1. **Create a table in SQL Server with 10 000 000 log entries (date + text). Search in the table by date range. Check the speed (without caching).**

USE master

GO

CREATE DATABASE PerformanceHomeworkDb;

GO

USE PerformanceHomeworkDb;

GO

CREATE TABLE Homeworks(

HomeworkID int NOT NULL PRIMARY KEY IDENTITY,

HomeworkName varchar(100),

TimeSent datetime

);

GO

DECLARE @counter int = 0,

@homework varchar(100) = 'homework',

@date datetime = '2014.1.1';

INSERT INTO Homeworks(HomeworkName, TimeSent)

VALUES (@homework, @date);

WHILE (SELECT COUNT(\*) FROM Homeworks) < 1000000

BEGIN

INSERT INTO Homeworks(HomeworkName, TimeSent)

SELECT HomeworkName, TimeSent;

SET @counter = @counter + 1;

END

GO

SELECT COUNT(\*)

FROM Homeworks

WHERE year(TimeSent) BETWEEN 2014 AND 2015

GO

1. **Add an index to speed-up the search by date. Test the search speed (after cleaning the cache).**

USE PerformanceHomeworkDb;

CREATE INDEX IX\_DateSent

ON Homeworks (TimeSent);

GO

SELECT COUNT(\*)

FROM Homeworks

WHERE year(TimeSent) BETWEEN 2014 AND 2015

GO

1. **Add a full text index for the text column. Try to search with and without the full-text index and compare the speed.**

CREATE FULLTEXT CATALOG ft AS DEFAULT;

CREATE UNIQUE INDEX UIX\_HomeworkID ON Homeworks(HomeworkID);

CREATE FULLTEXT INDEX ON Homeworks(HomeworkName)

KEY INDEX UIX\_HomeworkID

WITH STOPLIST = SYSTEM;

GO

SELECT COUNT(\*)

FROM Homeworks

WHERE year(TimeSent) BETWEEN 2014 AND 2015

GO

1. **Create the same table in MySQL and partition it by date (1990, 2000, 2010). Fill 1 000 000 log entries. Compare the searching speed in all partitions (random dates) to certain partition (e.g. year 1995).**

**CREATE DATABASE DatabasePerformance;**

**USE DatabasePerformance;**

**CREATE TABLE Logs(**

**LogId int NOT NULL AUTO\_INCREMENT,**

**LogDate datetime,**

**LogText nvarchar(100),**

**CONSTRAINT PK\_Logs\_LogId PRIMARY KEY(LogId)**

**);**

**DROP TABLE Logs;**

**DELIMITER $$**

**CREATE PROCEDURE PopulateDB()**

**BEGIN**

**DECLARE counter INT DEFAULT 0;**

**DECLARE minDate datetime;**

**DECLARE maxDate datetime;**

**SET minDate = '1980-01-01 00:00:00';**

**SET maxDate = '2014-01-01 00:00:00';**

**START TRANSACTION;**

**WHILE counter < 1000000 DO**

**INSERT INTO Logs(LogDate, LogText)**

**VALUES(TIMESTAMPADD(SECOND, FLOOR(RAND() \* TIMESTAMPDIFF(SECOND, minDate, maxDate)), minDate),**

**CONCAT('Text', CAST(counter as CHAR)));**

**SET counter = counter + 1;**

**END WHILE;**

**COMMIT;**

**END $$**

**DROP PROCEDURE PopulateDB**

**CALL PopulateDB()**

**CREATE DATABASE PartitioningDB;**

**USE PartitioningDB;**

**CREATE TABLE Logs(**

**LogId int NOT NULL AUTO\_INCREMENT,**

**LogDate datetime,**

**LogText nvarchar(100),**

**CONSTRAINT PK\_Logs\_LogId PRIMARY KEY(LogId, LogDate)**

**) PARTITION BY RANGE(YEAR(LogDate))(**

**PARTITION p0 VALUES LESS THAN (1990),**

**PARTITION p2 VALUES LESS THAN (2000),**

**PARTITION p3 VALUES LESS THAN (2010),**

**PARTITION p4 VALUES LESS THAN MAXVALUE**

**);**

**CALL PopulateDB()**

**SELECT \* FROM Logs PARTITION (p0)**

**USE DatabasePerformance;**

**SELECT \* FROM Logs**

**WHERE LogDate < '1990-01-01'**

**USE PartitioningDB**

**SELECT \* FROM Logs PARTITION(p0)**

**WHERE LogDate < '1990-01-01'**