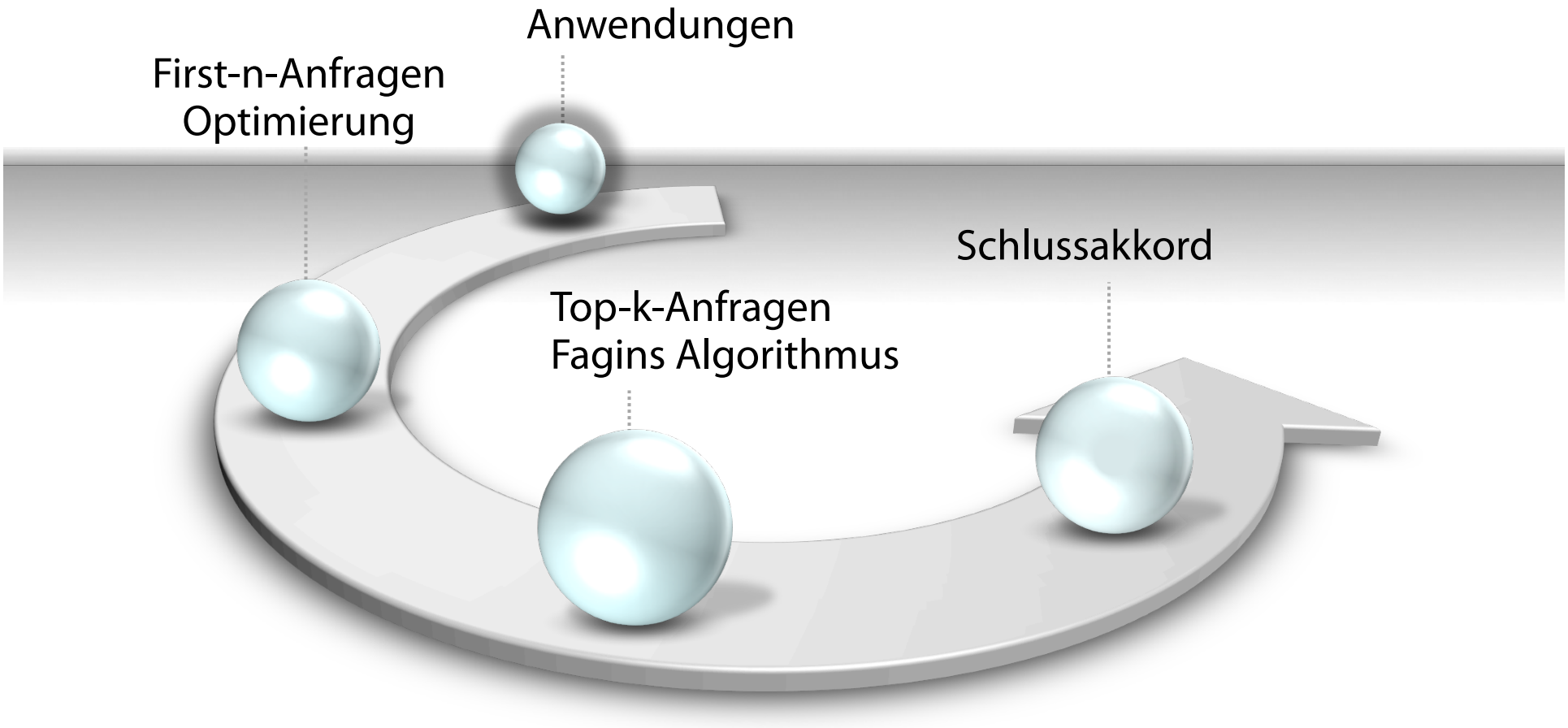
Non-Standard-Datenbanken

First-n- und Top-k-Anfragen

Prof. Dr. Ralf Möller
Universität zu Lübeck
Institut für Informationssysteme

Non-Standard-Datenbanken

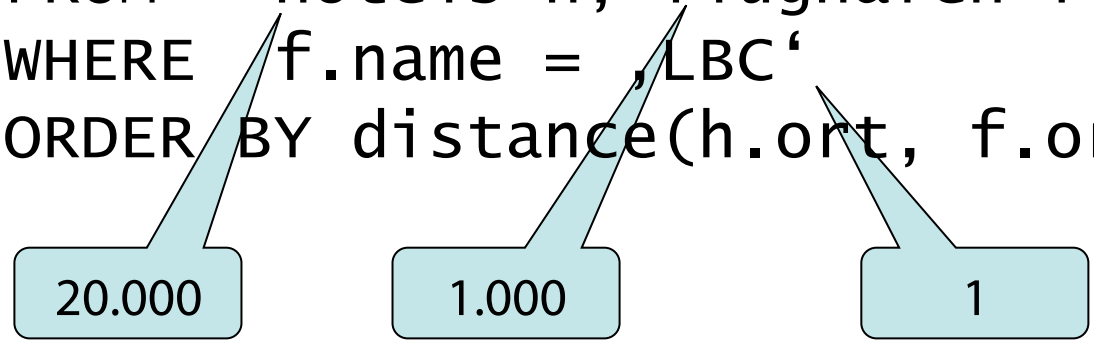
First-n und Top-k-Anfragen



Danksagung an Felix Naumann für Material
