DATAGRILLEN 2026



BABELFISH: THE SQL SERVER KILLER?

Gianluca Sartori



Founder of

QUANTUMDATIS

Independent SQL Server consultant

Data platform MVP

Works with SQL Server since 7.0

DBA @ Scuderia Ferrari

@spaghettidba.bsky.social

spaghettidba.com

quantumdatis.com

DATA SATURDAYS



AGENDA

- MAGIC
- WHAT IS BABELFISH?
- HOW DO I GET IT?
- DOES IT WORK?
- DOES IT __*REALLY*__ WORK????
- SHOULD I USE IT?





WHAT IS BABELFISH

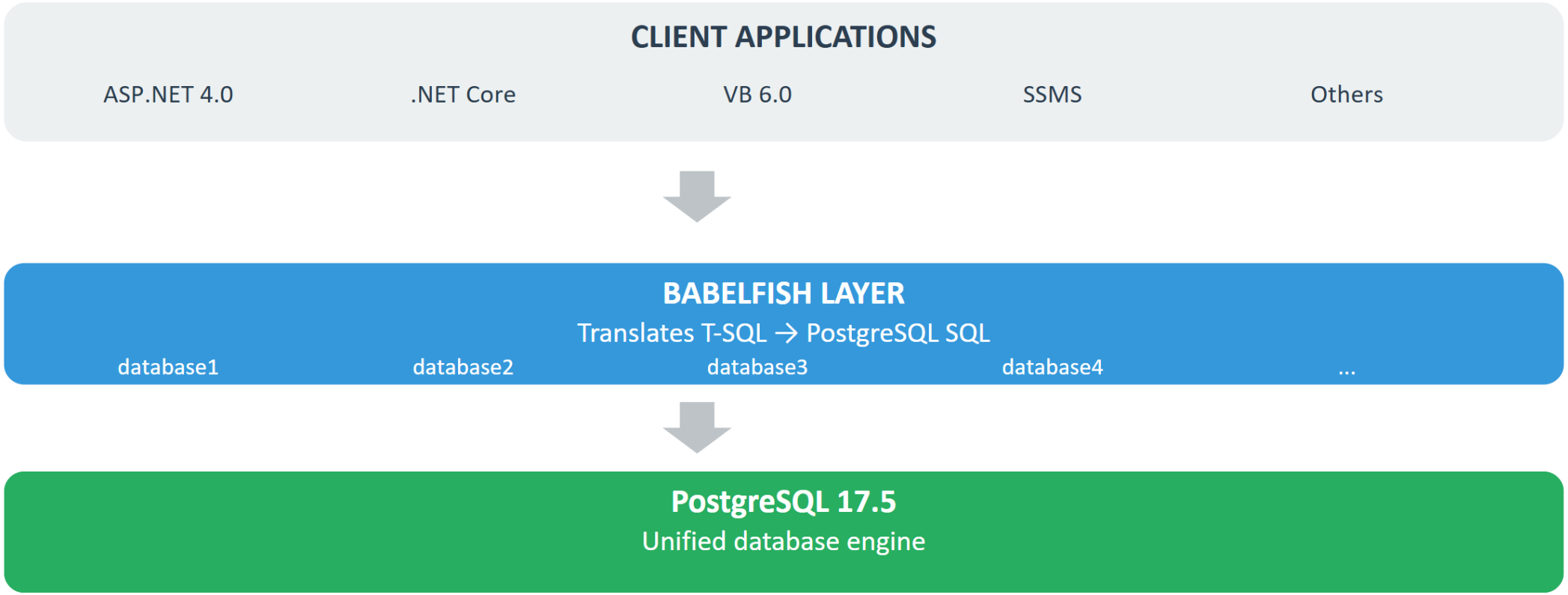
Babelfish promises drop-in replacement for SQL Server

- PostgreSQL extension
- Developed ad maintained by AWS
- Open-source, active development
- Speaks T-SQL over TDS
- Offers low-friction migration path from SQL Server to PostgreSQL
- Provides migration tools and docs
- Not a silver bullet



BABELFISH ABSTRACTION LAYER

You speak T-SQL, it speaks PGSQL



HOW DO I GET IT?

