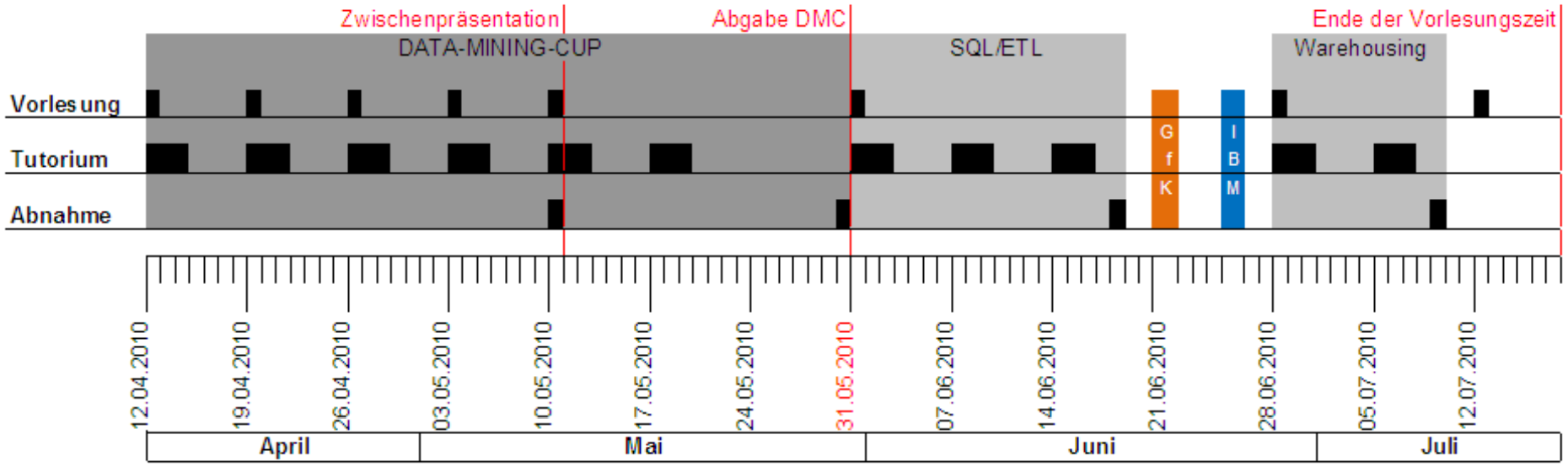
Data Warehousing (I): SQL/ETL

Praktikum: Data Warehousing und Data Mining

Weitere Termine



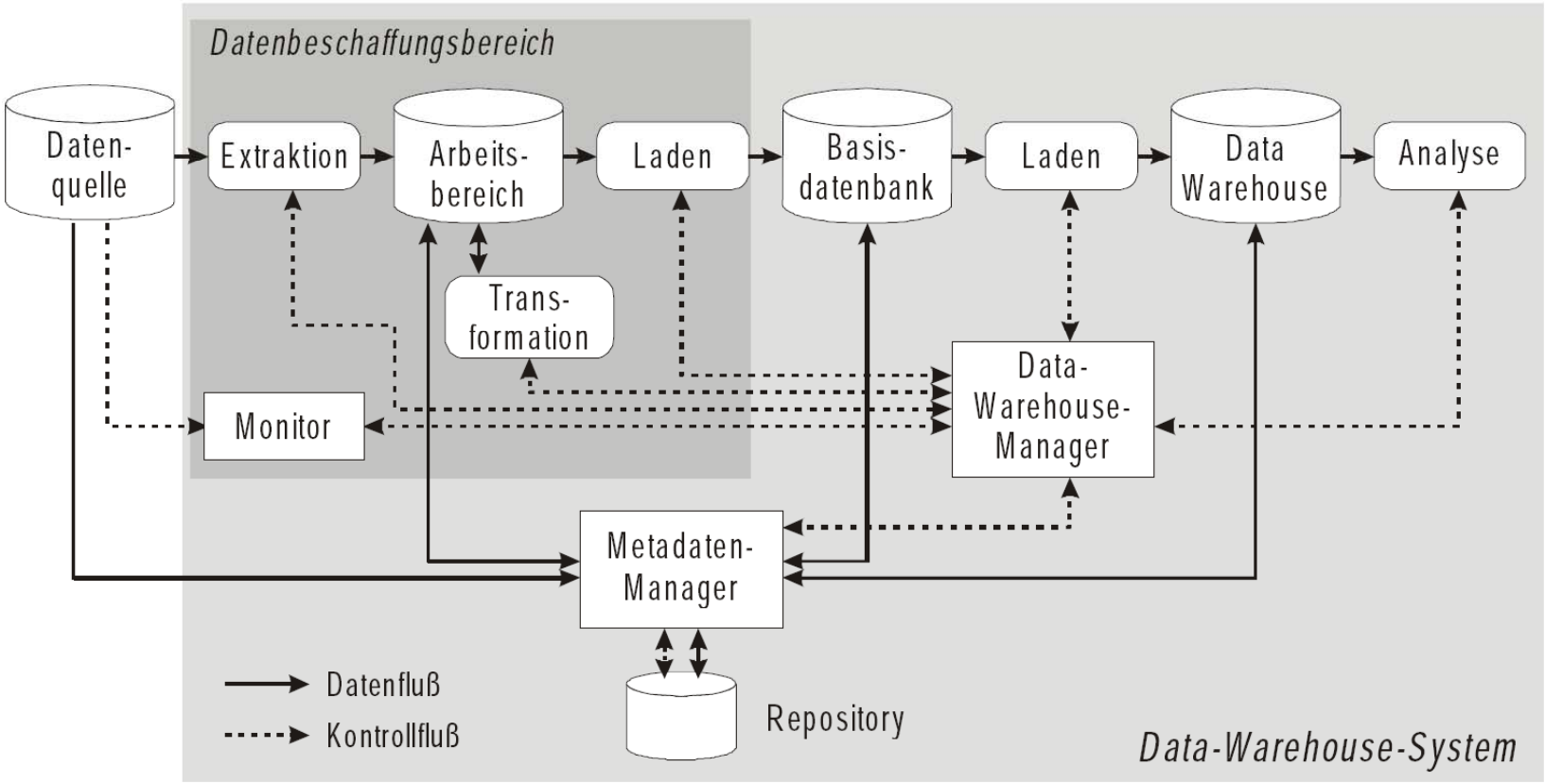
Agenda

- Einführung Data Warehouses
  - Referenzarchitektur
  - Datenbeschaffung
- Online Transactional Processing (OLTP)
  - Datenmanipulation mit SQL
  - Anfragen mit SQL
- Nächste Aufgabe
  - Analyse von Graphen
  - Projekt LogoTakt
  - SQL, ETL, Visualisierung

Eigenschaften eines Data Warehouse

- Integrierte Sicht auf beliebige Daten
  - …aus verschiedenen Datenbanken
  - …Integration von Schemata und Daten aus Quellen
- Analyseaspekt
  - …multidimensionales Datenmodell
  - …Online Analytical Processing (OLAP)
- Stabile Datenbasis
  - Eingebrachte Daten werden nicht mehr modifiziert
  - Neue Daten können aufgenommen werden
- Data-Warehouse-System
  - Komponenten zur Integration und Analyse + Data Warehouse

