TP162, 05/02

Rational. software

UML Data Modeling Profile

Davor Gornik

Table of Contents

Database Technology

Database is the collection of data organized in a way to be easily stored and retrieved using programs. A database includes the methods to store and retrieve the information.

The information and the requirement on organizing the information are different for different types of the application. However most of the market and most of the common requirements can be satisfied by relational databases. Other types of databases like hierarchical, object-oriented, and hypertext databases cover distinct market niches.

Relational databases realize a very simple principle of entities, which can be seen as tables, and relationships between entities, which are references to other entities.

Every other concept supported by the relational databases is used for easier access, speed, and security.

Relational Data Base Modeling System technology is one of the most reliable technologies on the market. The basic idea is more then 3 decades old, the first products have been developed 25 years ago.

The demand on standardizing was extremely high, which lead to SQL, the standardized language for data definition and data manipulation. There are three versions of the standard called SQL-1, SQL-2, and SQL-3. Although standardized in 1992, the SQL-2 standard is the commonly used one for the recent implementations of the major vendors. They extend their implementations with own language and structural constructs to cover market requirements.

Unified Modeling Language

The unified modeling language is compared to the SQL as a relatively new technology. UML was standardized in 1997, and since then has had some minor reviews. However the sources of the UML are from the 80s, and early 90s, when different modeling languages tried to establish a way to describe and design better applications.

The scope of the language was targeted towards software development. It was however visionary enough not to limit it in any direction. UML implies the concept of adaptability, which can be used to describe software-related and unrelated areas of expertise. **Profiles** customize the UML to a domain without leaving the standard of the language.

The strength of the unified modeling, is unification of all areas of expertise into a unified platform. It does not matter how different the technologies are, all of them can be described using the same language.

UML Data Modeling Profile

The Relational Data Base Management Systems are the most used form of databases. IBM Rational's UML Data Modeling Profile provides an easy to use and understand adoption of UML for the need of database modeling, and database design.

The concept of tables and relationships used in a database maps to the concept of classes and associations in the core UML. However there are additional constructs and constraints in the database modeling, which also have to be modeled visually, like the database, and schema.

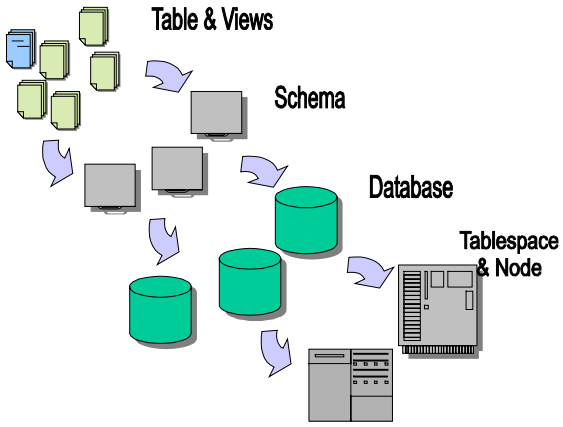

Figure 1 The variety of the database implementation

The figure 1 shows the variety of the database deployment. The complex assignments of tables and views to the schema, schemas to the database, and databases to tablespaces and nodes, confuse every DBA, who needs a simple representation of the underlying architecture. It is essential to be able to plan the distribution and configuration of the database.

Node

The representation of the physical entity (computer) where the database can be located is a node. The representation is part of the core UML.

The nodes are used in a deployment diagram, which represents the physical configuration of the software deployment. The diagram includes the nodes and the connections between nodes. The connections represent the communication protocols.

