

Datenbanksystem wechseln?

Oracle/MS SQL $\Rightarrow$ PostgreSQL

Was spricht dafür? Was spricht dagegen?
Womit muss ich rechnen?

GFU Semicolon
<https://www.gfu.net/s3160.html>

Holger Jakobs
<https://plausibolo.de>



Gängige SQL Datenbanksysteme



kommerziell/closed source

Oracle DB

Sehr lange im Markt, große Userbase, aufwendig in der Administration, eingeschränkte ISO-Kompatibilität, sehr gute Parallelität

MS SQL Server

Lange im Markt, Ableger von Sybase, große Userbase, sehr gute Integration ins MS-Universum, eingeschränkte ISO-Kompatibilität, passable Parallelität

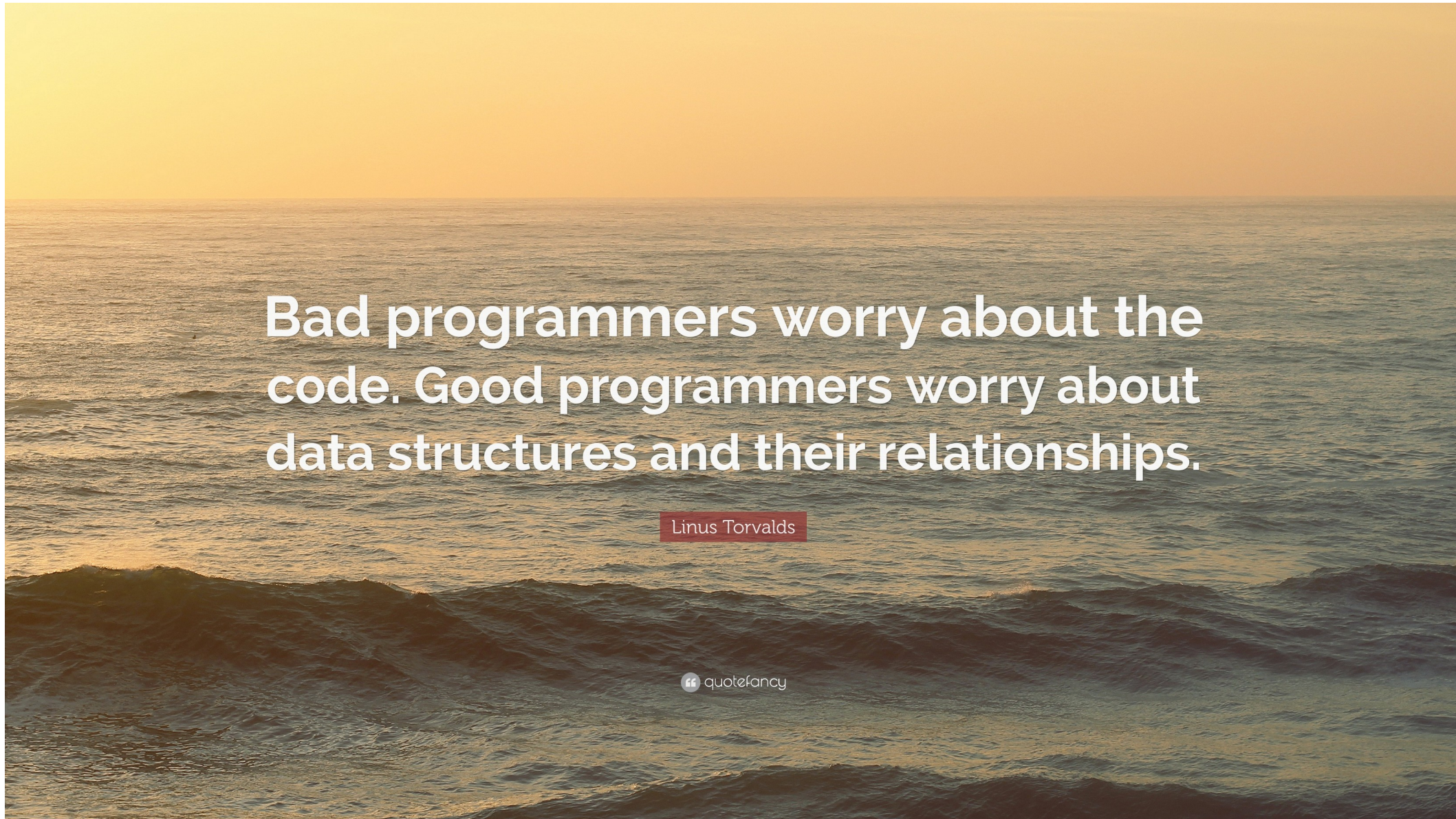
IBM DB2

Sehr lange im Markt, schwindende Userbase, aufwendig in der Administration, gute ISO-Kompatibilität, schwache Parallelität

