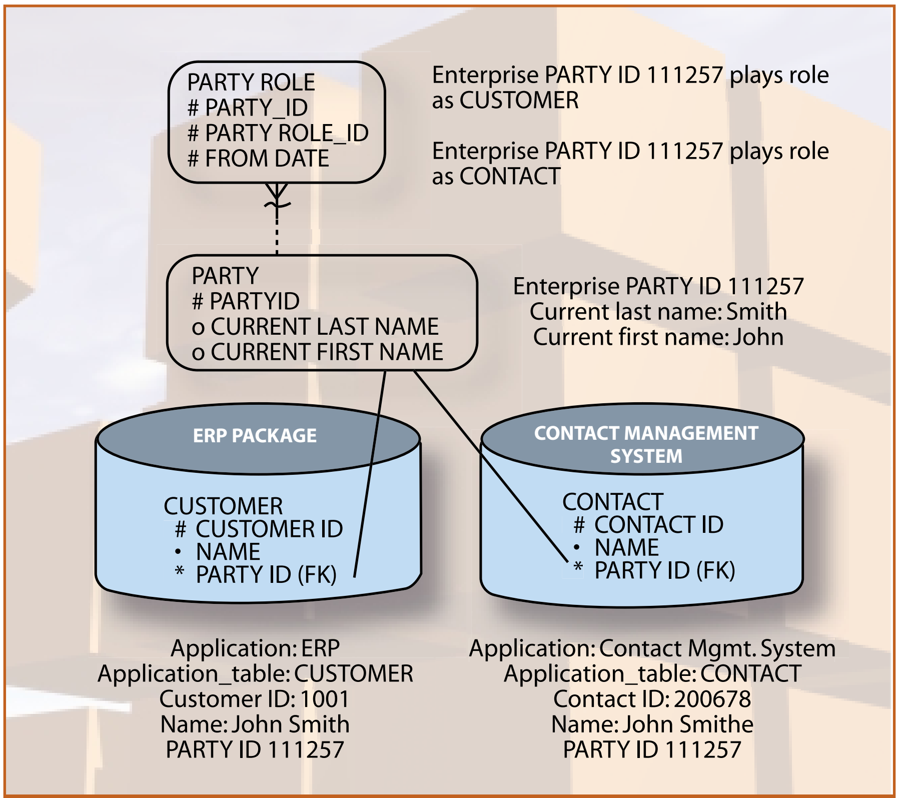


Figure 2: Cross-Referencing Applications to Universal Data Models Using Foreign Keys

Cross-Reference Table

The second strategy involves creating a cross-reference table, linking each enterprise-wide key with the associated application key. Figure 3 illustrates how these tables could be set up to cross-reference an enterprise-wide key with application keys. Notice that each instance of keys for John Smith is now linked to his enterprise-wide key.

This strategy requires that application keys are stored redundantly in this cross-reference table. This requires substantial overhead because every time a new application key for a party role is generated or changed, the cross-reference table needs to be updated. Why would one use this strategy versus just updating each table with a foreign key? The answer is that it is not always possible or easy to update each application with a foreign key. For instance,

Notes about the data modeling notations in this article:

- A crow's foot (three prongs at the end of the relationship line) indicates that there are many occurrences of the entity near the crow's foot for each entity that is not near the crow's foot. For example, each PARTY may be *acting in* ***one or more*** PARTY ROLES (entity names will be shown in caps in this article).

- The dotted line indicates optionality (as opposed to mandatory) for each side of the relationship. Each ROLE TYPE ***may be*** (since this is dotted part of the line) *used to identify* one or more PARTY ROLES. Reading the other way, each PARTY ROLE ***must be*** of one and only one ROLE TYPE.

- A '#' in front of an attribute indicates that this attribute is a key.
  A '*' indicates that the attribute is a mandatory attribute.
  An 'O' before an attribute indicates that the attribute is optional.

- Boxes within boxes indicate subtypes or subentities.

