SQL Server 2017 New Feature

宋沄剑 2017/12/16

自我介绍

阿里云 技术专家（RDS for SQL Server团队）

Microsoft Data platform MVP（Since 2012）

多年SQL Server经验，很多企业培训/调优/技术书籍翻译/博客分享

博客：http://www.cnblogs.com/careyson

Disclaimer: 本次分享仅代表个人观点，与本人所在的公司与岗位没有关系

SQL Server 2017新功能

Cross platform

QueryStore&Automatic tunning

Python&R Integration

Security

Misc

QueryStore&Automatic Tuning

Why Query Store

性能调优，过去需要基线，需要专业人士考虑下面的点：

- 捕捉什么数据（DMV/性能计数器/执行计划/高消耗语句）
- 如何捕捉这些数据（T-SQL/PowerShell/第三方工具/各种自定义脚本）
- 何时捕捉这些数据（定期执行/特定时间执行）
- 存在哪（SQL Server/ElasticSearch等）
- 如何查看和分析（T-SQL/Grafana/ELK等）
- 数据保留期（7天/1个月等）

上面这些都需要专业人士花费大量时间才能完成

QueryStore&Automatic Tuning

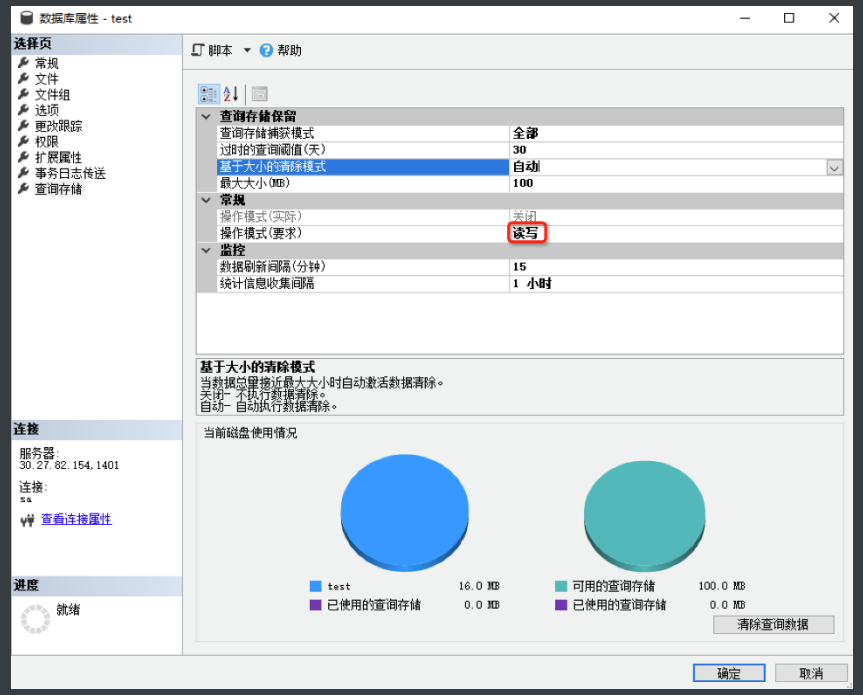
QueryStore提供了简单的方法
- 用于记录数据
- 为查询执行提供信息
- 捕捉执行计划与性能数据
- 在数据库层面启用
- 所有的SQL Server版本可用

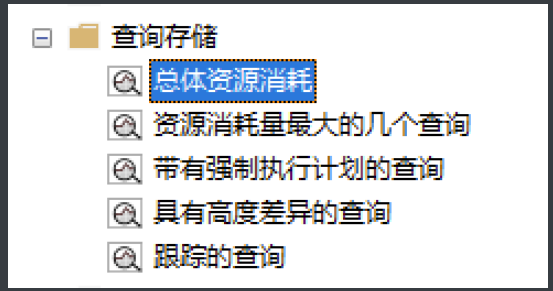
Query Store捕捉的信息
- 编译时间
- 上次执行时间点
- 执行花费时间
- CPU
- 逻辑&物理读
- 查询语句
- 执行计划

QueryStore&Automatic Tuning

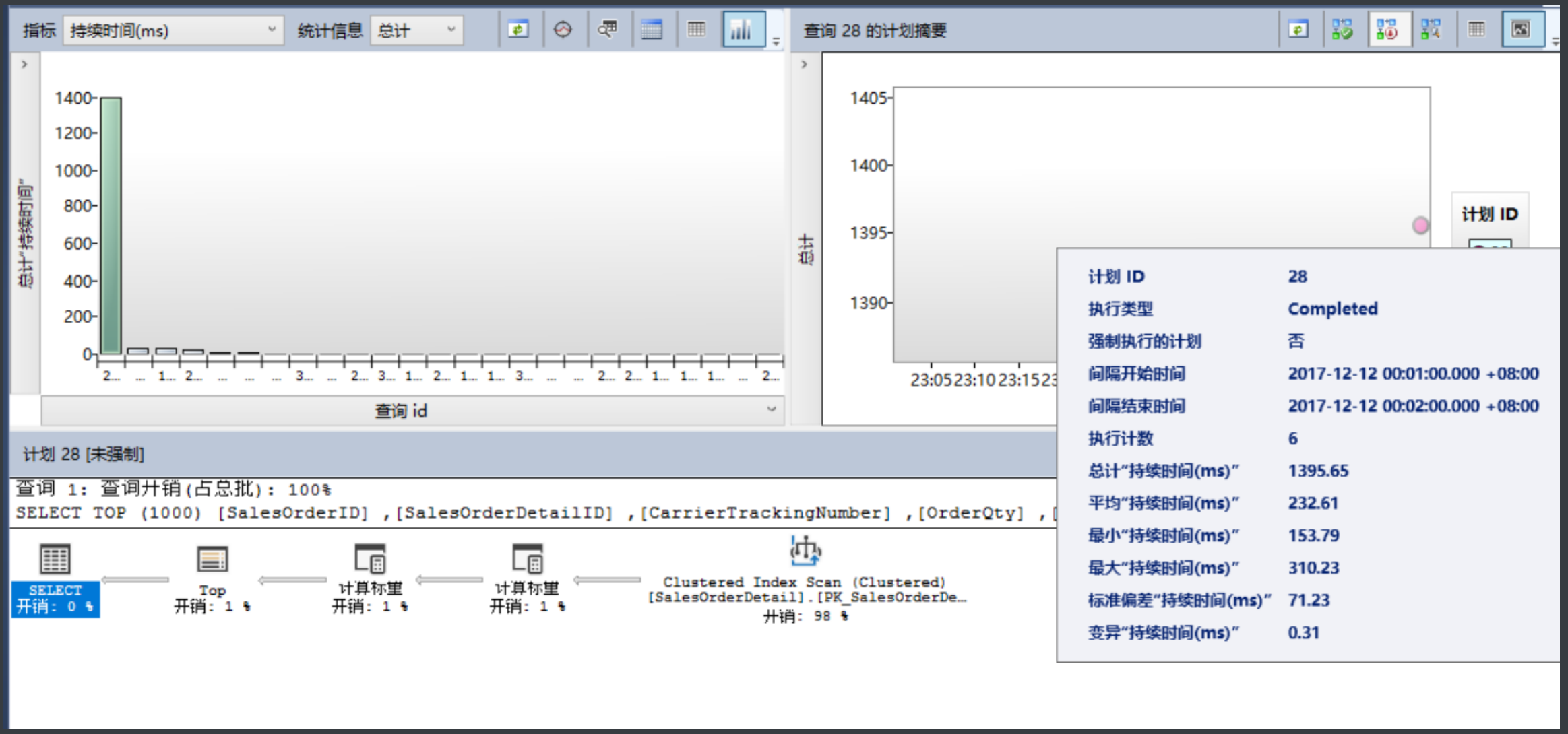
QueryStore提供了简单的方法
数据库级别启用QueryStore
存储与保留期
OPERATION_MODE
QUERY_CAPTURE_MODE
MAX_PLANS_PER_QUERY
MAX_STORAGE_SIZE_MB
CLEANUP_POLICY
SIZE_BASED_CLEANUP_MODE

