Create Database

```
CREATE DATABASE CHA2;
```

- The CREATE command will create a data file with the name provided and a .mdf file extension, as well as a transaction log with an .ldf extension.

Transaction Log file

- Many types of operations are recorded in the transaction log. These operations include:
- Every data modification (insert, update, or delete). This includes changes by system stored procedures or data definition language (DDL) statements to any table, including system tables.
- Creating or dropping a table or index.

Configuring file growth

- Enable Auto-growth :
  - In percent
  - In megabytes
  - Maximum file size:

Using multiple files

- secondary, data files have an .ndf file extension by default.
- it does not enable control over the location of tables or indexes.
- this technique does reduce the I/O load on each disk subsystem.

Using multiple files

```
CREATE DATABASE NewDB
ON
PRIMARY
(NAME = NewDB,
FILENAME = ‘e:\SQLData\NewDB.mdf’),
(NAME = NewDB2,
FILENAME = ‘f:\SQLData\NewDB2.ndf’)
LOGON
(NAME = NewDBLog,
FILENAME = ‘g:\SQLLog\NewDBLog.ldf’),
(NAME = NewDBLog2,
FILENAME = ‘h:\SQLLog\NewDBLog2.ldf’)
```

Creating a database with filegroups

```
CREATE DATABASE NewDB
ON
PRIMARY
(NAME = NewDB,
FILENAME = ‘d:\SQLData\NewDB.mdf’,
SIZE = 50MB,
MAXSIZE = 5Gb,
FILEGROWTH = 25MB),
FILEGROUP Data DEFAULT
(NAME = NewDBData,
FILENAME = ‘e:\SQLData\NewDBData.ndf’,
SIZE = 100MB,
MAXSIZE = 50Gb,
FILEGROWTH = 100MB)
LOG ON
(NAME = NewDBLog,
FILENAME = ‘f:\SQLLog\NewDBLog.ndf’,
SIZE = 100MB
```

Schemas

- A schema is an object that exists purely to own database objects, most likely to segment a large database into manageable modules, or to implement a segmented security strategy.

- Server.database.schema.object;

Schemas

- Create Schema Myschema
- Assign user login to schema and give permissions.
- Include a table in specific schema.

- ALTER SCHEMA dbo TRANSFER MySchema.Table_1;

Create table

- Using SSMS
- Using T-SQL

```
CREATE TABLE dbo.ProductCategory (
ProductCategoryID UNIQUEIDENTIFIER NOT NULL
ROWGUIDCOL DEFAULT (NEWID()) PRIMARY KEY
NONCLUSTERED,
ProductCategoryName NVARCHAR(50) NOT NULL,
ProductCategoryDescription NVARCHAR(100) NULL
)
ON [Data];
```

Creating data columns

```
CREATE TABLE TableName (
ColumnName DATATYPE Attributes,
ColumnName DATATYPE Attributes
);
```

column name, data type, and any

column attributes such as constraints, null-ability, or default value

Column data type

- The column’s data type serves two purposes:

1-The data type is a valuable data-validation tool that should not be overlooked.

2-It determines the amount of disk storage allocated to the column.

