die Algorithmen hinter XING, Facebook und den Navigationsgeräten

mit dem SQL Server nachgebaut

Torsten Ahlemeyer, arelium GmbH
Projektleiter, IT-Berater

Microsoft Partner
Gold Data Analytics
Silver Application Development
Silver Cloud Platform

Über mich

Torsten Ahlemeyer
arelium GmbH
40764 Langenfeld
torsten.ahlemeyer@arelium.de
@KurzarmSQLer
www.arelium.de

Dipl. Wirtschaftsinformatiker
IT-Berater, Projektleiter
>16 Jahre Berufserfahrung mit dem SQL Server

Microsoft Partner
Gold Data Analytics
Silver Application Development
Silver Cloud Platform

Prominentes Thema…

- ~13 Millionen monatlich aktive Nutzer (Deutschland)
- >1.500 Mitarbeiter
- 40% Steigerung des B2B E-Recruiting (2019)
- EBITDA steigt auf 75,2 Mio€ (+23%) (2018 auf 2019)

XING

- ~384 Millionen monatlich aktive Nutzer (Europa)
- >35.000 Mitarbeiter
- Europaumsatz $3,6 Mrd. (nur in Q1/2019)
- Börsenwert $512 Mrd. (Mai 2018)

- Umsatz $766 Millionen (Q1/2019)
- davon $127 Millionen (Q1/2019) mit PKW-Navigationsgeräten / Navi-Software

GARMIN

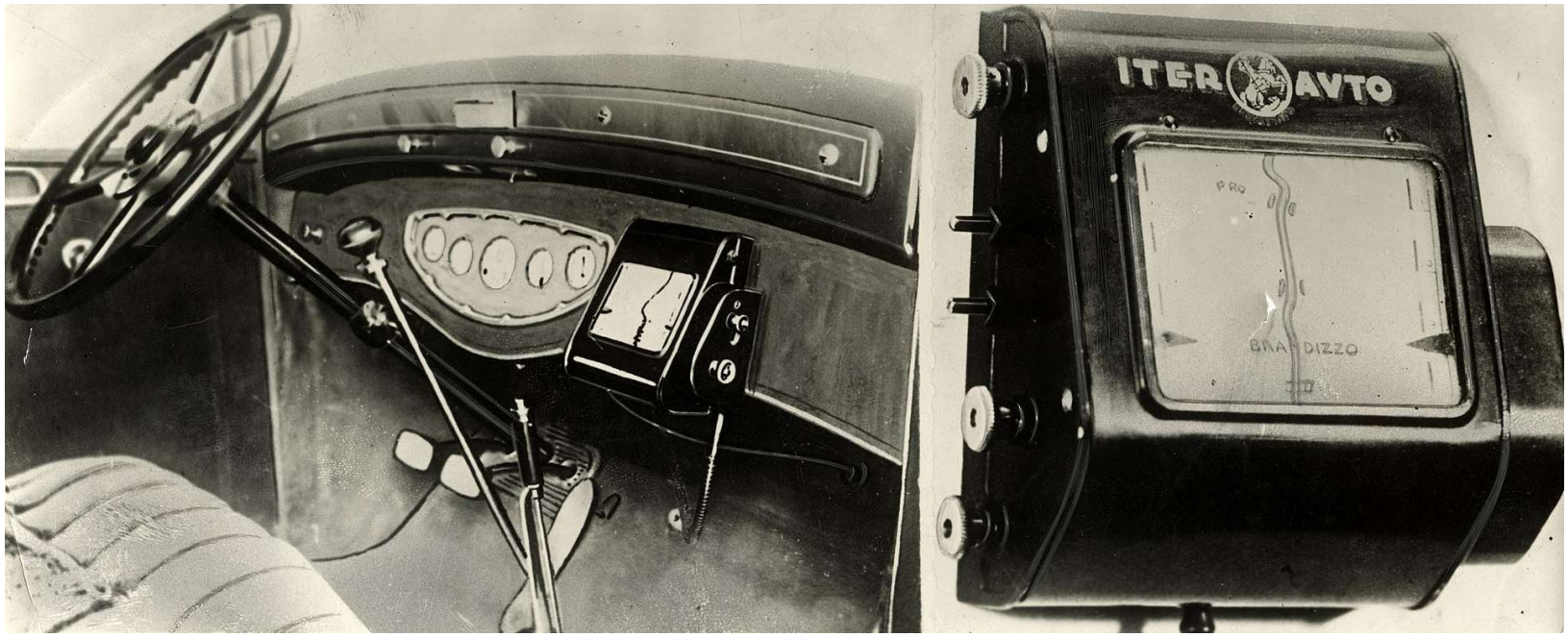
Was verkaufen diese Unternehmen?

