



Table Constraints



Constraints

- A Constraint is a restriction placed on a table
- Constraints can be defined at:
 - ☐ Column level, or
 - ☐ Table level
- Constraints are rules that ensure that your data meets certain rules
- List of Constraints include:
 - ☐ Primary key constraints
 - ☐ Foreign key constraints
 - ☐ Unique constraints (also known as alternate keys)
 - ☐ Check constraints
 - ☐ Default constraints



Type of Constraints

- Constraints can be categorized as:
 - Domain constraints
 - Entity constraints and
 - Referential integrity constraints

- Domain Constraints
 - Deal with one or more columns
 - Ensure that a particular column or set of columns meets particular criteria
 - Applied independent of any other row in the table
 - Check constraints and default constraints are examples of domain constraints



Type of Constraints (cont.)

■ Entity Constraints

- Entity constraints are all about individual rows
- The constraint requires that every row has a unique value for a column or combination of columns.
- The primary key and unique constraints are examples of Entity Constraints

■ Referential Integrity Constraints

- Are created when a value in one column must match the value in another column (in another table or from the same table)
- Allows us to build what we would call a domain or lookup table.
 - A lookup table is a table that provides a limited list of acceptable values
- FOREIGN KEY Constraints are example of Referential integrity



Constraint Names

- Recommended that you specify a constraint name
 - use a combination of the constraint type with a phrase to indicate what it does or the name(s) of the column(s) it affects
 - Example
 - A Check Constraint on a column that ensures that phone numbers are formatted properly can be named as **CKCustomerPhoneNo**
- SQL Server will auto-generate a constraint name, if it is not given one
 - Example: PK__Employees__145C0A3F
 - The “PK” is for primary key
 - the Employees is for the Employees table, and
 - the rest is a randomly generated value to ensure uniqueness
- It is important that you use a consistent style in naming constraints



Check Constraints

- The constraint is defined using the same rules that you would use in a WHERE clause
- CHECK constraints are not restricted to a particular column.
 - They can be related to a column
 - they can also be table related in that they can check one column against another
- They can also check that any combination of column values meets a criterion.
-



Check Constraints (cont.)

- Example of Check Constraint expressions

BETWEEN 1 AND 12

LIKE '[0-9][0-9][0-9]-[0-9][0-9]'

IN ('Full Time', 'Part Time', 'Other')

UnitPrice >= 0

ShipDate >= OrderDate

- Adding a CHECK constraint:

```
ALTER TABLE Employee  
ADD CONSTRAINT CN_DoB CHECK ( DoB <= GETDATE () )
```



Primary Key

- Primary keys are the unique identifiers for each row
- A unique identifier allows you to accurately identify (or reference) a record from table
- Primary keys
 - must contain unique values
 - Cannot be NULL
- You can create the primary key either in
 - CREATE TABLE command or
 - ALTER TABLE command



Primary Key (cont.)

- Creating a PK at Table Creation as a Column Constraint

```
CREATE TABLE Customers (  
    CustomerNo int IDENTITY NOT NULL PRIMARY KEY  
    , CustomerName varchar(30) NOT NULL  
    , .....  
)
```

- Creating a PK at Table Creation as a Table Constraint

```
CREATE TABLE Customers (  
    CustomerNo int IDENTITY NOT NULL  
    , CustomerName varchar(30) NOT NULL  
    , CONSTRAINT [PK_ Customers]  
        PRIMARY KEY CLUSTERED ([CustomerNo] ASC )  
)
```



Primary Key (cont.)

- Creating a PK on an Existing Table

ALTER TABLE Customers

ADD CONSTRAINT PK_ Customers **PRIMARY KEY** (CustomerNo);

- The ALTER command indicates that:
 - We we are adding something to the table
 - We're adding a constraint
 - The name the constraint
 - The type of constraint
 - The column(s) that the constraint applies to



UNIQUE Constraints

- UNIQUE constraints require a unique value throughout the named column (or combination of columns) in the table.
- The column(s) on which a UNIQUE constraint is defined on are referred to as **Alternate Keys**
- The major differences between a PK and a UNIQUE constraint is that
 - you can have more than one UNIQUE constraint
 - UNIQUE constraint allow a NULL



UNIQUE Constraints (cont.)

- Creating UNIQUE Constraints on individual columns

```
CREATE TABLE Employee
```

```
(
```

```
    EmployeeID int IDENTITY NOT NULL PRIMARY KEY
```

```
    , EmpCode char(10) NOT NULL UNIQUE
```

```
    , PhoneNo varchar(14) NOT NULL UNIQUE
```

```
)
```

- Creating UNIQUE Constraints on an existing table

```
ALTER TABLE Employees
```

```
ADD CONSTRAINT UQ_EmployeeSSN UNIQUE (SSN);
```



FOREIGN KEY Constraints

- Foreign keys are used as a primary method of establishing relationships between tables
- A foreign key creates a dependency between two tables:
 - The referencing table and
 - The referenced table
- The column that is being referenced must have either a PRIMARY KEY or a UNIQUE constraint defined on it
- After adding a foreign key, any record you insert into the referencing table must either
 - have a matching value in the referenced column(s) in the referenced table, or
 - the value of the foreign key column(s) must be set to NULL.



