SQL Server Tables, Keys, and Relationships

Tovartech.org

Beginner Hands-On Lab Manual Using SSMS and T-SQL

Learn how to design related SQL Server tables using primary keys, foreign keys, UNIQUE constraints, NOT NULL rules, and JOIN queries.

Audience	Beginners learning SQL Server table design
Estimated Time	2 to 3 hours
Tool	SQL Server Management Studio (SSMS)
Lab Database	RelationshipTrainingDB
Goal	Create normalized tables and enforce relationships with keys

Training Safety: Use a test SQL Server instance only. Do not run practice scripts against production databases.

Lab Objectives

- Understand tables, rows, columns, data types, and schemas.
- Create SQL Server tables using clean naming standards.
- Create primary keys to uniquely identify rows.
- Create UNIQUE and NOT NULL constraints to improve data quality.
- Create foreign keys to enforce relationships between tables.
- Understand one-to-many and many-to-many relationships.
- Test referential integrity by attempting valid and invalid inserts.
- Use JOIN queries to read data across related tables.

Quick Concept Reference

Concept	What It Means	Example
Table	Stores rows and columns of related data	dbo.Servers
Primary Key	Uniquely identifies each row	ServerID
Foreign Key	Links a child table to a parent table	Applications.ServerID references Servers.ServerID
UNIQUE	Prevents duplicate values in a column or column group	ServerName must be unique
NOT NULL	Requires a value before a row can be saved	Environment cannot be blank

Module 1: Tables and Relational Thinking

A table should represent one subject. For an IT inventory example, servers, applications, owners, and tickets should not all be stored in one large spreadsheet-style table. Instead, related subjects should be split into separate tables and connected with keys.

- Servers table: one row per server.
- Applications table: one row per application installed on a server.
- Owners table: one row per application or system owner.
- ServerApplications bridge table: connects servers and applications when many servers can host many applications.

Design Tip: A table should avoid repeated groups such as App1, App2, App3. Repeating groups usually mean another related table is needed.

```
One-to-many example:
Owners 1 ---- many Applications

Many-to-many example:
Servers many ---- many Applications
Use a bridge table:
Servers 1 ---- many ServerApplications many ---- 1 Applications
```

Module 2: Create the Training Database

Open SSMS, connect to a test SQL Server instance, select New Query, and run the setup script below.

```
CREATE DATABASE RelationshipTrainingDB;
GO

USE RelationshipTrainingDB;
GO
```

Module 3: Create Parent Tables with Primary Keys

Parent tables are referenced by other tables. In this lab, Owners, Servers, and Applications are parent tables.

```
CREATE TABLE dbo.Owners
(
    OwnerID INT IDENTITY(1,1) CONSTRAINT PK_Owners PRIMARY KEY,
    OwnerName VARCHAR(100) NOT NULL,
    TeamName VARCHAR(100) NOT NULL,
    EmailAddress VARCHAR(150) NULL,
    CONSTRAINT UQ_Owners_EmailAddress UNIQUE (EmailAddress)
);
GO

CREATE TABLE dbo.Servers
(
    ServerID INT IDENTITY(1,1) CONSTRAINT PK_Servers PRIMARY KEY,
    ServerName VARCHAR(100) NOT NULL,
    OperatingSystem VARCHAR(100) NOT NULL,
    Environment VARCHAR(30) NOT NULL,
    CONSTRAINT UQ_Servers_ServerName UNIQUE (ServerName)
);
GO

CREATE TABLE dbo.Applications
(
    AppID INT IDENTITY(1,1) CONSTRAINT PK_Applications PRIMARY KEY,
    AppName VARCHAR(100) NOT NULL,
    VersionNumber VARCHAR(50) NULL,
    OwnerID INT NOT NULL,
    CONSTRAINT UQ_Applications_AppName UNIQUE (AppName),
    CONSTRAINT FK_Applications_Owners
        FOREIGN KEY (OwnerID)
        REFERENCES dbo.Owners(OwnerID)
);
GO
```

Primary keys enforce row uniqueness. SQL Server automatically creates a unique index to enforce a primary key constraint.