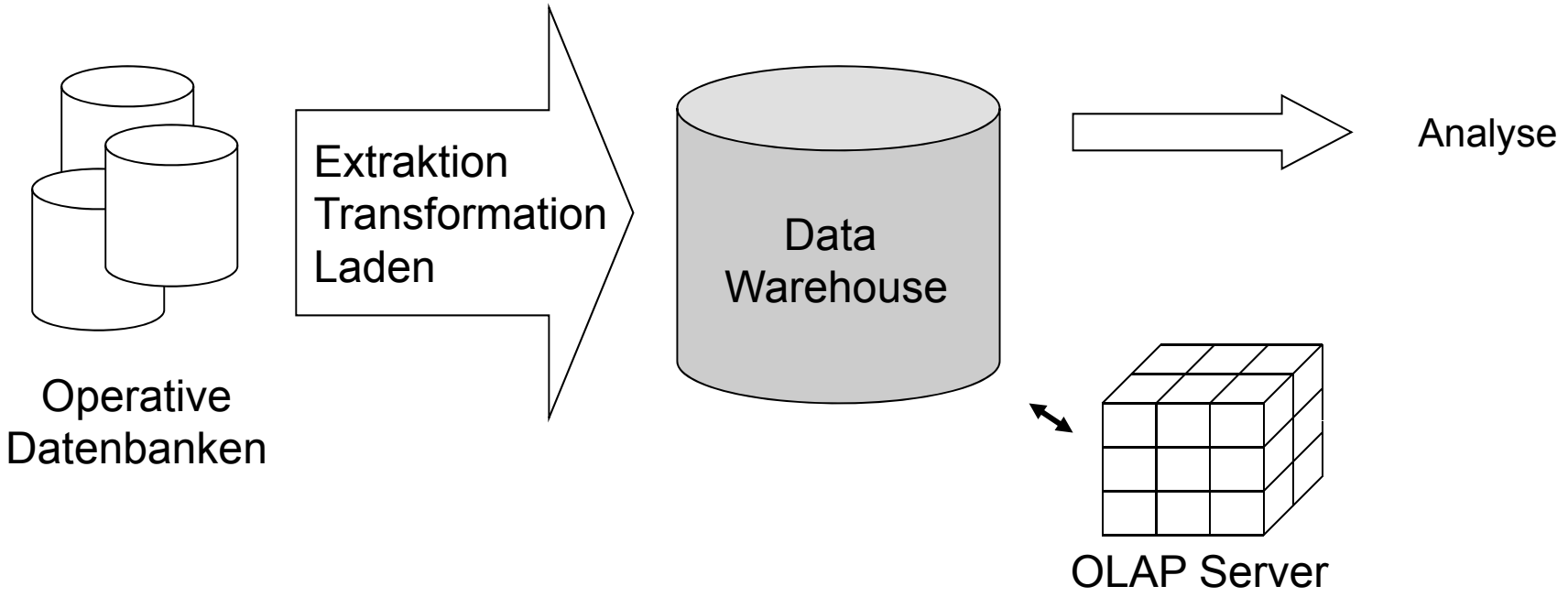
Referenzarchitektur



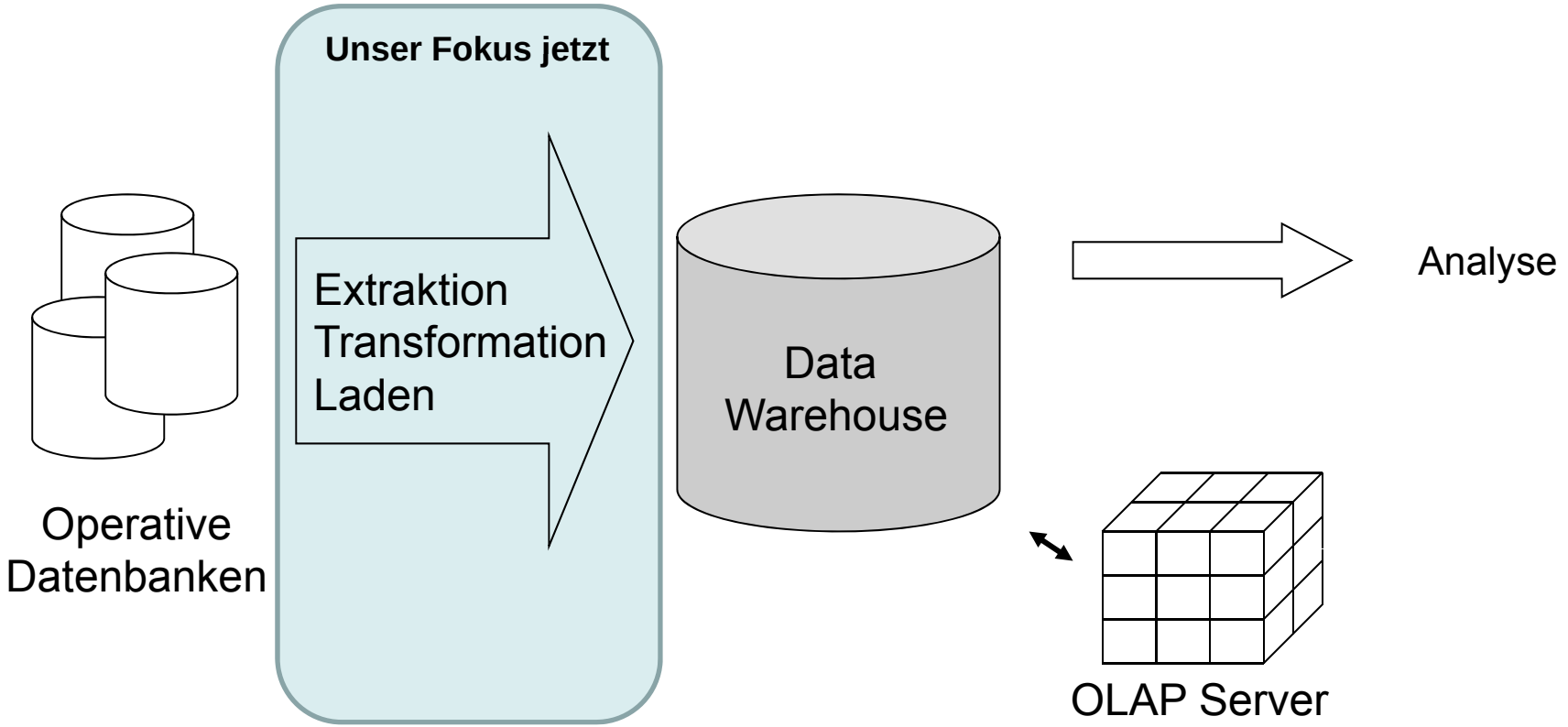
Data-Warehouse-Prozess

- Monitoring
  - Entdecken und Melden von Änderungen in den Quellen (*triggerbasiert, replikationsbasiert, zeitstempelbasiert, logbasiert, snapshotbasiert*)
- Extraktion
  - Selektion und Transport von Daten aus den Quellen in den Arbeitsbereich (*periodisch, auf Anfrage, ereignisgesteuert, sofort*)
- Transformation
  - Vereinheitlichung, Bereinigung, Integration, Konsolidierung, Aggregierung und Ergänzung der Daten im Arbeitsbereich
- Laden
  - Laden der Daten aus dem Arbeitsbereich in die Basisdatenbank bzw. ins Data Warehouse
- Analyse
  - Analyse und Präsentation der Daten im Data Warehouse

Vereinfachte Sicht auf die Referenzarchitektur



Fokus jetzt im Praktikum



Agenda

- Einführung Data Warehouses
  - Referenzarchitektur
  - Datenbeschaffung
- Online Transactional Processing (OLTP)
  - Datenmanipulation mit SQL
  - Anfragen mit SQL
- Nächste Aufgabe
  - Analyse von Graphen
  - Projekt LogoTakt
  - SQL, ETL, Visualisierung

Folie aus Veranstaltung 1

OLTP vs. OLAP
(Datenbank vs. Data Warehouse)