聚合时间点
INTERVAL_LENGTH_MINUTES
DATA_FLUSH_INTERVAL_SECONDS





QueryStore&Automatic Tuning



QueryStore&Automatic Tuning（DMV）

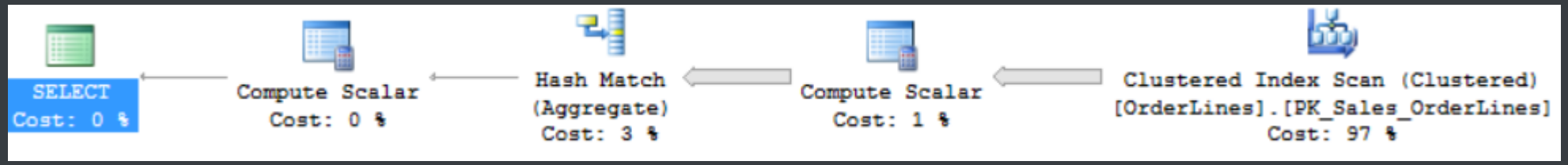
```
SELECT TOP 10
    RUNTIMESTATS.avg_physical_io_reads,
    QUERYTEXT.query_sql_text,
    QUERY.query_id,
    QUERYPLAN.plan_id,
    RUNTIMESTATS.runtime_stats_id,
    RUNTIMESTATSINT.start_time,
    RUNTIMESTATSINT.end_time,
    RUNTIMESTATS.avg_rowcount,
    RUNTIMESTATS.count_executions
FROM
    sys.query_store_query_text     AS QUERYTEXT INNER JOIN
    sys.query_store_query          AS QUERY        ON QUERYTEXT.query_text_id = QUERY.query_text_id INNER JOIN
    sys.query_store_plan           AS QUERYPLAN    ON QUERY.query_id = QUERYPLAN.query_id INNER JOIN
    sys.query_store_runtime_stats AS RUNTIMESTATS ON QUERYPLAN.plan_id = RUNTIMESTATS.plan_id INNER JOIN
    sys.query_store_runtime_stats_interval AS RUNTIMESTATSINT ON RUNTIMESTATSINT.runtime_stats_interval_id = RUNTIMESTATS.runtime_stats_interval_id
WHERE RUNTIMESTATSINT.start_time >= DATEADD(hour, -24, GETUTCDATE())
ORDER BY RUNTIMESTATS.avg_physical_io_reads DESC;
```

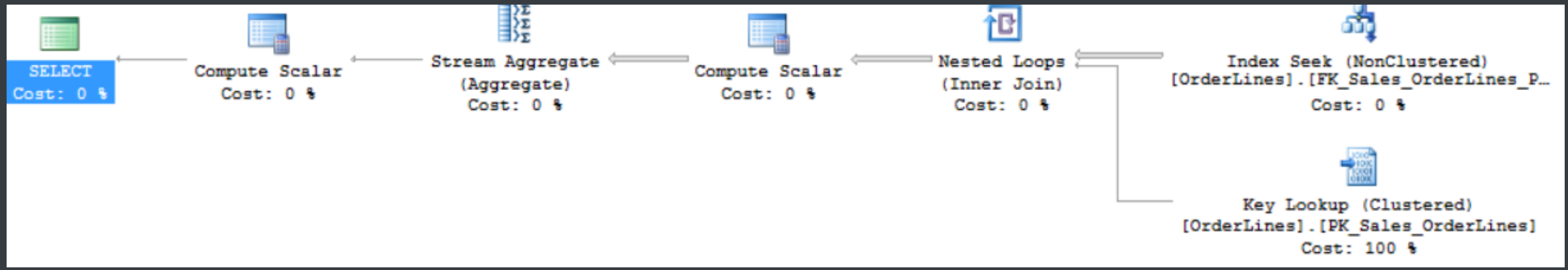
结果 消息

avg_physical_io_reads	query_sql_text	query_id	plan_id	runtime_stats_id	start_time	end_time
212.833333333333	SELECT TOP (1000) [SalesOrderID] ,[Sales...	28	28	28	2017-12-11 16:00:00.0000000 +00:00	2017-12-11 17:00:00.0000000 +0
4	(@_msparam_0 nvarchar(4000),@_msparam_1 nvarcha...	4	4	4	2017-12-11 16:00:00.0000000 +00:00	2017-12-11 17:00:00.0000000 +0
0	(@_msparam_0 nvarchar(4000),@_msparam_1 nvarcha...	11	11	11	2017-12-11 16:00:00.0000000 +00:00	2017-12-11 17:00:00.0000000 +0
0	(@_msparam_0 nvarchar(4000),@_msparam_1 nvarcha...	10	10	10	2017-12-11 16:00:00.0000000 +00:00	2017-12-11 17:00:00.0000000 +0
0	INSERT INTO #tmp_extended_remote_data_archive_t...	9	9	9	2017-12-11 16:00:00.0000000 +00:00	2017-12-11 17:00:00.0000000 +0
0	IF EXISTS(SELECT 1 FROM master.sys.syscolumns W...	8	8	8	2017-12-11 16:00:00.0000000 +00:00	2017-12-11 17:00:00.0000000 +0
0	(@_msparam_0 nvarchar(4000),@_msparam_1 nvarcha...	7	7	7	2017-12-11 16:00:00.0000000 +00:00	2017-12-11 17:00:00.0000000 +0
0	(@_msparam_0 nvarchar(4000),@_msparam_1 nvarcha...	6	6	6	2017-12-11 16:00:00.0000000 +00:00	2017-12-11 17:00:00.0000000 +0

QueryStore&Automatic Tuning

性能衰退例：



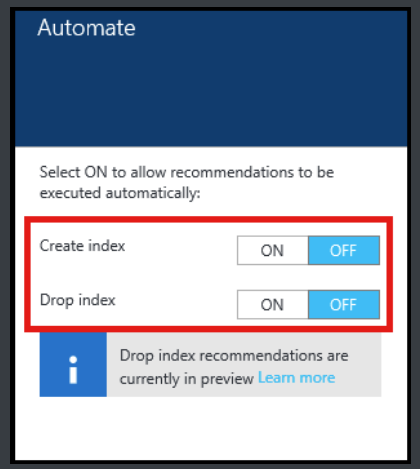


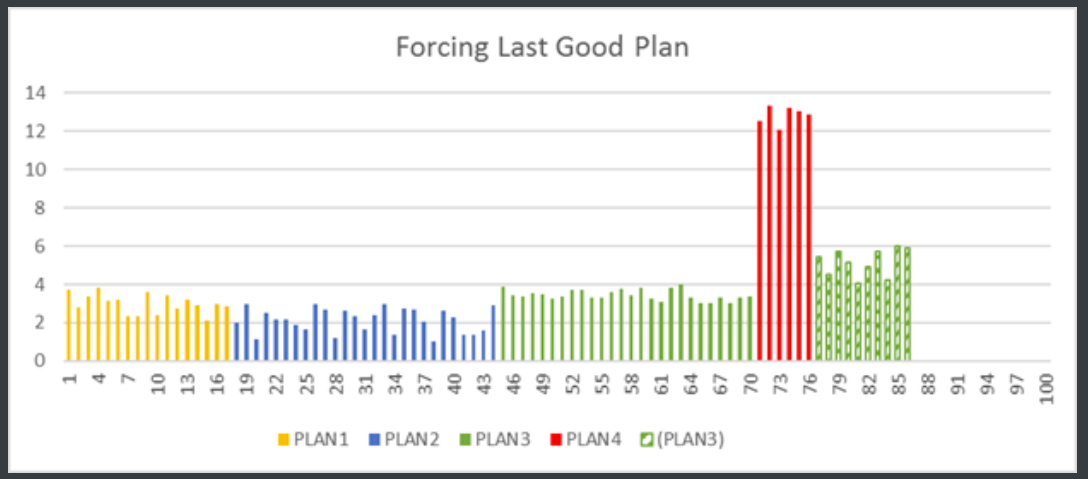
QueryStore&Automatic Tuning

数据库级别启用自动调优：
alter database test set automatic_tuning(force_last_good_plan = on)

