RedGres

MSSQL到PostgreSQL的应用迁移

Migration Application from MSSQL to PostgreSQL

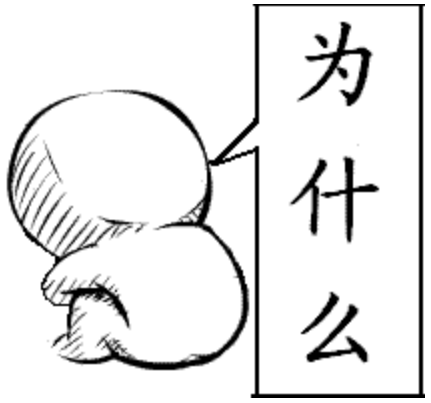
Jerray.Huang

主要内容

1. *迁移总体分析*
2. 数据迁移-类型与函数
3. 语法与存储过程迁移
4. 数据提供程序迁移

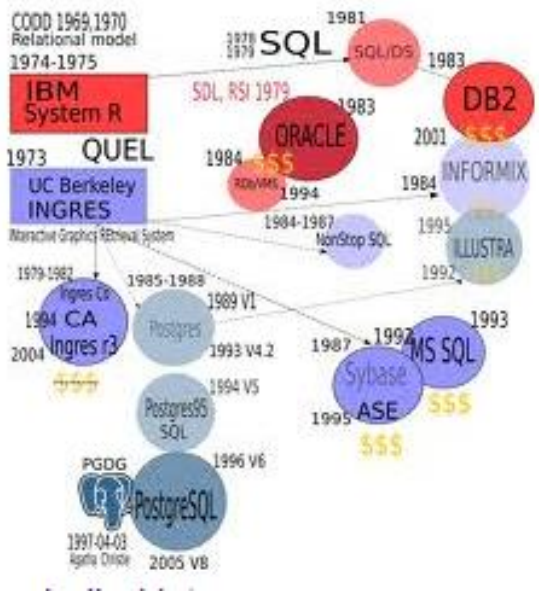
为什么要迁移

- 操作系统的问题
- 版权问题
- 经济问题
- 稳定性问题
- 客户要求
- 政府要求
- 硬件升级
- 其它

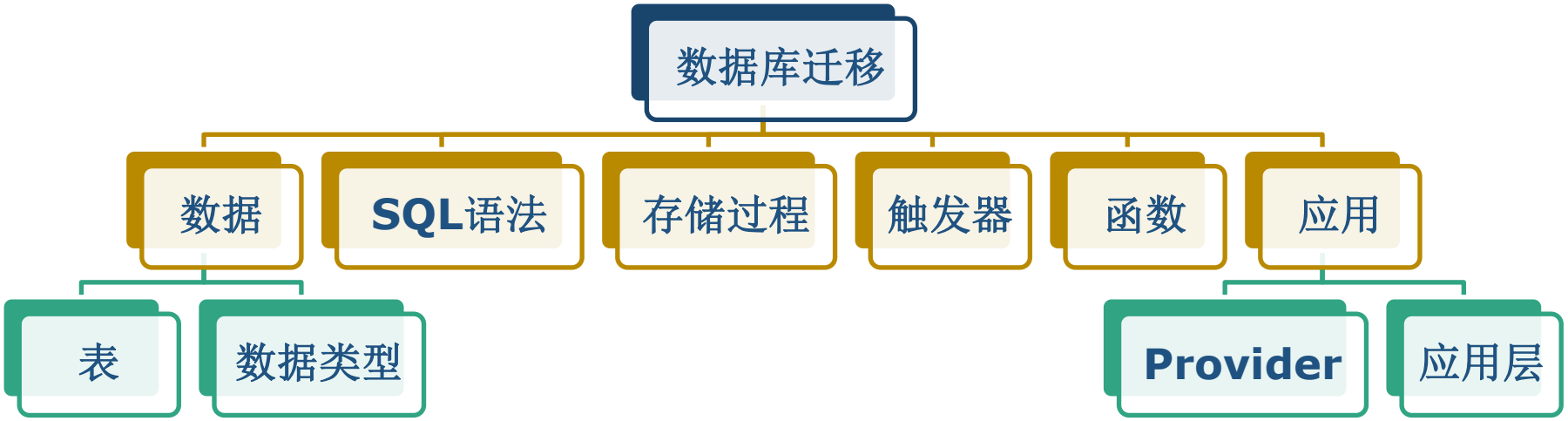


MS-SQL与PostgreSQL相似度

- SQL文类似吗？
- 数据类型类似吗？
- 存储过程类似？
- 函数类似吗？
- 调用方法类似吗？
- ……



迁移总体分析



主要内容

1. 迁移总体分析
2. *数据迁移-类型与函数*
3. 语法与存储过程迁移
4. 数据提供程序迁移

数据类型区别

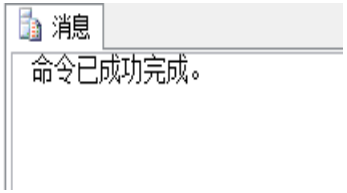
- 据统计，SQL SERVER共有28种类型，对应Postgresql 9.0，有13种类型无对应名字

