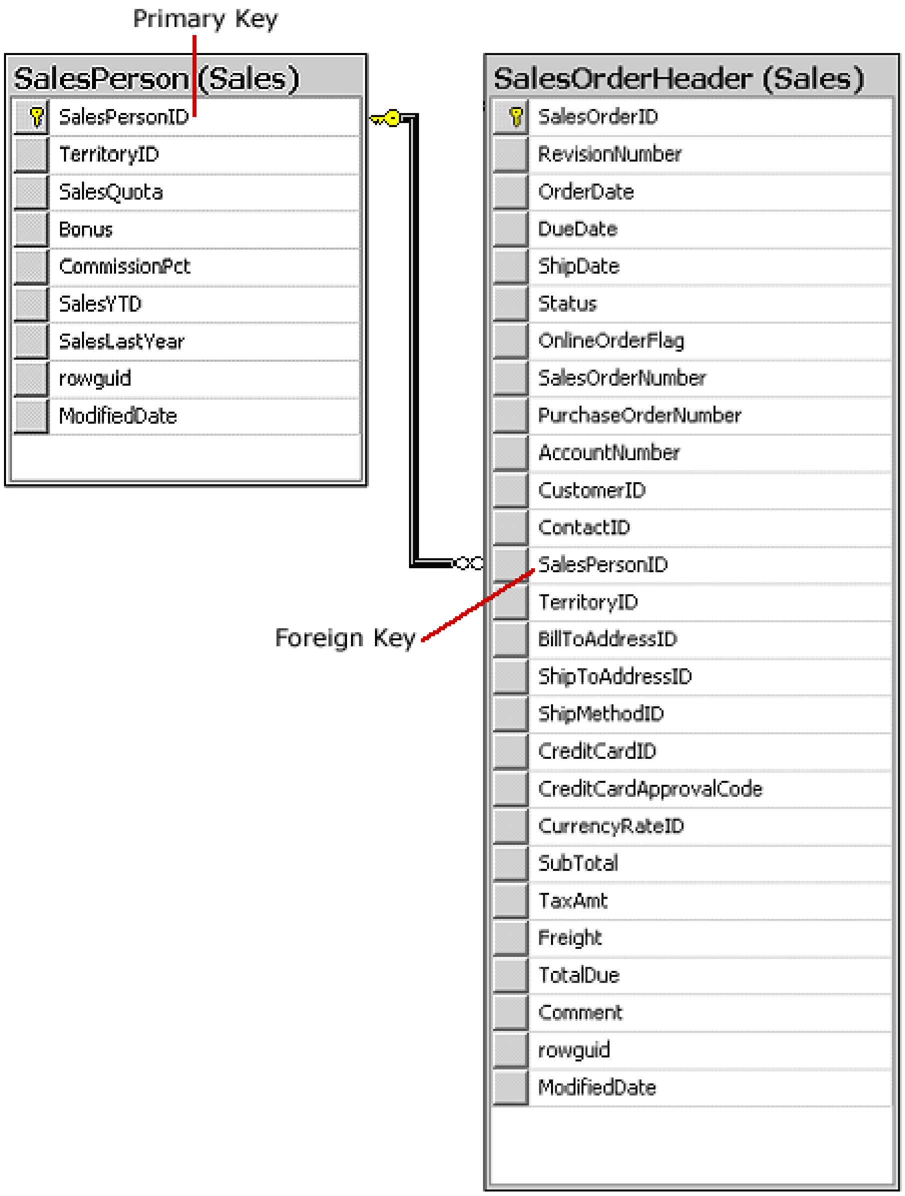
FOREIGN KEY Constraints

A foreign key (FK) is a column or combination of columns that is used to establish and enforce a link between the data in two tables. You can create a foreign key by defining a FOREIGN KEY constraint when you create or modify a table.

In a foreign key reference, a link is created between two tables when the column or columns that hold the primary key value for one table are referenced by the column or columns in another table. This column becomes a foreign key in the second table.

For example, the **Sales.SalesOrderHeader** table in the AdventureWorks2008R2 database has a link to the **Sales.SalesPerson** table because there is a logical relationship between sales orders and salespeople. The **SalesPersonID** column in the **SalesOrderHeader** table matches the primary key column of the **SalesPerson** table. The **SalesPersonID** column in the **SalesOrderHeader** table is the foreign key to the **SalesPerson** table.