open source

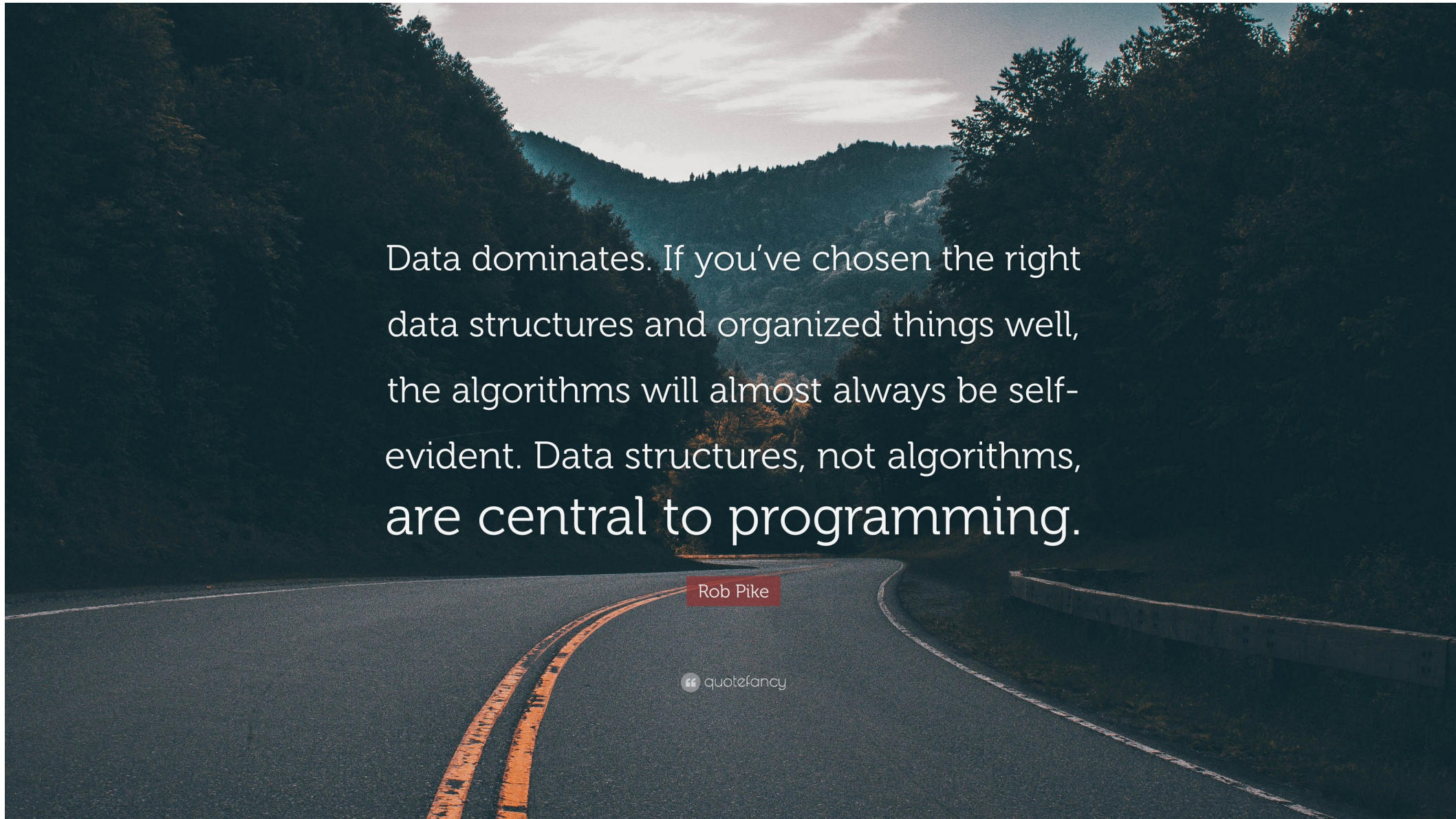
PostgreSQL

Lange im Markt, Open Source, große Userbase, sehr gute ISO-Kompatibilität, sehr gute Parallelität, sehr viele Datentypen

A full-page background image showing a sunset over the ocean. The sky is a warm, golden-orange color, and the water is a deep blue with white-capped waves in the foreground.

Bad programmers worry about the
code. Good programmers worry about