aus VL Informationsintegration, WS 05/06

Beispiel

```
SELECT h.name, h.adresse, h.tel
FROM   hotels h, flughäfen f
WHERE  f.name = ‚LBC‘
ORDER BY distance(h.ort, f.ort)
```



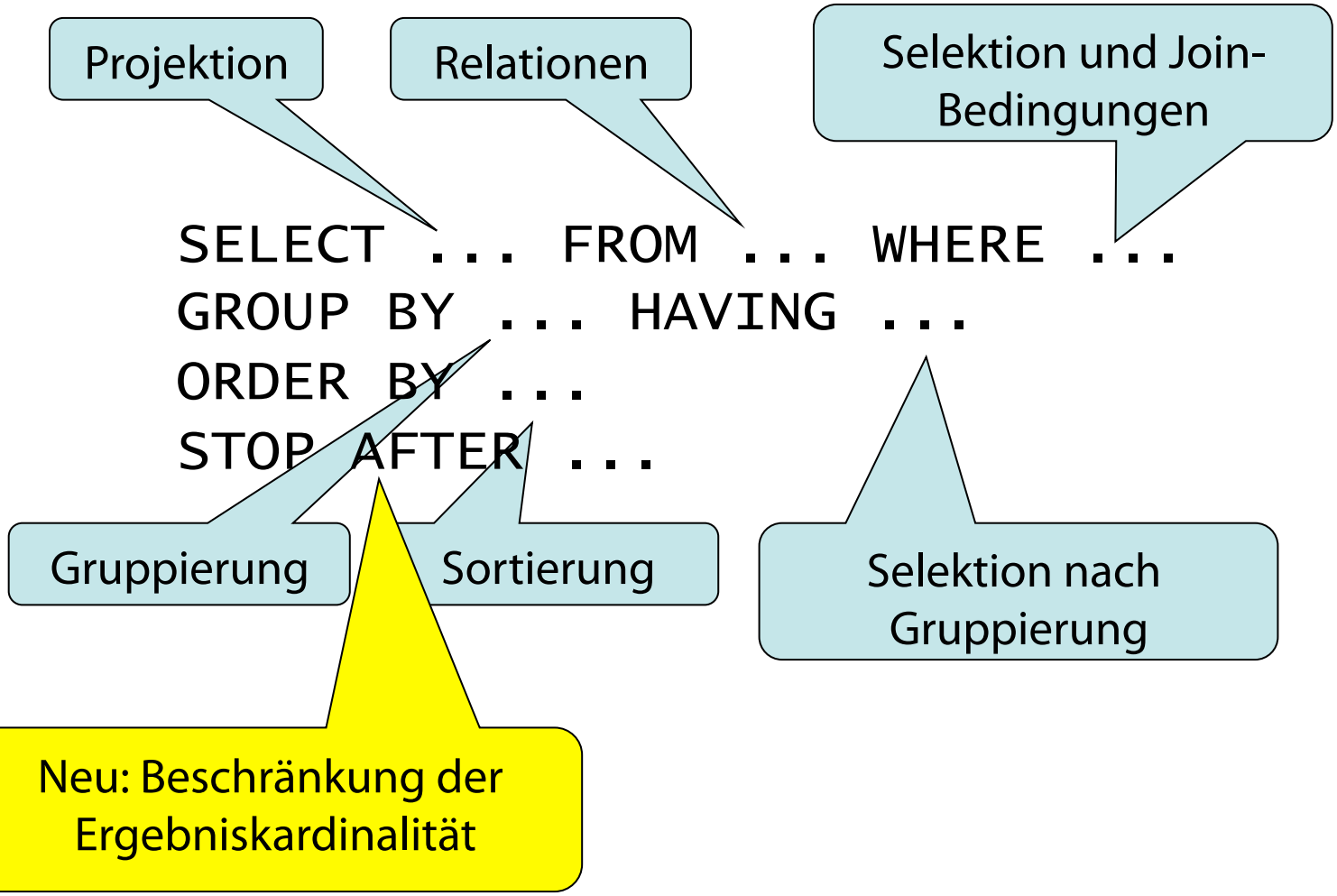
- Naive Auswertung: 20.000 Hotels mit aufsteigender Entfernung zu LBC
- Zur Sortierung: mind. 20.000 Mal distance() ausführen