- The tilde "~" on the relationship line represents foreign key inheritance. This means that the primary key of the entity closest to the tilde includes, as part of its key, the primary key of the entity without the crow's foot.

it may be very difficult or even impossible to add foreign key fields to legacy applications, custom-developed applications or certain application packages.

Another solution is to add foreign key fields when possible and use a cross-reference table when it is not possible or practical. This allows the cross-reference table to be smaller and reduces the overhead for the applications that use the foreign key; however, the downside to this strategy is that the routines for obtaining a consolidated view of the party's information are much more complex because the routine would call for foreign key lookups combined with lookups from the cross-reference table.

Regardless of the implementation chosen, one can implement the universal party models by cross-referencing the party enterprise key with application keys to gain a complete profile of information about parties.

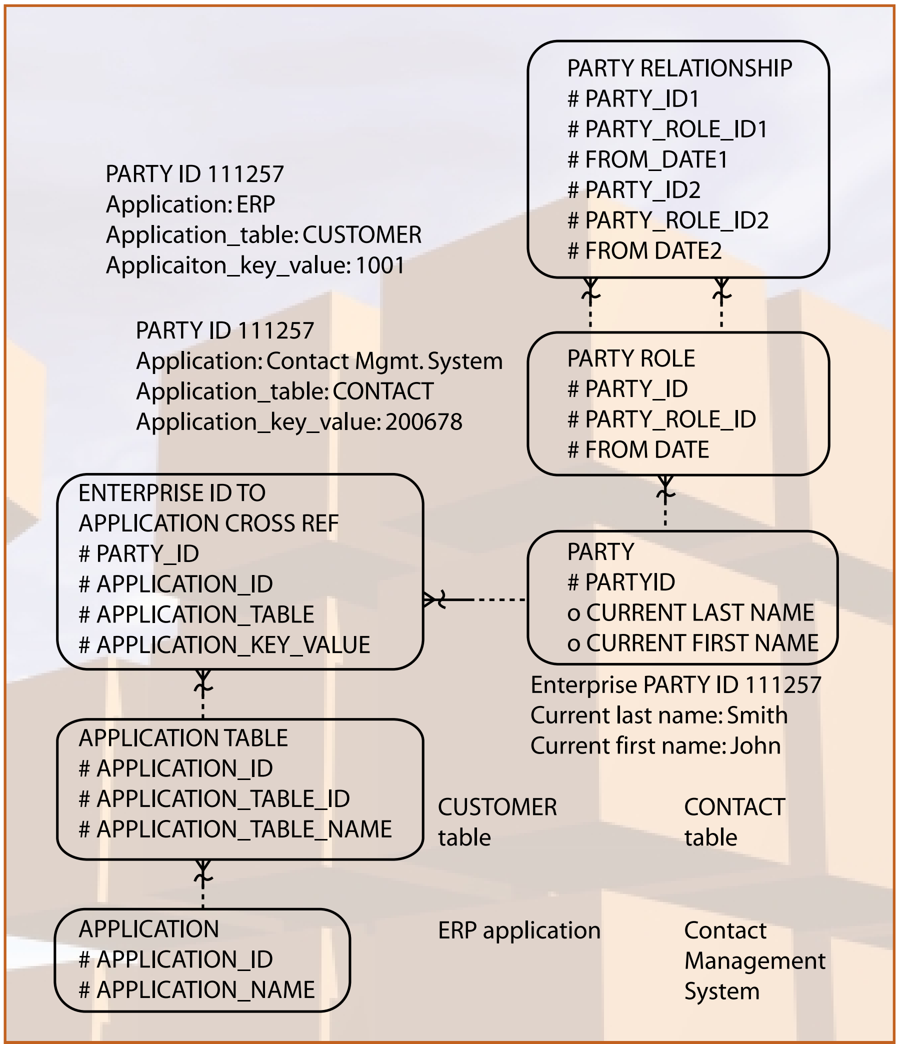


Figure 3: Cross-Reference Table to Map Application Keys to Enterprise Keys

Inconsistent Data

What if the data from two source systems is different? I once reviewed an enterprise data model and noticed that the person table had two columns for "blood type," namely "blood type 1" and "blood type 2." According to what I know about the human anatomy, a person can only have one blood type, so I asked how a person could have two blood types. The analyst explained that when a person in the model had two blood types listed, one type came from system A and the other from system B. Because the group managing each system insisted their data was correct, they decided to store both of them.