data structures and their relationships.

Linus Torvalds

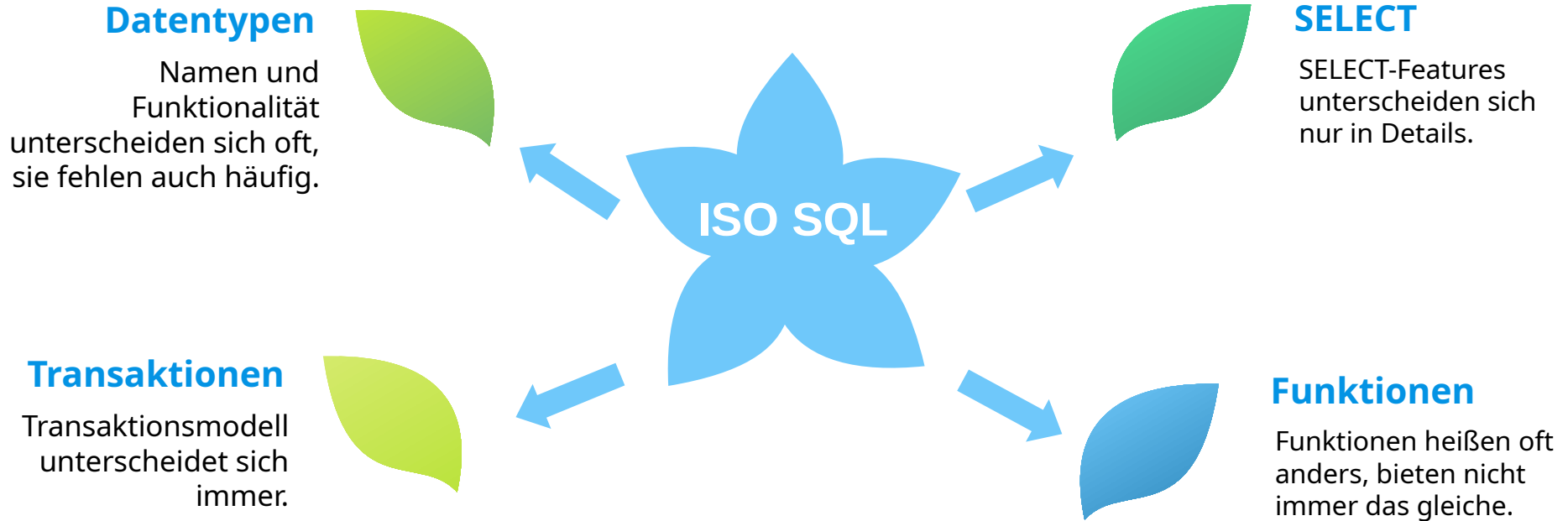
A scenic photograph of a two-lane asphalt road winding through a lush, green valley. The road has a double yellow line in the center and a white line on the right. The valley is filled with dense trees, and in the distance, rolling hills or mountains are visible under a soft, overcast sky. The overall mood is peaceful and contemplative.