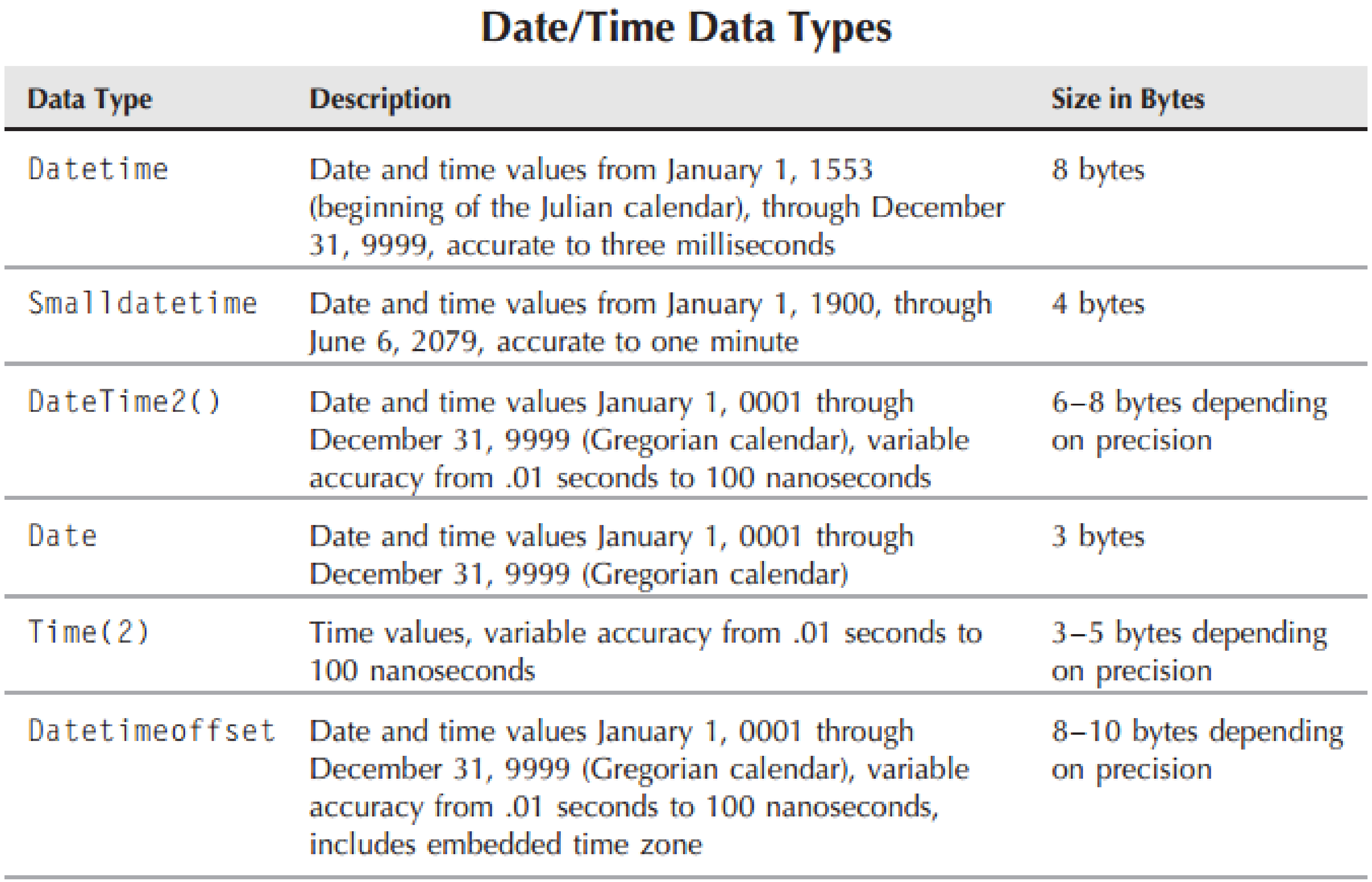
Character data types

Data type	Description	size
Char(n)	Fixed-length character data up to 8,000 characters long using collation character set	Defined length *1byte
Nchar(n)	Unicode fixed-length character data Defined length *2bytes	Defined length *2bytes
VarChar(n)	Variable-length character data up to 8,000 characters long using collation character set	1 byte per character
nVarChar(n)	Unicode variable-length character data up to 8,000 characters long using collation character set	2 bytes per character
Sysname	A Microsoft user-defined data type used for table and column names that is the equivalent ofnvarchar(128)	2 bytes per character

Numeric data types

Data type	Description	size
Bit	1 or 0	1bit
Tinyint	Integers from 0 to 255	1 byte
Smallint	Integers from -32,768 to 32,767	2 bytes
Int	Integers from -2,147,483,648 to 2,147,483,647	4 bytes
Bigint	Integers from -2 ˆ 63 to 2 ˆ 63-1	8 bytes
Decimal or Numeric	Fixed precision and scale numeric data from -10^38 +1 through 10^38 –1	Varies according to length
Money	Numbers from -2 ˆ 63 to 2 ˆ 63, accuracy to one ten-thousandths (.0001)	8 bytes
SmallMoney	Numbers from -214,748.3648 through +214,748.3647, accuracy to ten thousandths (.0001)	4bytes
Float	Floating-point numbers ranging from -1.79E+308 through 1.79E +308, depending on the bit precision	4or8bytes
Real	Float with 24-bit precision	4 bytes