数据类型区别

❖ 例1 类型datetime

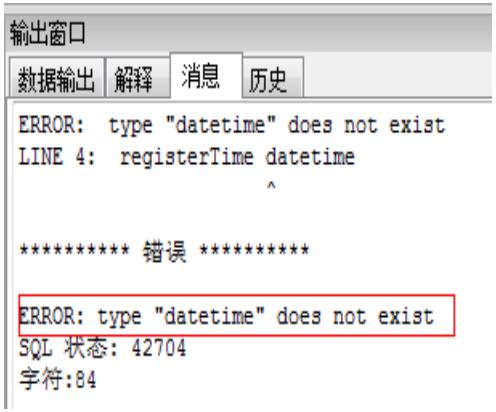
```
/*--------- Microsoft SQL Server 2005  ---------*/
CREATE TABLE datetime_test
(
    registerTime datetime
);
```

执行成功




```
/*--------- PostgreSQL 9.0  ---------*/
CREATE TABLE datetime_test
(
    registerTime datetime
);
```

执行失败



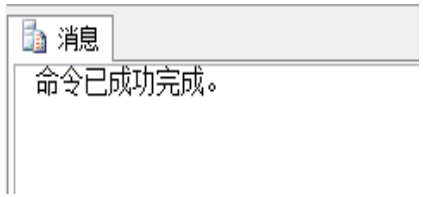
PostgreSQL
不存在*datetime*类型

数据类型区别

❖ 例2 类型nvarchar

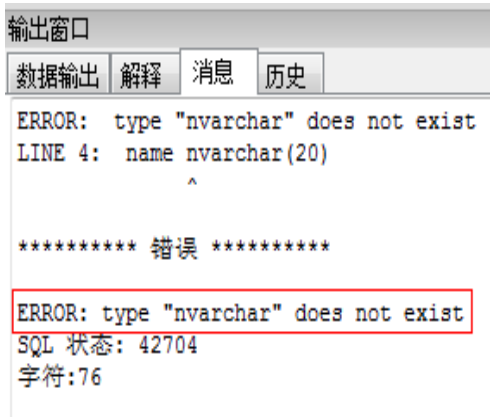
```
/*--------- Microsoft SQL Server 2005  ---------*/
CREATE TABLE nvarchar_test
(
    name nvarchar(20)
);
```

执行成功




```
/*--------- PostgreSQL 9.0  ---------*/
CREATE TABLE nvarchar_test
(
        name nvarchar(20)
);
```

执行失败



PostgreSQL
不存在*nvarchar*类型

数据类型迁移

- 要进行两类数据库的迁移，目前有两种迁移的方法
  - 手动替换
  - 使用sed/awk脚本自动替换

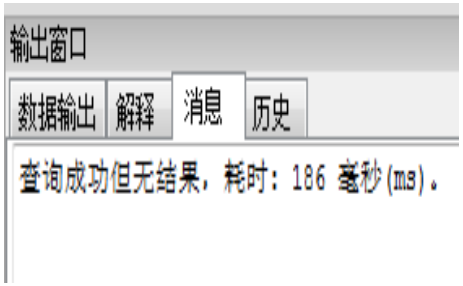
数据类型迁移

✱手动替换

- 例1 datetime
- 方法：将datetime类型修改为PostgreSQL支持的timestamp类型

```
/*--------- PostgreSQL 9.0  ---------*/
CREATE TABLE datetime_test
(
        registerTime /*datetime*/ timestamp
);
```

修改后执行成功

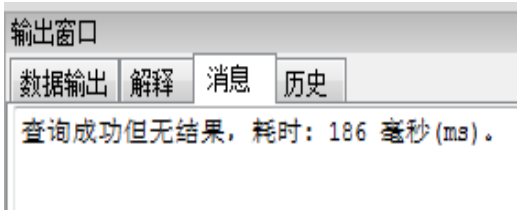


数据类型迁移

- 手动替换
  - 例2 nvarchar
  - 方法：将nvarchar类型修改为PostgreSQL支持的varchar类型

```
/*--------- PostgreSQL 9.0  ---------*/
CREATE TABLE nvarchar_test
(
        name /*nvarchar(20)*/ varchar(20)
);
```

修改后执行成功



注：由于PostgreSQL不分本地文本类型及国际文本类型，是否国际化取决于数据库的文本编码

数据类型迁移

- 脚本替换
  - 例 datetime nvarchar
  - 方法：通过sed脚本制定规则进行类型替换

```
# change datatypes
s/ bit / smallint /g
s/ datetime / timestamp /g
s/ smalldatetime / timestamp /g
s/ nvarchar / varchar /g
s/ ntext / text /g
s/ image / bytea /g
s/ tinyint / smallint /g
```

  - 将其保存为一脚本，在linux下输入命令：sed -f <规则替换脚本名> <需要替换的TSQL脚本>，即完成TSQL脚本的自动类型转换

函数区别

- 据统计，SQL SERVER中的函数共204个，有164个函数是无法直接使用的.

