**SQL Commands and Information**

SQL (Structured Query Language)

T-SQL (TRANSACT-SQL) – an extension of the SQL database language

Default SQL Server Port 1433

**DDL (Data Definition Language)**

CREATE, ALTER, USE, DELETE, DROP, TRUNCATE

**DCL (Data Control Language)**

GRANT, REVOKE, DENY

**Datatypes**

|  |  |  |
| --- | --- | --- |
| tinyint | Integer (0 to 255) | 1 byte |
| smallint | Integer (-32,768 to 32,767) | 2 bytes |
| int | Integer (-2^31 to 2^31-1), about +/-2.14 billion | 4 bytes |
| bigint | Integer (-2^63 to 2^63-1), about +/-9.2 quadrillion | 8 bytes |
| numeric | Fixed precision/scale, +/-(10^38) | Varies |
| decimal | Fixed precision/scale, +/-(10^38) | Varies |
| float | 4 bytes – 7 digit precision, 8 bytes – 15 digit prec | 4/8 bytes |
| smallmoney | +/- 214K, four decimal places | 4 bytes |
| money | +/- 9 Trillion, four decimal places | 8 bytes |
| datetimeoffset | Jan 1st 1AD – Dec 31st 9999, 100 nanosecond accuracy, timezone sensitive |  |
| datetime | Jan 1st 1753 – Dec 31st 9999, .000/.003/.007 sec accuracy | 8 bytes |
| smalldatetime | Jan 1st 1900 – Jun 6th 2079, 1 min accuracy | 4 bytes |
| datetime2 | Jan 1st 1AD – Dec 31st 9999, 100 nanosecond accuracy | Varies |
| date | Jan 1st 1AD – Dec 31st 9999, no time, 1 day accuracy | 3 bytes |
| time | 24 hour clock, 100 nanosecond accuracy | 5 bytes |
| char | Fixed length character string | 1/char |
| varchar | Variable length character string | 1/char |
| varchar(max) | 1 – 8000 characters | varies |
| nchar | Fixed length Unicode character string | 2/char |
| nvarchar | Variable length Unicode character string | 2/char |
| nvarchar(max) | Maximum 2 GB data | Varies |
| binary | Stores binary data, fixed length | Varies |
| varbinary | Stores binary data, variable length | Varies |
| UUID | Unique alphanumeric code, 32 digits long | 16 bytes |

**DML (Data Manipulation Language)**

SELECT, INSERT, UPDATE, DELETE, MERGE

**SELECT syntax**

SELECT \*DISTINCT <Columns…> (DISTINCT used to eliminate duplicate results)

FROM <Table(s)…>

ON <Table(s).Column(s)> (If JOIN is used)

WHERE <Conditions…>

GROUP BY <Columns…>

HAVING <Conditions…> (If GROUP BY used)

ORDER BY <Columns…> ASC/DESC

**User-Defined Functions**

* May be passed one or more variables
* Must return a value
* Single return value (Scalar function) or table (Tabular function)
* Code compiled each time function is called
* May be called within a program/script

**Syntax:**

CREATE FUNCTION <Function-name> ([Input Variables])

RETURNS <datatype>

AS

BEGIN

DECLARE [Output Variable] <datatype>

RETURN [Output Variable]

END

**User-Defined Procedures**

* May be passed one or more variables
* May return a value
* Code is pre-compiled when procedure created
* Called using EXEC command
* Can protect against ‘SQL Injection’ attacks

**Syntax:**

CREATE PROCEDURE <Procedure-name> ([Input Variables])

AS

BEGIN

<Script Body>

END

**Types of Keys**

PRIMARY – UNIQUE, NOT NULL

FOREIGN – NOT NULL, points to PRIMARY KEY in another table (or same table)

SUPERKEY – a combination of attributes which can uniquely identify records

CANDIDATE – a SUPERKEY with no unnecessary information

COMPOSITE – A PRIMARY KEY made up of two or more attributes

**Creating a PRIMARY KEY (Syntax):**

ALTER TABLE <Database Name>.<Table Name>

ADD CONSTRAINT <Key Name> PRIMARY KEY (<Column Name>)

**Constraints**

PRIMARY KEY, FOREIGN KEY, UNIQUE, IDENTITY, NOT NULL, CHECK, DEFAULT

**Database Normalization (5 types of normalization)**

1st Normal Form

* 1NF Eliminates duplicate columns
* There are no repeating groups
* A Primary Key has been identified

2nd Normal Form

* Removes subsets of data, placing them in separate tables
* Uses Foreign Keys to link tables
* No non-prime attribute in table is partially dependant on Primary Key

3rd Normal Form

* There are no Transitive dependencies

4th Normal Form

* There are no multi-valued dependencies

5th Normal Form

* No non-trivial join dependencies exist

**Indexes (Used to increase speed of queries)**

Clustered

* Stored on a B-Tree
* Highest level (Root), middle levels (Intermediate), and lowest level (Leaf)
* Lowest level (Leaf level) contains the data itself
* Only 1 clustered index allowed per table

Non-clustered

* Stored on a Heap
* Lowest level (Leaf level) contains index pages or pointers to the rows
* If a non-clustered index is created and there is no clustered index, it may be stored on a B-Tree instead
* Syntax: CREATE INDEX <IndexName> ON <Table> (<Column1>, <Column2>,…)

Covering Index

* Includes all columns in a query

Filtering Index

* Contains a WHERE clause
* Gives query results from a well-defined subset of data

Full Text Index

* For queries of text ‘strings’, only one per table (covers all text in table)