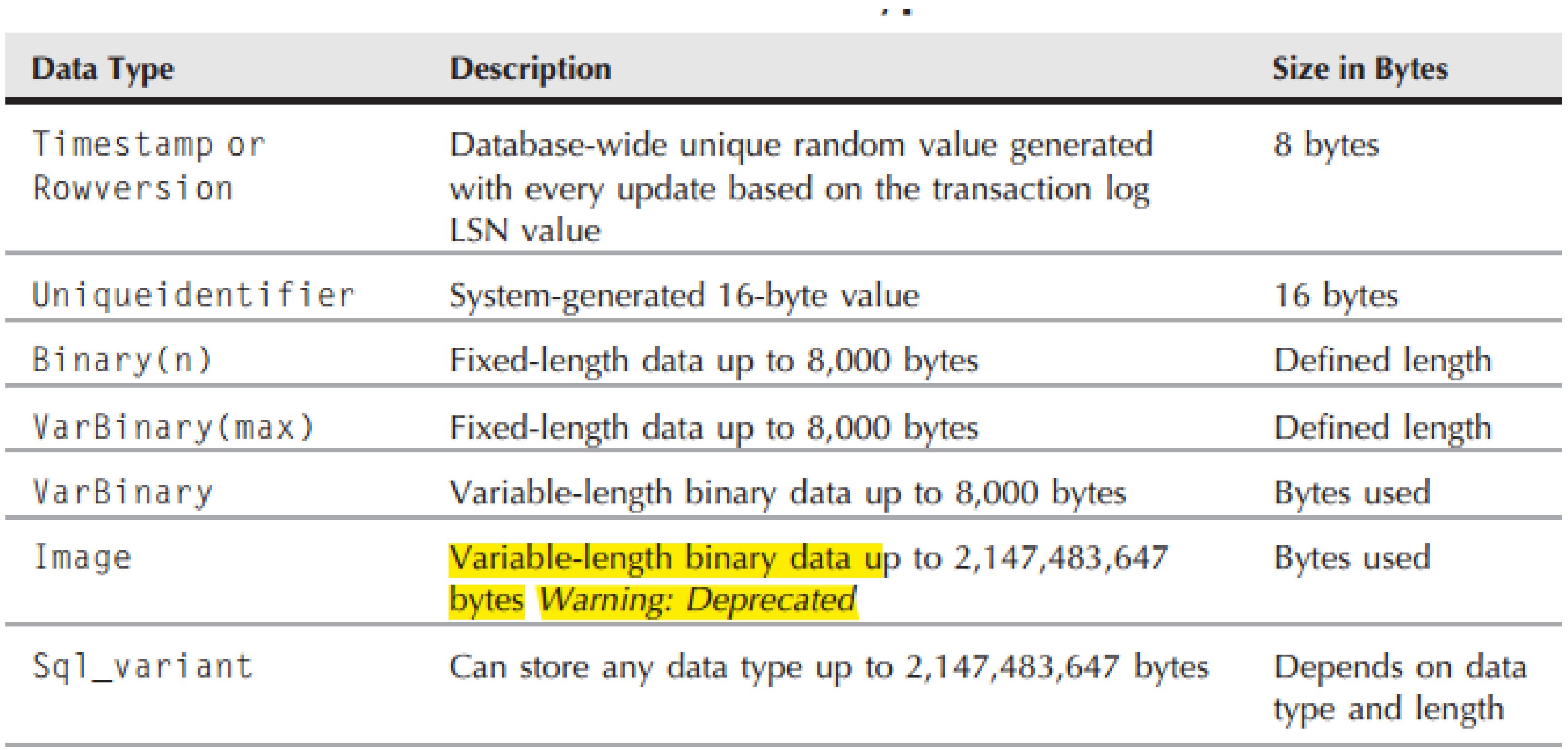
Date time data types



Date/Time Data Types

Data Type	Description	Size in Bytes
`Datetime`	Date and time values from January 1, 1553 (beginning of the Julian calendar), through December 31, 9999, accurate to three milliseconds	8 bytes
`Smalldatetime`	Date and time values from January 1, 1900, through June 6, 2079, accurate to one minute	4 bytes
`DateTime2()`	Date and time values January 1, 0001 through December 31, 9999 (Gregorian calendar), variable accuracy from .01 seconds to 100 nanoseconds	6–8 bytes depending on precision
`Date`	Date and time values January 1, 0001 through December 31, 9999 (Gregorian calendar)	3 bytes
`Time(2)`	Time values, variable accuracy from .01 seconds to 100 nanoseconds	3–5 bytes depending on precision
`Datetimeoffset`	Date and time values January 1, 0001 through December 31, 9999 (Gregorian calendar), variable accuracy from .01 seconds to 100 nanoseconds, includes embedded time zone	8–10 bytes depending on precision