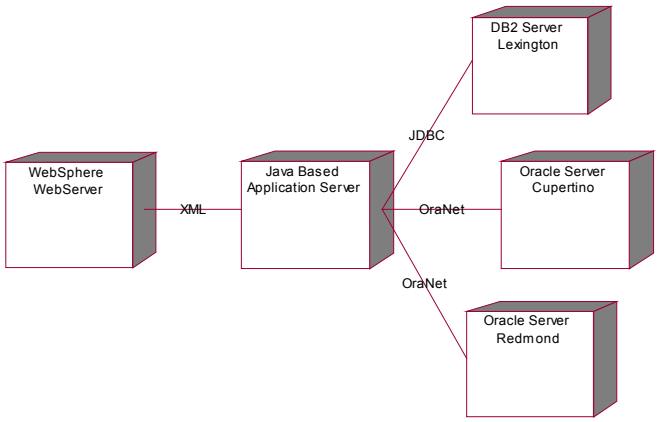

Figure 2 The deployment diagram

The nodes “DB2 Server Lexington”, “Oracle Server Cupertino”, “Oracle Sever Redmond” represent nodes, the XML, JDBC, OraNet represent communication protocols. All of the software and database has to be deployed on the physical node.

The deployment diagram is important for the data administrator for the configuration of servers, and problem tracking (start with the deployment and dive into details).

Tablespace

The Tablespace is the storage for data, which represents a database system. It is the link between the user transparent physical structure called Database (description follows) and the node. The Tablespace is a stereotyped component in the UML Data Modeling Profile.

Tablespace can be best understood as an area on the physical storage, which is maintained by a database. Database itself can be distributed to several Tablespaces, which is decided on the size of the data, data access requirements, and security requirements.

The Tablespace is associated from the Database using a dependency, and is optional in the design of the database implementation. If not used, a default Tablespace is assumed for the implementation; maintained by the Database.

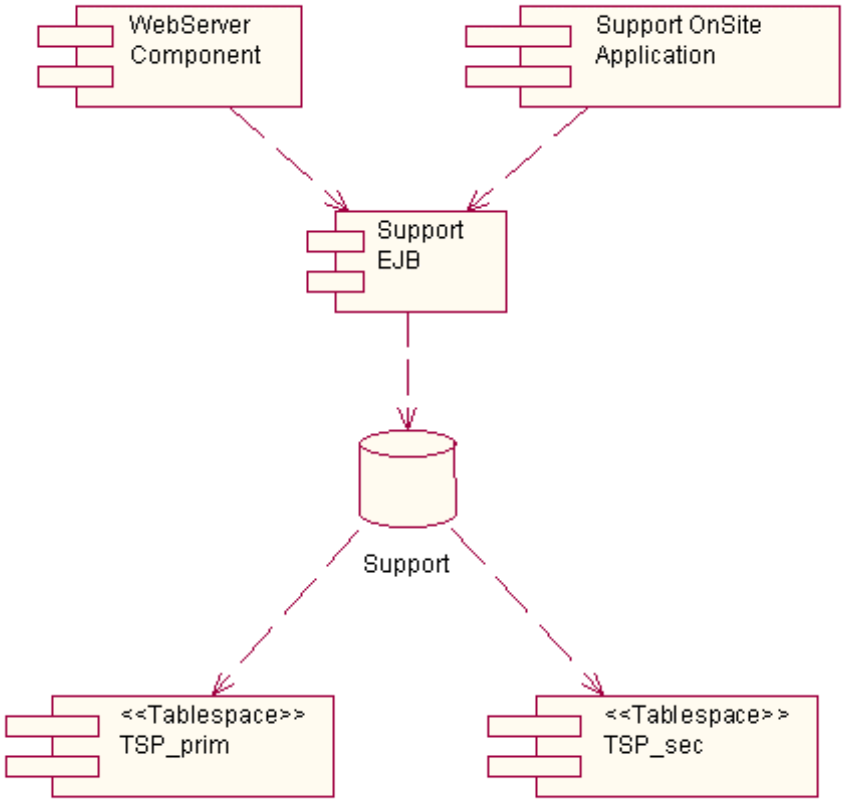

Figure 3 Database realization in two tablespaces

The value of Tablespaces in the realization of the database is in planning of the node environment and establishing the requirements for the nodes. Problems with parts of the database can be traced down easier with help of the component diagrams. Tables can be realized either with the Database or with the Tablespace. In case of the realization with the Database, default Tablespace is assumed by the implementation.