- Eine riesige Sammlung von Daten, mit denen sie sich beschäftigen
  (bspw. Nutzerdaten)
- Ihr Wissen um die Beziehungen der einzelnen Instanzen dieser Entitäten untereinander *(bspw. Bekanntenkreis, Verkehrsführung)*

- **Die Möglichkeit diese Daten effizient zu durchsuchen und zu filtern, wobei diese Operationen nicht nur auf die Daten sondern auch ihre Beziehungen untereinander angewandt werden können!**

- **Eine grafische Aufbereitung der nutzerbezogenen Daten**
  (bspw. Timeline, Kontakte, Fahrroute …)

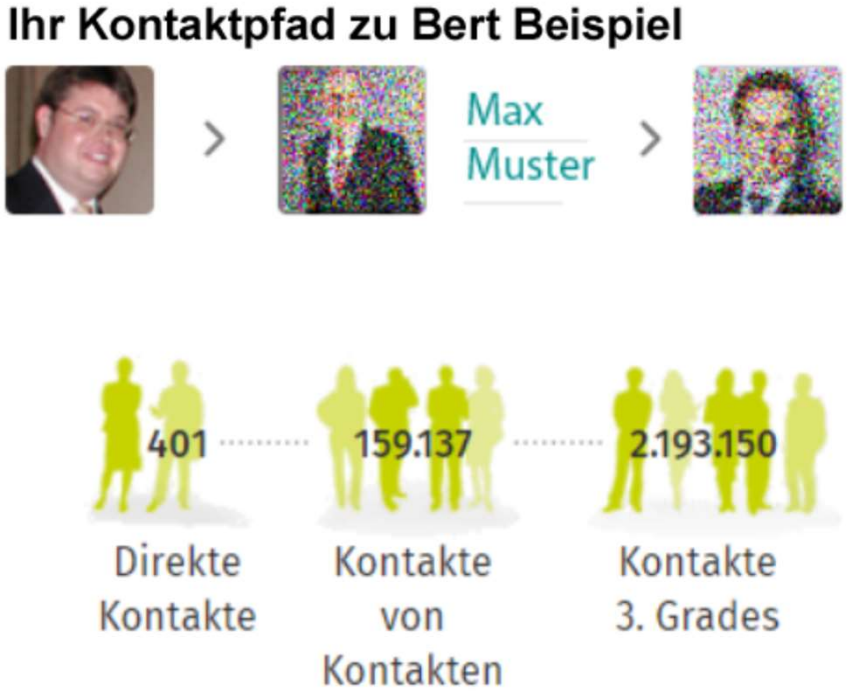
Vorgänger moderner Navigationsgeräte



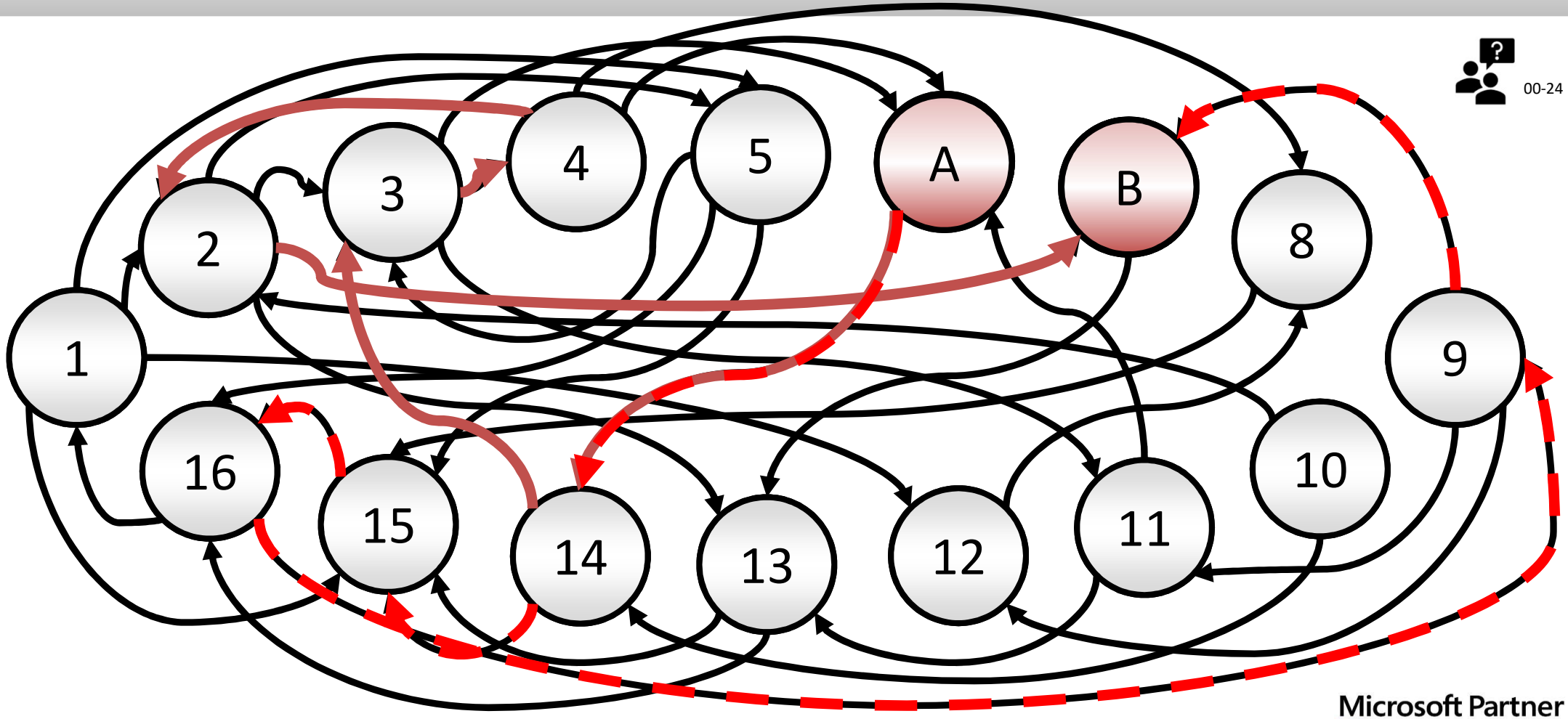
In Italien wurde 1932 das ***Iter Avto*** eingeführt. Es konnte eine **vorher festgelegte** Strecke in Abhängigkeit von der Fahrtgeschwindigkeit von einer Papierrolle abspulen und in einem Sichtfenster anzeigen.