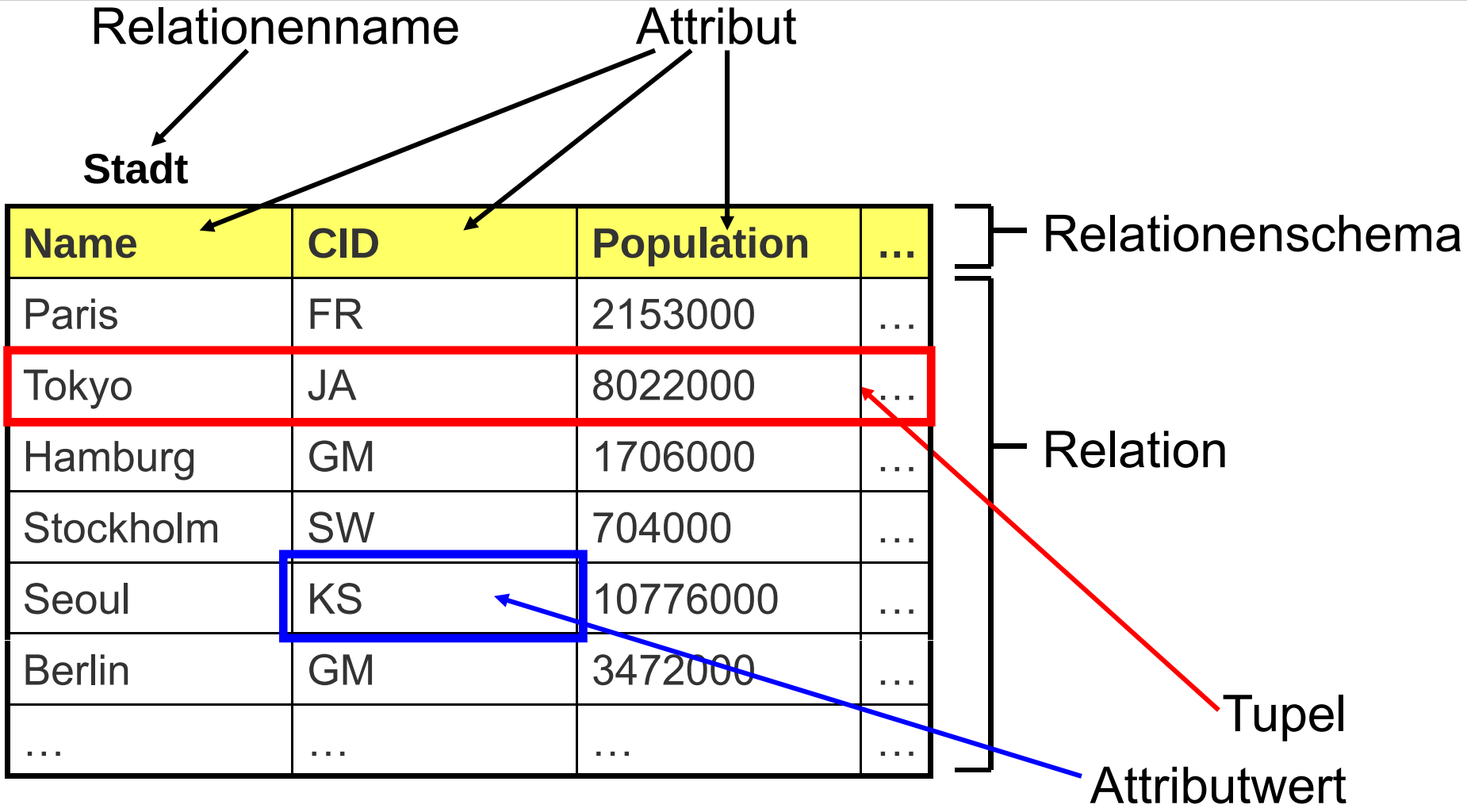
- Anfragecharakteristika

	transaktional	analytisch
Fokus	Lesen, Schreiben, Modifizieren, Löschen	Lesen, periodisches Hinzufügen
Transaktionsdauer und -typ	Kurze Lese- / Schreibtransaktionen	Lange Lesetransaktionen
Anfragestruktur	Einfach strukturiert	komplex
Datenvolumen einer Anfrage	Wenige Datensätze	Viele Datensätze

nach Bauer, Günzel (Hrsg):

Data Warehouse Systeme – Architektur, Entwicklung, Anwendung

Relationenmodell – Kurze Wiederholung



Integritätsbedingungen

- Primärschlüssel
  - Menge von Attributen zur eindeutigen Identifikation eines Tupels
  - Nötig, um eindeutig auf Tupel zugreifen zu können
- Fremdschlüssel
  - Referenziert von einem Tupel auf ein Tupel einer anderen Relation
  - Nötig zur Speicherung von Abhängigkeiten

SQL

- Eigenschaften
  - *die* Sprache für relationale Datenbanken
  - mengenorientiert & deklarativ
- Konstrukte zur Datendefinition (SQL-DDL)
  - **`CREATE, ALTER, DROP`**
- Konstrukte zur Datenmanipulation (SQL-DML)
  - **`INSERT, UPDATE, DELETE`**
- Konstrukt für Datenabfragen
  - **`SELECT`**

Datentypen

- Zeichenketten
  - **`CHARACTER(n), CHAR(n)`**
  - **`VARCHAR(n)`**
- Zahlen
  - **`INTEGER, INT`**
  - **`NUMERIC(p, s)`**
  - **`FLOAT`**
- Datum und Uhrzeit
  - **`DATE`**

Agenda

- Einführung Data Warehouses
  - Referenzarchitektur
  - Datenbeschaffung
- Online Transactional Processing (OLTP)
  - Datenmanipulation mit SQL
  - Anfragen mit SQL
- Nächste Aufgabe
  - Analyse von Graphen
  - Projekt LogoTakt
  - SQL, ETL, Visualisierung

SQL – Create

- Anlegen von Relationen

- Syntax

```
CREATE TABLE <Relation> (
    <Attribut><Datentyp>,
    …
    PRIMARY KEY (<Attribut>[, …])
    FOREIGN KEY <Attribut>
    REFERENCES <Relation>(<Attribut>)
    [, …]
)
```

SQL – Insert und Update

- Einfügen von Tupeln in Relation

- Syntax

```
INSERT INTO <Relation> VALUES
    (<Datum1>, <Datum2>, …)
```

- Ändern von Tupeln

- Syntax

```
UPDATE <Relation>
    SET <Attribut> = <Datum>
    WHERE <Selektionsbedingung>
```

SQL – Delete und Drop

- Löschen von Tupeln aus einer Relation
- Syntax

```
DELETE FROM <Relation>
    WHERE <Attribut> = <Datum>
```

- Löschen von Relationen
- Syntax

```
DROP TABLE <Relation>
```

Agenda

- Einführung Data Warehouses
  - Referenzarchitektur
  - Datenbeschaffung
- Online Transactional Processing (OLTP)
  - Datenmanipulation mit SQL
  - Anfragen mit SQL
- Nächste Aufgabe
  - Analyse von Graphen
  - Projekt LogoTakt
  - SQL, ETL, Visualisierung

Anfragen - Grundgerüst

- Anfragen an den Datenbestand

- Syntax

```
SELECT <Attribut>, …
    FROM <Relation>
    WHERE <Selektionsbedingung>
```

Projektion

- Auswahl von Spalten einer Relation
- Syntax

```
SELECT <Attribut>, …
FROM <Relation>
```

Name	CID	Population
Paris	FR	2153000
Tokyo	JA	8022000
Hamburg	GM	1706000
Stockholm	SW	704000
Seoul	KS	10776000
Berlin	GM	3472000

Selektion

- Auswahl von Tupeln einer Relation
- Syntax

```
SELECT * FROM <Relation>
WHERE <Selektionsbedingung>
```

Name	CID	Population
Paris	FR	2153000
Tokyo	JA	8022000
~~Hamburg~~	~~GM~~	~~1706000~~
Stockholm	SW	704000
Seoul	KS	10776000
Berlin	GM	3472000

Verbund

- Kombination mehrerer Relationen

- Syntax

```
SELECT <Attribut>, …
FROM <Relation1>, <Relation2>
WHERE
    <Relation1>.<Attribut> =
    <Relation2>.<Attribut>
```

Aggregatfunktionen

- Berechnung von Aggregaten auf Relationen

- Syntax

```
SELECT <Aggregat>(<Attribut>) AS <Name>
FROM <Relation>
```

- Wichtige Aggregatfunktionen:
  - COUNT
  - SUM
  - MIN
  - MAX
  - AVG

Gruppierung

- Gruppierung von gleichen Attributwerten

- Syntax

```
SELECT <Attribut>
    FROM <Relation>
    GROUP BY <Attribut>
    HAVING <Gruppenbedingung>
```