函数区别

例1 时间日期函数

getdate()

```
/*---------- Microsoft SQL Server 2005  ----------*/
SELECT getdate();
```

执行成功

结果 | 消息

	(无列名)
1	2011-07-11 00:24:31.563

```
/*---------- PostgreSQL 9.0  ----------*/
SELECT getdate();
```

失败，提示函数不存在

数据输出 | 解释 | 消息 | 历史

```
ERROR:  function getdate() does not exist
LINE 2: SELECT getdate();
```

day()

```
/*---------- Microsoft SQL Server 2005  ----------*/
SELECT day('2010-12-12');
```

执行成功

结果 | 消息

	(无列名)
1	12

```
/*---------- PostgreSQL 9.0  ----------*/
SELECT day('2010-12-12');
```

失败，提示函数不存在

数据输出 | 解释 | 消息 | 历史

```
ERROR:  function day(unknown) does not exist
LINE 2: SELECT day('2010-12-12');
```

函数区别

❖例2 字符串函数

- len()

```
/*---------- Microsoft SQL Server 2005  ----------*/
SELECT len('Database');
```

执行成功

结果 | 消息

	(无列名)
1	8

```
/*---------- PostgreSQL 9.0  ----------*/
SELECT len('Database');
```

失败，提示函数不存在

数据输出 | 解释 | 消息 | 历史

```
ERROR:  function len(unknown) does not exist
LINE 2: SELECT len('Database');
```

- left()

```
/*---------- Microsoft SQL Server 2005  ----------*/
SELECT left('Database',4);
```

执行成功

结果 | 消息

	(无列名)
1	Data

```
/*---------- PostgreSQL 9.0  ----------*/
SELECT left('Database',4);
```

失败，提示函数不存在

数据输出 | 解释 | 消息 | 历史

```
ERROR:  function left(unknown, integer) does not exist
LINE 2: SELECT left('Database',4);
               ^
```

函数迁移

- 目前，能够手动修改的函数主要为非系统级函数，主要方法为：
  - 通过 CREATE FUNCTION 创建SQL Server同名功能函数

函数迁移

❖ 例1 时间日期函数迁移

- getdate()

```
/*--------- PostgreSQL 9.0  ---------*/
create or replace function getdate()
returns timestamp
as $$
select cast(current_timestamp as timestamp);
$$
language sql;

select getdate();
```

修改后执行成功

数据输出	解释	消息	历史
	getdate timestamp without time zone		
1	2011-07-11 00:53:30.066		

- day()

```
/*--------- PostgreSQL 9.0  ---------*/
create or replace function day(timestamp)
returns integer
as
$$
select cast(date_part('day',$1) as integer);
$$
language sql;

select day('2011-02-03');
```

修改后执行成功

数据输出	解释	消息	历史
	day integer		
1	3		

函数迁移

❖ 例2 字符串函数迁移

- len()

```
/*--------- PostgreSQL 9.0  ---------*/
create or replace function len(varchar)
returns int
as $$
select length(rtrim($1,' '));
$$
language sql;

select len('Database');
```

修改后执行成功

数据输出	解释	消息	历史
	len integer		
1	8		

- left

```
/*--------- PostgreSQL 9.0  ---------*/
create or replace function left(varchar,integer)
returns varchar
as $$
select substr($1,1,$2);
$$
language sql;

select left('Database',4);
```

修改后执行成功

数据输出	解释	消息	历史
	left character varying		
1	Data		

函数迁移

❖ 例3 reverse函数迁移

```
/*--------- PostgreSQL 9.0  ---------*/
select reverse('Database');
```

修改前执行失败

数据输出 | 解释 | 消息 | 历史

```
ERROR:  function reverse(unknown) does not exist
LINE 2: select reverse('Database');
               ^
```

```
/*--------- PostgreSQL 9.0  ---------*/
create or replace function reverse(TEXT) RETURNS TEXT AS $$
  DECLARE
    original ALIAS FOR $1;
    reversed TEXT := '';
    onechar  VARCHAR;
    mypos    INTEGER;
  BEGIN
    SELECT LENGTH(original) INTO mypos;
    LOOP
      EXIT WHEN mypos < 1;
      SELECT substring(original FROM mypos FOR 1) INTO onechar;
      reversed := reversed || onechar;
      mypos := mypos -1;
    END LOOP;
    RETURN reversed;
  END
$$ LANGUAGE plpgsql;

select reverse('Database');
```