Data dominates. If you've chosen the right data structures and organized things well, the algorithms will almost always be self-evident. Data structures, not algorithms, are central to programming.

Rob Pike

“ quote fancy

Wie einheitlich/genormt ist ISO SQL?



Das Hauptproblem

Die meisten DBAs und Entwickler kennen nur ein RDBMS wirklich gut und halten dessen Verhalten daher für „normal“.

Dass die kommerziellen Systeme oft wesentliche Teile des ISO-Standards nicht umsetzen oder Elemente völlig anders benennen, ist den meisten gar nicht bewusst.

Dies kann verglichen werden mit der Denkweise in der Muttersprache, die oft auf Fremdsprachen übertragen wird → Das nennt man Interferenzfehler.

PostgreSQL hat als erklärtes Ziel, den ISO-Standard so weit wie möglich umzusetzen. Darüber hinaus bietet es aber auch sehr viele zusätzliche Features. Solche gibt es bei den anderen RDBMS auch, aber deutlich weniger.

Die Entwickler von PostgreSQL arbeiten selbst täglich damit. Wenn etwas nicht gut erscheint, wird das von ihnen direkt geändert.

Zitat

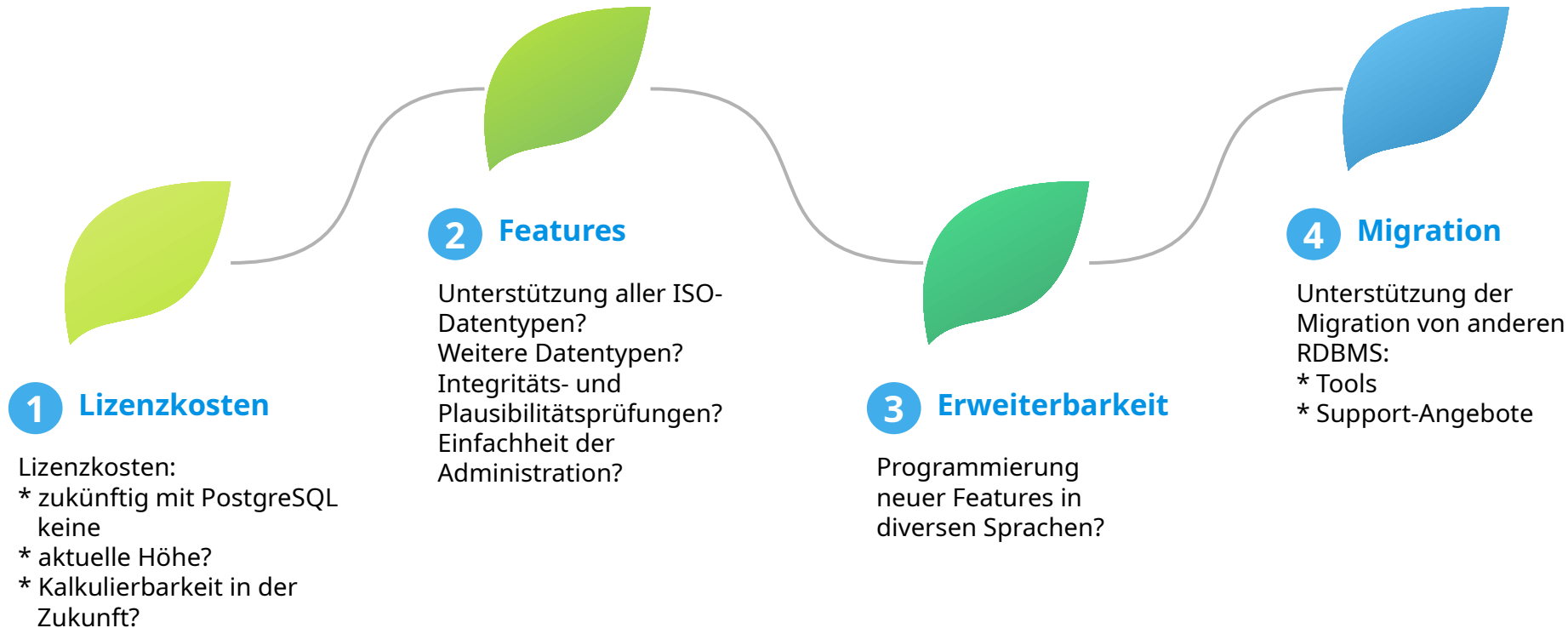
Aus einem Interview mit Yohann Gabory (Python Django Expert)