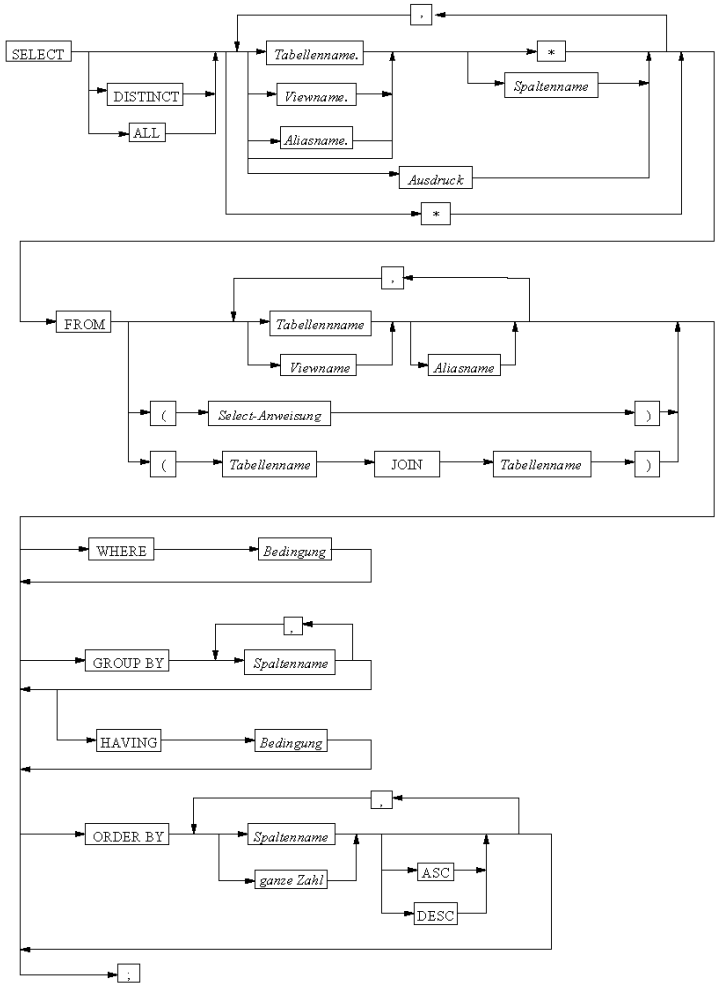
Mengenoperationen

- Mengenoperationen auf Anfrageergebnissen

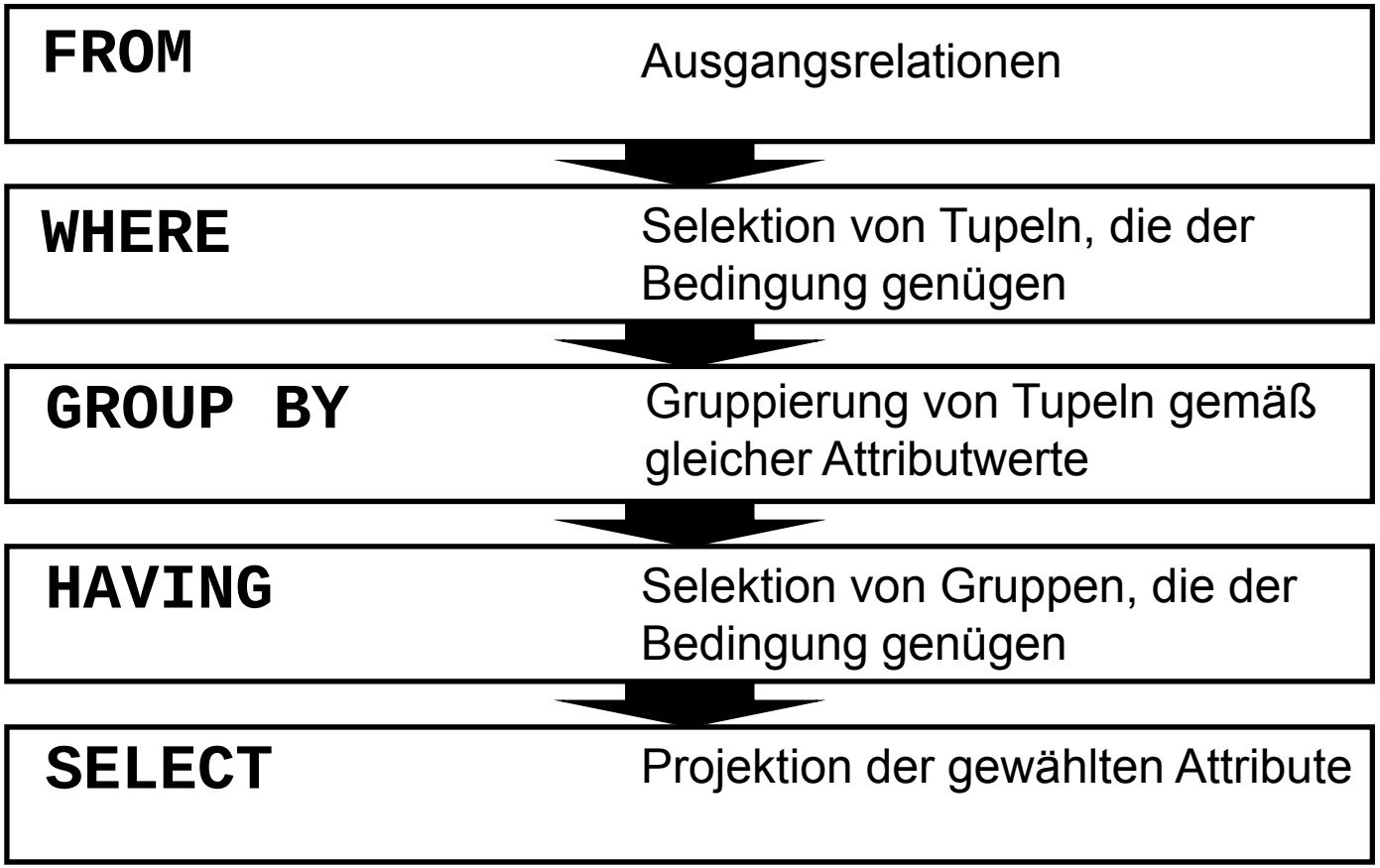
```
(SELECT <Attribut>, …
 FROM <Relation>)
INTERSECT | UNION | MINUS
(SELECT <Attribut>, …
 FROM <Relation>)
```

SELECT-Syntax

- Syntaxdiagramm des SQL-SELECT-Befehls (vereinfacht…):



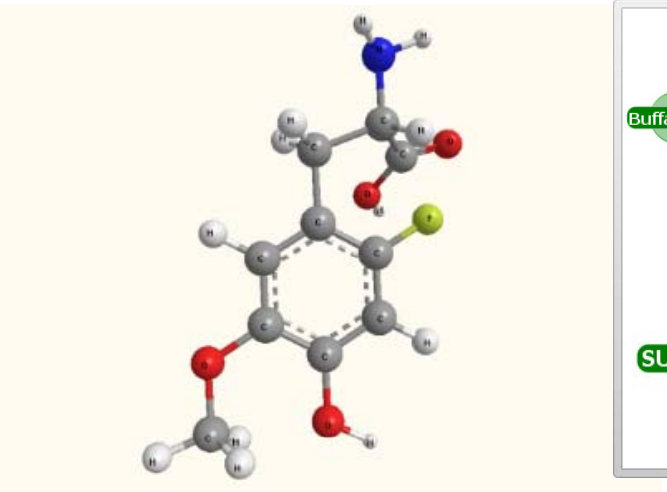
Vorgehen bei der Definition von Anfragen