Navigation in großen Datenmengen

- Selbst einfachste Aufgaben, wie das finden des „kürzesten Weges“ von A nach B sind bei entsprechenden Datenmengen eine echte Herausforderungen an Algorithmus und Infrastruktur.
- Die Art der Daten und vor allem ihrer Verknüpfungen sorgt für exponentiales Komplexitätswachstum.



Beispiel „Pfadfinder“



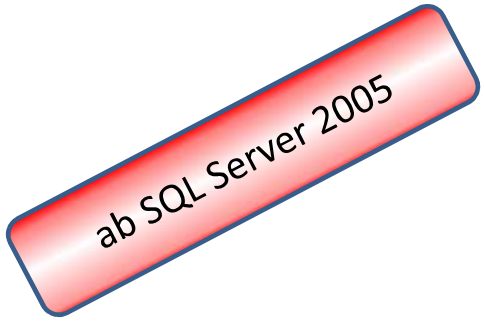
00-24

Nachteile relationaler Ansätze

- Änderung des Schemas sehr aufwändig (ALTER TABLE …)
- keine Unterstützung für strukturierte Daten (JSON ab 2016 / XML ab 2008)
- keine einfache Möglichkeit der Datensatz-Versionierung
- Darstellung von n:m-Beziehungen nur über Mappingtabellen
- Verknüpfung von Tabellen über rechenintensive JOINs
- keine rekursiven JOINs möglich
- adjazente (fachlich benachbarte, direkt verbundene) Knoten verteilen sich irgendwo im Datenmeer
- hoher Bedarf an Indizierung

Umstieg auf Window-Functions

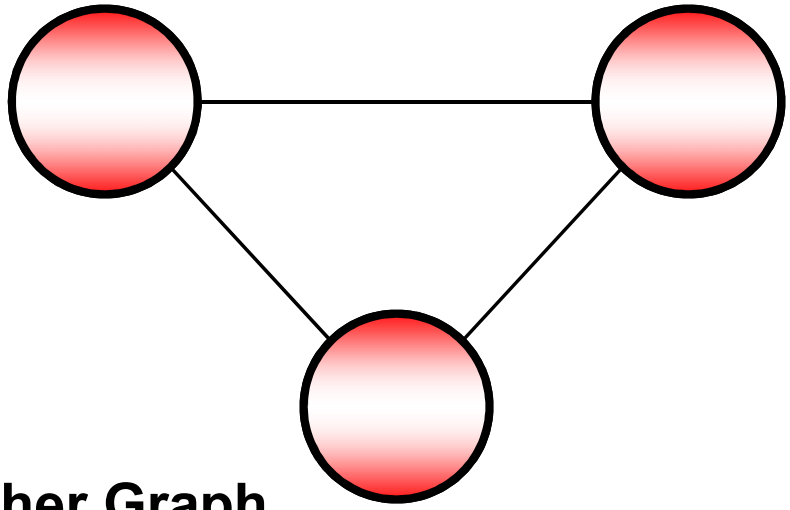
- stetig durch Microsoft mit weiteren Funktionen ausgebaut
- Unterstützung für neue Konzepte wie „Reihenfolgen“, „Vorher/Nachher“ und die Kombination eines Detailwertes mit Aggregaten in einer Ergebniszeile (bspw. „running total“)
- Schlüsselwort OVER (… PARTITION BY…)
- einfache Nutzung der Idee der Rekursion