The basic structure of a Tablespace as a physical storage unit is implemented differently with every database vendor. They give more or less control to the Tablespace on storage requirements and structuring inside of the storage.

Database

The Database is the system for physical data storage and controlled access to stored data. It is the biggest specialized element for data modeling. The Database is a stereotyped component and a part of the UML Data Modeling Profile.

The database defines the type of the database, and the constraints for the data modeling like data types, stored procedures, syntax, etc. The level of database is the basic level of access to information, which can be refined on any of the deeper levels.

Database is used together with other types of components in a component diagram to define the dependencies between the application and databases.

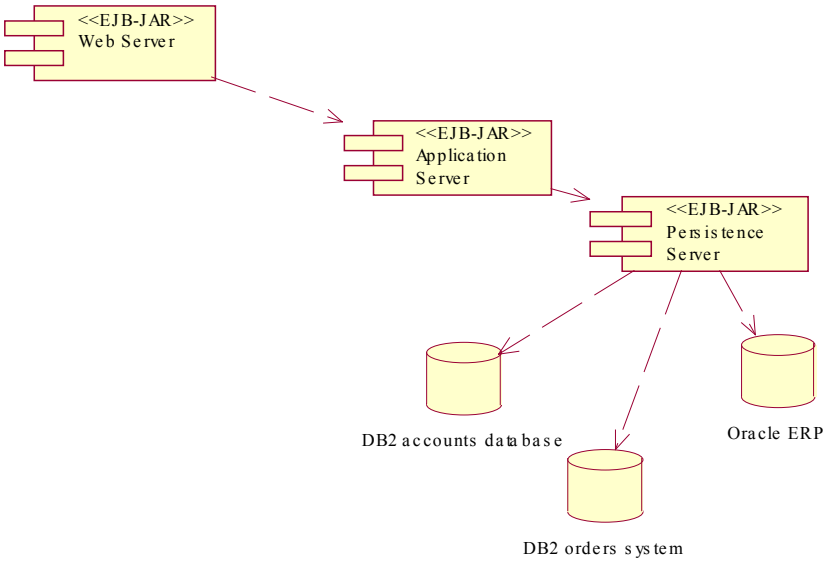

Figure 4 Databases in a Component Diagram

The value of the database component for the designer is in planning the accessibility of the database. The assignment of schemas to the database defines the basic structure of the information storage.

The database administrator uses the deployment diagram to recognize the communication problems between the application and the database, and to define the physical deployment of data together with deployment diagrams.

Schema

The basic unit of organization of tables is the schema. A schema is represented by a package, which is the organizational unit of UML. A Schema is a stereotyped package and a part of the UML Data Modeling Profile.

A schema is the basic unit an application works with. It is also a unit privileges can be granted for. Schemas are assigned to a database component on the next level of details.

Schemas are organized in a class diagram.

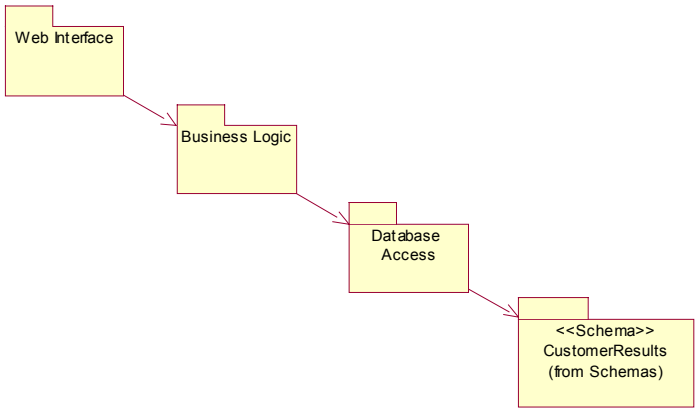

Figure 5 A class diagram explains schema dependencies

A schema should be assigned to a database, as the database defines the constraints on the language, data types, triggers available, possible database constraints, and types of stored procedures.