修改后执行成功

数据输出 | 解释 | 消息 | 历史

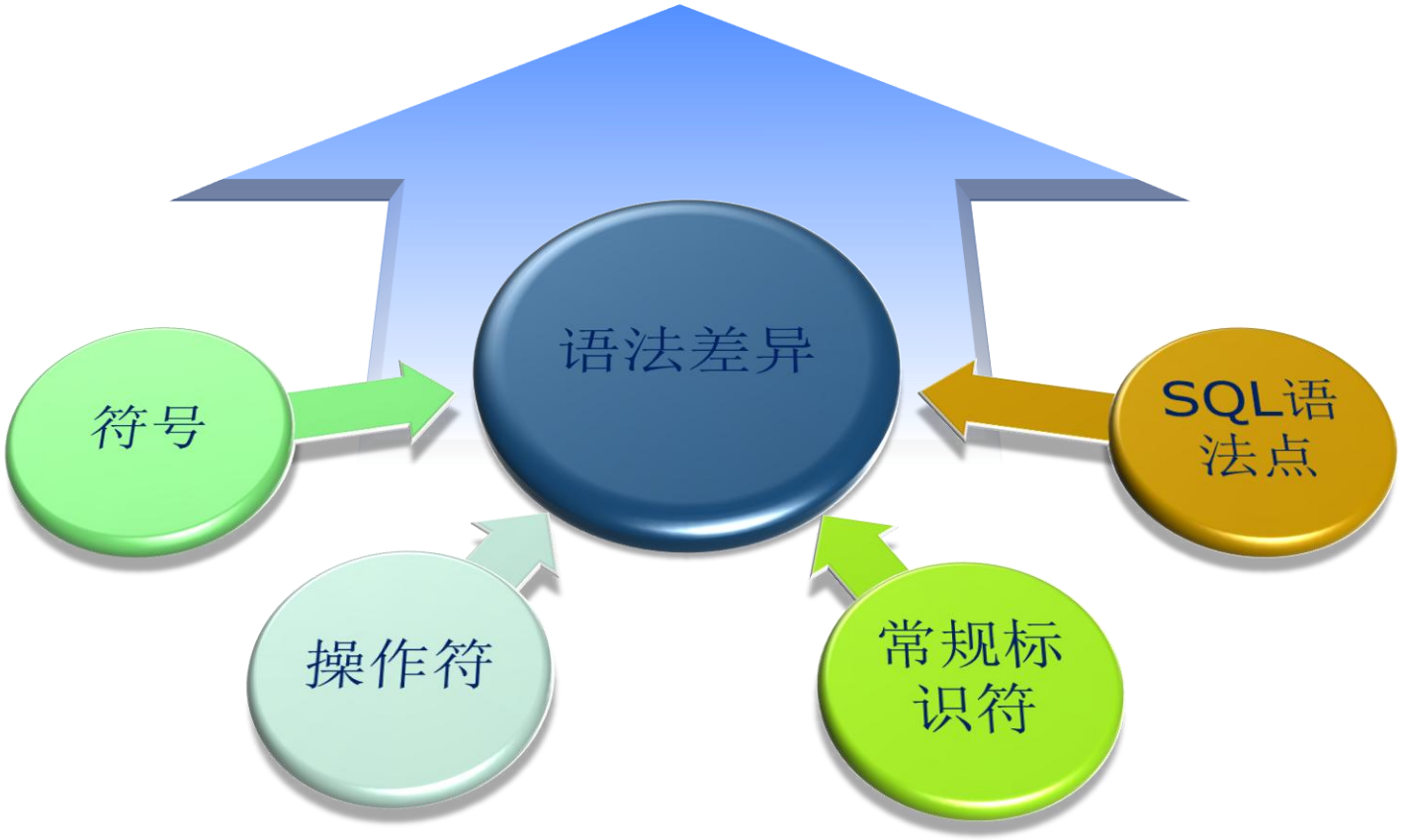
	reverse text
1	esabataD

主要内容

1. 迁移总体分析
2. 数据迁移-类型与函数
3. *语法与存储过程迁移*
4. 数据提供程序迁移

语法迁移

SQL Server和PostgreSQL语法迁移



语法差异--常规标识符

```
CREATE PROCEDURE Identifier_Test
@parameter  int
AS
    RETURN
GO
```

```
CREATE FUNCTION Identifier_Test(parameter int)
RETURNS VOID
AS
$$
    BEGIN
    END;
$$LANGUAGE plpgsql;
```

MS-变量名定义对比-PG

语法差异--符号

- 一些符号在SQL Server和PostgreSQL两边具有不同的功能，如中括号([])、数字符号(#)

```
CREATE TABLE [Brackets]
(
    brackets [INT]
)
```

```
CREATE TABLE Brackets
(
    brackets int[]
);
```

MS-中括号对比-PG

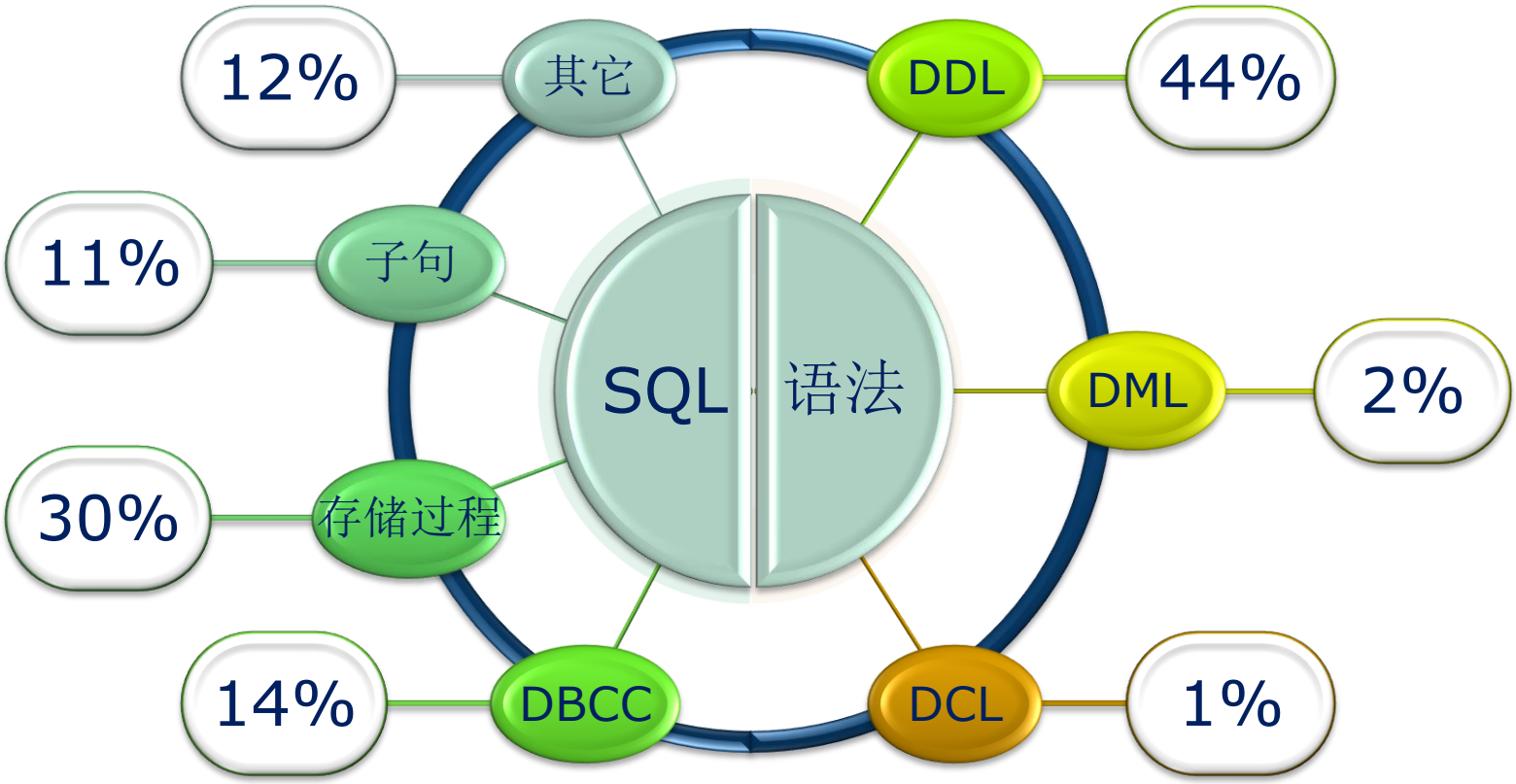
语法差异--操作符

```
CREATE PROCEDURE sqlserver_StringConcatenation
AS
DECLARE @sql varchar(100)
SET @sql = 'SELECT *' + 'from student'
GO
```

```
CREATE FUNCTION postgre_StringConcatenation()
RETURNS text
AS
$$
DECLARE sql text;
BEGIN
        sql = 'SELECT *' || 'from student';
        RETURN sql;
END;
$$LANGUAGE plpgsql;
```