Question: What do you think of supporting several RDMS solutions in your applications?

Answer: Sorry, but I have to admit that back in the days I believed in such a tale. Now as a grown-up I know two things: Santa and RDBMS agnosticism do not really exist.

Gründe für einen Wechsel zu PostgreSQL



Was bietet PostgreSQL? (1)



Sicherheit

Benutzer- und
Rechte-Verwaltung,
Autorisierungs-
Methoden



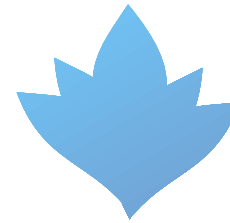
Zeichenketten

Speicherung auch
sehr langer
Zeichenketten,
sehr gute
UNICODE-
Unterstützung



Datentypen

ISO-Typen und
viele weitere,
inet, isbn, ean,
IBAN, Geometrie,
Bereichstypen



Indexe

B-Tree
Hash
GIN
GIST
BRIN



Integrität

Constraints,
Trigger,
Funktionen,
Prozeduren

Was bietet PostgreSQL? (2)



Erweiterbarkeit

Viele Extensions
verfügbar, Erstellen
in C und in Rust
möglich



Programmierung

Prozeduren und
Funktionen in
vielen Sprachen
erstellbar



Treiber

Treiber für quasi
alle Programmier-
Sprachen
verfügbar



Strukturierung

Datenbanken,
Schemas,
Tabellen,
Rollen (Benutzer
und Gruppen)



Transaktionen

MVCC-Verfahren
(wenige Sperren),
Savepoints,
ISO Isolation Levels

Standard-Features

ISO-Eigenschaft	Oracle	MS SQL	PostgreSQL
GENERATED ... AS IDENTITY	ja	nein (ähnlich)	ja
ON UPDATE CASCADE	nein	ja	ja
CURRENT_DATE	ja	GETDATE()	ja
CURRENT-Funktionen in CHECK?	nein	ja	ja
-Operator, Zeichenkettenverkettung	ja	nein (+)	ja
SIMILAR TO (mächtigeres LIKE)	nein	nein	ja
EXTRACT-Funktion	ja	nein	ja
Datumseingabe ohne : und -	nein	ja	ja
Funktion SUBSTRING(zk from 4 for 7)	nein	nein	ja
CHARACTER_LENGTH(), OCTET_LENGTH()	nein	nein	ja
NATURAL- und JOIN USING	ja	nein	ja

Zusätzliche Typen und Eigenschaften

Nicht-ISO-Datentyp / Eigenschaft	Oracle	MS SQL	PostgreSQL
IPv4/IPv6/MAC-Adressen	nein	nein	ja
Bereichstypen + Exclusion Constraints ¹	nein	nein	ja
Aufzähltypen	nein	nein	ja
EAN, ISBN, IBAN (Prüfziffern!)	nein	nein	ja
Integritätsprüfung bei VIEWS ²	nein	nein	ja
RETURNING-Klausel ³	nein	nein	ja
SELECT *, a+b AS summe	nein	ja	ja
per default ISO8601-Datum/Zeit	nein	ja	ja
ISO Information Schema	nein	ja	ja