Other data types



Data Type	Description	Size in Bytes
Timestamp or Rowversion	Database-wide unique random value generated with every update based on the transaction log LSN value	8 bytes
Uniqueidentifier	System-generated 16-byte value	16 bytes
Binary(n)	Fixed-length data up to 8,000 bytes	Defined length
VarBinary(max)	Fixed-length data up to 8,000 bytes	Defined length
VarBinary	Variable-length binary data up to 8,000 bytes	Bytes used
Image	Variable-length binary data up to 2,147,483,647 bytes *Warning: Deprecated*	Bytes used
Sql_variant	Can store any data type up to 2,147,483,647 bytes	Depends on data type and length

Sparse

- Sparse make your data taking no space if they are empty and more space if they have data. In other words they optimize storage for NULL values.
- SQL Server essentially writes the list of sparse columns that have data into a list for the row (5 bytes +2–4 bytes for every sparse column with data).

Sparse

- To create a sparse column, add the SPARSE keyword to the column definition. The sparse column must be nullable.

Calculated column

- A computed column is a virtual column that is not physically stored in the table, unless the column is marked PERSISTED
- A computed column expression can use data from other columns to calculate a value for the column to which it belongs.

Synonym

- A synonym is a database object that serves the following purposes:

1- Provides an alternative name for another database object, referred to as the base object, that can exist on a local or remote server.

2- Provides a layer of abstraction that protects a client application from changes made to the name or location of the base object.

Data integrity

- Enforcing data integrity ensures the quality of the data in the database.
- The ability to ensure that persisted data can be retrieved without error
- Data integrity can be defined in multiple ways:

1- Entity integrity

2- Domain integrity

3- Referential integrity

4- User-defined integrity

Entity integrity

- If the primary key is unique
- all attributes are scalar and fully dependent on the primary key
- the table’s primary key enforces entity integrity.

Domain integrity

- Domain integrity is the validity of entries for a given column
- You can enforce domain integrity by restricting the type (through data types)
- In the physical schema, the data type and nullability of the row enforce domain integrity.

Referential integrity

- Domain integrity means that if an attribute has a value, then that value must be in the domain.
- In the case of the foreign key, the domain is the list of values in the related primary key.
- Referential integrity ensures that key values are consistent across tables.

User-Defined Integrity

- User-defined integrity allows you to define specific business rules that do not fall into one of the other integrity categories.
- All of the integrity categories support user-defined integrity

How to force data integrity?

Integrity type	Constraint type	Description
Domain	DEFAULT	Specifies default value for column
	CHECK	Specifies allowed value for column
	FOREIGN KEY	Specifies column in which values must exist
	NULL	Specifies whether NULL is permitted
Entity	PRIMARY KEY	Identifies each row uniquely
	UNIQUE	Prevents duplication of nonprimary keys
Referential	FOREIGN KEY	Defines columns whose value must match the primary key of this table
	CHECK	Specifies the allowed value for a column based on the contents of another column

Check constraint

- Check constraints may affect INSERT and UPDATE operations.
- Each table column may have multiple check constraints.

```
Constraint Myconstraint CHECK ( logical_expression )
```

Column level constraint

- In this type the constraint is checked when the value of the column changed.

```
CREATE TABLE [COLUMNLEVEL]
(
    [ID] INT PRIMARY KEY,
    [STARTDATE] DATE NOT NULL,
    [ENDDATE] DATE NOT NULL,
    [CHECKED] DATE NOT NULL,
    CONSTRAINT COLUMNLEVELCONSTRIANT CHECK( [CHECKED] > '2012-01-01')
)
```

Table Level Constraints

- In this type the constraint is checked if there is any modification to a row, regardless the value of the column changed or not.

```
CREATE TABLE [TABLELEVEL]
(
    [ID] INT PRIMARY KEY,
    [STARTDATE] DATE NOT NULL,
    [ENDDATE] DATE NOT NULL,
    [CHECKED] DATE NOT NULL,
    CONSTRAINT TABLELEVELCONSTRIANT
CHECK( [CHECKED] BETWEEN [STARTDATE] AND [ENDDATE])
)
```