当发现性能衰退的时候，会自动回退到上一个Cost较低的执行计划（Great feature，No More参数嗅探）

注：Azure SQL Database支持自动索引创建，box的SQL Server不支持





Cross Platform

所支持的平台：RHEL Ubuntu Docker(Docker镜像基于Ubuntu)
基于YUM与APT包/Docker镜像的安装文件
数据引擎层面支持绝大多数Windows下SQL Server的功能
Windows下的SSMS可以直接连接Linux/Docker的SQL Server
命令行：sqlcmd/bcp/sqlpackage
支持SQL Server Agent
使用Pacemaker作为可用性组与集群的基础（心跳）



Cross Platform(Docker)

Why Docker:

将环境/代码从一个环境部署到另一个环境过于复杂
其次才是省硬件资源

分析DB：
hadoop+hive

网站：
nginx+iis+logagent

缓存：Redis

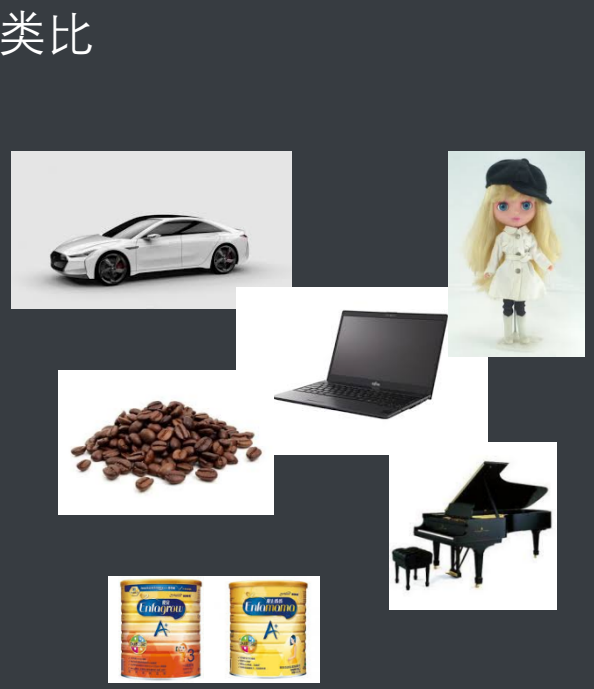
数据库：SQL Server,
dotnet 4.6,SSIS

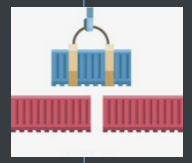
API：
Python+Java+mysql
client

队列：
kafka+zookeeper

Cross Platform(docker)

类比





Cross Platform(docker)

因此开发人员只需要关注如何打包到Container，具体如何处理Container由提供方实现

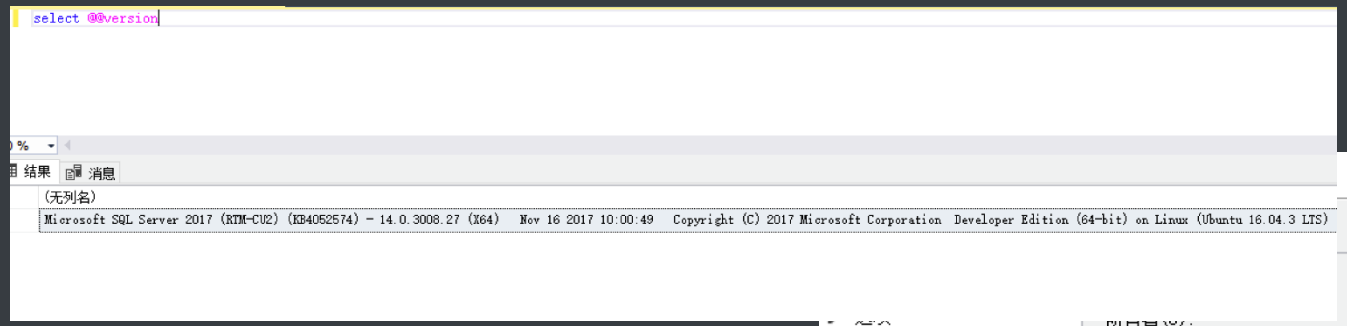
出现了新的运输工具（新型号货车），如何处理Container是货车需要适配的

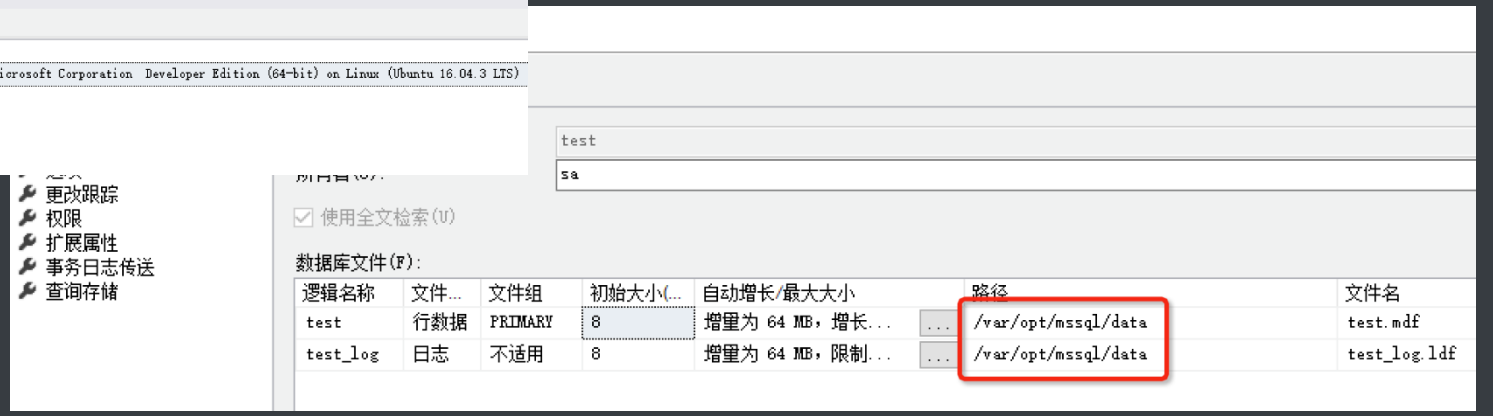
同样，出现新的平台，处理Container是平台需要时适配的（eg：Windows 2016支持Docker，相当于新出现的货车适应了Container）

Cross Platform(docker)