MS-字符串连接对比-PG

语法差异--SQL语法点



❖举例:TOP关键字

```
SELECT TOP 2 * FROM sqlserver_Table
```

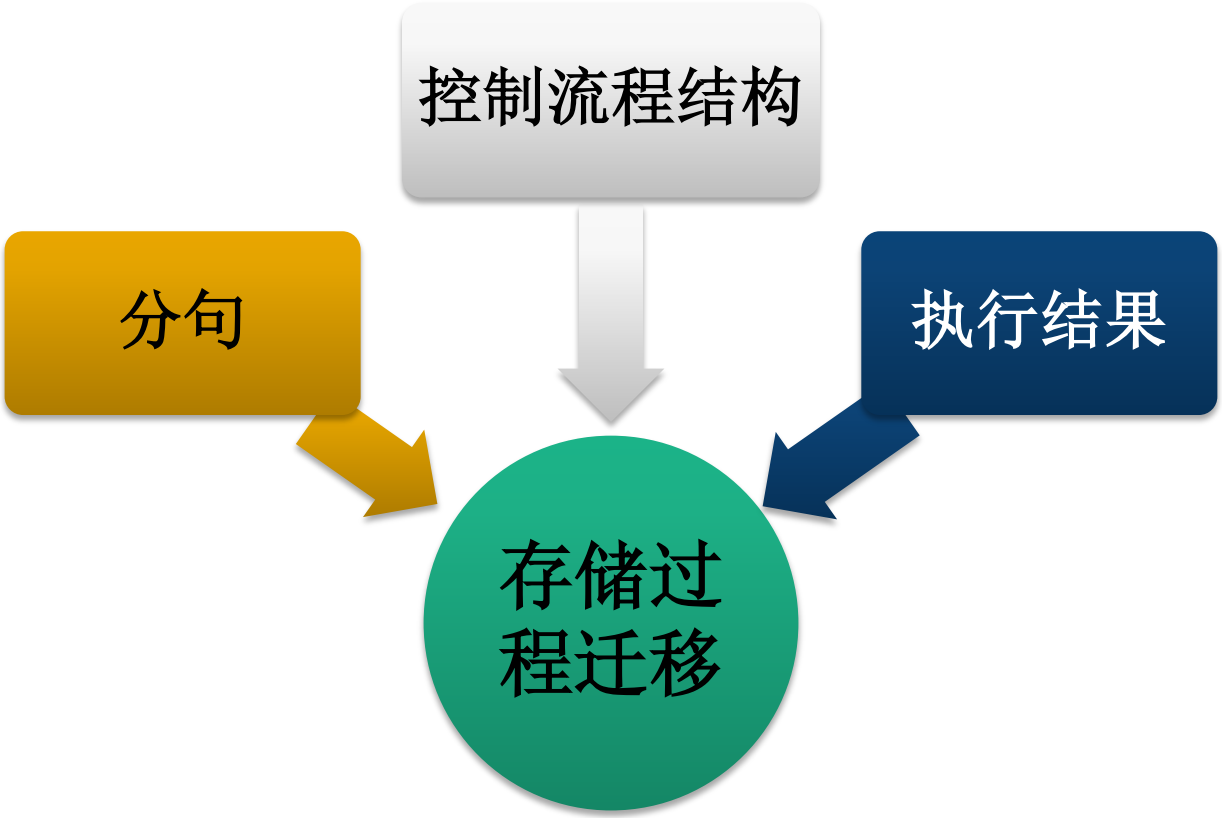
SQL Server

```
SELECT  TOP 2 *  FROM postgre_Table
```

