Database Management Systems

CREATING DATABASE SCHEMAS & MANIPULATING DATA IN T-SQL

September 2019

Contents

MS SQL Server architecture

- Instances
- Databases
- Schemas

Creating database schemas

- DDL statements
- Databases
- Schemas
- Tables
- Constraints

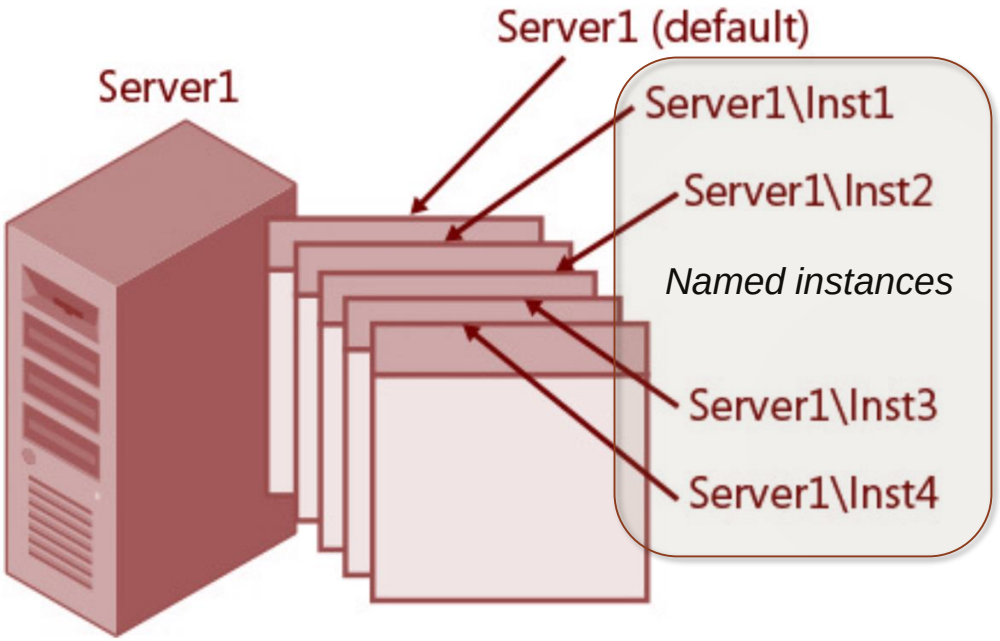
Manipulating data

- Insert
- Update
- Delete

1

MS SQL SERVER ARCHITECTURE

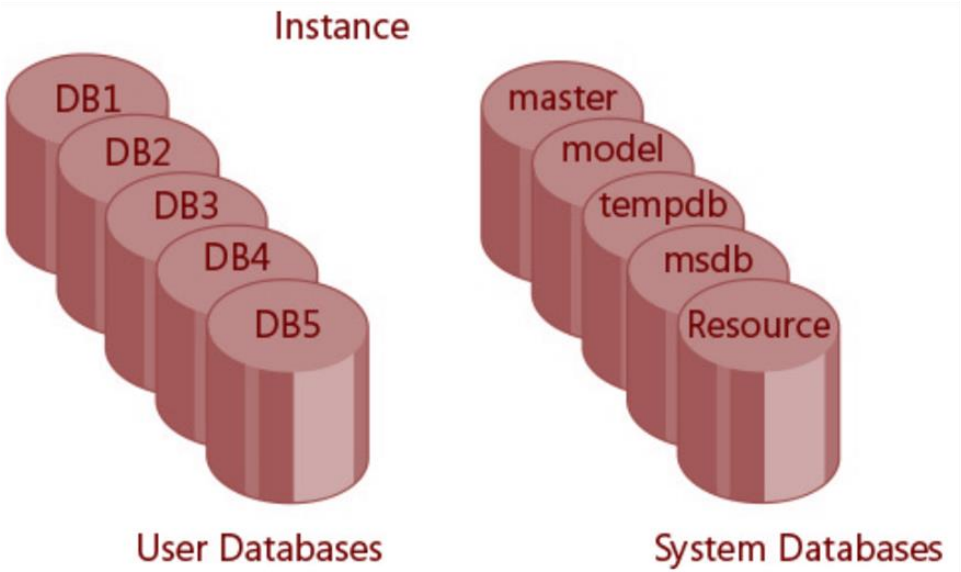
Instances



Instance: an installation of SQL Server database engine / service

- Multiple independent (e.g. security, data) instances can be installed on the same computer
- There must be one default instance