Module 4: Create Child and Bridge Tables with Foreign Keys

Foreign keys enforce valid relationships. The child table stores the foreign key column, and the parent table stores the referenced key.

```
CREATE TABLE dbo.ServerApplications
(
    ServerID INT NOT NULL,
    AppID INT NOT NULL,
    InstalledDate DATE NOT NULL DEFAULT (GETDATE()),
    InstallStatus VARCHAR(30) NOT NULL DEFAULT ('Installed'),
    CONSTRAINT PK_ServerApplications PRIMARY KEY (ServerID, AppID),
    CONSTRAINT FK_ServerApplications_Servers
        FOREIGN KEY (ServerID)
        REFERENCES dbo.Servers(ServerID),
    CONSTRAINT FK_ServerApplications_Applications
        FOREIGN KEY (AppID)
        REFERENCES dbo.Applications(AppID)
);
GO

CREATE TABLE dbo.ServiceTickets
(
    TicketID INT IDENTITY(1,1) CONSTRAINT PK_ServiceTickets PRIMARY KEY,
    ServerID INT NOT NULL,
    TicketTitle VARCHAR(150) NOT NULL,
    TicketStatus VARCHAR(30) NOT NULL,
    CreatedDate DATE NOT NULL DEFAULT (GETDATE()),
    CONSTRAINT FK_ServiceTickets_Servers
        FOREIGN KEY (ServerID)
        REFERENCES dbo.Servers(ServerID)
);
GO
```

The ServerApplications table uses a composite primary key. The combination of ServerID plus AppID must be unique, which prevents assigning the same application to the same server twice.

Module 5: Insert Valid Parent and Child Data

Insert into parent tables first. Then insert into child tables that depend on parent keys.

```
INSERT INTO dbo.Owners (OwnerName, TeamName, EmailAddress)
VALUES
('Noe Tovar', 'ET Applications', 'noe.tovar@example.com'),
('Gary Ballard-Toney', 'Management', 'gary.ballard@example.com'),
('Teresa Example', 'Application Support', 'teresa.example@example.com');
GO

INSERT INTO dbo.Servers (ServerName, OperatingSystem, Environment)
VALUES
('APPWEB01', 'Windows Server 2022', 'Production'),
('APPLNX01', 'Red Hat Enterprise Linux 9', 'Production'),
('APPTEST01', 'Windows Server 2019', 'Test');
GO

INSERT INTO dbo.Applications (AppName, VersionNumber, OwnerID)
VALUES
('SQL Server', '2022', 1),
('Apache HTTP Server', '2.4', 1),
```

```
('Tomcat', '10.1', 3);
GO

INSERT INTO dbo.ServerApplications (ServerID, AppID, InstalledDate)
VALUES
(1, 1, '2026-07-01'),
(2, 2, '2026-07-02'),
(2, 3, '2026-07-02'),
(3, 3, '2026-07-03');
GO

INSERT INTO dbo.ServiceTickets (ServerID, TicketTitle, TicketStatus)
VALUES
(1, 'Review SQL Server backup configuration', 'Open'),
(2, 'Investigate Apache service restart', 'In Progress'),
(3, 'Validate test application deployment', 'Open');
GO
```

Module 6: Test Data Integrity Rules

The next scripts intentionally test constraints. Failed statements are useful because they prove SQL Server is protecting the data model.

Lab 6.1: Test UNIQUE Constraint

```
-- This should fail because APPWEB01 already exists.
INSERT INTO dbo.Servers (ServerName, OperatingSystem, Environment)
VALUES ('APPWEB01', 'Windows Server 2022', 'Production');
```

Lab 6.2: Test NOT NULL Constraint

```
-- This should fail because Environment is required.
INSERT INTO dbo.Servers (ServerName, OperatingSystem, Environment)
VALUES ('APPBAD01', 'Windows Server 2022', NULL);
```

Lab 6.3: Test Foreign Key Constraint

```
-- This should fail because ServerID 999 does not exist.
INSERT INTO dbo.ServiceTickets (ServerID, TicketTitle, TicketStatus)
VALUES (999, 'Invalid server relationship test', 'Open');
```

Foreign key constraints help prevent orphan records, such as a ticket assigned to a server that does not exist.

Module 7: Query Related Tables with JOINs

Once relationships are in place, JOIN queries can combine meaningful data from related tables.