A FOREIGN KEY constraint does not have to be linked only to a PRIMARY KEY constraint in another table; it can also be defined to reference the columns of a UNIQUE constraint in another table. A FOREIGN KEY constraint can contain null values; however, if any column of a composite FOREIGN KEY constraint contains null values, verification of all values that make up the FOREIGN KEY constraint is skipped. To make sure that all values of a composite FOREIGN KEY constraint are verified, specify NOT NULL on all the participating columns.

Note

A FOREIGN KEY constraint can reference columns in tables in the same database or within the same table. These are called *self-referencing* tables. For example, consider an employee table that contains three columns: **employee_number**, **employee_name**, and **manager_employee_number**. Because the manager is also an employee, there is a foreign key relationship from the **manager_employee_number** column to the **employee_number** column.

Referential Integrity

Although the main purpose of a FOREIGN KEY constraint is to control the data that can be stored in the foreign key table, it also controls changes to data in the primary key table. For example, if the row for a salesperson is deleted from the **Sales.SalesPerson** table, and the salesperson's ID is used for sales orders in the **Sales.SalesOrderHeader** table, the relational integrity between the two tables is broken; the deleted salesperson's sales orders are orphaned in the **SalesOrderHeader** table without a link to the data in the **SalesPerson** table.

A FOREIGN KEY constraint prevents this situation. The constraint enforces referential integrity by guaranteeing that changes cannot be made to data in the primary key table if those changes invalidate the link to data in the foreign key table. If an attempt is made to delete the row in a primary key table or to change a primary key value, the action will fail when the deleted or changed primary key value corresponds to a value in the FOREIGN KEY constraint of another table. To successfully change or delete a row in a FOREIGN KEY constraint, you must first either delete the foreign key data in the foreign key table or change the foreign key data in the foreign key table, which links the foreign key to different primary key data.

Indexing FOREIGN KEY Constraints

Creating an index on a foreign key is often useful for the following reasons:

- Changes to PRIMARY KEY constraints are checked with FOREIGN KEY constraints in related tables.
- Foreign key columns are frequently used in join criteria when the data from related tables is combined in queries by matching the column or columns in the FOREIGN KEY constraint of one table with the primary or unique key column or columns in the other table. An index enables the Database Engine to quickly find related data in the foreign key table. However, creating this index is not required. Data from two related tables can be combined even if no PRIMARY KEY or FOREIGN KEY constraints are defined between the tables, but a foreign key relationship between two tables indicates that the two tables have been optimized to be combined in a query that uses the keys as its criteria. For more information about using FOREIGN KEY constraints with joins, see Join Fundamentals and Query Types and Indexes.

Number of FOREIGN KEY Constraints in a Table

SQL Server does not have a predefined limit on either the number of FOREIGN KEY constraints a table can contain (which reference other tables), or the number of FOREIGN KEY constraints owned by other tables that reference a specific table. Nevertheless, the actual number of FOREIGN KEY constraints is limited by your hardware configuration and by the design of your database and application. We recommend that a table contain no more than 253 FOREIGN KEY constraints, and that it be referenced by no more than 253 FOREIGN KEY constraints. Consider the cost of enforcing FOREIGN KEY constraints when you design your database and applications.

See Also

Reference

CREATE TABLE (Transact-SQL)
ALTER TABLE (Transact-SQL)
DROP TABLE (Transact-SQL)

Concepts

Creating and Modifying FOREIGN KEY Constraints
Indexes

Community Additions

The last statement needs more explaining or may be an error.

I am referring to the statement "We recommend that a table contain no more than 253 FOREIGN KEY constraints, and that it be referenced by no more than 253 FOREIGN KEY constraints. Consider the cost of enforcing FOREIGN KEY constraints when you design your database and applications." . It is not really clear what the recommendation means. Is it that a table not contain more than 253 Foreign key columns per table row? Or that the table itself have at most 253 foreign keys defined? Also, not sure what it means that a table be referenced no more than 253 Foreign key constraints.

DataByter
11/30/2012

© 2016 Microsoft