Can be very easy, can be spectacularly hard

Two options:

1) AWS Aurora PostgreSQL



1) On-prem



BUILDING BABELFISH

Oh, you really want to do it?

- No public official container image
- Some unofficial images
- Build it yourself
  - Create a linux VM with gcc and *A LOT* of dependencies
  - Build both Babelfish AND PostgreSQL
  - Build 4 additional extensions (pg_money, pg_common, pg_TDS, pg_TSQL)
  - Build will probably fail unless you have exact same distro + version in docs → ubuntu 24 LTS works
  - If latest version fails, pick an older release





GETTING STARTED

- **Assess**
  - Inspect the source SQL Server database
  - Discover unsupported objects
- **Create**
  - Set up the Babelfish database ← it is ONE Babelfish database per PG instance
- **Migrate schema**
- **Migrate data**



BABELFISH SINGLE-DB MODE

One database in SQL Server, one database in PostgreSQL

T-SQL View (port 1433)	PostgreSQL View (port 5432)
Database Structure: **Databases: MyDatabase** **Schemas:** • dbo • sales • hr **Example Query:** USE database1; SELECT * FROM dbo.customers; SELECT * FROM sales.orders; SELECT * FROM hr.employees;	**Database Structure:** **Database: babelfish_db** **Schemas:** • dbo • sales • hr **Example Query:** SELECT * FROM dbo.customers; SELECT * FROM sales.orders; SELECT * FROM hr.employees;



BABELFISH MULTI-DB MODE

Many databases in SQL Server, many schemas in PostgreSQL

T-SQL View (port 1433)	PostgreSQL View (port 5432)
Database Structure: **Multiple Databases:** • database1 (with dbo, sales, hr) • database2 (with dbo, sales, hr) • database3 (with dbo, sales, hr) **Example Query:** USE database1; SELECT * FROM dbo.customers; SELECT * FROM sales.orders; USE database2; SELECT * FROM dbo.customers;	**Database Structure:** **Database: babelfish_db (single!)** Schemas (prefixed with DB name): • database1_dbo • database1_sales • database1_hr • database2_dbo • database2_sales • database2_hr **Example Query:** SELECT * FROM database1_dbo.customers; SELECT * FROM database1_sales.orders; SELECT * FROM database2_dbo.customers;

DEMO

Babelfish Compass – Compatibility Assessment



ALL THAT GLITTERS IS GOLD?