Databases



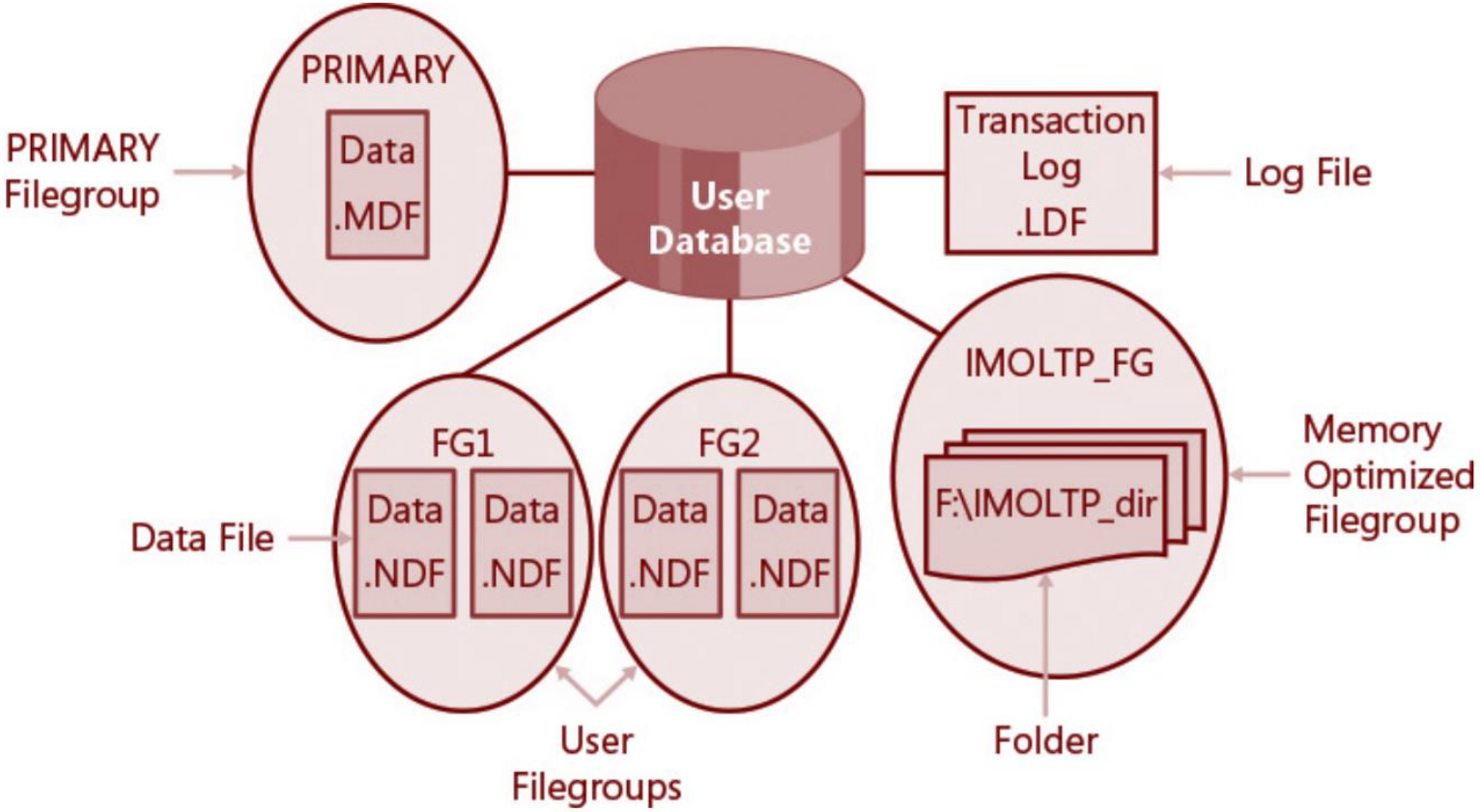
Database – container of objects

- Tables, views, stored procedures, others

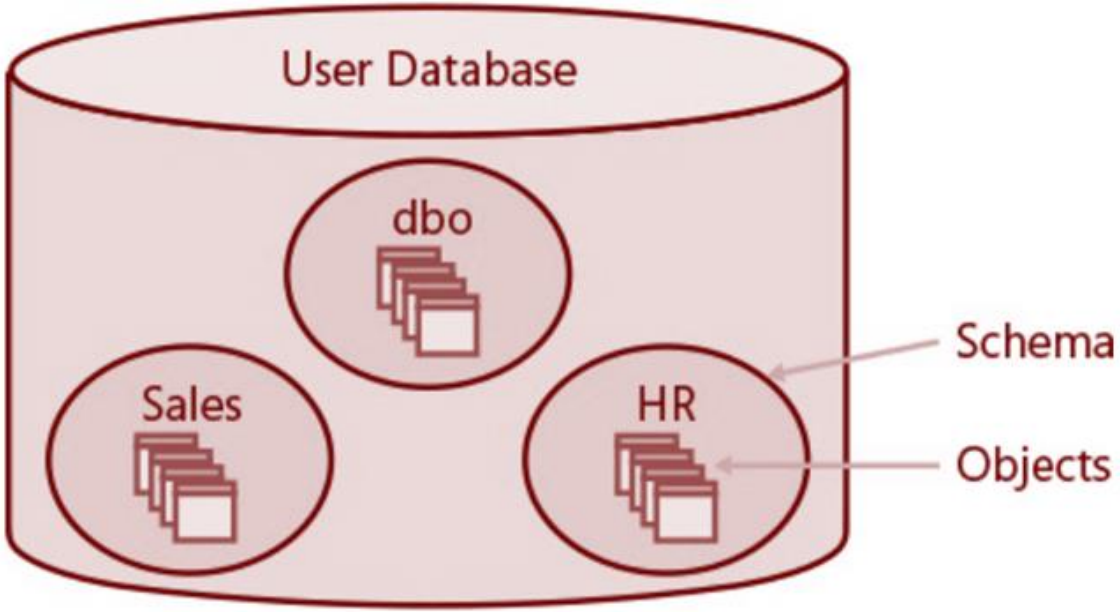
An instance can contain multiple databases

- System databases automatically created for system data & internal purposes
- User databases for application data

Physical Database Layout



Schemas and Objects



Database contains schemas

- Schemas contain objects (views, tables, …)
- Permissions can be controlled at schema level
- Schema also a namespace, e.g. Sales.Orders (schema qualified object name)

Connecting to Databases

Instance login

- Windows authenticated
- SQL Server authenticated

Database user

- Entity granted permissions to objects in the database

Contained databases

- User fully contained within specific database
- Cannot subsequently switch to other user databases

2

CREATING DATABASE SCHEMAS

DDL Statements

CREATE statement: creating objects

ALTER statement: modifying objects

DROP statement: deleting objects

Create / Alter / Drop Databases

```
CREATE DATABASE company;
```

Create a new empty database with default options

```
CREATE DATABASE company
ON PRIMARY (FILENAME =
    'C:\Database\company.mdf')
FOR ATTACH;
```

Create a database with an existing database file

```
ALTER DATABASE company
 SET Read_Only;
```