1 Beispiel: <https://www.postgresql.org/docs/current/rangetypes.html#RANGETYPES-CONSTRAINT>

2 Verhinderung der Löschung von Spalten der Basistabelle und Anpassung bei Umbenennung

3 Beispiel: <https://www.postgresql.org/docs/current/queries-with.html#QUERIES-WITH-MODIFYING>

Stichwort: Datentypen

ISO-Datentyp	Oracle	MS SQL	PostgreSQL
BOOLEAN	fehlt	Ersatz: BIT	vorhanden
Zeichenkettenlänge	max. 32767	Gigabyte	Gigabyte
NUMERIC > 38 Stellen	nein	nein	ja
NULL ungleich " (leere Zeichenkette)	nein	ja	ja
reines Datum	fehlt	vorhanden	vorhanden
reine Uhrzeit	fehlt	vorhanden	vorhanden
INTERVAL-Typ (Zeitdauer)	vorhanden	fehlt	vorhanden
Zeitzone pro Benutzer/Sitzung einstellbar	ja	nein	ja
IANA-Zeitzone	ja	nein	ja
CREATE DOMAIN	nein	nein	ja

Gute Eigenschaften

Nicht-ISO-Eigenschaften	Oracle	MS SQL	PostgreSQL
MVCC, readers don't block writers ...	ja	ja (konfigurierbar)	ja
MVCC in RR und SERIALIZABLE	ja	nein	ja
Deferrable Constraints	ja	nein	ja
sprechend benannte Constraints	nein	ja	ja
UTF-8-Unterstützung	eingeschränkt	fehlerhaft	ja
Schemanamen ohne eigenen User	nein	ja	ja
korrekte Datum/Zeit-Arithmetik	eingeschränkt	eingeschränkt	ja
reguläre Ausdrücke beim Suchen	ja	nein	ja
gnädige Eingabeformate Datum ...	nein	ja	ja
transaktionale DDL-Kommandos	nein	ja	ja

Transaktionsmodelle

Oracle

- Automatisches Beginnen einer TA bei DML. Ohne explizites COMMIT wird nicht geschrieben.
- Fehler in einer TA haben keine Auswirkung.
- SAVEPOINT fritz; ROLLBACK TO fritz;
- DDL-Kommandos führen COMMIT durch. Immer!
- MVCC: Writers don't block Readers. Readers don't block Writers.
- 2 Isolation Levels
- Dirty reads unmöglich.

MS SQL

- Autocommit
Jedes Statement wird sofort geschrieben.
- Fehler in einer TA haben keine Auswirkung.
- SAVE TRAN fritz; ROLLBACK TRAN fritz;
- DDL-Kommandos können Bestandteil einer TA sein.
- Sperren: Writers block Readers. Readers block Writers.
- 4 Isolation Levels
- Dirty reads möglich.

PostgreSQL

- Autocommit
Jedes Statement wird sofort geschrieben.
- Fehler in einer TA → TA tot! COMMIT führt zu ROLLBACK
- SAVEPOINT fritz; ROLLBACK TO fritz;
- DDL-Kommandos können Bestandteil einer TA sein.
- MVCC: Writers don't block Readers. Readers don't block Writers.
- 4 Isolation Levels (mit 3 verschiedenen Verhalten)
- Dirty reads unmöglich.

Diese wesentlichen Unterschiede müssen von den Anwendungen berücksichtigt werden. Evtl. kümmern sich ORMs (teilweise) darum.

Bei der Umstellung

PostgreSQL kann vieles, was die anderen RDBMS nicht bieten. Zum Zeitpunkt der Umstellung wird man dies alles noch nicht nutzen können, weil bislang viel mehr als eigentlich nötig im Anwendungscode gemacht wird.

Daher werden die Vorteile erst einmal kaum auffallen. Lediglich wenn die Entwickler geschult werden und eingearbeitet sind, zeigen sich diese.