Die Idee der Graphendatenbank

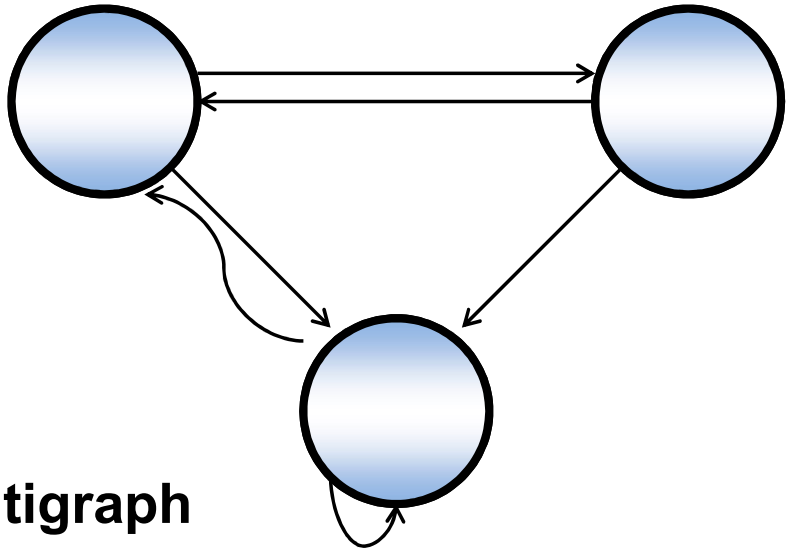
- man nehme eine (relationale) Datenbank…
- … füge eine abstrakte Struktur hinzu, die Objekte („Knoten“) und ihre Verbindungen („Kanten“) untereinander repräsentiert
- … und ergänze einige spezialisierte, hoch effiziente Algorithmen, die bspw.
  - Muster finden („Graph Pattern“)
  - Graphen traversieren („finde alle (in-)direkten Nachbarn eines gegebenen Knotens)
  - kürzeste Pfade zwischen zwei Knoten ermitteln
  - Hotspots (besonders stark vernetzte Regionen) finden
  - bekannte Graphenstrukturen, wie bspw. „Cliquen“ (Graphenteilmenge, bei der jedes Knotenpaar durch eine Kante verbunden ist), entdecken

Graphentypen



Einfacher Graph

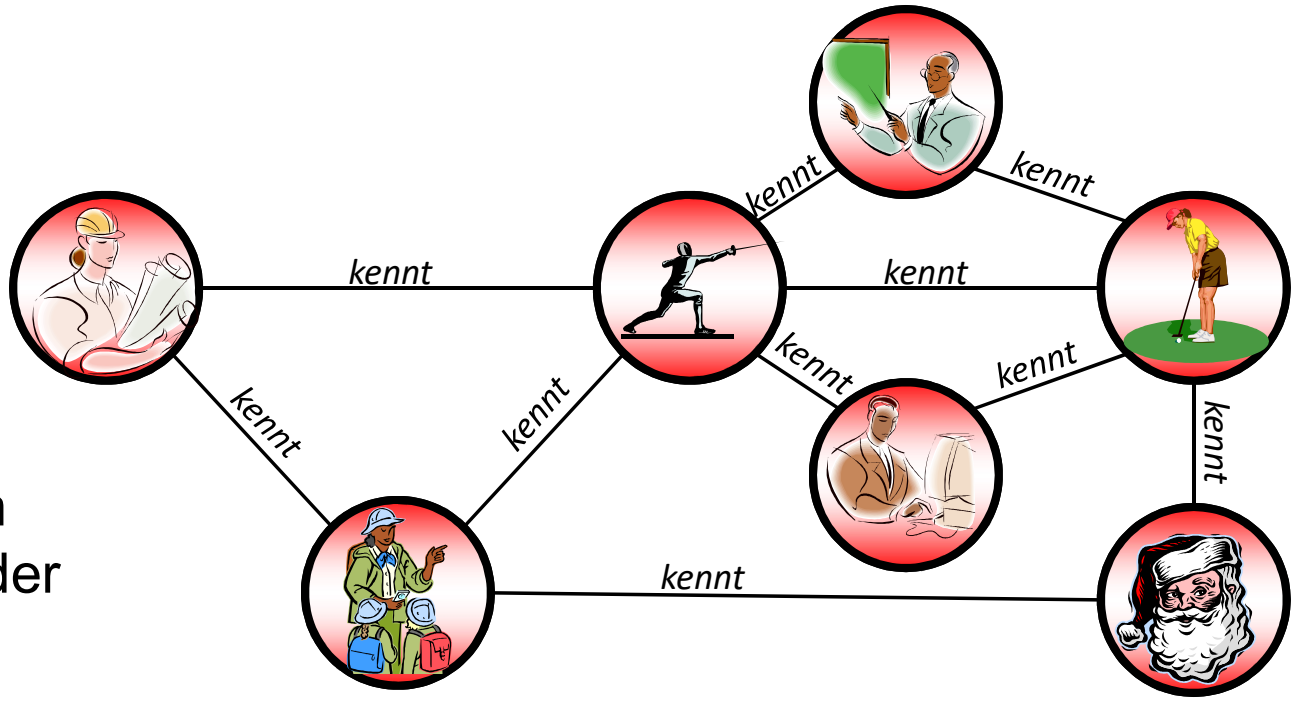
Ein Knotenpaar ist stets durch maximal eine Kante miteinander verbunden. Die Verbindungen („Kanten“) gelten für beide Richtungen. Selbstbezüge sind nicht erlaubt



Multigraph

Ein Knotenpaar darf auch durch mehrere Kanten miteinander verbunden sein. Diese können gerichtet sein, also nur in eine Richtung gelten. Selbstbezüge sind explizit erlaubt.