Set the database read_only option

```
DROP DATABASE company;
```

Remove the database

Create / Alter / Drop Schemas

```
CREATE SCHEMA hr AUTHORIZATION hr;
```

Create hr schema for hr database user

```
ALTER SCHEMA hr
    TRANSFER sales.person;
```

Transfer person table from sales schema to hr schema

```
DROP SCHEMA hr;
```

Remove the hr schema

Create Tables

Table name

Single column attributes

Column name & type

Entire table attributes

```
CREATE TABLE department (
  dname VARCHAR(20) not null,
  dnumber NUMERIC(1) not null,
  mgrssn NUMERIC(9) not null,
  mgrstartdate DATE not null,
  CONSTRAINT pk_department PRIMARY KEY (dnumber),
  CONSTRAINT ck_department UNIQUE (dname)
);
```

Column and Table Constraints

Could be specified at two different levels:

- Column-level constraints (single column only)
- Table-level constraints

Tested before a row is added or updated.

Column and Table Constraints (cont.)

Constraint	Used as a column-level constraint	Used as a table-level constraint
`NOT NULL`	Prevents null values from being stored in the column.	n/a
`PRIMARY KEY`	Requires that each row in the table have a unique value in the column. Null values are not allowed.	Requires that each row in the table have a unique set of values over one or more columns. Null values are not allowed.
`UNIQUE`	Requires that each row in the table have a unique value in the column.	Requires that each row in the table have a unique set of values over one or more columns.
`CHECK`	Limits the values for a column.	Limits the values for one or more columns.
`[FOREIGN KEY] REFERENCES`	Enforces referential integrity between a column in the new table and a column in a related table.	Enforces referential integrity between one or more columns in the new table and one or more columns in the related table.

Key Constraint

Column level

```
CREATE TABLE department (
  dname VARCHAR(20) not null UNIQUE,
  dnumber NUMERIC(1) not null PRIMARY KEY,
  mgrssn NUMERIC(9) not null,
  mgrstartdate DATE not null
);
```

Table level, named for easy modification later on

```
CREATE TABLE department (
  dname VARCHAR(20) not null,
  dnumber NUMERIC(1) not null,
  mgrssn NUMERIC(9) not null,
  mgrstartdate DATE not null,
  CONSTRAINT pk_department PRIMARY KEY (dnumber),
  CONSTRAINT ck_department UNIQUE (dname)
);
```

Check Constraints

Column level

```
CREATE TABLE Invoices(
    InvoiceID INT NOT NULL IDENTITY PRIMARY KEY,
    InvoiceTotal MONEY NOT NULL CHECK (InvoiceTotal >= 0),
    PaymentTotal MONEY NOT NULL DEFAULT 0 CHECK
        (PaymentTotal >= 0)
);
```

Table level

```
CREATE TABLE Invoices(
    InvoiceID INT NOT NULL IDENTITY PRIMARY KEY,
    InvoiceTotal MONEY NOT NULL,
    PaymentTotal MONEY NOT NULL DEFAULT 0,
    CONSTRAINT chk_InvoiceTotal_PaymentTotal
        CHECK ((InvoiceTotal >= 0) AND (PaymentTotal >= 0))
);
```

Foreign Key Constraint

Column level

```
CREATE TABLE Invoices(
    InvoiceID INT NOT NULL PRIMARY KEY,
    VendorID INT NOT NULL REFERENCES Vendors(VendorID),
    InvoiceTotal MONEY NULL);
```

Table level

```
CREATE TABLE Invoices(
    InvoiceID INT NOT NULL PRIMARY KEY,
    VendorID INT NOT NULL,
    InvoiceTotal MONEY NULL,
    CONSTRAINT fk_Invoices_Vendors
        FOREIGN KEY (VendorID)
        REFERENCES Vendors(VendorID)
);
```

Foreign Key Constraints (cont.)