Unsupported statements

ALTER DATABASE

ALTER DATABASE SCOPED CONFIGURATION

ALTER DATABASE SCOPED CREDENTIAL

ALTER DATABASE SET HADR

ALTER INDEX

ALTER PARTITION FUNCTION

ALTER PARTITION SCHEME

ALTER SCHEMA

ALTER SERVER CONFIGURATION

ALTER SERVICE, BACKUP/RESTORE SERVICE MASTER KEY clause

BEGIN CONVERSATION TIMER

BEGIN DISTRIBUTED TRANSACTION

BEGIN DIALOG CONVERSATION

BULK INSERT

CREATE COLUMNSTORE INDEX

CREATE EXTERNAL FILE FORMAT

CREATE EXTERNAL TABLE

CREATE, ALTER, DROP APPLICATION ROLE

CREATE, ALTER, DROP ASSEMBLY

CREATE, ALTER, DROP ASYMMETRIC KEY

CREATE, ALTER, DROP CREDENTIAL

CREATE, ALTER, DROP CRYPTOGRAPHIC PROVIDER

CREATE, ALTER, DROP ENDPOINT

CREATE, ALTER, DROP EVENT SESSION

CREATE, ALTER, DROP EXTERNAL LANGUAGE

CREATE, ALTER, DROP EXTERNAL RESOURCE POOL

CREATE, ALTER, DROP FULLTEXT CATALOG

CREATE, ALTER, DROP FULLTEXT INDEX

CREATE, ALTER, DROP FULLTEXT STOPLIST

CREATE, ALTER, DROP MESSAGE TYPE

CREATE, ALTER, DROP, OPEN/CLOSE, BACKUP/RESTORE MASTER KEY

CREATE, ALTER, DROP QUEUE

CREATE, ALTER, DROP RESOURCE GOVERNOR

CREATE, ALTER, DROP RESOURCE POOL

CREATE, ALTER, DROP ROUTE

CREATE, ALTER, DROP SEARCH PROPERTY LIST

CREATE, ALTER, DROP SECURITY POLICY

CREATE, ALTER, DROP SELECTIVE XML INDEX clause

CREATE, ALTER, DROP SERVICE

CREATE, ALTER, DROP SPATIAL INDEX

CREATE, ALTER, DROP TYPE

CREATE, ALTER, DROP XML INDEX

CREATE, ALTER, DROP XML SCHEMA COLLECTION

CREATE/DROP RULE

CREATE, DROP WORKLOAD CLASSIFIER

CREATE, ALTER, DROP WORKLOAD GROUP

ALTER TRIGGER

CREATE TABLE... GRANT clause

CREATE TABLE... IDENTITY clause

CREATE USER

DENY

END, MOVE CONVERSATION

EXECUTE with AS LOGIN or AT option

GET CONVERSATION GROUP

GROUP BY ALL clause

GROUP BY CUBE clause

GROUP BY ROLLUP clause

INSERT... DEFAULT VALUES

MERGE

READTEXT

REVERT

SELECT TOP x PERCENT WHERE x <> 100

SELECT TOP... WITH TIES

SELECT... FOR BROWSE

SELECT... FOR XML AUTO

SELECT... FOR XML EXPLICIT

SELECT... FOR XML PATH

SEND

SET DATEFORMAT

SET DEADLOCK_PRIORITY

SET FMTONLY

SET FORCEPLAN

SET NUMERIC_ROUNDABORT ON

SET OFFSETS

SET REMOTE_PROC_TRANSACTIONS

SET SHOWPLAN_TEXT

SET SHOWPLAN_XML

SET STATISTICS

SET STATISTICS PROFILE

SET STATISTICS TIME

SET STATISTICS XML

SHUTDOWN statement

UPDATE STATISTICS

UPDATETEXT

Using EXECUTE to call a SQL function

VIEW... CHECK OPTION clause

VIEW... VIEW_METADATA clause

WAITFOR DELAY

WAITFOR TIME

WAITFOR, RECEIVE

WITH XMLNAMESPACES construct

WRITETEXT

XPATH expressions

DEMO

What works and what doesn’t



AUTHENTICATION

Different scenarios, different outcomes

- **SQL Server auth:** ok

- **Windows auth:** not implemented.

  No SSPI, no NTLM, no Kerberos

  Usually a hard blocker

- **PostgreSQL auth:** all supported methods are available



TRANSACTION ISOLATION

Looks the same, it's not the same

SQL Server	Babelfish	PostgreSQL	Notes
READ UNCOMMITTED	READ UNCOMMITTED	READ COMMITTED	Upgraded to MVCC
READ COMMITTED	READ COMMITTED	READ COMMITTED	Upgraded to MVCC
READ COMMITTED SNAPSHOT	READ COMMITTED SNAPSHOT	READ COMMITTED	MVCC (not identical to SQL Server)
SNAPSHOT	SNAPSHOT	REPEATABLE READ	Equivalent
REPEATABLE READ	REPEATABLE READ	REPEATABLE READ	SS: pessimistic PG: MVCC
SERIALIZABLE	SERIALIZABLE	SERIALIZABLE	SS: pessimistic PG: MVCC

REPEATABLE READ and SERIALIZABLE must be explicitly enabled:

```
EXECUTE sp_babelfish_configure 'isolation_level_repeatable_read', 'pg_isolation'
EXECUTE sp_babelfish_configure 'isolation_level_serializable', 'pg_isolation'
```

DEMO

The AdventureWorks application



SHOULD I USE IT?