If there are data inconsistencies, what should happen in the meantime? Should the system include all instances of parties with pointers to the various sources for the actual data, thus properly reflecting the various versions of the data? Should the integrated store include only data that has passed the preset business rules and send the other data that cannot be resolved into another data store that is reviewed by data stewards and business representatives for data resolution?

System of Record

Business rules need to be established to determine when to use which data from what system and how data inconsistencies should be handled. For instance, under certain circumstances, the blood type from system A may be determined to be correct source for data. Under other circumstances, the data from system B may be determined to be the correct data. Another business rule could be that inconsistent data values need to be recorded and reported to business representatives who need to resolve these data inconsistencies. Of course, this requires cooperation from the various groups maintaining data. That's the real challenge – ownership of data and cooperation between various parts of the enterprise to resolve data differences. In order to have consolidated, integrated information, it is necessary to work as a coordinated team across various parts of the enterprise to resolve these types of data issues.

A common and often effective solution for resolving data inconsistencies is to define the "system of record," or source, that will be used as the overriding source if there are conflicts. Many enterprises define a "system of record," tagging each data field and possible rows within that field with the application that is defined as the "system of record" for that data.

Let's take a look at another example. Suppose the contact management system stored a record for "John Smith," Social Security number "222-22-2222," with an address of "100 Main Street." Further suppose that the ERP system has a record for a "John Smithe," Social Security number of "222-22-2222" and an address of "101 Main Street."

If there is a business rule that says that when the Social Security number is the same, the records will be consolidated into a single record in the enterprise integrated data store, which name and which address should be stored? How should the system work in this case?

One possible scenario is that the enterprise could establish a business rule stating that the contact management system is the system of record for personal names and addresses. In this case, the information that is reported (and that may be stored in an enterprise data store) would be the data from the contact management system, namely, "John Smith," Social Security number "222-22-2222" at "100 Main Street."

To resolve the data discrepancy between the source systems, the system could either automatically update the ERP system with the name and address from the contact management system because it was defined as the system of record (and record this in an audit log) or issue an alert that the data is inconsistent and needs to be resolved by business representatives.

Data Architecture

Figure 4 illustrates a possible architecture showing how data could be synchronized and integrated across an enterprise. An enterprise could create routines that synchronize and integrate data across the application and use the universal data models as a basis for the integrated enterprise schema. Note that the enterprise-wide schema may physically implement both the cross-reference tables and the PARTY, PARTY ROLE and PARTY RELATIONSHIP tables for convenient access to integrated roles, relationships and other enterprise-wide information or, alternatively, just implement the foreign keys and/or cross-reference tables. This schema could feed a data warehouse and associated data marts, or the data warehouse could be fed within the synchronization and integration routines. Notice that the integration and synchronization routines output data issues to a database that data stewards can use to resolve data discrepancies. Furthermore, a meta data repository can define data about the source systems, synchronization and integration routines, common data schema and associated data warehouse and/or marts.

While universal data models can help jump-start database design efforts, another use for them is to help integrate data. There are numerous choices in integrating data, and enterprises need to define their strategies and business rules for synchronization and integration. Cross-referencing source systems to universal data models and identifying the system of record for each data item can be an effective strategy to identify and resolve data discrepancies while providing an integrated view of information.