The schema is not just an organizational unit; it is also a security mechanism. The class diagram allows the database administrator and the analyst to recognize the dependencies between application-based packages and the data, which results in the usage patterns for the database.

Table

A table is the basic modeling structure of a relational database. It represents a set of records of the same structure, also called rows. Each of these records contains data. The information about the structure of a table is stored in the database itself.

A Table is a stereotyped class and part of the UML Data Modeling Profile.

Tables are represented in the **data model diagram**.

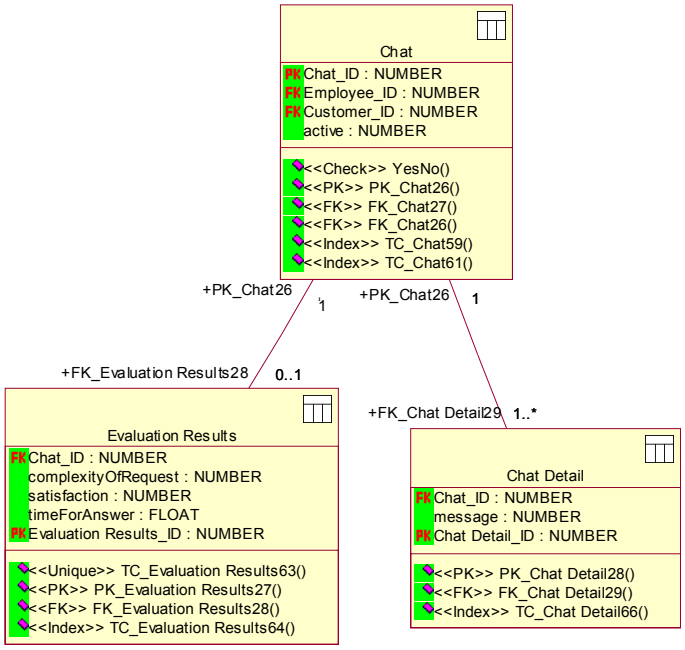

Figure 6 A data model diagram represents a view on tables and relationships

As the diagram is just a view of the model, it can represent a solution oriented focus on tables. This prevents from building a huge diagram, without any possibility to find the scope of the physical data model you are looking for.

The data model diagram with tables, views, relationships between tables, dependencies of views, and stored procedure containers is the exact representation of a part of the data dictionary. The data administrator can recognize the structure of the database in a more readable graphical representation.

On the design side, the tuning of the database is much easier with a graphical representation, because you are able to see the content of a table and the documentation of every detail. As tuning is always a manual process, the moving of data between tables is an essential capability, which can be realized just with knowledge of all constraints of the model.

The architect does not care in detail about the data model diagrams, but he can check if all of the information is represented in the database.

View

A View is a virtual Table. It represents a set of records of the same structure, exactly like a table, only with the difference that the physical source of the data is in other tables.

A View is a stereotyped class and part of the UML Data Modeling Profile.

Views are represented in the **data model diagram**.