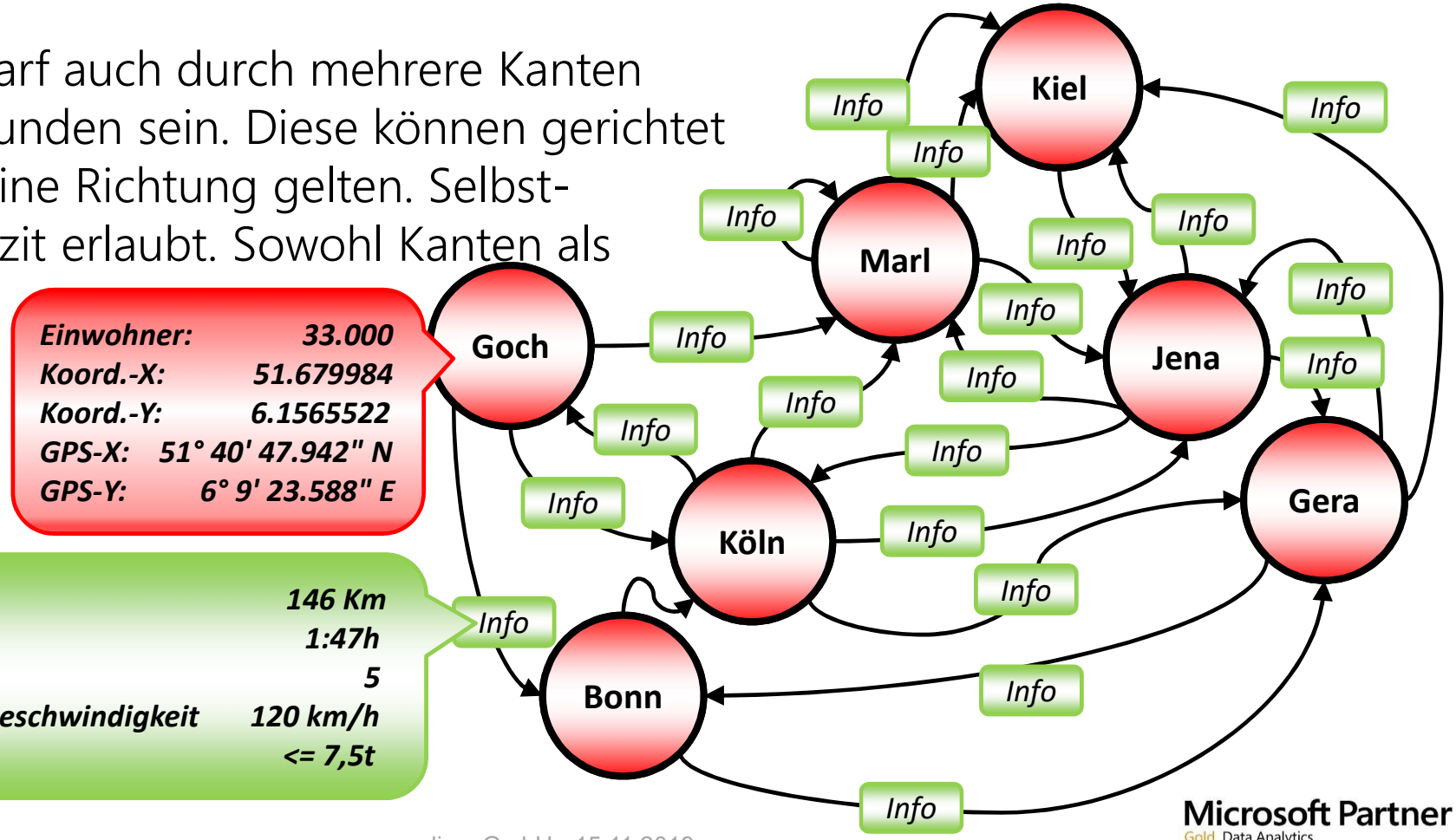
Anwendungsbeispiel „einfacher Graph“



Ein Knotenpaar ist stets durch maximal eine Kante miteinander verbunden. Die Verbindungen („Kanten“) gelten für beide Richtungen. Selbstbezüge sind nicht erlaubt.

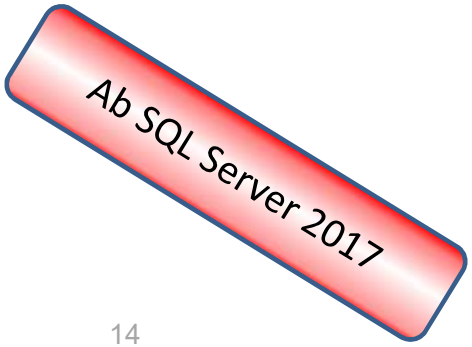
Anwendungsbeispiel „Multigraph“

Ein Knotenpaar darf auch durch mehrere Kanten miteinander verbunden sein. Diese können gerichtet sein, also nur in eine Richtung gelten. Selbstbezüge sind explizit erlaubt. Sowohl Kanten als auch Knoten tragen Informationsattribute.



