**Working with Table Relationships**

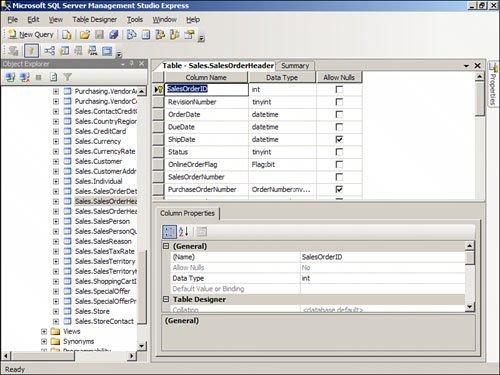
It is easy to view all the foreign key relationships in which a table is participating. Follow these

steps:

**1.** Right-click the table and select Modify. The design of the table appears (see Figure 5.9).

**Figure 5.9. While the design of the table is visible, you are able to**

**select the Relationships tool on the toolbar.**

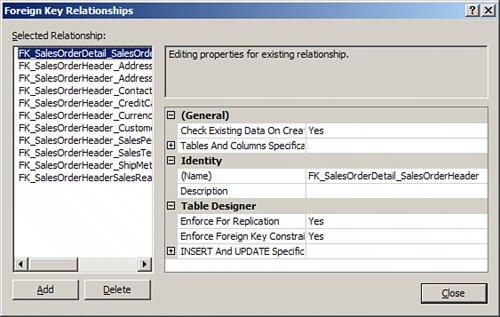


**2.** Click the Relationships tool on the toolbar. The Foreign Key Relationships dialog appears (see

Figure 5.10).

**Figure 5.10. The Foreign Key Relationships dialog enables you to**

**work with the relationships associated with a table.**



[View full size image]

**3.** Click a relationship to select it. The properties of that relationship appear.

**Adding a Foreign Key Relationship**

The Foreign Key Relationships dialog also allows you to add an index. Simply click the Add button.

A new relationship appears with a default name and without a description. Before you take any

further action you should supply the Tables and Columns Specification covered in the section "How

to Designate Table and Column Specifications." You must designate the table and column

specification before SQL Server Express will accept the new relationship.

**Deleting a Foreign Key Relationship**

Deleting a foreign key relationship is easy. Follow these steps:

**1.** While in the Foreign Key Relationships dialog, select the relationship you wish to

remove.

**2.** Click the Delete button. SQL Server Express removes the relationship without warning.

**How to Designate Table and Column Specifications**

By entering a Tables and Columns specification, you designate the foreign key table that will

participate in the relationship, the field in the foreign key table that will participate in the

relationship, and the field in the current table that will participate in the relationship. To work with

the Tables and Columns Specification, follow these steps:

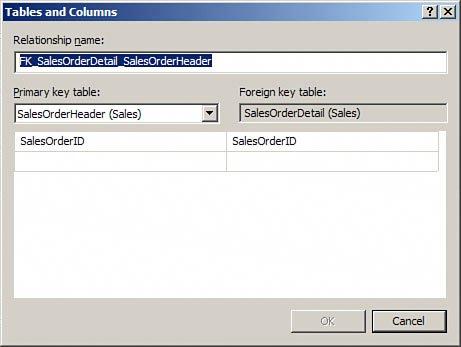
**1.** In the Foreign Key Relationships dialog, click to select Tables and Columns Specification.

**1.** Click the Build button (...) that appears to the right. The Tables and Columns dialog appears

(see Figure 5.11).

**Figure 5.11. The Tables and Columns dialog enables you to designate**

**what tables and fields participate in the relationship.**



**3.** If you want to, modify the relationship name. You will generally want to rename the

relationship to more accurately reflect the relationship that you are creating (for example,

FK\_Customers\_Orders\_CustomerID).



**4.** Click to select the primary key table that will participate in the relationship. For example, if

you are creating foreign keys for the Orders table, you would designate the Customer table as

the primary key table.

**5.** Use the drop-down on the left (under the primary key table) to select the field(s) that will

participate in the relationship. For example, in the foreign key relationship between Orders and

Customers, the CustomerID in the Customers table participates in the relationship.

**6.** Use the drop-down on the right (under the foreign key table) to select the field(s) in the

current table that will participate in the relationship. The complete dialog appears in Figure

5.12. In the relationship between the Orders table and the Customers table, the foreign key

field participating in the relationship would be the CustomerID field.

**Figure 5.12. The Completed Tables and Columns dialog, showing the**

**relationship between the Customers table and the Orders table.**

**7.** Click OK to complete the process. SQL Server Express returns you to the Foreign Key

Relationships dialog.

**How to Add a Relationship Name and Description**

It is helpful to provide a descriptive name for each relationship that you add, and for you to provide

a brief description of each relationship. This way when you are viewing a relationship in the Foreign

Key Relationships window you can easily see the nature of the relationship that you have selected.

To enter or change a name for the relationship simple click the (Name) property for the

relationship. Enter or change the name as you desire.

To enter a description for the relationship, click the Description property for the index. Enter a short

description of your choice.

**How to Determine when Foreign Key Relationships**

**Constrain the Data Entered in a Column**

As you can see, establishing a relationship is quite easy. Establishing the right kind of relationship

is a little more difficult. When you attempt to establish a relationship between two tables, SQL

Server makes some decisions based on a few predefined factors:

It establishes a one-to-many relationship if one of the related fields is a primary key or has a

unique index.

It establishes a one-to-one relationship if both the related fields are primary keys or have

unique indexes.

It cannot create a relationship if neither of the related fields is a primary key and neither has

a unique index.