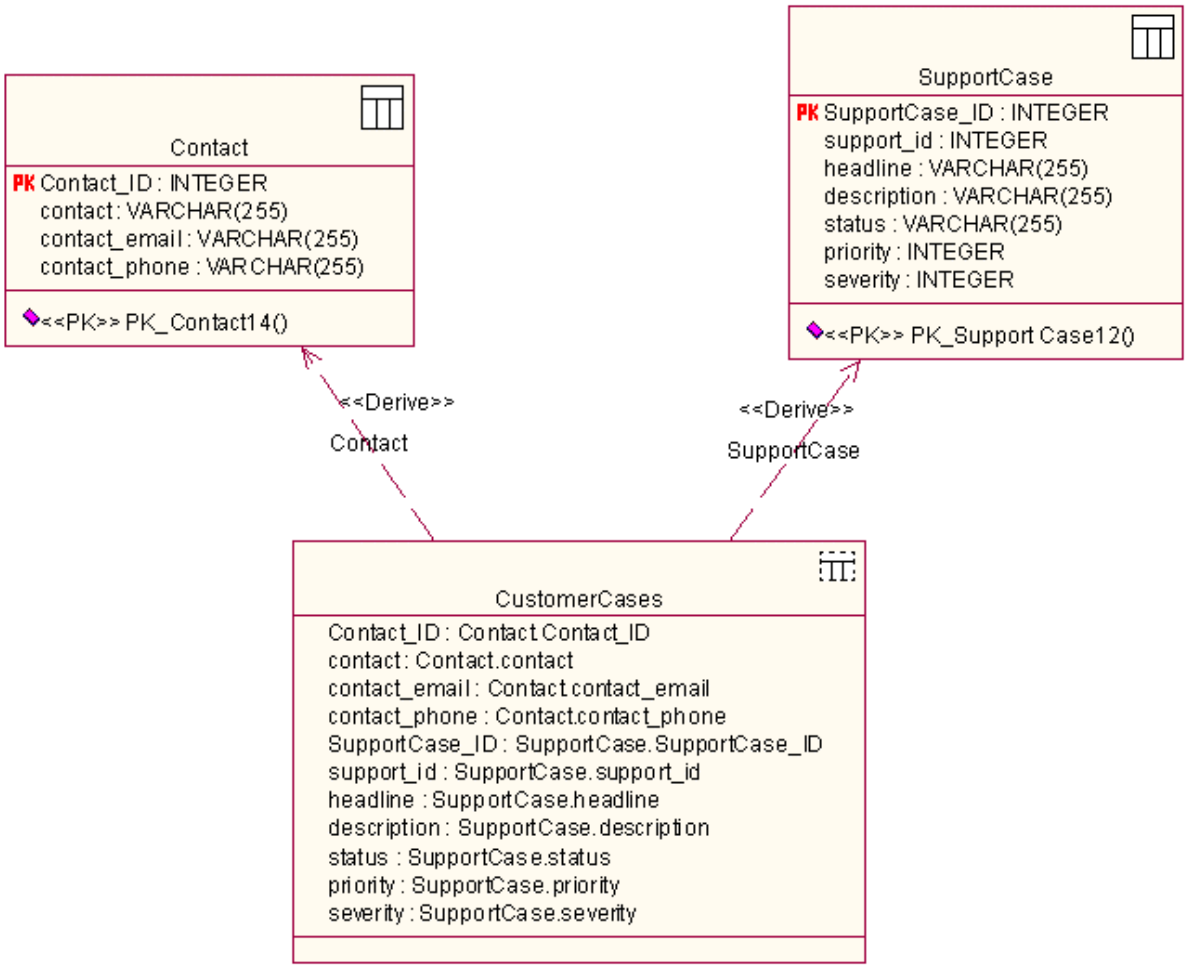

Figure 7 View - derived from two tables

As the diagram is just a view of the model, it can represent a solution oriented focus on views as well as tables.

The value of modeling of tables in views is not just in the definition of the data structure for the database. It is in the problem-oriented analysis of data, which cannot be done in the repository of the database itself. It is easy to discover dependencies between data structures and the source of the data.

Column

The column is the basic organizational element inside of the relational database. Every data has to be stored in a column of a row in a table. The columns are part of the UML Data Modeling Profile as stereotyped attributes.

The column adds the tagged value of the data type, which has to be specified. Also, the column data can be physically stored as an artifact in the database, or computed from other columns using an expression.