Vorteile Ansätze Graphentabellen

- Beziehungen über Kanten sind direkt am Knoten modellierbar, kein zusätzliches Mapping erforderlich
- keine Indizes für Relationen erforderlich
- adjazente Knoten können nah beieinander persistiert werden
- Performance hängt nicht (kaum) von der Größe des Graphen ab
- Eingebaute Unterstützung von State-of-the-art-Schnittstellen (bspw. JSON)
- konzipiert für „Verteilte Systeme“ und den Einsatz in der Cloud



Cypher-Language

Cypher ist eine deklarative Abfragesprache für Graphen, die expressive und effiziente Abfragen und Aktualisierungen auf sogenannten Property-Graphen ermöglicht. Cypher ist eine relativ einfache und dennoch mächtige Sprache.

[Wikipedia]

```
MATCH (nicole:Actor {name: 'Nicole Kidman'})-[:ACTED_IN]->(movie:Movie)
WHERE movie.year < $yearParameter
RETURN movie
```

Cypher in T-SQL

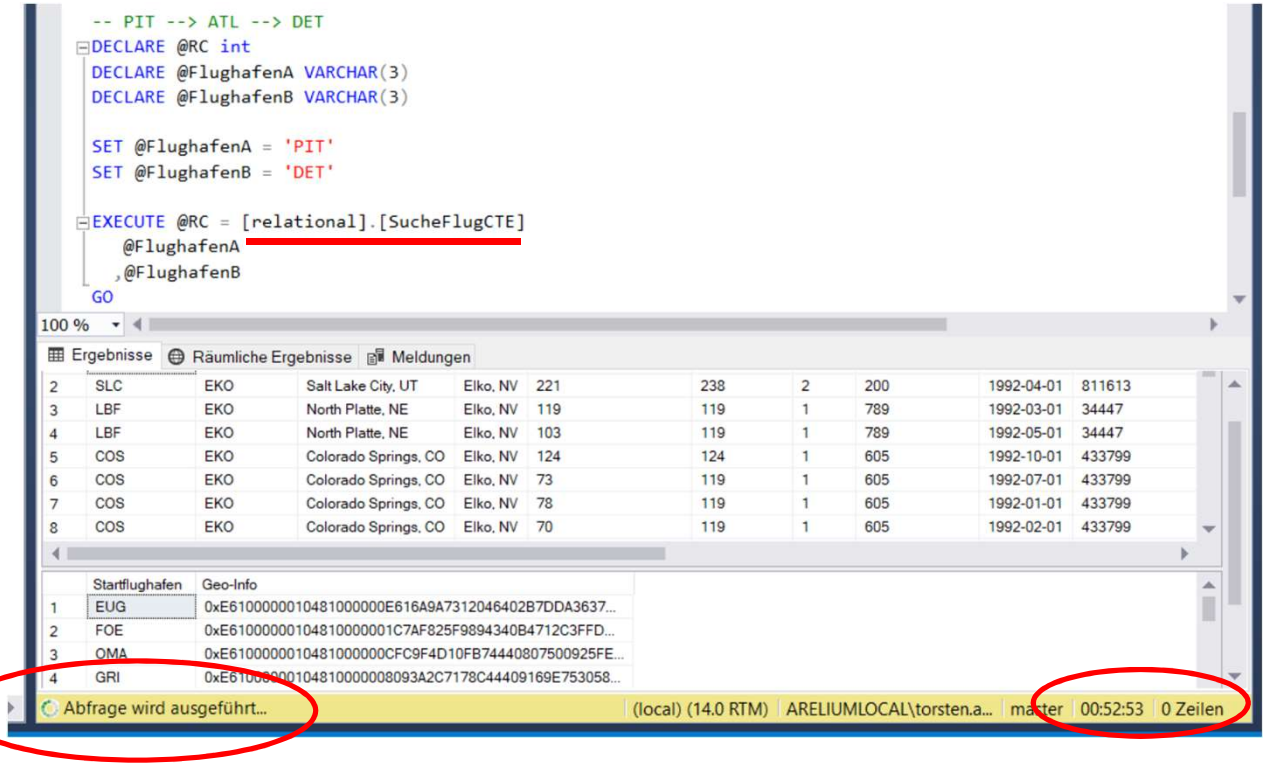
Microsoft hat Teile von Cypher in T-SQL integriert. Dies erlaubt leicht zu lernende und einfach zu debuggende Statements:

```
SELECT
    [p].[Nachname]
  , [p].[Vorname]
  , [r].[Name]
FROM [Person] AS [p], [mag], [Restaurant] AS [r]
WHERE MATCH [p] - [mag] -> [r]
```

keine JOINS

Der angedeutete Pfeil gibt die Beziehungsrichtung an

relationale CTE → lange Laufzeit



```
-- PIT --> ATL --> DET
DECLARE @RC int
DECLARE @FlughafenA VARCHAR(3)
DECLARE @FlughafenB VARCHAR(3)

SET @FlughafenA = 'PIT'
SET @FlughafenB = 'DET'

EXECUTE @RC = [relational].[SucheFlugCTE]
   @FlughafenA
  ,@FlughafenB
GO
```