Docker下简单部署（Only One Step）

```
[redacted]b:opt careyson$ sudo docker run -e 'ACCEPT_EULA=Y' -e 'MSSQL_SA_PASSWORD=[redacted]' -p 1401:1433 --name sql1 -d microsoft/mssql-server-linux:2017-latest
Unable to find image 'microsoft/mssql-server-linux:2017-latest' locally
2017-latest: Pulling from microsoft/mssql-server-linux
f6fa9a861b90: Pull complete
da7318603015: Pull complete
6a8bd10c9278: Pull complete
d5a40291440f: Pull complete
bbdd8a83c0f1: Pull complete
3a52205d40a6: Pull complete
6192691706e8: Pull complete
1a658a9035fb: Pull complete
97fa7291bda1: Pull complete
b27ed30c4cf6: Pull complete
Digest: sha256:0e36a3de0913a27227fd89e9f9ff8a6d6d969ce7a54128351226060df345f8a5
Status: Downloaded newer image for microsoft/mssql-server-linux:2017-latest
0e6fd8c69d4b61214b7f22de8a29ccde56e51e28c3ceaeb89c8e25ceaee1fb46
```





Cross Platform(docker)

Docker Make Change and Commit again

```
[redacted]:opt careyson$ docker exec -it 0e /bin/bash
root@0e6fd8c69d4b:/# ls /var/opt/mssql/data
master.mdf   model.mdf     msdbdata.mdf  tempdb.mdf   test.mdf
mastlog.ldf  modellog.ldf  msdblog.ldf   templog.ldf  test_log.ldf
root@0e6fd8c69d4b:/# ls /var/opt/mssql/data -l
total 202180
-rw-r----- 1 root root  4194304 Dec 12 06:11 master.mdf
-rw-r----- 1 root root  2097152 Dec 12 06:11 mastlog.ldf
-rw-r----- 1 root root  8388608 Dec 11 11:04 model.mdf
-rw-r----- 1 root root  8388608 Dec 11 11:04 modellog.ldf
-rw-r----- 1 root root 15400960 Dec 11 11:07 msdbdata.mdf
-rw-r----- 1 root root   786432 Dec 11 11:07 msdblog.ldf
-rw-r----- 1 root root  8388608 Dec 11 09:59 tempdb.mdf
-rw-r----- 1 root root  8388608 Dec 11 11:56 templog.ldf
-rw-r----- 1 root root 75497472 Dec 11 11:57 test.mdf
-rw-r----- 1 root root 75497472 Dec 11 12:46 test_log.ldf
root@0e6fd8c69d4b:/#
```

在官方的镜像基础上
还原来自MSSQL2012的库
修改了一些配置参数

```
ali-6c96cfd8fdfb:opt careyson$ docker commit sql1 testenvironment
sha256:054ff0152c15cd92c8f109dbf8c64bd00a001d28853b6122a2a359334a1b4565
```

```
[redacted]:opt careyson$ docker images
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
testenvironment               latest              054ff0152c15        About a minute ago  1.55GB
microsoft/mssql-server-linux  2017-latest         b0bb2e68f091        3 weeks ago         1.33GB
hello-world                   latest              725dcfab7d63        5 weeks ago         1.84kB
centos                        6.8                 6704d778b3ba        5 weeks ago         195MB
```

打包并提交images

Cross Platform

Linux不支持的功能和服务

Features

- DBMail
- Alters
- FileTable
- StretchDB
- XP
- Mirror
- VSS snapshot

Service

- SSAS
- SSIS
- SSRS
- R Service
- Master Data Service
- Data Quality Service

Cross Platform On Linux

Linux与Windows在安装习惯上有很大差别

Linux往往是先安装，再配置

Windows往往是先配置再安装

SQL Server 2017遵循与Linux平台一致的安装体验

Cross Platform On Linux(REHL7)

```
[root@dlp ~]# curl https://packages.microsoft.com/config/rhel/7/mssql-server-2017.repo -o
/etc/yum.repos.d/mssql-server-2017.repo
[root@dlp ~]# curl https://packages.microsoft.com/config/rhel/7/prod.repo -o
/etc/yum.repos.d/msprod.repo

[root@dlp ~]# yum -y install mssql-server mssql-tools unixODBC-devel
```

```
[root@dlp ~]# firewall-cmd --add-port=1433/tcp --permanent
success
[root@dlp ~]# firewall-cmd --reload
success
```

```
[root@dlp ~]# sqlcmd -S localhost -U SA
Password:   # admin password you set

# show system databases
1> select name,database_id from sys.databases;
2> go
name          database_id
------------- -----------
master                  1
tempdb                  2
model                   3
msdb                    4

(4 rows affected)
```

```
[root@dlp ~]# /opt/mssql/bin/mssql-conf setup
Choose an edition of SQL Server:
  1) Evaluation (free, no production use rights, 180-day limit)
  2) Developer (free, no production use rights)
  3) Express (free)
  4) Web (PAID)
  5) Standard (PAID)
  6) Enterprise (PAID)
  7) Enterprise Core (PAID)
  8) I bought a license through a retail sales channel and have a product key to enter.

Details about editions can be found at
https://go.microsoft.com/fwlink/?LinkId=852748&clcid=0x409

Use of PAID editions of this software requires separate licensing through a
Microsoft Volume Licensing program.
By choosing a PAID edition, you are verifying that you have the appropriate
number of licenses in place to install and run this software.

# select an edition you'd like to use
Enter your edition(1-8): 2
The license terms for this product can be found in
/usr/share/doc/mssql-server or downloaded from:
https://go.microsoft.com/fwlink/?LinkId=855862&clcid=0x409

The privacy statement can be viewed at:
https://go.microsoft.com/fwlink/?LinkId=853010&clcid=0x409

# agree to the license
Do you accept the license terms? [Yes/No]:y
```