There are other tagged values with columns, which specify details of the data model, like null ability, and uniqueness.

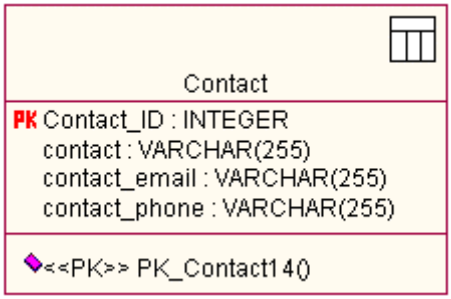

Figure 8 Table with four columns

The value of the column definition is in the specification of the data structure. Additionally it can be used for integration of different data sources and discovery of similarities and differences between implementations.

Key

Keys are used to access the table. Primary keys uniquely identify a row in a table, while foreign keys access data in other related tables.

The Primary Key is often content independent and automatically generated by the database to ease updates of data.

Foreign key is always derived from a relationship to other tables.

Keys are implementations of Key Constraints, which specify the content of the key (which columns build the key), as well as the physical implementation of the key. For easy recognition of the key columns in the table, they are tagged with either Primary Key (<<PK>>) or Foreign Key (<<FK>>) stereotype. In case a foreign key is used as a primary key, the combined key is marked with the (<<PFK>>) stereotype.

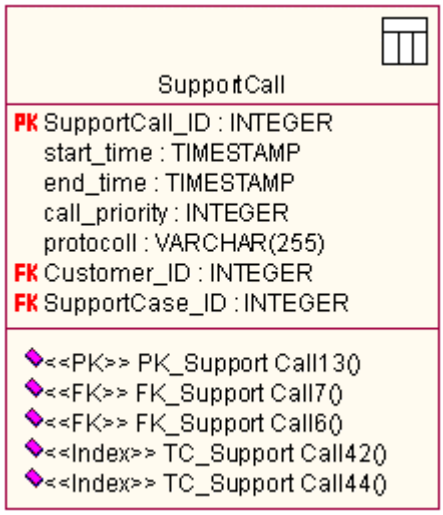

Figure 9 Table with Primary and Foreign Keys

Keys represent the identification of data. Therefore they are essential to recognize the complete structure of a database with all of the links between the data, to get information out of pure artifacts.

Index

An index is the physical data structure supporting faster data access. It does not change the quality of the data at all.

The Index is represented as a stereotype on the operation in the UML Data Modeling Profile.

An Index can, as well as the Key, contain several columns. The order of the columns in an index is essential.

The Index specification contains not only the columns of the index, but also the type of the index (uniqueness, etc.).

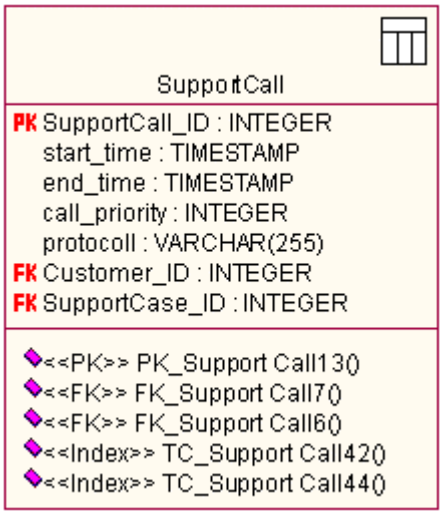

Figure 10 A Table with two Indexes

The value of an index is best seen when something keeps the application from good performance. Index is the first place to look at.

Constraint

A constraint is a rule applied to the database's structure. The rule can be applied to the column and/or table and can be limited to a schema or a database.

Several types of constraints are defined in the UML Data Modeling Profile, however all of them are implemented as stereotyped operations.