100 %

Ergebnisse | Räumliche Ergebnisse | Meldungen

2	SLC	EKO	Salt Lake City, UT	Elko, NV	221	238	2	200	1992-04-01	811613
3	LBF	EKO	North Platte, NE	Elko, NV	119	119	1	789	1992-03-01	34447
4	LBF	EKO	North Platte, NE	Elko, NV	103	119	1	789	1992-05-01	34447
5	COS	EKO	Colorado Springs, CO	Elko, NV	124	124	1	605	1992-10-01	433799
6	COS	EKO	Colorado Springs, CO	Elko, NV	73	119	1	605	1992-07-01	433799
7	COS	EKO	Colorado Springs, CO	Elko, NV	78	119	1	605	1992-01-01	433799
8	COS	EKO	Colorado Springs, CO	Elko, NV	70	119	1	605	1992-02-01	433799

	Startflughafen	Geo-Info
1	EUG	0xE6100000010481000000E616A9A7312046402B7DDA3637...
2	FOE	0xE6100000010481000000C7AF825F9894340B4712C3FFD...
3	OMA	0xE6100000010481000000CFC9F4D10FB74440807500925FE...
4	GRI	0xE6100000010481000000[illegible]8093A2C7178C44409169E753058...

Abfrage wird ausgeführt... | (local) (14.0 RTM) | ARELIUMLOCAL\torsten.a... | master | 00:52:53 | 0 Zeilen

55-59

Schon die Zahl der Datensätze führt im konventionellen Ansatz zu langen Laufzeiten, obwohl es mehrere (auch relativ triviale) Lösungen gibt.

Graphen - für wen und wann?

Ideal für die Anwendungsfälle:

- stark vernetzte, umfangreiche Daten
- viele m:n-Beziehungen
- stetiges Wachstum der Datenmenge
- strukturierter Datenaufbau

besonderer Vorteil:

- rekursive Verknüpfungen
- hoch optimierte Abfragen, schnelle Antwortzeiten

Graphen – nicht geeignet bei…

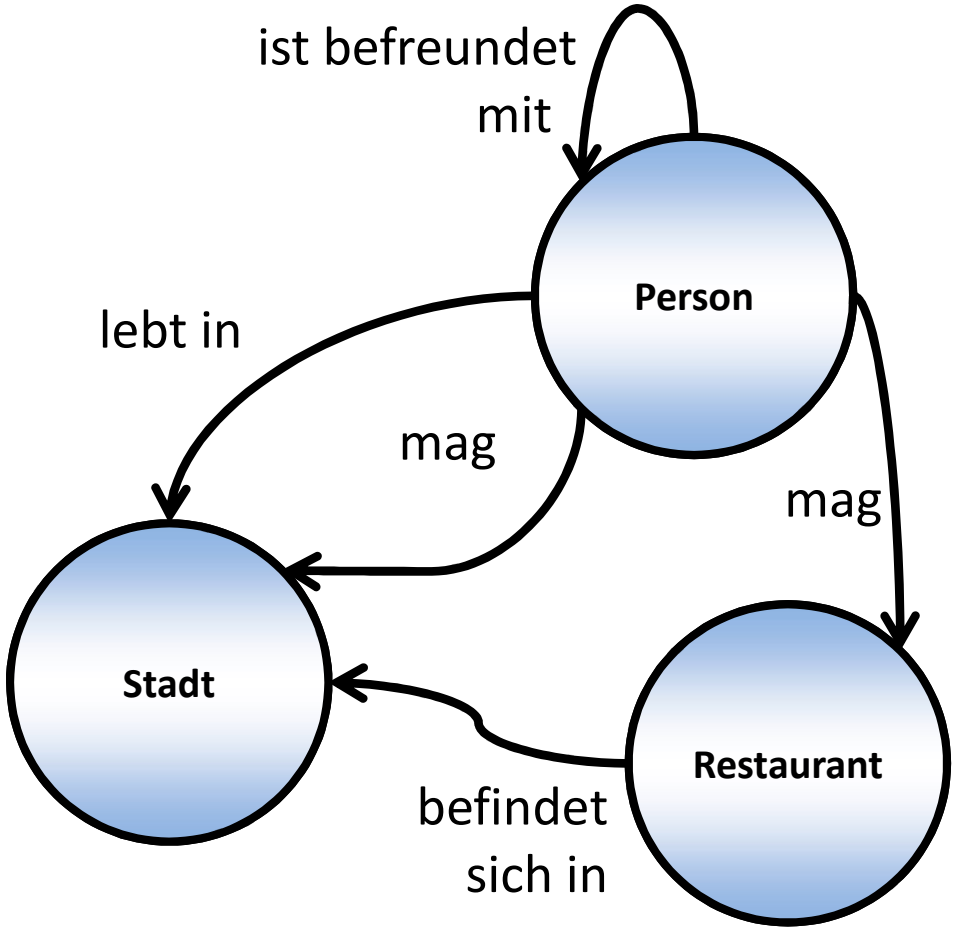
- **Fragestellungen nach allen Entitäten mit spezifischen Attributen**

hier ist dann der traditionelle, relationale Ansatz wahrscheinlich schneller und effektiver

Einschränkungen Graph-DB

- nur 1 Diagramm pro Datenbank
- keine temporären Knoten- oder Kantentabellen (weder lokal noch global) möglich
- datenbankübergreifende Abfragen werden nicht unterstützt
- keine Tabellenwertvariablen für Knoten- oder Kantentabellen realisierbar
- kein UPDATE auf „$from_id“- und „$to_id“-Kantentabellenzeilen
  (statt dessen Löschung und Neuanlage)

Neueste Möglichkeit ;-)



der Befehl **SHORTEST_PATH** vereinfacht die Programmierung nochmal.