图数据库

Why图数据库

用于描述复杂的多对多关系

关系数据库对于处理多对多关系非常吃力（各种Join，行转列，随着数据的增长和关系的增加，性能几何级下降）

数据模型

Nodes：实体-例如：人/产品/商店/帖子/城市

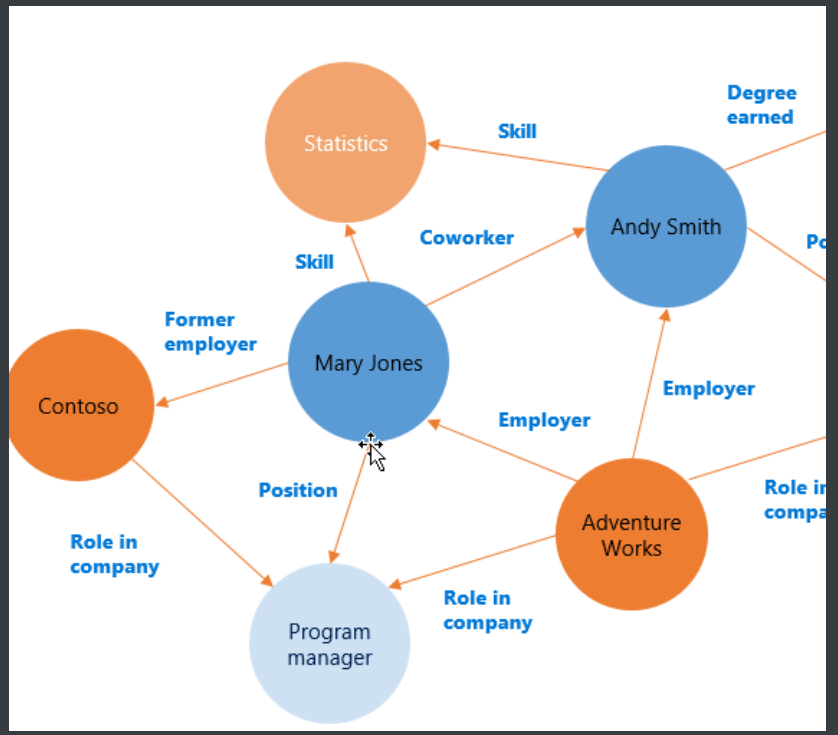
Edges：节点间的关系

PROPERTIES：描述特定Node与edge的关系

SQL Server

利用现有引擎，抽象出Node表与edge表

无需Join，像查询普通表一样查询Node与Edge表



图数据库

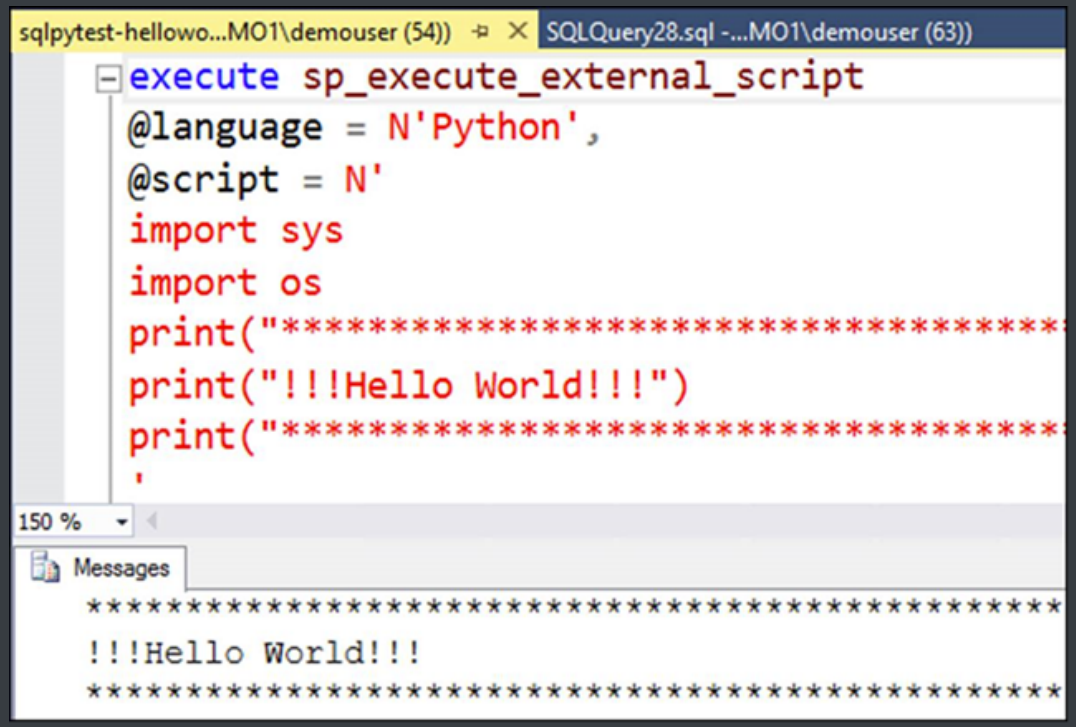
```
CREATE TABLE UniqueUser (UserId VARCHAR(80)) AS NODE;
GO

CREATE TABLE Likes (ListenCount BIGINT) AS EDGE;
GO

CREATE TABLE UniqueSong (
    SongId VARCHAR(50)
    ,SongTitle VARCHAR(500)
    ,ArtistName VARCHAR(500)
    ) AS NODE;
GO
```

```
-- Similar songs
SELECT TOP 10 SimilarSong.SongTitle, COUNT(*)
FROM    UniqueSong AS MySong,
        UniqueUser AS U,
        Likes AS LikesOther,
        Likes AS LikesThis,
        UniqueSong AS SimilarSong
WHERE   MySong.SongTitle LIKE '%Just Dance%' AND
        MATCH(SimilarSong<-(LikesOther)-U-(LikesThis)->MySong)
GROUP BY SimilarSong.SongTitle
ORDER BY COUNT(*) DESC;
```

外部代码运行Python&R

减少计算节点与数据节点间的数据移动
在SQL Server中执行R或Python可以利用SQL Server的优势
- 并行
- 权限控制
- 资源调控
- 统一的执行环境体验
- 易于部署

外部代码运行Python&R



Security

Always Encrypted（2016）

背景：如今部署环境各种复杂，数据链路较长，部分链路可能不受自己控制，可能导致敏感数据泄漏风险

- 公有云内部
- 跨公有云
- 混合云
- 私有云异地机房

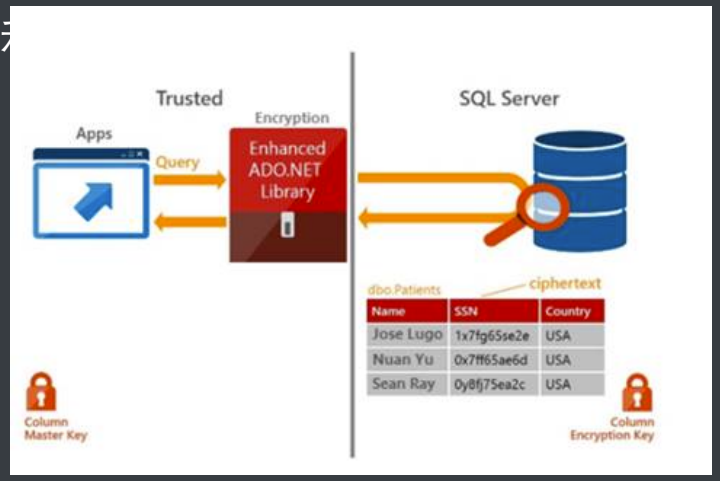
传统的Column Encryption与TDE加密证书和加密key在数据库，数据传输过程中依然是明文