Lab 7.1: Servers and Service Tickets

```
SELECT
    s.ServerName,
    s.Environment,
    t.TicketID,
    t.TicketTitle,
    t.TicketStatus
FROM dbo.Servers s
```

```
INNER JOIN dbo.ServiceTickets t
    ON s.ServerID = t.ServerID
ORDER BY s.ServerName, t.TicketID;
```

Lab 7.2: Applications with Owners

```
SELECT
    a.AppName,
    a.VersionNumber,
    o.OwnerName,
    o.TeamName
FROM dbo.Applications a
INNER JOIN dbo.Owners o
    ON a.OwnerID = o.OwnerID
ORDER BY a.AppName;
```

Lab 7.3: Full Server Application Inventory

```
SELECT
    s.ServerName,
    s.OperatingSystem,
    s.Environment,
    a.AppName,
    a.VersionNumber,
    o.OwnerName,
    sa.InstalledDate,
    sa.InstallStatus
FROM dbo.ServerApplications sa
INNER JOIN dbo.Servers s
    ON sa.ServerID = s.ServerID
INNER JOIN dbo.Applications a
    ON sa.AppID = a.AppID
INNER JOIN dbo.Owners o
    ON a.OwnerID = o.OwnerID
ORDER BY s.ServerName, a.AppName;
```

Module 8: View Relationships in SSMS

- In Object Explorer, expand RelationshipTrainingDB.
- Expand Database Diagrams.
- If prompted, allow SQL Server to create diagram support objects in the training database.
- Create a new database diagram.
- Add Owners, Servers, Applications, ServerApplications, and ServiceTickets.
- Review the key icons and relationship connector lines.

Visual diagrams help beginners understand parent tables, child tables, and bridge tables. Diagrams are especially helpful when explaining one-to-many and many-to-many designs.

Challenge Exercises

#	Exercise	Done

#	Exercise	Done
1	Create a Locations table with LocationID, LocationName, City, and StateCode.	☐
2	Add LocationID to dbo.Servers as a nullable column.	☐
3	Create a foreign key from dbo.Servers.LocationID to dbo.Locations.LocationID.	☐
4	Insert at least two locations and update each server with a location.	☐
5	Write a JOIN query that displays ServerName, Environment, LocationName, and City.	☐
6	Attempt to update a server to LocationID 999 and observe the foreign key error.	☐
7	Create a view named vwServerApplicationInventory using the full inventory JOIN query.	☐
8	Draw or describe the final relationship diagram in your own words.	☐

Mini Cheat Sheet

```
-- Primary key
CONSTRAINT PK_TableName PRIMARY KEY (IDColumn)

-- Unique constraint
CONSTRAINT UQ_TableName_ColumnName UNIQUE (ColumnName)

-- Foreign key
CONSTRAINT FK_ChildTable_ParentTable
    FOREIGN KEY (ParentID)
    REFERENCES dbo.ParentTable(ParentID)

-- Composite primary key
CONSTRAINT PK_BridgeTable PRIMARY KEY (ColumnA, ColumnB)

-- Join parent and child
SELECT p.ParentName, c.ChildName
FROM dbo.ParentTable p
INNER JOIN dbo.ChildTable c
    ON p.ParentID = c.ParentID;
```

Final Lab Completion Checklist

Skill	Complete
Created the training database	☐
Created parent tables	☐
Created child and bridge tables	☐
Used identity columns	☐
Created primary keys	☐
Created UNIQUE constraints	☐
Created NOT NULL columns	☐
Created foreign keys	☐
Inserted valid parent data before child data	☐
Tested unique constraint failure	☐
Tested not-null constraint failure	☐
Tested foreign key constraint failure	☐
Queried one-to-many relationships	☐
Queried many-to-many relationships	☐
Viewed relationships in SSMS	☐
Completed challenge exercises	☐

References

Microsoft Learn: Primary and foreign key constraints - https://learn.microsoft.com/en-us/sql/relational-databases/tables/primary-and-foreign-key-constraints

Microsoft Learn: Create foreign key relationships - https://learn.microsoft.com/en-us/sql/relational-databases/tables/create-foreign-key-relationships

For more information and resources, please visit https://www.tovartech.org