Different scenarios, different outcomes

- **Migration:** I'm in a PostgreSQL shop and I inherit a legacy SQL Server app
- **Licensing:** It's my core application, I need performance, I'm actively developing it but I don't want to pay for licensing
- **Dwarf databases:** It's tiny databases and I don't want to pay for licenses



THE “OPEN-SOURCE TAX”

- Lack of official support
  - Community-centric support – AKA “RTFM”
- Sparse or outdated documentation
- Dependency hell
  - “Works on my machine” syndrome
- DIY Troubleshooting
  - StackOverflow, Reddit, GitHub issues, Source Code
- Mitigated by the cloud
  - All is taken care by pros





PERFORMANCE

Performance tuning

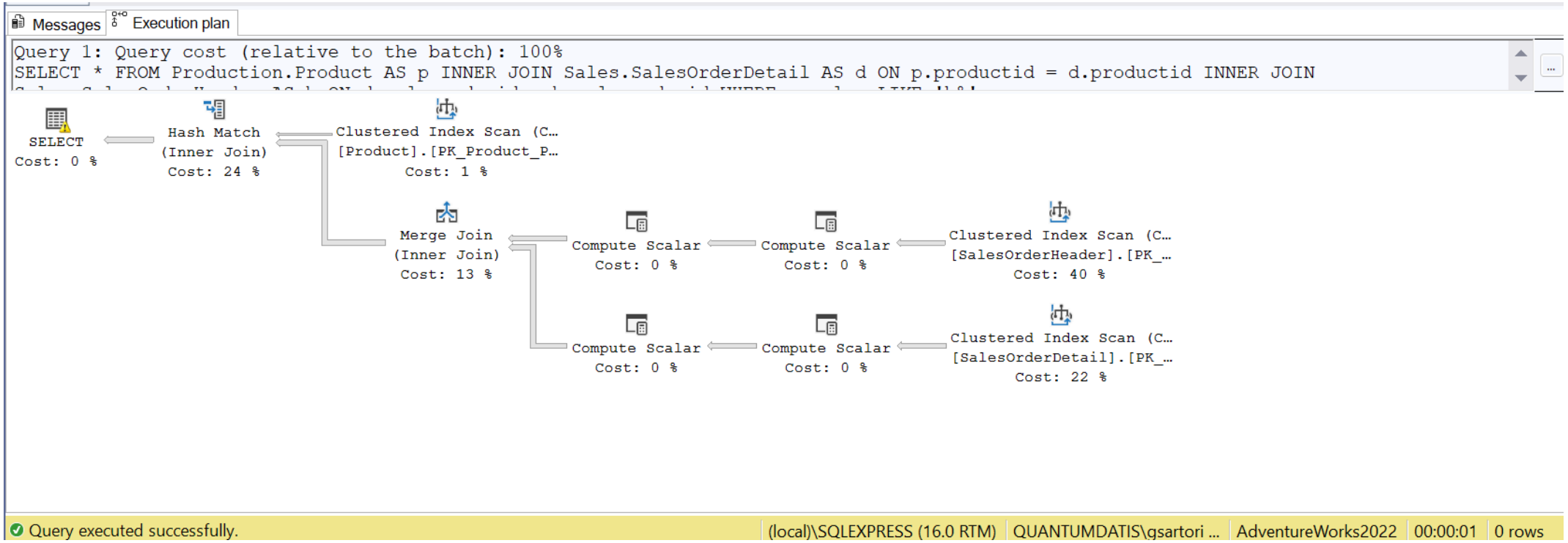
SQL Server superior performance tuning tools

- Rich query execution plans
- Live execution plans
- Query Store
- Extended Events
- Automatic tuning
- Countless DMVs to observe every possible entity