方案：Client通过驱动（.net 4.6+,ODBC Driver 13.1+,JDBC Driver 6.0+）加密

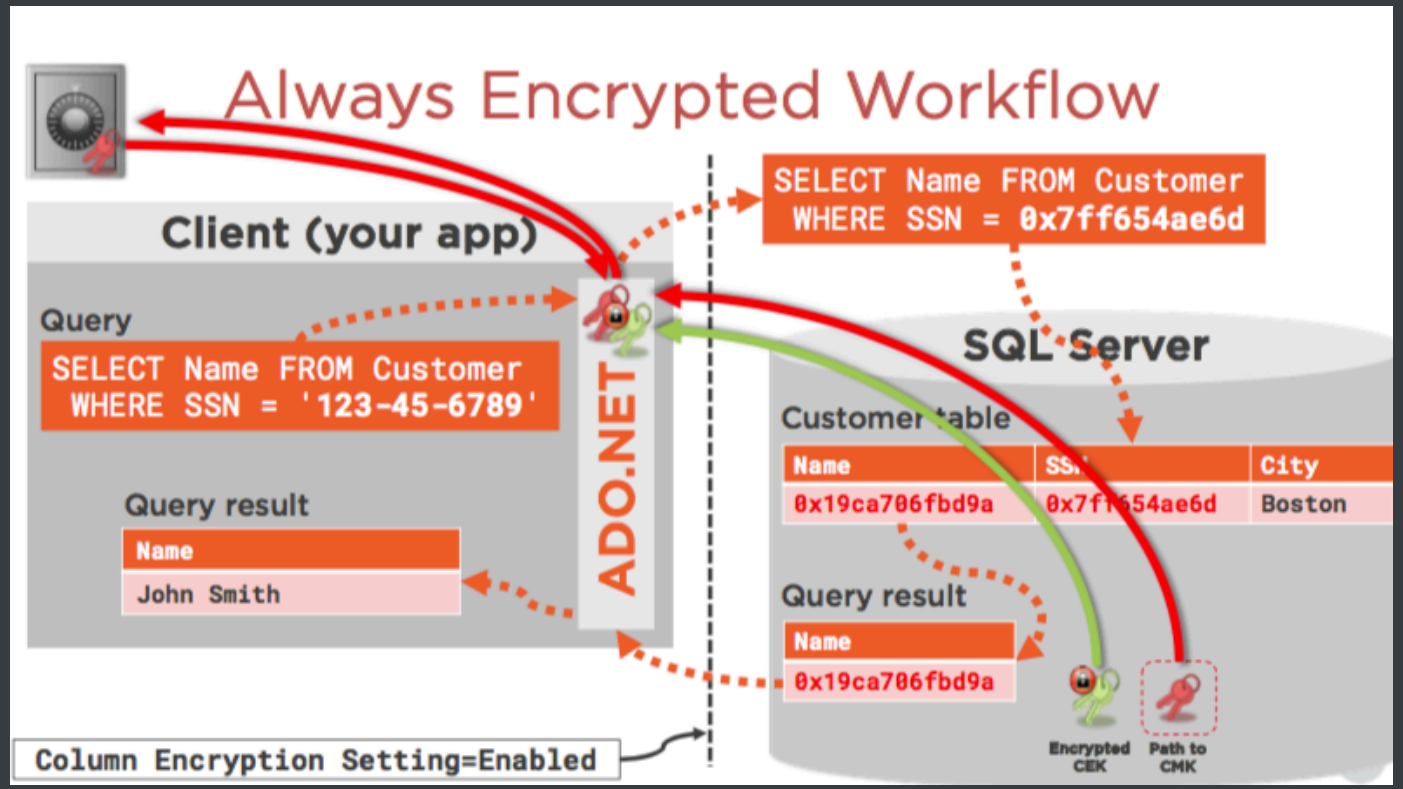
SQL Server永远不知道真正数据是什么

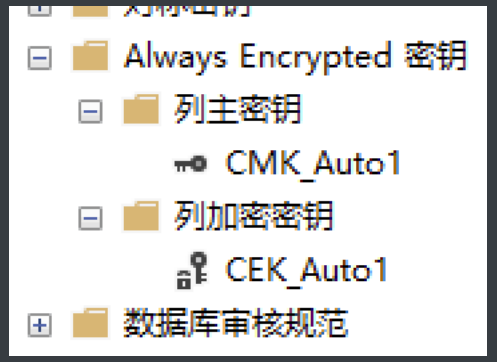
加密Key存储在第三方，SQL Server中仅存储加密文件的元数据

性能开销：5% 左右



Always Encrypted（2016）





Row Level Security（2016）

出现背景：SQL Server一直没有行级权限控制，如果希望实现这种粒度的权限控制过去只能通过View或Sp实现，这种方式需要修改程序，成本较高
行级安全可以通过函数过滤的方式定义行集权限，有下述优势：

- 应用程序透明
- 细粒度权限控制
- 中心化的安全逻辑控制

可以实现多租户源头控制数据库，例如：

- 每个办事处只能看到自己的数据
- BI/Report用户只分析部分与自己相关的数据子集

Application

```
Select * from Customer
```

Row Level Security（2016）

```
SELECT * FROM Sales
```

110 %

结果 消息

	OrderID	SalesUsername	Product	Qty
1	1	SalesUser1	Valve	5
2	2	SalesUser1	Wheel	2
3	3	SalesUser1	Valve	4
4	4	SalesUser2	Bracket	2
5	5	SalesUser2	Wheel	5
6	6	SalesUser2	Seat	5

```
-- 启用安全策略，将函数作为过滤条件
CREATE SECURITY POLICY sec.SalesPolicyFilter
    ADD FILTER PREDICATE sec.fn_securitypredicate(SalesUsername)
    ON dbo.Sales
    WITH (STATE = ON)
```

```
--过滤函数
CREATE FUNCTION sec.fn_securitypredicate(@Username AS varchar(50))
    RETURNS TABLE
    WITH SCHEMABINDING
AS
    RETURN
        -- 返回1或0
        SELECT
            1 AS fn_securitypredicate_result
        WHERE
            DATABASE_PRINCIPAL_ID() = DATABASE_PRINCIPAL_ID(@Username) OR
            DATABASE_PRINCIPAL_ID() = DATABASE_PRINCIPAL_ID('ManagerUser')
GO
```

```
-- SalesUser1只能看到自己的用户
EXECUTE AS USER = 'SalesUser1'
SELECT * FROM Sales
REVERT
```

110 %

结果 消息

	OrderID	SalesUsername	Product	Qty
1	1	SalesUser1	Valve	5
2	2	SalesUser1	Wheel	2
3	3	SalesUser1	Valve	4

```
EXECUTE AS USER = 'SalesUser2'
SELECT * FROM Sales
REVERT
```

110 %

结果 消息

	OrderID	SalesUsername	Product	Qty
1	4	SalesUser2	Bracket	2
2	5	SalesUser2	Wheel	5
3	6	SalesUser2	Seat	5

Dynamic Data Mask（2016）

出现背景：数据库内的部分数据较为敏感，需要：

- 满足监管需求
- 数据在不同环境中，敏感数据需要对开发人员/DBA/测试人员 等隔离
- 需要在不影响底层数据的情况下将这些数据脱敏
  - 列加密的方式污染底层数据且需要应用程序作出修改

没有Unmask权限的用户导出/备份 后的数据是Mask的数据

- 数据库无需脱敏可用于BI/测试环境

Dynamic Data Mask (2016)

```
CREATE TABLE Membership(
    MemberID int IDENTITY PRIMARY KEY,
    FirstName varchar(100) MASKED WITH (FUNCTION = 'partial(2, "...", 2)') NULL,
    LastName varchar(100) NOT NULL,
    Phone varchar(12) MASKED WITH (FUNCTION = 'default()') NULL,
    Email varchar(100) MASKED WITH (FUNCTION = 'email()') NULL)
```

```
-- 数据库权限为MASK
EXECUTE AS USER = 'TestUser'
SELECT * FROM Membership
REVERT
GO
```

110 %

结果 消息

	MemberID	FirstName	LastName	Phone	Email
1	1	Ro...to	Tamburello	xxxx	RXXX@XXXX.com
2	2	Ja...ce	Galvin	xxxx	JXXX@XXXX.com
3	3	...	Mu	xxxx	ZXXX@XXXX.com
4	4	...	Smith	xxxx	JXXX@XXXX.com
5	5	Da...ny	Jones	xxxx	DXXX@XXXX.com

MISC

Resumable Online Index Operation

非常适用大型索引

索引创建失败(磁盘空间满，故障转移)依然可以从失败点重试

随时开始和停止索引创建

大索引重建不会再导致日志爆掉，在rebuild的过程中可以收缩日志

例：

创建：

```
ALTER INDEX test_idx on test_table REBUILD WITH (ONLINE=ON, MAXDOP=1,
RESUMABLE=ON)
```