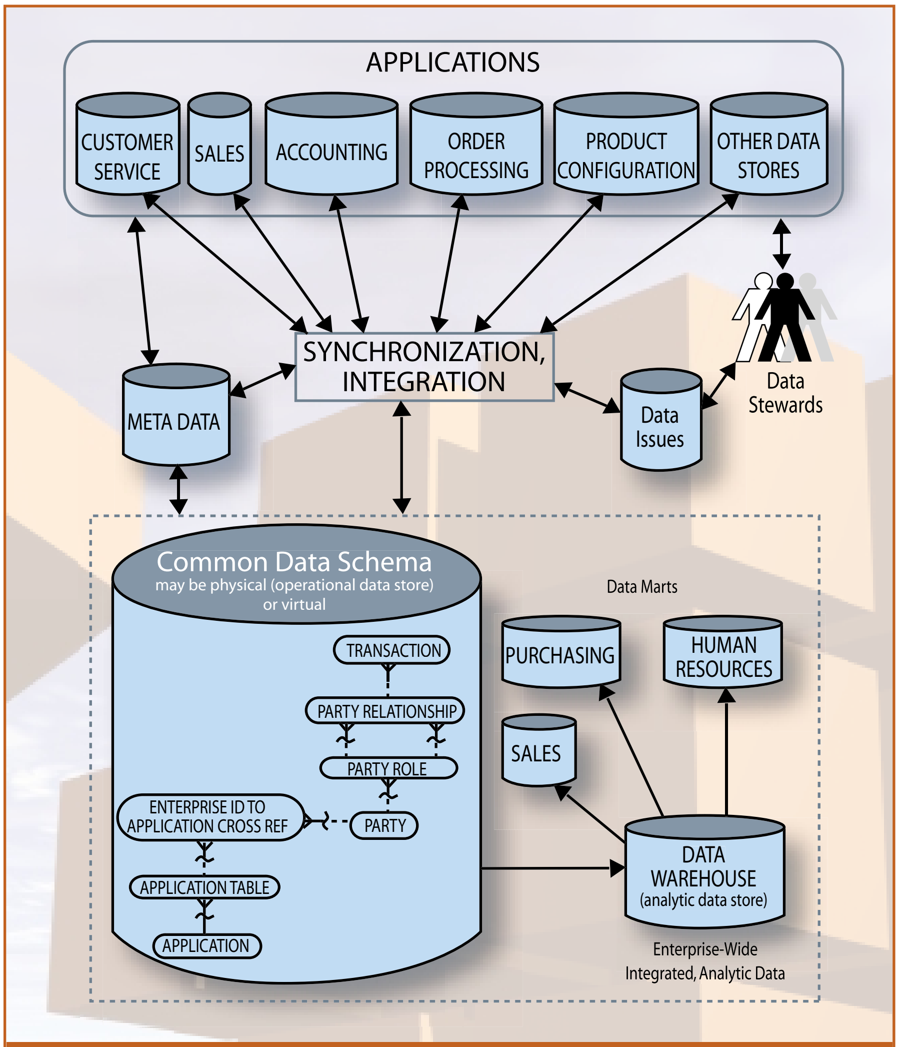


Figure 4: Synchronizing and Integrating Data Across Applications

*Len Silverston is a data management consultant with more than 20 years of experience in helping enterprises integrate data. He is the author of the best-selling **The Data Model Resource Book** series (Wiley, 2001), which describes more than 230 integrated, reusable generic and industry-specific data models. Silverston has developed extensive software versions of these data models some of which are now licensed worldwide by Microsoft and some that are available for licensing directly. Silverston's company, Universal Data Models, provides consulting, training and software to jump-start data modeling and data warehouse design efforts while increasing design quality and facilitating data integration. Silverston can be reached at lsilverston@univdata.com.*

www.dataWarehouse.com

If you haven't been to www.dataWarehouse.com, it may behoove you to check it out! This is a community site, and members are BI/DW professionals. Visit the online trade show; read columns by David Ray Fuller, Tom Johnston, David Loshin and Cody Bateman and articles by industry practitioners; or join a forum discussion.