**Common Data Structures for Indexes**

* Bitmap Index – stores data in bit arrays (most common is B-Tree)
* Dense Index – works with pairs of keys and pointers to each record
* Sparse Index – works as dense index, but points to a block of data; less costly, less effective
* Reverse Index – reverses the key value (e.g. 12345 -> 54321); useful when keys in sequence

**SQL Security Levels**

* LOGIN (Server Roles)
* USER (Database Roles)
* OBJECT (Permissions)

**Fixed Server Roles**

* Bulkadmin – allows for bulk INSERT operations
* Dbcreator – CREATE, ALTER, DROP, and RESTORE databases
* Diskadmin – CREATE, ALTER, and DROP disk files
* Processadmin – can kill running SQL server
* Securityadmin – manages logins for the server
* Serveradmin – can configure server-wide settings, set-up full-text searches, shutdown server
* Setupadmin – can configure linked servers, extended stored procedures, and start-up stored procedure
* Sysadmin – can perform any activity regardless of permissions, can override denied permissions on any object

**Fixed Database Roles**

* db\_accessadmin – authorizes user to access DB, but not manage DB security
* db\_backupoperator – user can perform backups, checkpoints, and DBCC commands, but not restores (only Sysadmin)
* db\_datareader – allows user to read all data in DB, similar to a GRANT ALL command; DENY can override
* db\_datawriter – user can write to all data in DB, similar to GRANT ALL command; DENY can override
* db\_ddladmin – user can issue DDL commands (CREATE, ALTER, DROP)
* db\_denydatareader – denies a user to read from any table in DB; overrides any GRANT
* db\_denydatawriter – denies a user to write to any table in DB; overrides and GRANT
* db\_owner – has all permissions on the DB; not like Sysadmin role as an Object level DENY will override
* db\_securityadmin – permits a user to manage DB level security (roles and permissions)

**OBJECT Permissions**

* Select – right to select data, can apply to specific columns
* Insert – right to insert data
* Update – can modify existing data, requires Select permission if WHERE is used, can apply to specific columns
* Delete – right to delete data
* DRI – right to create foreign keys
* Execute – right to execute stored procedures or user-defined functions

**SQL GRANT Syntax**

GRANT <Permissions>

ON <Table>

TO <User/Role>

WITH GRANT OPTION (Optional: a user can grant the same permissions to others)

**Database Backup Types**

* Full Backup – all data in a database/set of filegroups is backed-up
  + Complete copy of all database files
  + Restoring is relatively simple
  + Time-consuming
  + Expensive storage costs (money/space)
  + System may be down during backup procedure
* Differential Backup – only backs-up any changes since last full backup
  + Faster than Full Backup
  + Costs less than Full Backup
  + Restore process can be faster than Full Backup
  + Best balance between time and cost
  + May need full backup to restore database completely
* Incremental Backup – only backs up any changes since last incremental or full backup
  + Fastest type of backup
  + Cost is less than Full Backup
  + Restore process can be much faster than Full Backup
  + Restore process can be more complicated
  + Full Backup often needed to restore database completely
* Replication Services (Real-time Mirroring)
  + Used to backup/replicate a database running on multiple servers
  + Can be Full or Incremental
  + Database is never unavailable
  + Restore can be performed on one server while others still operate
  + Costs are very high (multiple servers, bandwidth)
  + Physical backups still also required for safety
  + Viruses may propagate across the servers

**Useful Information/Commands**

AVG (<Values>) – returns the average of all the values

COUNT (<Values>) – returns the number of rows

MAX (<Values >) – returns the largest value

MIN (<Values >) – returns the smallest value

SUM (<Values >) – returns the total of all the values

LEN (<Values >) – returns the length of a string/text field

ROUND (<Values>, x) – rounds a numeric field to the specified number of places

UPPER (<String>) – puts the input into all upper case

LOWER (<String>) – puts the input into all lower case

LEFT (<String>, x) – returns the leftmost x characters of the string

RIGHT (<String>, x) – returns the rightmost x characters of the string

YEAR (<Date>) – returns the year from the date

MONTH (<Date>) – returns the month from the date

DAY (<Date>) – returns the day from the date

LTRIM (<String>)/RTRIM (<String>) – removes spaces at start/end of a string

SET DATEFORMAT DMY – sets the date format to day/month/year

SELECT DATEADD (DD, 0, DATEDIFF(DD, 0, GETDATE()) -> To return today’s date without the time

CONCAT - Joins strings together

CONCAT (‘My name is ‘, <First\_name>, ’ ‘ ,<Last\_name>) -> Result: ‘My name is John Smith’

CAST – used to change one datatype to another datatype (ANSI compliant)

CAST (<Source Value> AS <Destination Type>)

e.g. CAST (count AS float) {converting an integer type (count) to a float }

CONVERT - used to change one datatype to another datatype (not ANSI compliant)

CONVERT (datatype [(length)], expression, [style])

e.g. CONVERT (nvarchar(10), OrderDate, 101)

{Converts a datetime datatype to a varchar string of length 10, using USA date format (101)}

**VARIABLES**

DECLARE @variablename <Datatype>

**TRIGGERS**

Used to execute stored procedures in response to an event

Can trigger on DML or DDL commands

**Syntax:**

CREATE TRIGGER <Trigger Name>

ON <Database Name>.<Table Name>

AFTER <Command>

AS

EXEC <Procedure Name>

**TRANSACTIONS**

* Used to execute many writes/changes to the database at once
* Can be used to undo errors

BEGIN TRAN – begin recording actions to the transaction file

COMMIT TRAN – commit all actions since BEGIN TRAN

ROLLBACK TRAN – undo all actions since BEGIN TRAN