FOREIGN KEY - Examples

■ Column-level Foreign Key definition

```
CREATE TABLE Orders
(
    OrderID int IDENTITY NOT NULL PRIMARY KEY
    , CustomerNo int NOT NULL FOREIGN KEY REFERENCES
                                                Customers(CustomerNo)
    , OrderDate date NOT NULL
)
```

■ Table-level Foreign Key definition

```
CREATE TABLE Employees(
    [EmpID] [int] IDENTITY(1,1) NOT NULL,
    [FirstName] [varchar](20) NULL,
    [ManagerEmpID] [int] NULL,
    , CONSTRAINT FK_EmployeeHasManager FOREIGN KEY (ManagerEmpID)
REFERENCES Department(ManagerID);
)
```



FOREIGN KEY - Examples (cont.)

- Adding a Foreign Key to an Existing Table

```
ALTER TABLE Orders  
ADD CONSTRAINT FK_EmployeeCreatesOrder  
FOREIGN KEY (EmployeeID) REFERENCES Employees(EmployeeID)
```

- Adding a self-referencing Foreign Key

```
ALTER TABLE Employees  
ADD CONSTRAINT FK_EmployeeHasManager  
FOREIGN KEY (ManagerEmpID) REFERENCES Employees(EmployeeID)
```



Referential Integrity Rules

- Foreign keys are bidirectional;
 - They restrict the child table to values that exist in the parent, and
 - They check for child rows whenever an update or delete is done on the parent
- Referential Integrity Rules ensures that **NO ORPHAN** records are created
- Different option exist
 - Restrict
 - Cascade
 - Set NULL
 - Set DEFAULT



Referential Integrity Rules (cont.)

■ SQL Server

- The default behavior for SQL Server to “restrict” the parent row from being **Deleted** or **Updated** if there are any dependent child row(s)

■ Foreign key - Syntax

FOREIGN KEY REFERENCES <table name> (<column name>)

[ON **DELETE** {CASCADE|NO ACTION|SET NULL|SET DEFAULT}]

[ON **UPDATE** {CASCADE|NO ACTION|SET NULL|SET DEFAULT}]



Referential Integrity Rules – On Update

ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }

When a referenced column is updated in the referenced (parent) table, then:

- **NO ACTION:** The update action is denied (Restricted) if there are any rows that are referencing it in the child table
- **CASCADE:** The FK column is updated in the referencing child table
- **SET NULL**
 - The foreign key of the rows that are referencing the deleted row are set to NULL
 - The foreign key columns must be NULLABLE
- **SET DEFAULT**
 - The foreign key of the rows that are referencing the updated column are set to their default values
 - All foreign key columns must have default definitions.
 - NULL will be used if a column is nullable, and there is no default value set



Referential Integrity Rules – On Delete

ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }

When a row is deleted in the referenced (parent) table, then:

- **NO ACTION:** The delete action is denied (Restricted) if there are any rows that are referencing it in the child table
- **CASCADE:** The rows that are referencing it in the child table will be deleted
- **SET NULL**
 - The foreign key of the rows that are referencing the deleted row are set to NULL
 - The foreign key columns must be NULLABLE
- **SET DEFAULT**
 - The foreign key of the rows that are referencing the deleted row are set to their default values
 - All foreign key columns must have default definitions.
 - NULL will be used if a column is nullable, and there is no default value set



Referential Integrity Rules (cont.)

■ Example:

```
CREATE TABLE Album
```

```
(
```

```
    AlbumID    INT    PRIMARY KEY,
```

```
    Name       VARCHAR(50)
```

```
)
```

```
CREATE TABLE Track
```

```
(
```

```
    TrackID    INT    PRIMARY KEY,
```

```
    Title       VARCHAR(50),
```

```
    AlbumID    INT    REFERENCES Album(AlbumID)
```

```
                ON DELETE SET NULL
```

```
                ON UPDATE CASCADE
```

```
)
```



Disabling Constraints

- You can deactivate (Disable) a FOREIGN KEY or CHECK constraint without deleting (**DROP**ing) them
- You cannot disable PRIMARY KEY or UNIQUE constraints
- Any reason for disabling a constraint should be a temporary thing
 - Otherwise you would DROP them
- The usual reason to disable a constraint is **BAD** data:
 - BAD data that is already in your database when you create the constraint
 - BAD data that you want to add after the constraint is already built



Disabling Constraints (cont.)

Ignoring Bad Data when you create the Constraint

- To add a constraint that won't apply to existing data
 - use of the **WITH NOCHECK** option when you execute the ALTER TABLE statement that adds your constraint

- Example

```
ALTER TABLE Student
WITH NOCHECK
ADD CONSTRAINT CK_StudentCode
    CHECK (StudCode LIKE '([A-Z][A-Z])[0-9][0-9][0-9]-[0-9]')
```



Disabling Constraints (cont.)

Temporarily disabling an existing constraint

- You can disable constraints to import data that doesn't match up with the existing rules
 - Use an ALTER TABLE statement with an option called **NOCHECK** that turns off the constraint in question
- Example:

```
ALTER TABLE Customers
NOCHECK
CONSTRAINT CN_CustomerPhoneNo;
```
- When we are ready for the constraint to be active again, use the **CHECK** option:

```
ALTER TABLE Customers
CHECK
CONSTRAINT CN_CustomerPhoneNo;
```