EXECUTION PLANS

Rich, graphical execution plans





EXECUTION PLANS

XML execution plans loaded with information

Execution plan.xml | SQLQuery3.sql...sartori (83))* | SQLQuery2.sql...ks (demo (0))* | SQLQuery1.sql...ks (demo (0))*

```
<?xml version="1.0" encoding="utf-16"?>
<ShowPlanXML xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="
  <BatchSequence>
    <Batch>
      <Statements>
        <StmtSimple StatementCompId="1" StatementEstRows="52540.3" St
          <StatementSetOptions ANSI_NULLS="true" ANSI_PADDING="true"
          <QueryPlan NonParallelPlanReason="NoParallelPlansInDesktopO
            <Warnings>
              <PlanAffectingConvert ConvertIssue="Cardinality Estimat
            </Warnings>
            <MemoryGrantInfo SerialRequiredMemory="1024" SerialDesire
            <OptimizerHardwareDependentProperties EstimatedAvailableM
            <TraceFlags IsCompileTime="true">
              <TraceFlag Value="8017" Scope="Global" />
            </TraceFlags>
            <RelOp AvgRowSize="664" EstimateCPU="0.624811" EstimateIO
              <OutputList>
                <ColumnReference Database="[AdventureWorks2022]" Sche
                <ColumnReference Database="[AdventureWorks2022]" Sche
```



EXECUTION PLANS

Feels like SQL Server 6.5

Grid
Text
Record

```
QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------------------------------------------
Nested Loop  (cost=19.46..408.87 rows=246 width=494) (actual time=1.147..20.208 rows=1218 loops=1)
  Output: p.productid, p.name, p.productnumber, p.makeflag, p.finishedgoodsflag, p.color, p.safetystocklevel, p.reorderpoint, p.standardcost, p.l
  Inner Unique: true
  ->  Hash Join  (cost=19.17..157.17 rows=246 width=267) (actual time=1.095..7.284 rows=1218 loops=1)
        Output: p.productid, p.name, p.productnumber, p.makeflag, p.finishedgoodsflag, p.color, p.safetystocklevel, p.reorderpoint, p.standardcos
        Inner Unique: true
        Hash Cond: (d.productid = p.productid)
        ->  Seq Scan on sales.salesorderdetail d  (cost=0.00..126.28 rows=4428 width=122) (actual time=0.045..1.772 rows=5998 loops=1)
              Output: d.salesorderid, d.salesorderdetailid, d.carriertrackingnumber, d.orderqty, d.productid, d.specialofferid, d.unitprice, d.un
        ->  Hash  (cost=18.82..18.82 rows=28 width=145) (actual time=1.030..1.031 rows=119 loops=1)
              Output: p.productid, p.name, p.productnumber, p.makeflag, p.finishedgoodsflag, p.color, p.safetystocklevel, p.reorderpoint, p.stand
              Buckets: 1024  Batches: 1  Memory Usage: 30kB
              ->  Seq Scan on production.product p  (cost=0.00..18.82 rows=28 width=145) (actual time=0.038..0.959 rows=119 loops=1)
                    Output: p.productid, p.name, p.productnumber, p.makeflag, p.finishedgoodsflag, p.color, p.safetystocklevel, p.reorderpoint, p
                    Filter: (((p.color)::text ~~* 'b%'::text COLLATE sys.bbf_unicode_cp1_cs_as) AND (p.color >= 'b'::sys.nvarchar) AND (p.color <
                    Rows Removed by Filter: 385
  ->  Index Scan using pk_salesorderheader_salesorderid on sales.salesorderheader h  (cost=0.29..1.02 rows=1 width=227) (actual time=0.008..0.008
        Output: h.salesorderid, h.revisionnumber, h.orderdate, h.duedate, h.shipdate, h.status, h.onlineorderflag, h.salesordernumber, h.purchase
        Index Cond: (h.salesorderid = d.salesorderid)
Planning Time: 1.102 ms
Execution Time: 20.584 ms
```

CONCLUSIONS



IS SQL SERVER READY FOR RETIREMENT?

Get off my lawn

- Losing traction? → Maybe
- PostgreSQL stealing the spotlight? → Possible
- SQL Server applications are here to stay
- You will retire before SQL Server does
- Still extremely solid in Enterprise class
- The reports of SQL Server's death are highly exaggerated
- Babelfish is not the threat you may think



FEEDBACK PLEASE!





Thanks!