PostgreSQL

```
ERROR:  syntax error at or near "2"
LINE 1: SELECT  TOP 2 *  FROM postgre_Table
                ^
```

存储过程迁移



存储过程迁移--分句

- SQL Server 的存储过程语句之间分句可以使用分号(';')，也可以是空格
- PostgreSQL的语句之间分句必须使用分号(';')

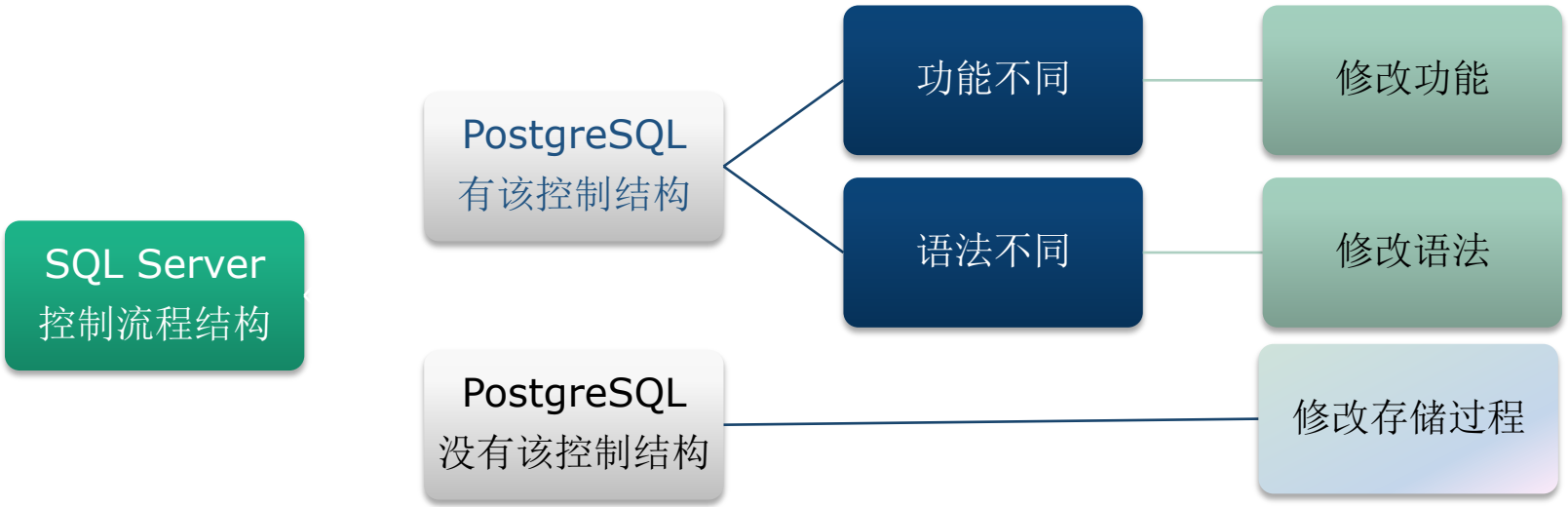
❖举例：分句

```
CREATE PROCEDURE Expressions_Separate_No_Semicolon
@flag int
AS
DECLARE @sql varchar(100)
IF @flag > 0
    BEGIN
    SET @sql = 'SELECT * FROM student'
    EXECUTE(@sql)
    END
ELSE
    BEGIN
    SET @sql = 'SELECT * FROM course'
    EXECUTE(@sql)
    END
GO
```

MS-分句对比-PG

```
CREATE FUNCTION Expressions_Separate_With_Semicolon(flag int) RETURNS SETOF record
AS
$$
BEGIN
        IF flag > 0
                THEN RETURN QUERY SELECT * FROM pg_language;
        ELSE
                RETURN QUERY SELECT * FROM pg_proc;
        END IF;
END;
$$LANGUAGE plpgsql;
```

存储过程迁移—控制流程结构



```
CREATE PROCEDURE Control_If_Else
@count int
AS
IF @count <> 0
    SELECT * FROM [msdb].[dbo].[sysjobs]
ELSE
    SELECT * FROM [msdb].[dbo].[sysjobservers]
GO
```

MS-控制流程结构-PG

```
CREATE FUNCTION Control_If_Else(param int)
RETURNS SETOF RECORD
AS
$$
BEGIN
    IF param <> 0
        THEN RETURN QUERY SELECT * FROM pg_proc;
    ELSE
        RETURN QUERY SELECT * FROM pg_language;
    END IF;
END;
$$LANGUAGE plpgsql;
```

存储过程迁移--执行结果

PostgreSQL

SQL Server

SQL Server存储过程执行结果

```
CREATE PROCEDURE ProcedureResult
@inparameter int,
@outparameter int OUTPUT
AS
IF @inparameter >= 0
    BEGIN
        SET @outparameter = @inparameter
        SELECT * FROM [msdb].[dbo].[sysjobs]
        RETURN 1
    END
ELSE
    BEGIN
        SET @outparameter = -1 * @inparameter
        SELECT * FROM [msdb].[dbo].[sysjobservers]
        RETURN -1
    END
GO
```

OUT参数

存储过程执行状态

存储过程内

PostgreSQL函数执行结果

```
CREATE FUNCTION ProcedureResult(inparam int, out outparam int)
RETURNS int
AS
$$
BEGIN
        outparam = 3 * inparam;
        RETURN outparam;
END;
$$LANGUAGE plpgsql;
```