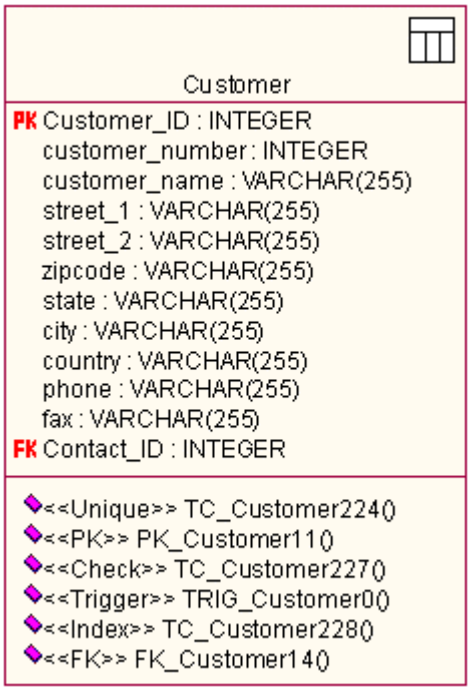

Figure 11 Table with Constraints

The value of defined constraints is in the details of the specification. Constraints describe the dynamic behavior of the database, whereas columns and tables do not.

Primary Key

The Primary Key Constraint defines a primary key of a table. Only one Primary Key is allowed per table.

The Primary Key Constraint uses the stereotype <<PK>> in the UML Data Modeling Profile.

Foreign Key

The Foreign Key is the constraint implementing a relationship. The constraint is always implemented on the child table.

The Foreign Key Constraint uses the stereotype <<FK>> in the UML Data Modeling Profile.

Trigger

An activity automatically executed as a result of other activity is a Trigger. It is always a side effect of the modification of data in the database and in most cases ensures a consistent behavior of the database.

A Trigger Constraint uses the stereotype <<Trigger>> in the UML Data Modeling Profile.

Value Verification

Values in the column can be verified using a trigger, which can not only use a fixed range of values to compare to, but also compare to other data in the database.

The Value Verification Constraint uses the stereotype <<Check>> in the UML Data Modeling Profile.

Uniqueness

The Unique constraint assures that all of the values of the specified column are different.

The Unique Constraint uses the stereotype <<Unique>> in the UML Data Modeling Profile.

Relationship

A dependency of any kind between tables in a data model is called a relationship.

A relationship is a summary of a stereotyped association and a set of primary and foreign keys. Every relationship is between a parent and child table, where a parent table must have a primary key defined. The child table creates a foreign key column and foreign key constraint to address the parent table.

A non-identifying association represents a relationship between two independent tables. The foreign key of the child table does not contain all of the primary key columns.