Check constraint with alter table

- alter table ConstraintTest add constraint NameConstraint check (name like 'ma%')

Default Constraint

- you can use a DEFAULT constraint to supply that column with an anticipated or non-NULL value.
- In create table:

    DEFAULT constant_expression

- With alter:

alter table ConstraintTest add constraint DefaultSalary Default 1100 for salary

alter table ConstraintTest add constraint DefaultSalary Default 1100 for salary

Entity Integrity

- Primary Key constraint
- A combination of one or more columns that uniquely identifies each row in a table
- Integrity is enforced during inserts and updates
- Limit of one primary key constraint per table

Primary key constraint

```
create table ConstraintTest
(Id int identity primary key)
```

Or

```
alter table ConstraintTest add constraint PK_constraint primary
key (Id)
```

Or

```
alter table ConstraintTest add constraint PK_constraint primary
key (Id,name)
```

Unique constraint

- A unique constraint also enforces entity integrity
- Integrity is enforced during inserts and updates
- Tables can have more than one unique constraint

Unique constraint

create table *table_name* (*column_name datatype* [NULL | NOT NULL | IDENTITY] **[constraint *constraint_name*]**
unique)

Or

alter table ConstraintTest add constraint SalaryUnique unique (Id)

Or

alter table ConstraintTest add constraint SalaryUnique unique (Id,salary)

References Constraint

- A references constraint enforces referential integrity
- Integrity is enforced during inserts, updates, and deletes
  - If an **insert** or **update** contains foreign key values that do not exist in the primary key column(s), the statement fails
  - If an **update** or **delete** attempts to remove a primary key value that exists in a corresponding foreign key, the statement fails

References Constraint

```
Create table Xtable
(
fk_id int constraint fk_table22 references [dbo].[fk_table] (id)
)
```

Or

```
alter table ConstraintTest add constraint fk_table22 foreign key (fk_id)
references [dbo].[fk_table] (id)
```

Cascading Referential Integrity

- you can define the actions that the SQL Server takes when a user tries to delete or update a key to which existing foreign keys point.
- When creating foreign key constraint
- [ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }]
- [ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT }]

Cascading Referential Integrity

<table>
<tr><th>Option</th><th>UPDATE Behavior</th><th>DELETE behavior</th></tr>
<tr><td><b>NO ACTION (default)</b></td><td colspan="2">Return error and roll back operation</td></tr>
<tr><td><b>CASCADE</b></td><td>Update foreign keys in referencing tables</td><td>Delete rows in referencing table</td></tr>
<tr><td><b>SET NULL</b></td><td colspan="2">Set foreign keys in referencing tables to NULL</td></tr>
<tr><td><b>SET DEFAULT</b></td><td colspan="2">Set foreign keys in referencing tables to DEFAULT values</td></tr>
</table>

Drop constraint

alter table *table_name*
drop constraint *constraint_name*

```
alter table roysched
 drop constraint chk_hirange_lorange
```

System-Defined Constraint Messages

- Msg 547, Level 16, State 0, Line 2
- The INSERT statement conflicted with the CHECK constraint "SalaryConstraint". The conflict occurred in database "ITI", table "dbo.ConstraintTest", column 'salary'.
- The statement has been terminated.

System Procedures for Constraints

- **sp_helpconstraint** *table_name*
  - Displays information about the constraints on the specified table
- **sp_rename** *old_constraint_name, new_constraint_name*
  - Changes the name of a check or references constraint

Default Object

- A default is a database object that supplies a value to a column during an **insert** statement if no value was specified for that column
- Create a SQL Server default.
- Bind or unbind an existing SQL Server default to a column or user-defined data type.

Default object

- Create default

CREATE DEFAULT default_name AS constant_expression

- Binding default to object

sp_bindefault default_name, ‘object_name’;

Example

- create default salary_default as 1100
- go
- sp_bindefault salary_default,'dbo.[ConstraintTest].[salary]'

Rule object

- Creates an object called a rule.
- When bound to a column or an alias data type
- a rule specifies the acceptable values that can be inserted into that column.

Rule object

- Create rule

create rule *rule_name* as *condition_expression*

- Example:

```
 CREATE RULE range_rule
 AS @range>= $1000 AND @range <$20000;
```

Rule object

- Binding Rule

sp_bindrule *rule_name*, *object_name*

- Example

sp_bindrule 'rule_ssn', 'ssn‘