Create and Alter Constraints

Section Objective:

This objective may include but is not limited to: PRIMARY KEY, FOREIGN KEY, UNIQUE, CHECK, cascading referential integrity, enabling/disabling, NOCHECK; SET IDENTITY_INSERT

Constraints Definition:

A referential CONSTRAINT definition defines an integrity condition, restrictions for columns values, which must be satisfied by all the rows in two tables. The resultant dependency between two tables affects changes to the rows contained in them.

Entity Integrity

Entity integrity ensures each row in a table is a uniquely identifiable entity. You can apply entity integrity to a table by specifying a PRIMARY KEY constraint. For example, the ProductID column of the Products table is a primary key for the table.

Referential Integrity

Referential integrity ensures the relationships between tables remain preserved as data is inserted, deleted, and modified. You can apply referential integrity using a FOREIGN KEY constraint. The ProductID column of the Order Details table has a foreign key constraint applied referencing the Orders table. The constraint prevents an Order Detail record from using a ProductID that does not exist in the database. Also, you cannot remove a row from the Products table if an order detail references the ProductID of the row.

Entity and referential integrity together form key integrity.

Domain Integrity

Domain integrity ensures the data values inside a database follow defined rules for values, range, and format. A database can enforce these rules using a variety of techniques, including CHECK constraints, UNIQUE constraints, and DEFAULT constraints. These are the constraints we will cover in this article, but be aware there are other options available to enforce domain integrity. Even the selection of the data type for a column enforces domain integrity to some extent. For instance, the selection of datetime for a column data type is more restrictive than a free format varchar field.

The following list gives a sampling of domain integrity constraints.

1. A product name cannot be NULL.
2. A product name must be unique.

3. The date of an order must not be in the future.
4. The product quantity in an order must be greater than zero.

Different Type(s) of Constraints

Constraint Type	Description
Primary Key	A primary key is used to uniquely identify each row in a table. It can either be part of the actual record itself , or it can be an artificial field (one that has nothing to do with the actual record). A primary key can consist of one or more fields on a table. When multiple fields are used as a primary key, they are called a composite key. Primary keys can be specified either when the table is created (using CREATE TABLE) or by changing the existing table structure (using ALTER TABLE).
Foreign Key	A foreign key is a field (or fields) that points to the primary key of another table. The purpose of the foreign key is to ensure referential integrity of the data. In other words, only values that are supposed to appear in the database are permitted.
Unique	You can use UNIQUE constraints to make sure that no duplicate values are entered in specific columns that do not participate in a primary key. Although both a UNIQUE constraint and a PRIMARY KEY constraint enforce uniqueness, use a UNIQUE constraint instead of a PRIMARY KEY constraint when you want to enforce the uniqueness of a column, or combination of columns, that is not the primary key. Multiple UNIQUE constraints can be defined on a table, whereas only one PRIMARY KEY constraint can be defined on a table. Also, unlike PRIMARY KEY constraints, UNIQUE constraints allow for the value NULL. However, as with any value participating in a UNIQUE constraint, only one null value is allowed per column. A UNIQUE constraint can be referenced by a FOREIGN KEY constraint.
Check	CHECK constraints enforce domain integrity by limiting the values that are accepted by a column. They are similar to FOREIGN KEY constraints in that they control the values that are put in a column. The difference is in how they determine which values are valid: FOREIGN KEY constraints obtain the list of valid values from another table, and CHECK constraints determine the valid values from a logical expression that is not based on data in another column. You can create a CHECK constraint with any logical (Boolean) expression that returns TRUE or FALSE based on the logical operators.