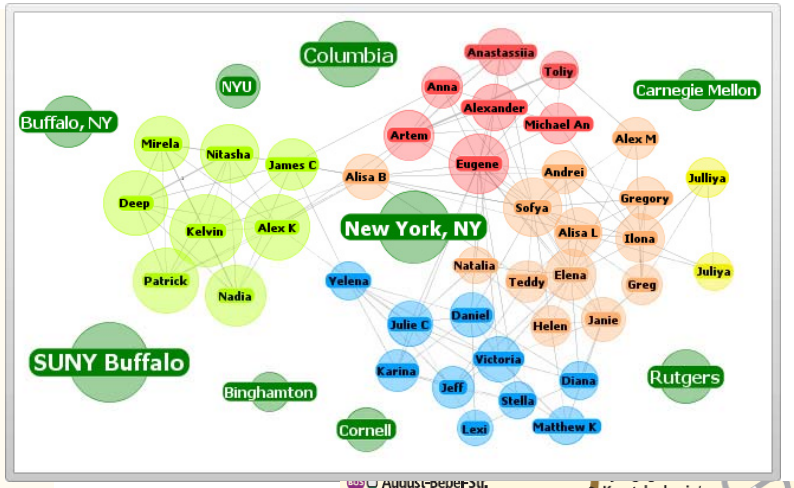
Agenda

- Einführung Data Warehouses
  - Referenzarchitektur
  - Datenbeschaffung
- Online Transactional Processing (OLTP)
  - Datenmanipulation mit SQL
  - Anfragen mit SQL
- Nächste Aufgabe
  - Analyse von Graphen
  - Projekt LogoTakt
  - SQL, ETL, Visualisierung

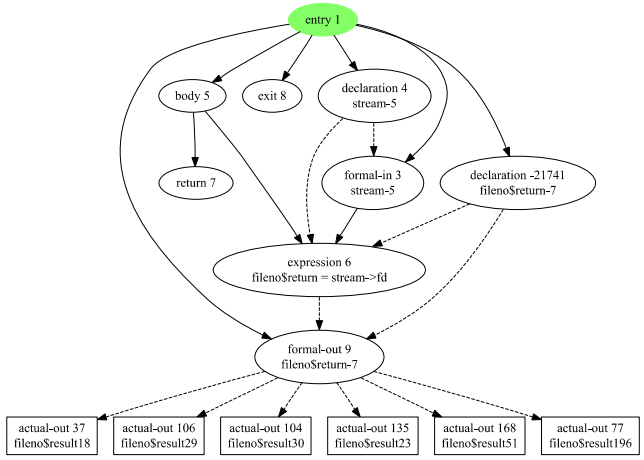
Graphen

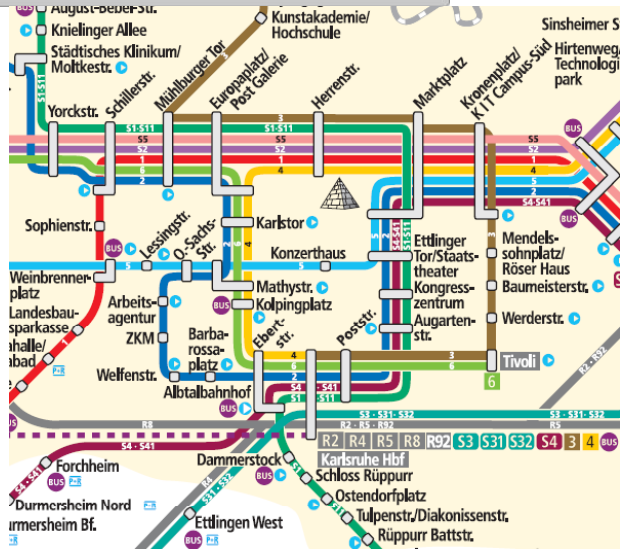


http://www.fzd.de/db/Pic?pOid=26054



www.touchgraph.com





www.kvv.de

Graphen: Ausgewählte Forschungsfragen

- Frequent Subgraph Mining
  - Auffinden unbekannter, häufiger Teilgraphen mit Support >= s
- Graphanfragesprachen
  - Spezifikation und Auffinden von Mustern mit bestimmten strukturellen Eigenschaften
- Gemeinsames zugrundeliegendes Problem: Subgraphisomorphie (NP-vollständig)

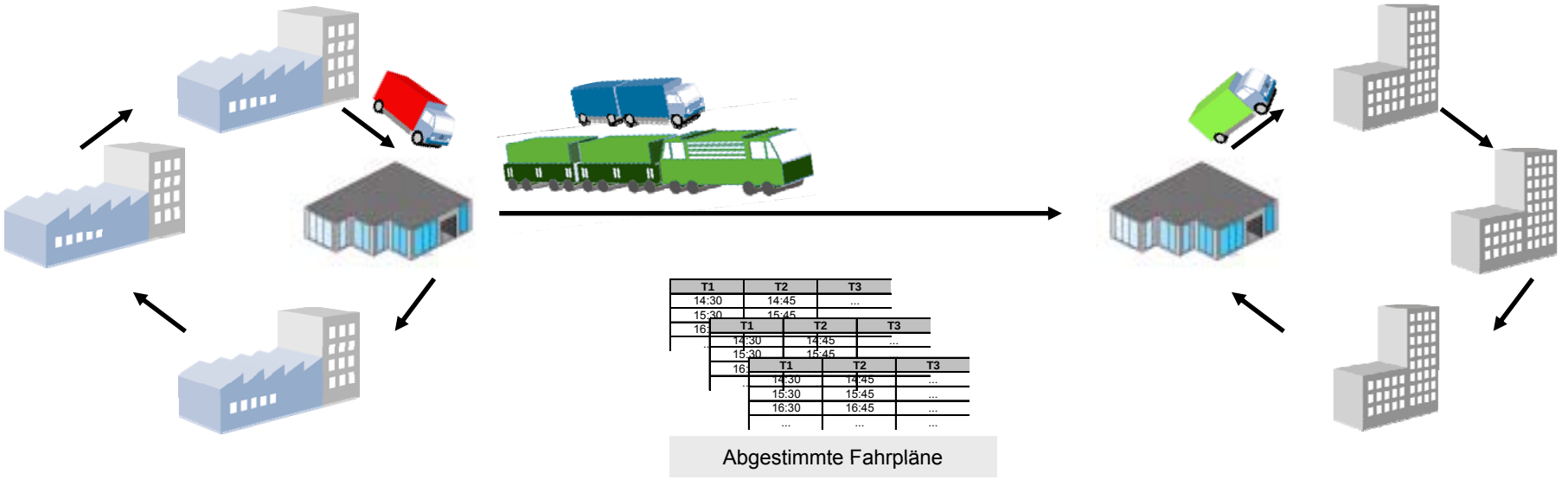
Projekt LogoTakt

- Ziel: Entwicklung von Technologien und Prozessen für robuste getaktete Logistiknetzwerke
- Vom BMWi gefördert

LogoTakt, BOSCH, FZI, Schweitzer Spedition, Die Bahn DB, LOCOM Consulting GmbH Software GmbH, traffic mobility logistics., KIT Karlsruhe Institute of Technology, Dönne+Hellwig Logistics D.Logistics Gruppe, Volkswagen Logistics

- Analyse der Robustheit abgelaufener Transportprozesse unter Einsatz von:
  - Graph Mining
  - Graphanfragesprachen