暂停：

```
ALTER INDEX test_idx on test_table PAUSE ;
```

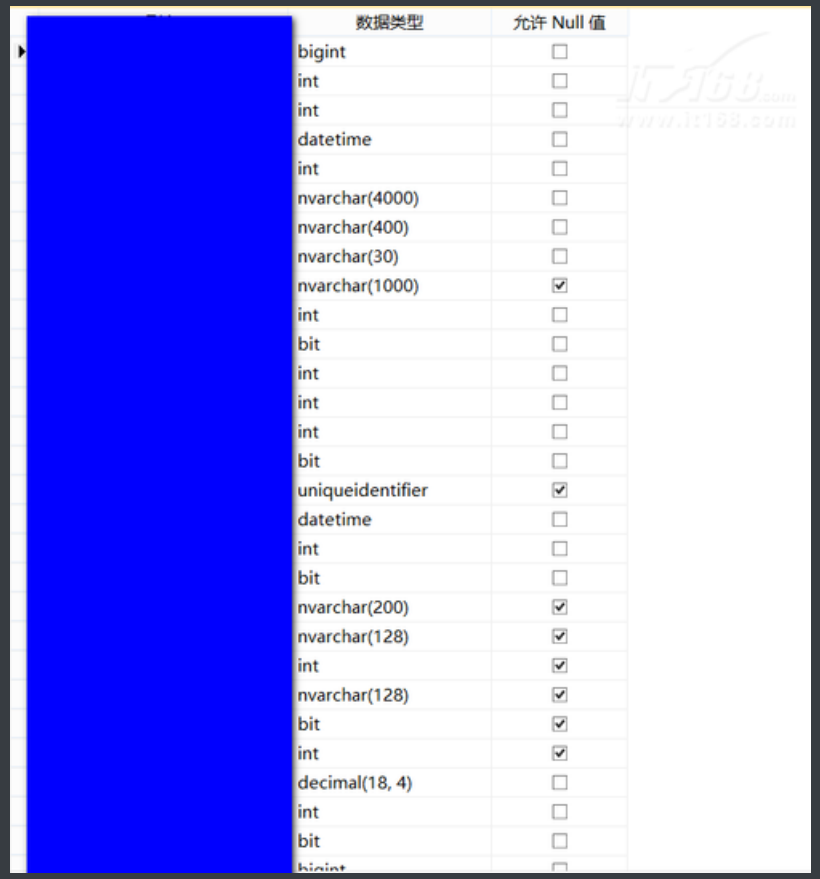
恢复并附加选项：

```
ALTER INDEX test_idx on test_table RESUME WITH (MAXDOP=4) ;
```

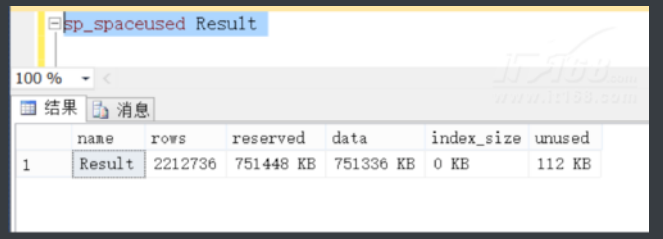
放弃：

```
ALTER INDEX test_idx on test_table ABORT;
```

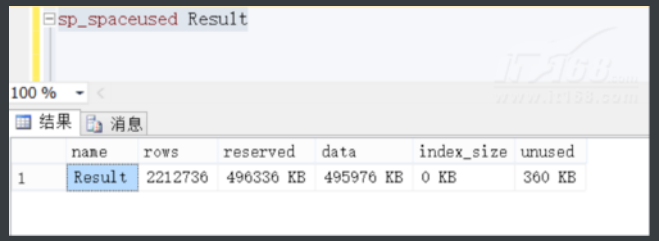
ColumnStore Index support LOB Data



未压缩2G
左右



列存储
751MB



归档列存
储压缩
495MB

现在支持LOB数据了！！！

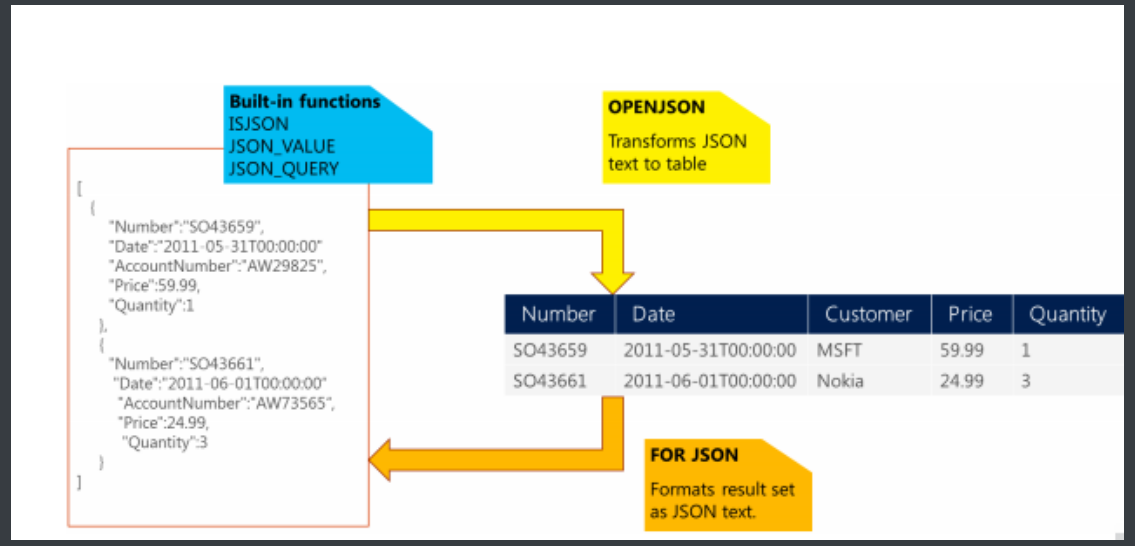
JSON Support

功能
- 从关系查询中导出或格式化JSON
- 在数据库内存储和查询JSON

类似于XML，但使用NVARCHAR（MAX）支持

为什么不使用JSON类型
- 容易从早期的JSON列迁移
- 跨功能兼容性（比如Hekaton）
- 客户端（ADO，JDBC）支持问题

没有自定义JSON索引
- 使用标准索引优化JSON查询



JSON Support

JSON -> 关系数据库

关系数据库 -> JSON

JSON函数

- ISJSON
- JSON_VALUE
- JSON_QUERY

```
DECLARE @JSON NVARCHAR(MAX);
SET @JSON = '{
                "Name": "Chocolate Cheesecake",
                "Description": "Chocolate cheesecake flavored ice cream with graham cracker pieces",
                "Cost": 7.4700,
                "Type": "Italian-style"
              }
             ,
SELECT * FROM OPENJSON(@JSON)
```

结果 消息

	key	value	type
	Name	Chocolate Cheesecake	1
	Description	Chocolate cheesecake flavored ice cream with gr...	1
	Cost	7.4700	2
	Type	Italian-style	1

```
select top 2  [SalesOrderID] as [sales.id],
[OrderQty] as [sales.qty],
[UnitPrice] as [price.price]
 from
[Sales].[SalesOrderDetail]
  FOR JSON PATH
```