	You can apply multiple CHECK constraints to a single column. You can also apply a single CHECK constraint to multiple columns by creating it at the table level.
NOCHECK	Special situations often arise in database development where it is convenient to temporarily relax the rules. For example, it is often easier to load initial values into a database one table at a time, without worrying with foreign key constraints and checks until all of the tables have finished loading. After the import is complete, you can turn constraint checking back on and know the database is once again protecting the integrity of the data. **Note:** The only constraints you can disable are the FOREIGN KEY constraint, and the CHECK constraint. PRIMARY KEY, UNIQUE, and DEFAULT constraints are always active. Disabling a constraint using SQL is done through the ALTER TABLE command.
SET IDENTITY_INSERT	Allows explicit values to be inserted into the identity column of a table. At any time, only one table in a session can have the IDENTITY_INSERT property set to ON. If a table already has this property set to ON, and a SET IDENTITY_INSERT ON statement is issued for another table, Microsoft® SQL Server™ returns an error message that states SET IDENTITY_INSERT is already ON and reports the table it is set ON for. If the value inserted is larger than the current identity value for the table, SQL Server automatically uses the new inserted value as the current identity value. The setting of SET IDENTITY_INSERT is set at execute or run time and not at parse time.
Default	DEFAULT constraints allow you to specify a value that the database will use to populate fields that are left blank in the input source. They're a replacement for the use of NULL values that provide a great way to predefine common data elements.
NULL	Although not a constraint in the strictest definition, the decision to allow NULL values in a column or not is a type of rule enforcement for domain integrity.

Example(s) : Create Constraints

Constraint Type	Example(s)

Primary Key	`CREATE TABLE #MyTempTable`
	`(`
	`  Col_1 INT PRIMARY KEY`
	`)`
	
	`ALTER TABLE #TargetTable`
	`  ADD CONSTRAINT PK_EmployeeID PRIMARY KEY (EmployeeID)`
	
	`ALTER TABLE #TargetTable`
	`  DROP CONSTRAINT PK_EmployeeID`

Foreign Key	CREATE
	
	`CREATE TABLE dbo.Employees`
	`(`
	`  EmployeeID …`
	`, …`
	`, TitleID  smallint  NOT NULL CONSTRAINT FK_Employees_JobTitle FOREIGN KEY(TitleID) REFERENCES dbo.JobTitles(TitleID)`
	`, …`
	`)`
	
	ALTER
	`ALTER TABLE dbo.Employees`
	`  ADD CONSTRAINT FK_Employees_JobTitle FOREIGN KEY(TitleID) REFERENCES dbo.JobTitles(TitleID)`
	
	DROP
	`ALTER TABLE dbo.Employees`
	`  DROP CONSTRAINT FK_Employees_JobTitle`

Unique	CREATE
	`ALTER TABLE dbo.Employees`
	`  ADD CONSTRAINT UnqConstraint_Employee UNIQUE (workphone)`

Check	`ALTER TABLE dbo.Employees` `  ADD CONSTRAINT ck_Salary CHECK (salary > 0.00)`
NOCHECK	`ALTER TABLE dbo.Employees` `  NOCHECK CONSTRAINT UnqConstraint_Employee`

SET IDENTITY_INSERT	`-- SET IDENTITY_INSERT to ON.`
	`SET IDENTITY_INSERT products ON`
	`GO`
	
	`-- Attempt to insert an explicit ID value of 3`
	`INSERT INTO products`
	`(`
	`  Id`
	`, product`
	`) VALUES (`
	`  3`
	`, 'garden shovel'`
	`)`
	`GO`

Default	**Create:**
	`CREATE TABLE Test_DefaultConstraints`
	`(`
	`  Col1  int CONSTRAINT Test_DefaultConstraints_Col1 DEFAULT -1`
	`, Col2  int CONSTRAINT Test_DefaultConstraints_Col2 DEFAULT -2`
	`, Col3  int CONSTRAINT Test_DefaultConstraints_Col3 DEFAULT -3`
	`)`

	Drop:
	`ALTER TABLE Test_DefaultConstraints`
	`  Drop CONSTRAINT Test_DefaultConstraints_Col1;`
	
	Alter:
	`ALTER TABLE Test_DefaultConstraints`
	`  Add CONSTRAINT Test_DefaultConstraints_Col1 DEFAULT (10) for Col1;`

NULL	**Create:**
	`CREATE TABLE Test_NOTNULLConstraints`
	`(`
	`    Col1   int          NOT NULL`
	`  , Col2   int          NOT NULL`
	`  , Col3   int          NOT NULL`
	`)`