ON DELETE / UPDATE clauses specify what happen when the referred row/key is deleted/updated:

- CASCADE
- SET NULL
- SET DEFAULT
- NO ACTION (#reject)

Alter Table

```
ALTER TABLE Vendors
ADD LastTranDate SMALLDATETIME NULL;
```

Add a column

```
ALTER TABLE Vendors
ALTER COLUMN LastTranDate DATETIME;
```

Change column data type

```
ALTER TABLE Vendors
DROP COLUMN LastTranDate ;
```

Drop the column

Alter / Drop Table (cont.)

Add a foreign key constraint without checking

```
ALTER TABLE employee
    WITH NOCHECK ADD CONSTRAINT fk_employee_department
        FOREIGN KEY (dno) REFERENCES department(dnumber);
```

```
ALTER TABLE employee
    CHECK CONSTRAINT fk_employee_department;
```

Enable constraint checking

```
ALTER TABLE employee
    DROP CONSTRAINT fk_employee_department ;
```

Drop the constraint

```
DROP TABLE employee;
```

Drop the table

3

MANIPULATING DATA

Select … Into

Quick way to create a new table with data from an existing table

```
SELECT VendorID, SUM(InvoiceTotal) AS SumOfInvoices
INTO VendorBalances
FROM Invoices
WHERE InvoiceTotal - PaymentTotal - CreditTotal <> 0
GROUP BY VendorID;
```

Insert Data

Without column list

```
INSERT INTO InvoiceCopy
    VALUES (97, '456789', '2016-04-01', 8344.50, 0,
            0, 1, '2016-04-30', NULL);
```

With column list

```
INSERT INTO InvoiceCopy (
    VendorID, InvoiceNumber, InvoiceTotal, PaymentTotal,
    CreditTotal, TermsID, InvoiceDate, InvoiceDueDate)
    VALUES
        (97, '456789', 8344.50, 0,
        0, 1, '2016-04-01', '2016-04-30');
```

Insert Data (cont.)

Insert data from another table

```
INSERT INTO InvoiceArchive(
    InvoiceID, VendorID, InvoiceNumber, InvoiceTotal,
    CreditTotal,PaymentTotal, TermsID, InvoiceDate,
    InvoiceDueDate)
SELECT
    InvoiceID, VendorID, InvoiceNumber, InvoiceTotal,
    CreditTotal, PaymentTotal, TermsID, InvoiceDate,
    InvoiceDueDate
FROM InvoiceCopy
WHERE InvoiceTotal - PaymentTotal - CreditTotal = 0;
```

Update Data

Update with an arithmetic expression

```
UPDATE InvoiceCopy
SET CreditTotal = CreditTotal + 100
WHERE InvoiceNumber = '97/522';
```

Update with results from a subquery

```
UPDATE InvoiceCopy
SET CreditTotal = CreditTotal + 100,
    InvoiceDueDate = (SELECT MAX(InvoiceDueDate) FROM
InvoiceCopy)
WHERE InvoiceNumber = '97/522';
```

Update Data (cont.)

```
UPDATE InvoiceCopy
SET CreditTotal = CreditTotal + 100
FROM
   (SELECT TOP 10 InvoiceID
    FROM InvoiceCopy
    WHERE InvoiceTotal - PaymentTotal - CreditTotal >= 100
    ORDER BY InvoiceTotal - PaymentTotal - CreditTotal DESC)
AS TopInvoices
WHERE InvoiceCopy.InvoiceID = TopInvoices.InvoiceID;
```

Delete Data

```sql
DELETE InvoiceCopy;
```

```sql
DELETE InvoiceCopy
WHERE InvoiceTotal - PaymentTotal - CreditTotal = 0;
```

```sql
DELETE InvoiceCopy
FROM InvoiceCopy JOIN VendorCopy
    ON InvoiceCopy.VendorID = VendorCopy.VendorID
WHERE VendorName = 'Blue Cross';
```