结果 消息

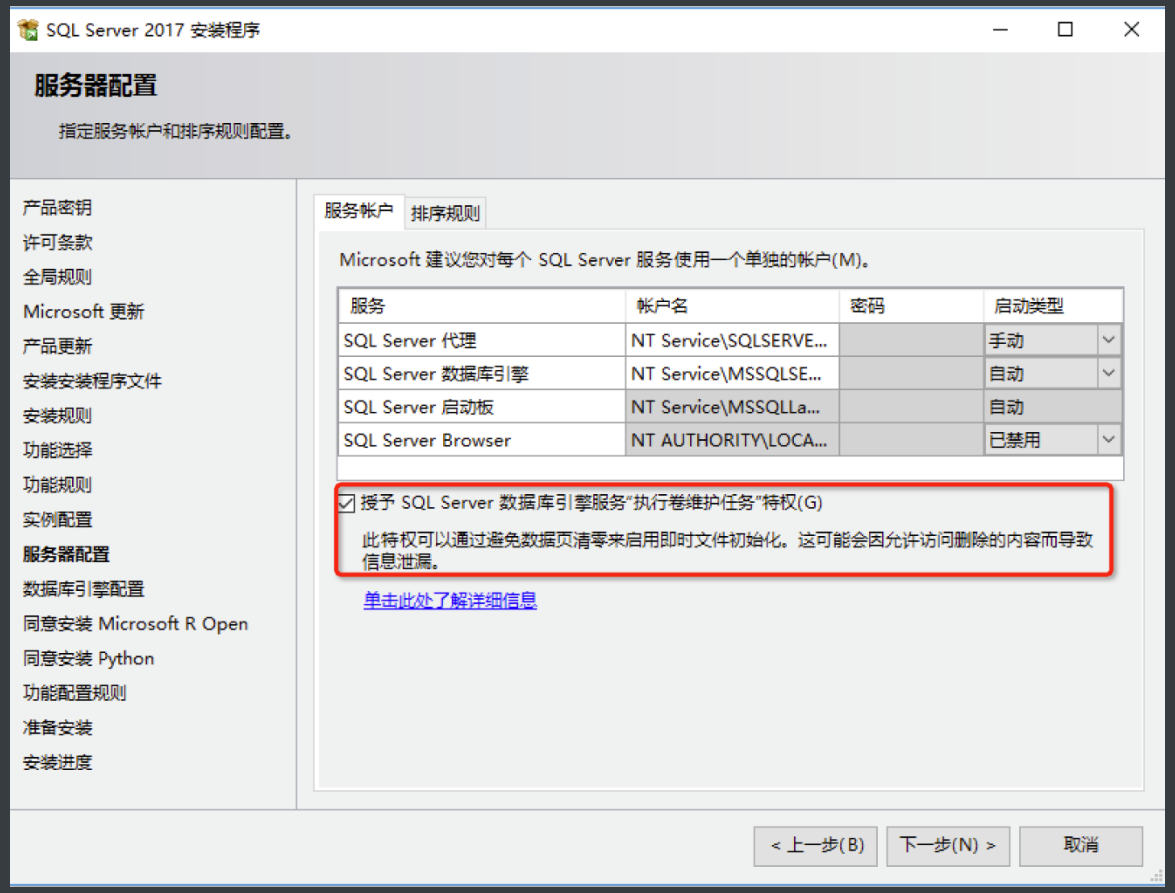
	JSON_F52E2B61-18A1-11d1-B105-00805F49916B
1	[{"sales":{"id":43659,"qty":1},"price":{"price":2024.9940}},{"sales":{"id":43659,"qty":3},"price":{"price":2024.9940}}]

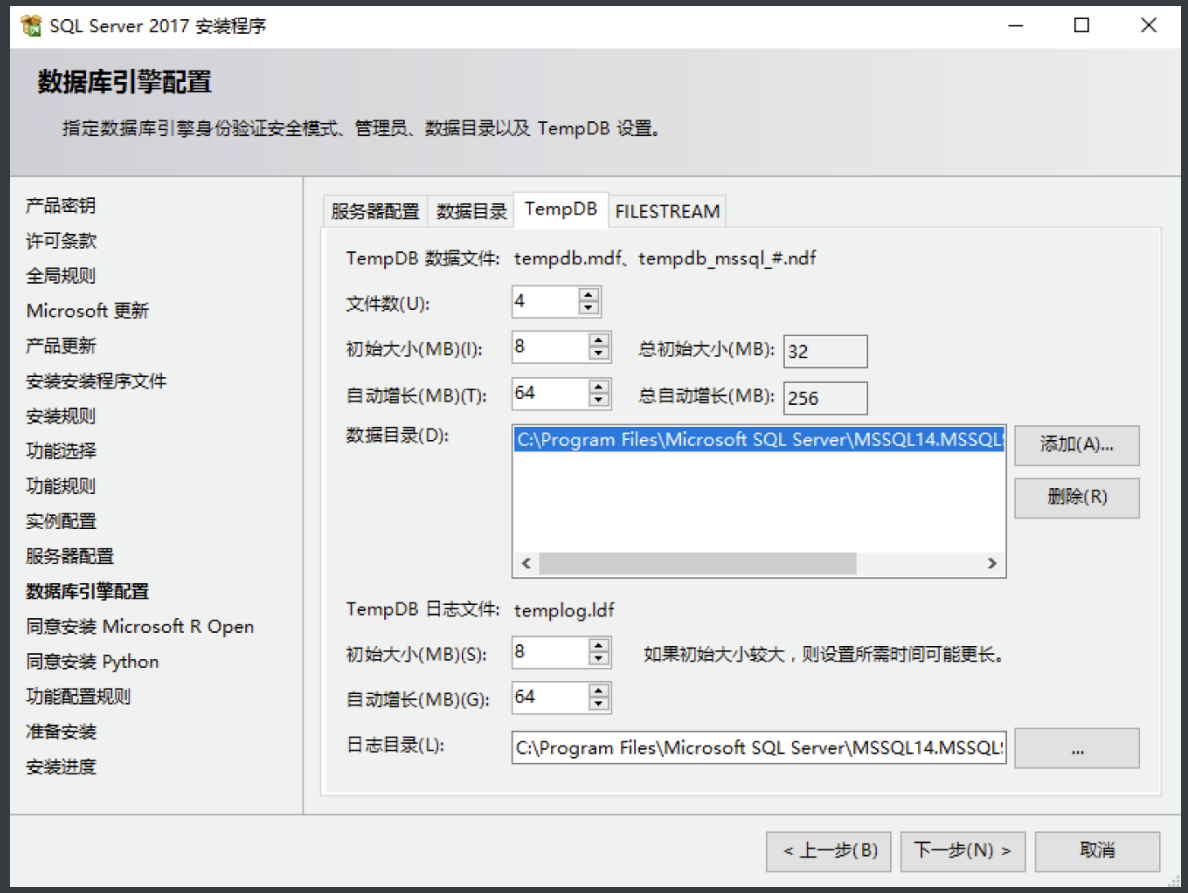
```
DECLARE @temp TABLE
(
[Id] INT,
[Value] NVARCHAR(MAX)
)

INSERT INTO @temp
VALUES
(1, '{
    "ProductName": "valid JSON"
  }'),
(2, '{
    "ProductName", "invalid JSON"
  }')

SELECT * FROM @temp
WHERE ISJSON([Value]) > 0
```

结果 消息

	Id	Value
1	1	{ "ProductName": "valid JSON" }

Installation Optimize





T-SQL Improvement(语法糖)

String_agg聚合函数
- 将一列值连接成一个字符串（可指定分隔符）

ConCat_WS,TRANSLATE,TRIM 字符串函数（语法糖）
- 拼接不通类型的字符串（varchar,nvarchar,char）
- TRANSLATE避免多次Replace函数
- TRIM避免两次LTRIM与RTRIM

日本字符集

Bulk insert支持FORMAT= ‘CSV’ (避免写很长的语句)

DROP IF EXISTS(2016，之前需要判断sys.objects)

TRUNCATE TALBE with PARTITION(2016，之前需要switch分区到空表，然后truncate)

谢谢

阿里云 | 为了无法计算的价值