Aus Entwicklersicht

PostgreSQL bietet aus Entwicklersicht große Vorteile. Der Umfang von Anwendungscode kann deutlich reduziert werden – abgesehen davon, dass Integritätsprüfungen im Anwendungscode nichts verloren haben.

Denn dann müssten sie in jeder Anwendung erneut implementiert werden und wären zudem bei Verwendung eines Administrations-Frontends nicht wirksam.

Aus Administrationssicht

PostgreSQL stammt aus der UNIX-Welt und fügt sich dort sehr gut ein.

PostgreSQL ist nicht so leicht komplett in eine Windows-Umgebung zu integrieren wie MS SQL Server.

Der Lieferumfang von PostgreSQL ist nicht vergleichbar mit dem kommerzieller Produkte.

Lieferumfang

PostgreSQL ist ein Datenbankserver. Punkt!
Mehr nicht!

- kein grafisches Frontend
- kein Scheduler
- kein automatisches Backup
- keine fertige Replikationslösung
- kein High Availability „out of the box“

PostgreSQL ist ein Datenbankserver.
Mehr nicht!

- kein grafisches Frontend
- kein Scheduler
- kein automatisches Backup
- keine fertige Replikationslösung
- kein High Availability „out of the box“

Lieferumfang

All das gibt es natürlich schon, aber von separaten Projekten:

GUI: pgAdmin, DBeaver

Scheduler: pg_cron, pg_timetable, pg_agent

Backup: Barman, pgProbackup, pgBackrest, WAL-G

HA: Patroni, pgAutoFailover, repmgr

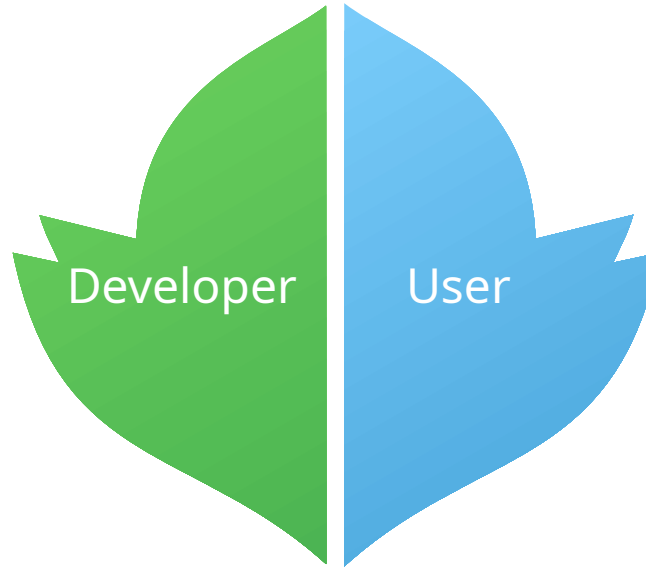
Aber: Die Auswahl und Integration sind zusätzlicher Aufwand!

In welcher Situation sind Sie?

Software entwickelndes Unternehmen

Umstellung bestehender Software auf PostgreSQL ist möglich, evtl. ist manches sogar leichter einzurichten.

Entwicklung neuer Software wird sehr erleichtert durch viele Features.



Software verwendendes Unternehmen

Austausch des Datenbanksystems ohne Unterstützung des Software-Lieferanten ist so gut wie unmöglich.

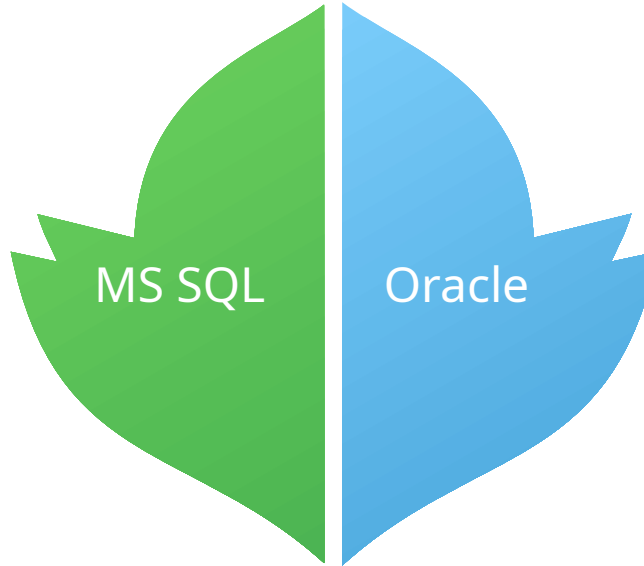
Worauf muss man bei PostgreSQL verzichten?