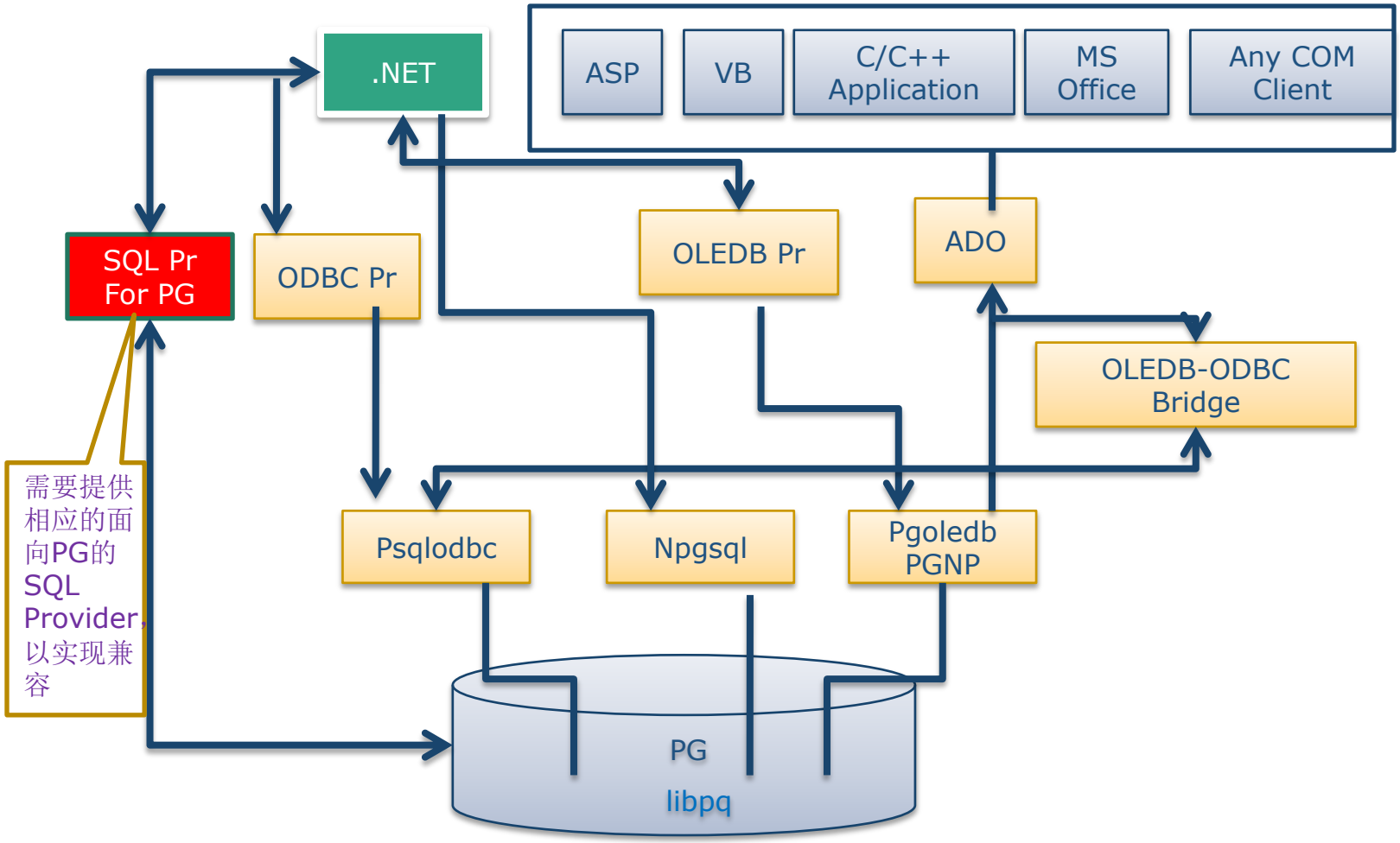
函数返回值，三者必须相对应
并且仅有一个返回值

主要内容

1. 迁移总体分析
2. 数据迁移-类型与函数
3. 语法与存储过程迁移
4. *数据提供程序迁移*

Postgresql Data Provider

- .NET 平台访问PG数据库有多种，其中Npgsql最为方便



Sqlclient与Npgsql Data Provider

- .NET 平台访问SQL Server数据库使用Sqlcient，与访问PG数据库的Npgsql比较

- *System.Data.SqlClient*
  - SqlClientFactory
  - SqlCommand
  - SqlCommandBuilder
  - SqlConnection
  - SqlConnectionStringBuilder
  - SqlDataAdapter
  - SqlDataReader
  - SqlParameter
  - ……

vs

- *Npgsql*
  - NpgsqlFactory
  - NpgsqlCommand
  - NpgsqlCommandBuilder
  - NpgsqlConnection
  - NpgsqlConnectionStringBuilder
  - NpgsqlDataAdapter
  - NpgsqlDataReader
  - NpgsqlParameter
  - ……

Sqlclient迁移Npgsql Data Provider

Sqlcliet连接SQL Server应用程序 → **Npgsql连接PG应用程序**

- 至少经过以下几个步骤：

Sqlclient迁移Npgsql Data Provider

```
Using  System.Data.SqlClient;
…
SqlConnection conn = new
SqlConnection(connStr);
conn.Open();
…
SqlCommand command = new
SqlCommand("insert into
dbo.table1 values(1, 1)", conn);
…
rowsaffected =
command.ExecuteNonQuery();
…
conn.Close();
```