- er gibt eine kürzeste Verbindung zwischen zwei gegebenen Knoten (auch wenn mehrere kürzeste existieren!) zurück
- die Rekursion muss nicht mehr selber formuliert werden
- nur innerhalb der MATCH-Klausel zulässig

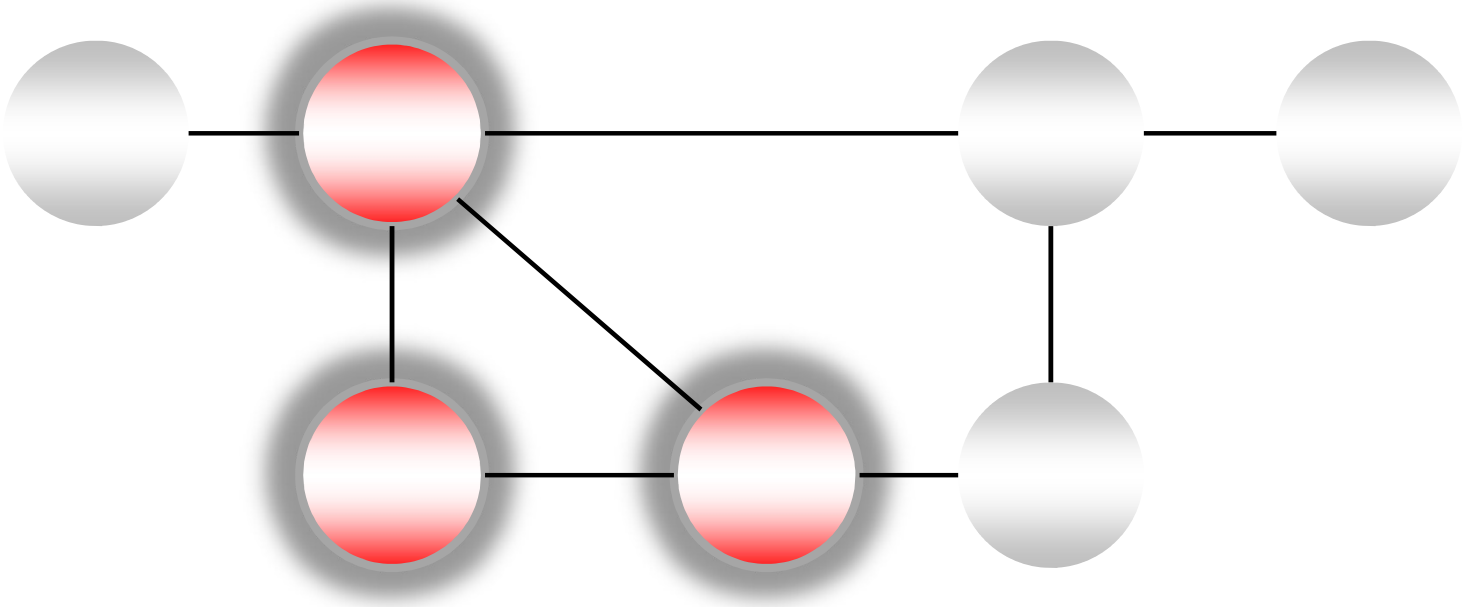
erst ab SQL Server 2019

Vielen Dank!

Ich freue mich auf Feedback!

„Clique“

- Teilmenge eines Graphen
- jedes Knotenpaar der Teilmenge ist durch genau eine Kante miteinander verbunden



Quellennachweise

- https://allfacebook.de/allgemeines/richtlinien-fur-die-nutzung-des-facebook-logos-und-anderen-warenzeichen
- https://en.facebookbrand.com/#brand-guidelines-assets
- https://corporate.xing.com/de/unternehmen/daten-und-fakten/
- https://allfacebook.de/toll/state-of-facebook
- https://www.navigation-professionell.de/garmin-quartalszahlen-q1-2019/
- Von Nationaal Archief - https://www.flickr.com/photos/nationaalarchief/4192749543/, No restrictions, https://commons.wikimedia.org/w/index.php?curid=53555690
- https://www.stern.de/digital/online/facebook-will-mit-diesen-gruseligen-methoden-herausfinden--ob-menschen-sich-kennen-7837602.html
- https://pixabay.com/de/photos/birne-strom-energie-glas-lampe-546859/
- https://pixabay.com/de/illustrations/fragezeichen-frage-antwort-1019820/