Variablen und Ablaufstrukturen in SQL

pl/pgsql verwenden, ggf. in
einem anonymen Block.

Verwendung mehrerer Datenbanken eines Servers in einem Kommando

Geht nicht, da Datenbanken
gegeneinander abgegrenzt
sind (Mandantenfähigkeit).



Autonome Transaktionen

Ersatz:
Separate Verbindung oder
Schreiben in Datei über
eine Funktion mit
"security definer"

Migrationstools (Auswahl)



pgLoader

von MySQL, SQLite oder
MS SQL Server



Pg Chameleon

PG Chameleon

von MySQL



ora2pg

von Oracle DB



NMIG

von MySQL, verwendet
node.js



sqlserver2pgsql

von MS SQL Server



Migrator

von Oracle DB

Amazon bietet das AWS Schema Conversion Tool an.

Konvertierung MS SQL → PostgreSQL

Beispiel: Tabellenerzeugung

```
CREATE TABLE [Mgmt].[LocalDraft](
    [ItemNumber] [bigint] NOT NULL,
    [ItemUnit] [nvarchar](3) NOT NULL,
    [ValidFrom] [date] NOT NULL,
    [ValidTo] [date] NOT NULL,
    [Price] [decimal](9, 2) NOT NULL,
    [NoStatistics] [bit] BOOLEAN NOT NULL,
    [PublishedPrice] [decimal](9, 2) NOT NULL,
    [RowVersion] [nvarchar](20) NULL,
    [LastModified] [datetime2] TIMESTAMP(3) NOT NULL,
    [ModifiedBy] [nvarchar](50) NOT NULL,
    CONSTRAINT [PK_LocalDraft] PRIMARY KEY CLUSTERED
(
    [ItemNumber] ASC,
    [ItemUnit] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
GO
;
```

Alles in Orange muss weg,
das Grüne wird hinzugefügt.

DBMS-Umstellungsprozess

- 1) Datenstrukturen ex- und importieren.
Anpassungen wie gerade gezeigt sind nötig.
Passende Datentypen wählen.
- 2) Anwendung auf neues System anpassen. Das ist
Entwicklungs- und Test-Aufwand.
- 3) Altes System vom Netz nehmen.
- 4) Daten ex- und importieren, evtl. unter Einsatz
von SQL/MED.
- 5) Neues System ans Netz nehmen.

Fazit

Der Wechsel eines RDBMS will gut überlegt sein. SQL ist noch lange nicht SQL.

Als anwendendes Unternehmen geht es nicht ohne Unterstützung durch den Software-Anbieter der Anwendungsprogramme.

Als entwickelndes Unternehmen können Sie zukünftig von den PostgreSQL-Features profitieren und den Entwicklungsaufwand merklich senken. Trotzdem darf der Umstellungsaufwand nicht unterschätzt werden. Die Vorteile stellen sich erst im Laufe der Zeit ein – und nur, wenn die Entwickler sich auch mental auf PostgreSQL umstellen.

Die Umstellung von Anwendungen, die mit den Features von PostgreSQL entwickelt wurden, auf ein anderes RDBMS wäre eine wesentlich größere Herausforderung und würde einen sehr viel größeren Entwicklungsaufwand nach sich ziehen.



Vielen Dank für Ihr Interesse an PostgreSQL

The world's most advanced ~~open source~~
relational database system