As covered earlier in this hour, *referential integrity* consists of a series of rules that SQL Server

applies to ensure that the relationships between tables are maintained properly. At the most basic

level, referential integrity rules prevent the creation of orphan records in the table on the many

side of the one-to-many relationship. After establishing a relationship between a Customers table

and an Orders table, for example, all orders in the Orders table must be related to a particular

customer in the Customers table. Before you can establish referential integrity between two tables,

the following conditions must be met:

The matching field on the one side of the relationship must be a primary key field or must

have a unique index.

The matching fields must have the same data types. They also must have the same size.

Number fields on both sides of the relationship must have the same size (int, for example).

Both tables must be part of the same database.

If you opt to set the Check Existing Data on Creation option to Yes, existing data within the

two tables cannot violate any referential integrity rules. All orders in the Orders table must

relate to existing customers in the Customers table, for example.

After you establish referential integrity between two tables, SQL Server applies the following rules:

You cannot enter a value in the foreign key of the related table that does not exist in the

primary key of the primary table. For example, you cannot enter a value in the CustomerID

field of the Orders table that does not exist in the CustomerID field of the Customers table.

You cannot delete a record from the primary table if corresponding records exist in the related

table. For example, you cannot delete a customer from the Customers table if related records



exist in the Orders table (records with the same value in the CustomerID field) unless you

designate a Delete Rule (see the section that follows).

You cannot change the value of a primary key on the one side of a relationship if

corresponding records exist in the related table. For example, you cannot change the value in

the CustomerID field of the Customers table if corresponding orders exist in the Orders table

unless you designate an Update rule in the Foreign Key Relationships dialog for the

relationship (see the section that follows).

If any of the previous three rules is violated and referential integrity is being enforced between the

tables, an appropriate error message is displayed, as shown in Figure 5.13.

**Figure 5.13. An appropriate error message appears if referential integrity**

**is violated.**

SQL Server's default behavior is to prohibit the deletion of parent records that have associated child

records and to prohibit the change of a primary key value of a parent record when that parent has

associated child records. You can override these restrictions by using the INSERT and UPDATE

specification, covered in the next section.

For now, let's see how you can establish referential integrity between the tables in your database.

The process is as follows:

**1.** From the Foreign Key Relationships dialog, select the relationship for which you want to

establish referential integrity.

**2.** Set the Enforce Foreign Key Constraint property to Yes. This step alone is all you need to

establish referential integrity.

**3.** If you want to check existing data when you save your changes to ensure that they do not

violate the referential integrity rules, set the Check Existing Data on Creation or Re-enabling

option to Yes.

**4.** If you are utilizing replication, and want to enforce referential integrity during the

synchronization process, set the Enforce for Replication property to Yes.

**Designating Insert and Update Specifications**

SQL Server enables you to define rules that dictate what will happen when the user deletes or

updates a record. You can find these rules under the INSERT and UPDATE Specification node of the

Foreign Key Relationships window. The text that follows explores this node, and why and how you

should use it.

**The Delete Rule**

By setting the Delete rule, you determine what happens when the user deletes a record on the one

side of a one-to-many relationship. For example, by setting the Delete rule to Cascade you

establish the rule so that the user can delete a record on the one side of a one-to-many

relationship, even if related records exist in the table on the many side of the relationship. The user

can delete a customer even if the customer has existing orders, for example. Referential integrity is

maintained between the tables because SQL Server automatically deletes all related records in the

child table.

If you attempt to delete a record from the table on the one side of a one-to-many relationship and

no related records exist in the table on the many side of the relationship, you are able to delete the

record. On the other hand, if you attempt to delete a record from the table on the one side of a

one-to-many relationship and related records exist in the child table, you will delete the record

from the parent table as well as any related records in the child table.

**Did you Know?: Tip**

Setting the Delete rule to Cascade is not always appropriate. It is an excellent feature,

but you should use it prudently. Although it is usually appropriate to cascade delete from

an Orders table to an Order Details table, for example, it generally is not appropriate to

cascade delete from a Customers table to an Orders table. This is because you generally

do not want all your order history deleted from the Orders table if for some reason you

want to delete a customer. Deleting the order history causes important information, such

as your profit and loss history, to change. It therefore is appropriate to prohibit this type

of deletion and handle the customer in some other way, such as marking him as inactive

or archiving his data. On the other hand, if you delete an order because it was canceled,

you probably want the corresponding order detail information to be removed as well. In

this case, the Cascade option is appropriate. You need to make the appropriate decision

in each situation, based on business needs. The important thing is to carefully consider

the implications of each option before making your decision.

**The Update Rule**

With the Update rule set to Cascade, the user can change the primary key value of the record on

the one side of the relationship. When the user makes an attempt to modify the field joining the

two tables on the one side of the relationship, the change is cascaded down to the foreign key field

on the many side of the relationship. This is useful if the primary key field is modifiable. For

example, a purchase number on a purchase order master record may be updateable. If the user

modifies the purchase order number of the parent record, you would want to cascade the change to

the associated detail records in the purchase order detail table.

**By the Way: Note**

There is no need to select the Cascade option when the related field on the one side of the

relationship is an identity field. An identity field can never be modified. The Cascade

option has no effect on identity fields.

**By the Way: Note**

Other options for the Delete and Update rules include No Action, Set Null, and Set

Default. No Action, the default value, does nothing, and therefore does not allow the

deletion of parent records that have children or the modification of the key field(s) of

parent records that have children. Set Null sets the value of the foreign key field to Null.

Finally, Set Default sets the value of the foreign key field to its default value.