Kernmerkmale LogoTakt



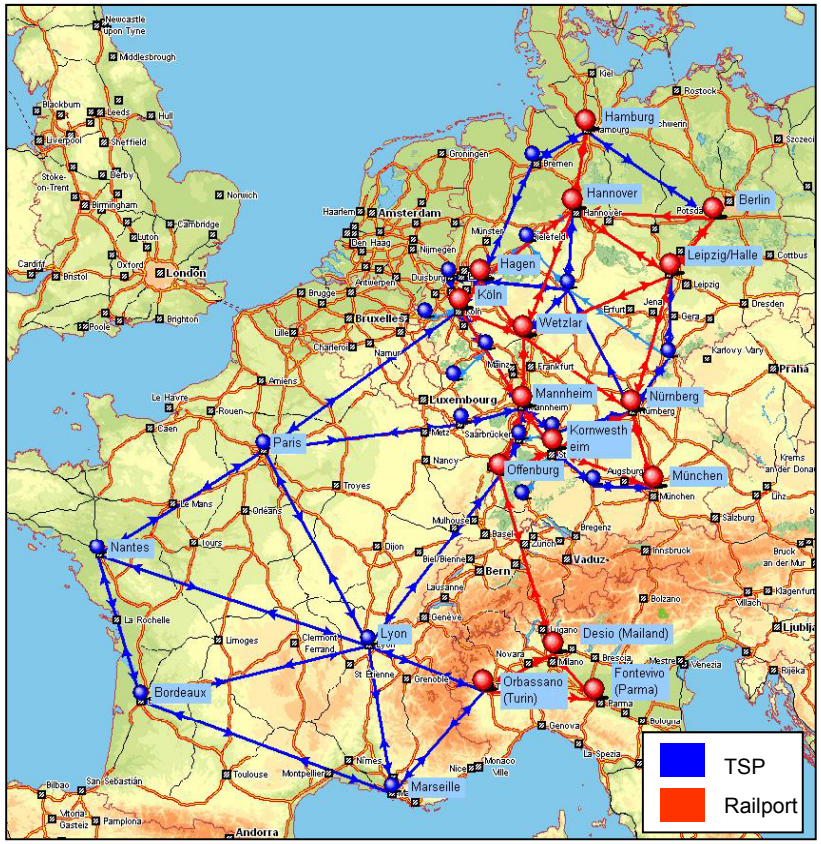
Abgestimmte Fahrpläne

Taktung der Verkehre in Vor- Haupt- und Nachlauf

Hohe **Robustheit** durch vorkalkulierte Zeit- und Mengenpuffer

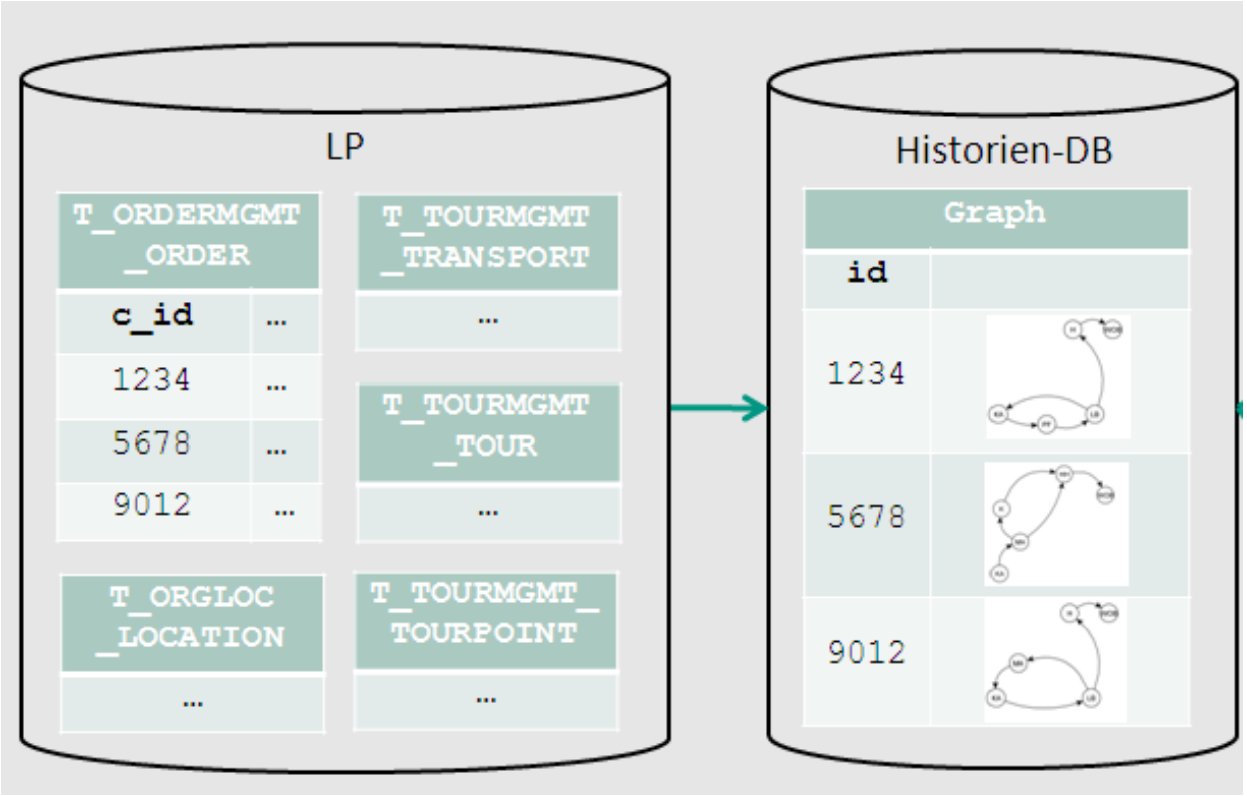
Intermodalität zur Verbesserung der Ökobilanz