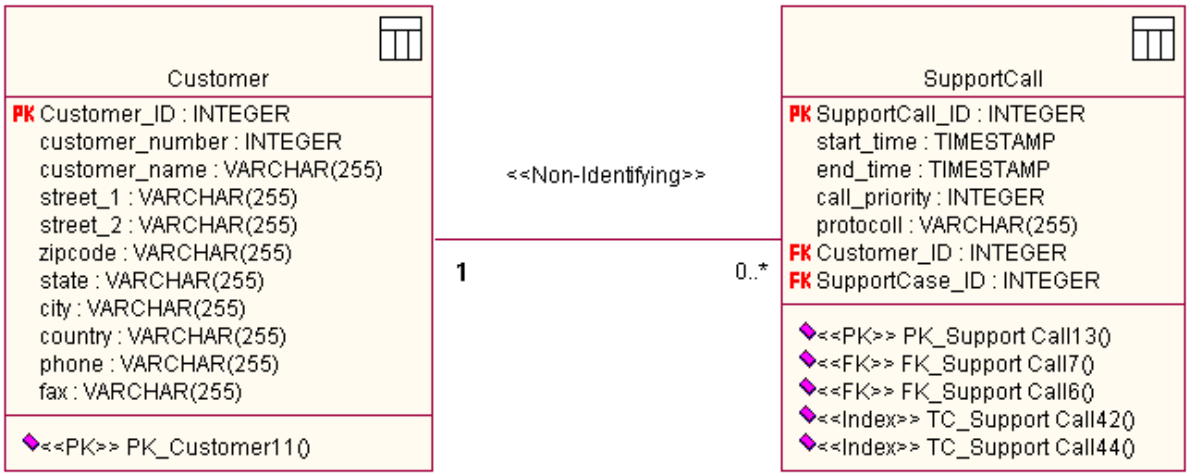

Figure 12 Non-Identifying Relationship

An identifying relationship is a relation between two dependent tables, where the child table cannot exist without the parent table. All of the primary keys of the parent table (Person, in this example) become both primary and foreign key columns in the child table (Account).

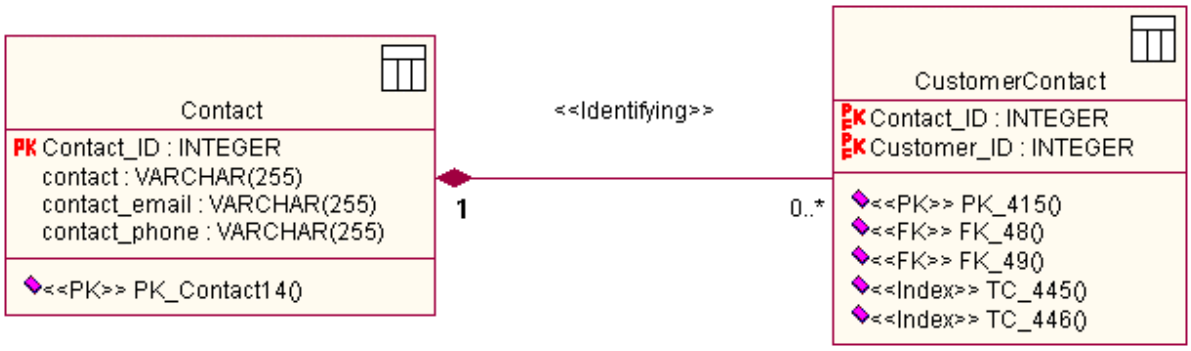

Figure 13 Identifying Relationship

A relationship has two roles associated with it. They define the role of one table in association with the other. It is possible to assign more than one relationship between two tables using different roles.

Each relationship creates migrated keys from the parent to the child table.

Summary

With the data modeling for UML profile, the UML fully supports data modeling needs. It allows the support of software development and data modeling with one unified language. Using the UML data modeling profile, IBM Rational Rose® Data Modeler unifies software development teams with a single, shared tool.

IBM software integrated solutions
IBM Rational supports a wealth of other offerings from IBM software. IBM software solutions can give you the power to achieve your priority business and IT goals.

- *DB2® software helps you leverage information with solutions for data enablement, data management, and data distribution.*
- *Lotus® software helps your staff be productive with solutions for authoring, managing, communicating, and sharing knowledge.*
- *Tivoli® software helps you manage the technology that runs your e-business infrastructure.*
- *WebSphere® software helps you extend your existing business-critical processes to the Web.*
- *Rational® software helps you improve your software development capability with tools, services, and best practices.*

Rational software from IBM
Rational software from IBM helps organizations create business value by improving their software development capability. The Rational software development platform integrates software engineering best practices, tools, and services. With it, organizations thrive in an on demand world by being more responsive, resilient, and focused. Rational's standards-based, cross-platform solution helps software development teams create and extend business applications, embedded systems and software products. Ninety-eight of the Fortune 100 rely on Rational tools to build better software, faster. Additional information is available at www.rational.com and www.therationaledge.com, the monthly e-zine for the Rational community.

IBM Corporation
Software Group
Route 100
Somers, NY 10589
U.S.A.

Printed in the United States of America
01-03 All Rights Reserved.
Made in the U.S.A.

The IBM home page on the Internet can be found at **ibm.com**