(2)修改命名空间

(3)修改相关类

(4)去除语法差异

```
using Npgsql;
…
NpgsqlConnection conn = new
NpgsqlConnection(connStr);
conn.Open();
…
NpgsqlCommand command =
new NpgsqlCommand("insert
into table1 values(1, 1)",
conn);
…
rowsaffected =
command.ExecuteNonQuery()
;
…
conn.Close();
```

Sqlclient迁移Npgsql Data Provider

```
...
SqlParameter  param = new
SqlParameter(ParamName,
SqlDbType. NVarChar, Size);
...
```

(5)修改db类型

```
...
NpgsqlParameter  param =
new NpgsqlParameter
(ParamName, NpgsqlDbType.
Text, Size);
...
```

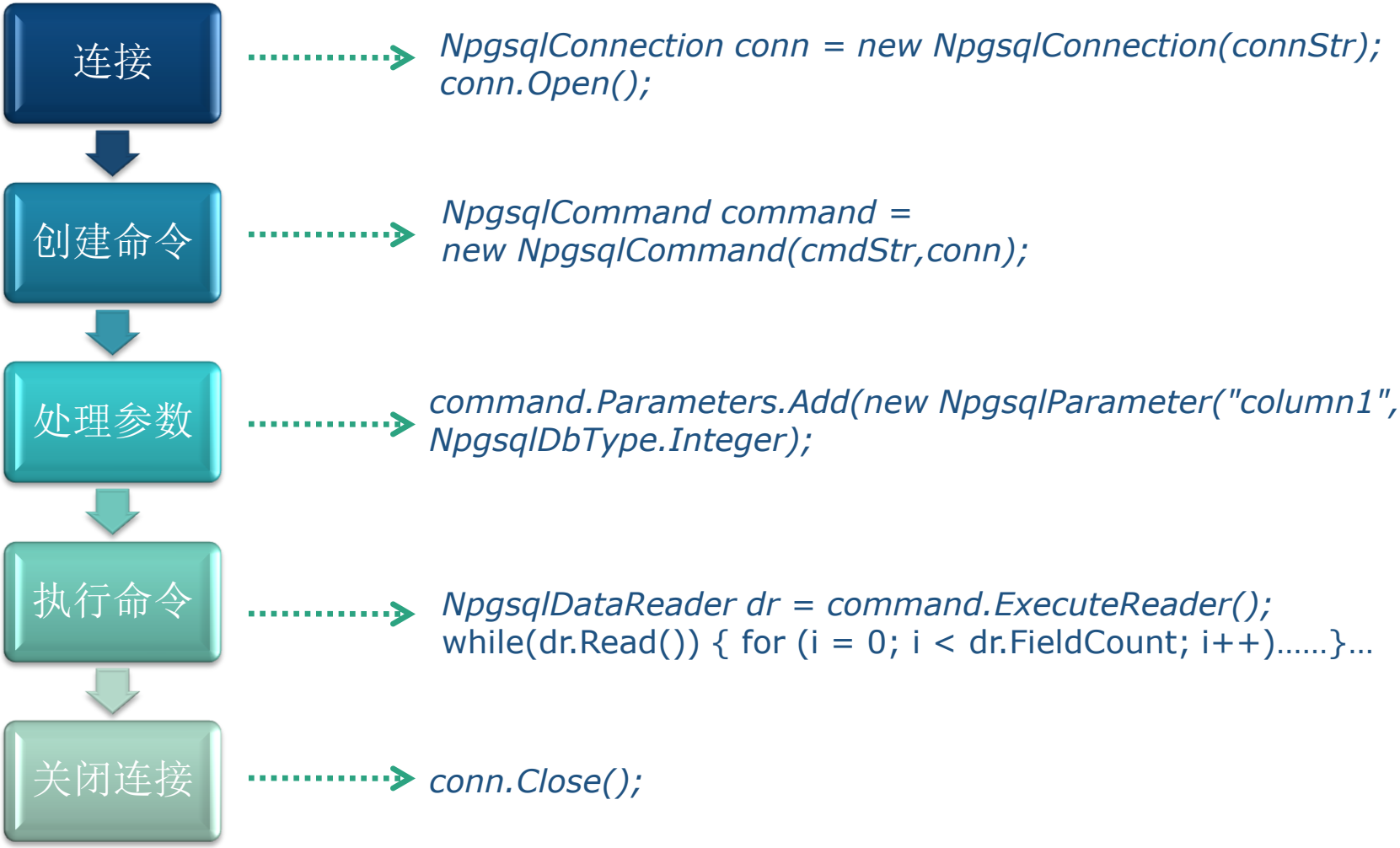
```
...
String ConnStr = "Data
Source=192.168.10.38;User
ID=sa;Password=111111;In
itial Catalog=dnt36;";
...
```

(6)修改连接字符串

```
...
String ConnStr =
"Server=127.0.0.1;Port=54
32;User
Id=joe;Password=secret;Da
tabase=joedata;";
...
```

直接使用Npgsql开发.NET程序

- 至少经过以下几个步骤：



还有没有更简单的方案

- 完全支持存储过程方式
- 支持TSQL的语法
- 支持类型的差异
- 支持最少的应用程序的变化
- 支持.....所有

- 一键式完成所有的所有！
- 请兴趣的朋友下来可以跟我交流


Thank You !