Netzwerkstruktur



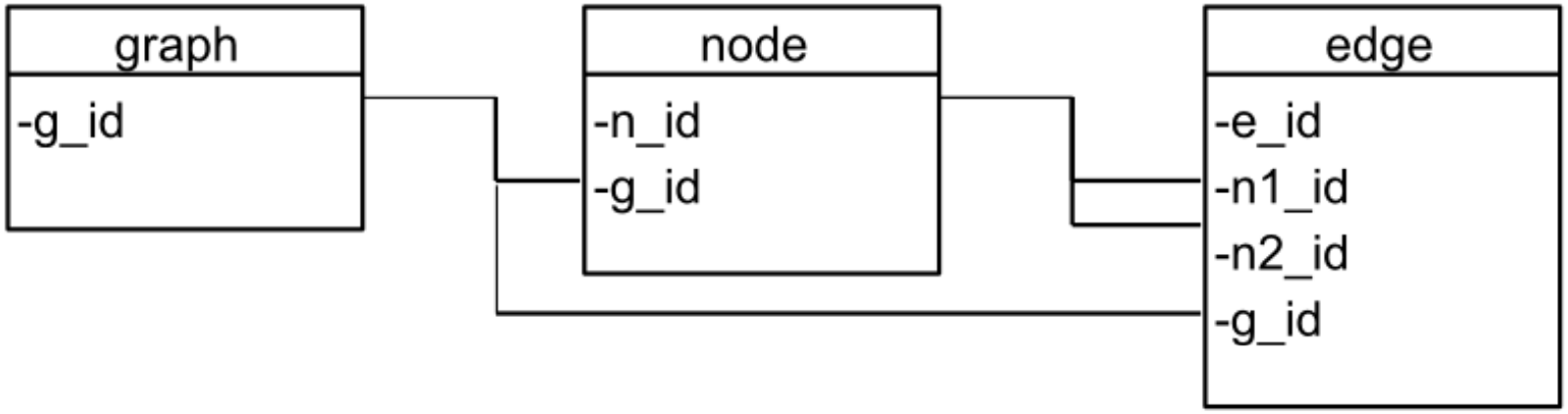
Transformationsprozess

- ETL-Aufgabe



Generische Graphdarstellung

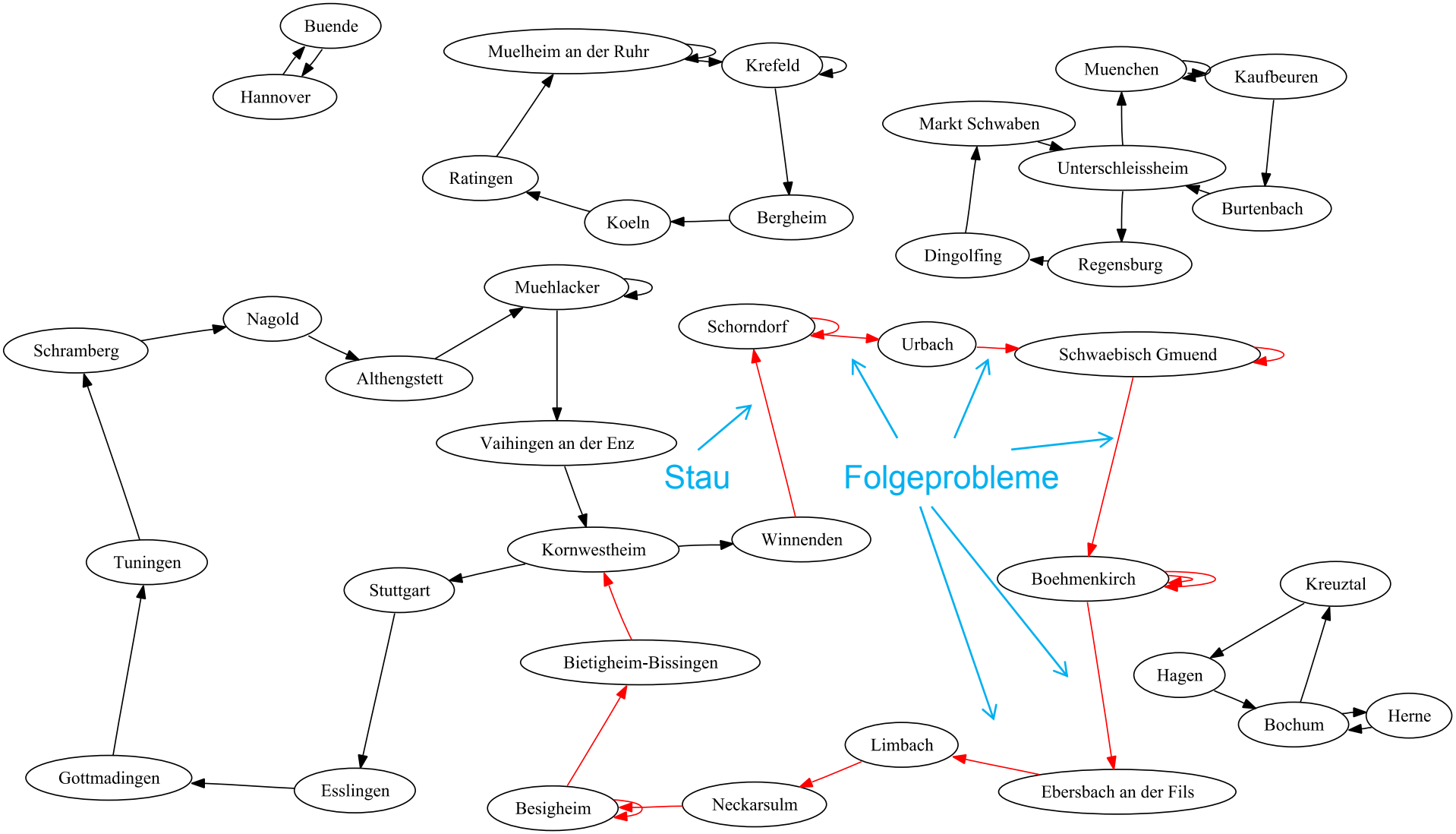
- Ziel:
  - Drei Relationen: Graph, Node, Edge.
  - Jeweils numerische ID für jedes Tupel
  - Zusätzlich: alle „interessanten“ Attribute
  - Jeweils ein Knoten pro Adresse (PLZ, Straße identisch)



Graph-Mining-Analysen in LogoTakt

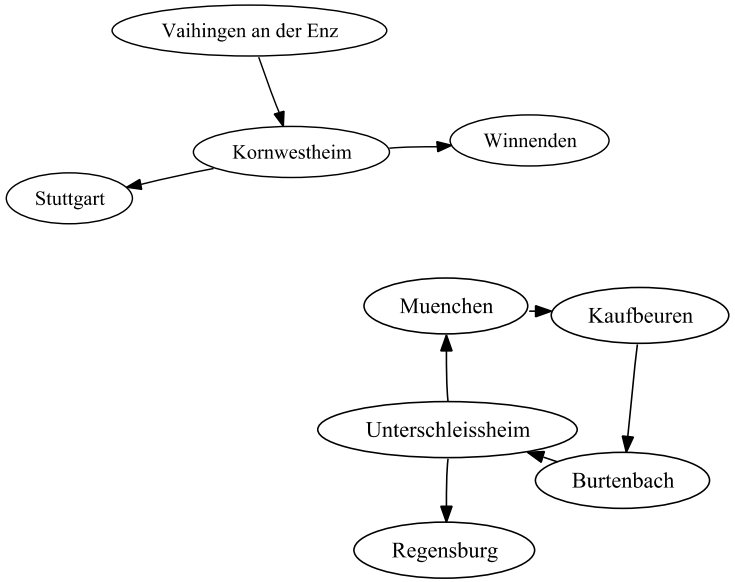
- Ziel:
  - Regelmäßiges Finden von interessanten Zusammenhängen in den Historien einer vergangenen Periode in einem manuellen Prozess.
  - Eingabe für die Optimierung des Gesamtnetzwerks.
- Demo:
  - Betrachtung der LKW-Fahrten des vergangenen Monats.
  - Jeder Tag bildet einen Graphen, jede Stadt einen Knoten, jede Fahrt eine Kante. Alternativ kann die Sicht auf eine Region/Spedition beschränkt werden.
  - Kanten werden rot eingefärbt, gdw. IST > SOLL.

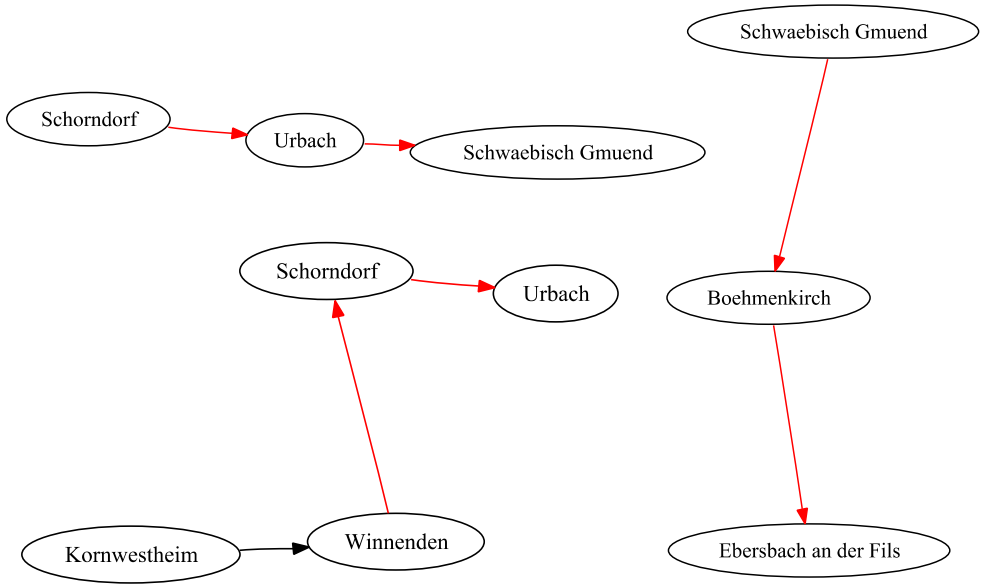
Beispiel-Graph, 2010-04-06_Dienstag



Graph-Mining-Ergebnis I

- Graph Mining mit allen 30 Graphen des April 2010 (finde alle Teilgraphen, die in mindestens 10% aller Graphen enthalten sind) liefert folgendes:



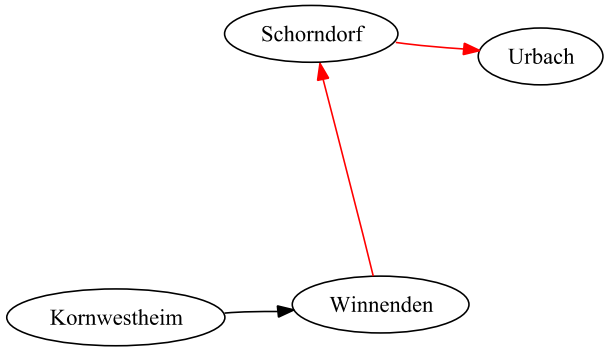


viele weitere…

In welchen Graphen ist dieses Teilgraph-Muster enthalten?

Graph-Mining-Ergebnis II

- In welchen Graphen ist dieses Teilgraph-Muster enthalten?
  - 2010-04-06_Dienstag
  - 2010-04-13_Dienstag
  - 2010-04-20_Dienstag
  - 2010-04-27_Dienstag



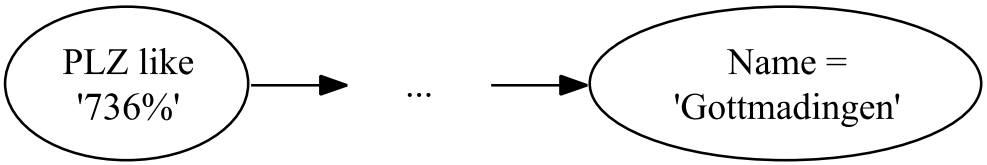
→ Probleme treten immer am Dienstag auf!

→ „Dienstags-Stau“ auf Relation Winnenden -> Schorndorf hat Auswirkungen auf das Gesamtnetz.

Graphanfragen

- Uns ist ein weiteres Problem bekannt: Ein Kunde aus Gottmadingen wird oft spät beliefert, Verzögerungen im Umkreis von Schorndorf (PLZ 736…).
  - Das Dienstags-Problem ist nicht schuld:
    - An diesen Tagen keine Fahrten zwischen diesen Orten.
  - Weitere Analyse dieser Problematik: Betrachte alle Fahrten aus PLZ-Bereich 736… mit Zwischenstopps nach Gottmadingen mittels S²QL.
  - Anfrage:

```
select g.name, a.name as ort
from graph g, node a, node b
where a.plz like '736%' and b.name = 'Gottmadingen'
structured 'a/+/b/';
```



Ergebnisse dieser Demo

- Uns ist ein weiteres Problem bekannt: Ein Kunde aus Gottmadingen wird oft spät beliefert, Verzögerungen im Umkreis von Schorndorf (PLZ 736…).
  - Weiteres Problem offensichtlich donnerstags!

2 Einträge gefunden, zeige alle Einträge.**1**

Name	Ort
2010_04_15_Donnerstag	Schorndorf
2010_04_22_Donnerstag	Schorndorf

Exportoptionen: CSV | XLS| XML | PDF

- Mit Graph Mining und Graphanfragen wurden diverse Probleme rund um Kornwestheim identifiziert.

- Fragestellungen für nächste Iteration der Netzwerkoptimierung:
  - Sollte für Fahrten in diesem Gebiet mehr Zeit eingeplant werden?
  - Kann Gebietszuschnitt optimiert werden?
  - Ist ein zusätzlicher Railport in BW hilfreich?

Nächstes Aufgabenblatt

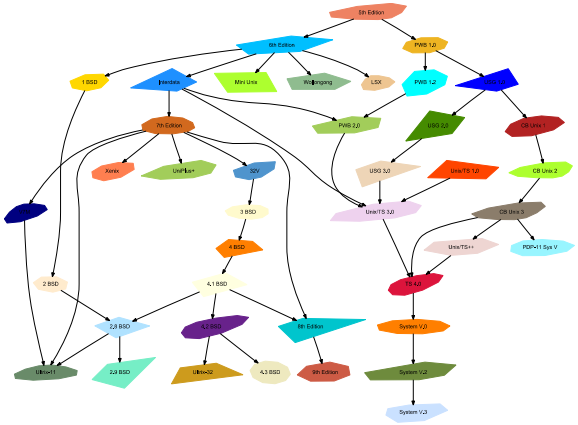
- Kennenlernen des Datenbestands mit SQL-Anfragen
- Durchführung eines (wiederholbaren) ETL-Prozesses: Überführung der Transportdaten in generisches Graphformat
- Visualisierung des Datenbestands mittels *dot*

Zu erzeugende Graphen

- Jede Zweiergruppe bearbeitet eine mögliche Kombination der folgenden drei Parameter:
  - Zeit: ein Graph pro Tag/pro Woche/pro Wochentag
  - Graph muss zusammenhängend sein: ja/nein
  - Mehrfachkanten zusammengefasst: ja/nein
- Visualisierung je nach Schwierigkeitsgrad der erzeugten Graphen
- Bewertung der Lösung u.a. je nach Umfang und „Sinn“ der zugeteilten Graph-/Knoten-/Kanten-Attribute

Graph-Visualisierung

- Graphen als Menge von Knoten und Kanten sind ein Konstrukt der diskreten Mathematik.
- Graph-Visualisierung
  - …ist oft sehr hilfreich!
  - …verfolgt (ästhetische) Ziele, z.B.
    - Minimierung von Kreuzungen, Fläche
    - Maximierung des kleinsten Winkels
    - Vereinheitlichung von Kantenlängen
  - …kann sehr aufwändig werden (NP-vollständig).
  - …ist ein eigenes Forschungsgebiet.
  - …kann von verschiedenen Tools erledigt werden.



http://www.graphviz.org/Gallery.php

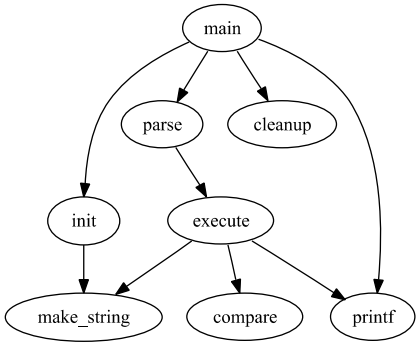
Graph-Visualisierung mit Graphviz

```
digraph G {
  main -> parse -> execute;
  main -> init;
  main -> cleanup;
  execute -> make_string;
  execute -> printf
  init -> make_string;
  main -> printf;
  execute -> compare;
}
```

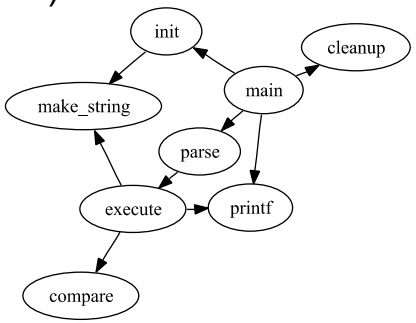
g.dot

Textuelle Graph-Darstellung als Kantenliste in DOT

Hierarchische, baumartige Visualisierung mit dot



Optimierungsbasierte Visualisierung mit neato (spring-model)



…

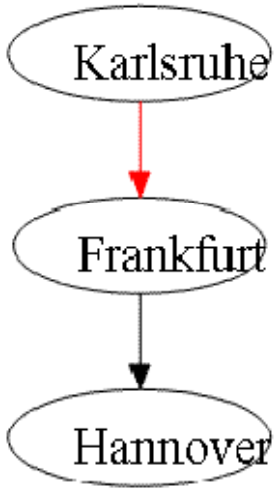
www.graphviz.org

Weiteres DOT-Beispiel

```
digraph G {
  0 [label="Karlsruhe"];
  1 [label="Frankfurt"];
  2 [label="Hannover"];
  0 -> 1 [color="red"];
  1 -> 2;
 }
```

example.dot

```
dot -Gcharset=latin1
 -O -Tgif example.dot
```



Textuelle Graph-Darstellung als gelabelte/ attributierte Knoten- und Kantenliste

- Für „schöne“ Ergebnisse empfiehlt sich der Export als SVG (`-Tsvg`); anschließende Konvertierung z.B. mit Inkscape (`www.inkscape.org`).
- Viele weitere Optionen und Beispiele in den Handbüchern zu *dot* bzw. *neato*.

DOT-Erzeugung mit SQL

```
SELECT 'digraph g {'
UNION ALL
SELECT DISTINCT n_id || ' [label="' || name || '"]'
   FROM node WHERE g_id = <g_id>
UNION ALL
SELECT DISTINCT n1_id || ' -> ' || n2_id
   FROM edge WHERE g_id = <g_id>
UNION ALL
SELECT '}'
```

- Beachte:
  - `||` in Projektionen verkettet Attribute als String
  - Manche DBMS verlangen die Angabe der FROM-Klausel, z.B. `SELECT 'digraph g {' FROM dual`

Quellenangaben

A. Bauer, H. Günzel: „Data Warehouse Systeme – Architektur, Entwicklung, Anwendung“, dpunkt.verlag, 2004.

K. Sattler, S. Conrad: Folien zur Vorlesung Data Warehouse Technologien, 2003

C. von der Weth: Folien zum Datenbankpraktikum, 2005

M. Stock und R. Pinger: Kleiner Leitfaden zur Anwendung von SQL-Anweisungen, 1997

Graphviz-Doku: http://www.graphviz.org/Documentation.php

Software (Graphviz, Inkscape) als Portable-Version unter /home/eichi/software im IPD-UNIX-Netz