M.J. Carey, Donald Kossmann: On Saying "Enough Already!" in SQL. SIGMOD Conference: 219-230, **1997**

Problem

- Was machen wir, wenn nur 5 Hotels gezeigt werden sollen?
  - Auswahl im Anwendungsprogramm
    - Großer Aufwand
      Eventuell werden sehr viele Daten kommuniziert
  - Auswahl durch Programmiersprache auf Seiten der DB angewendet auf Anfrageergebnis
    - Keine Optimierung
  - Gewünschte Anzahl in Anfrage spezifizieren
    - Optimierungsmöglichkeiten

STOP AFTER – Syntax



M.J. Carey, Donald Kossmann: On Saying "Enough Already!" in SQL. SIGMOD Conference: 219-230, **1997**

STOP AFTER – Semantik

- Ohne Sortierung
  - Willkürlich n Tupel aus dem SQL-Ergebnis
- Mit Sortierung
  - Erste n Tupel aus dem SQL-Ergebnis (sortiert)
  - Bei Gleichheit bzgl. des Sortierprädikats willkürliche Auswahl falls n+1-tes Tupel usw. gleich n-tem Tupel

- Daraus folgt:
  Falls Ergebnis nur bis zu n Tupel → keine Wirkung

STOP AFTER – Beispiel

```
SELECT h.name, h.adresse, h.tel
FROM   hotels h, flughäfen f
WHERE  f.name = ‚LBC‘
ORDER BY distance(h.ort, f.ort)
STOP AFTER 5
```

- Ergebnis: 5 Hotels mit aufsteigender Entfernung zu LBC
- Können wir Indexstrukturen für distance() nutzen?

STOP-AFTER: Berechnete Anzahl

- Liste Name und Umsatz der 10% umsatzstärksten Softwareprodukte

```
SELECT p.name, v.umsatz
FROM   Produkte p, Verkäufe V
WHERE  p.typ = ‚software‘
AND    p.id = v.prod_id
ORDER BY v.umsatz DESC
STOP AFTER (
```

)

Praktische Ausprägung in PostgreSQL 9.3

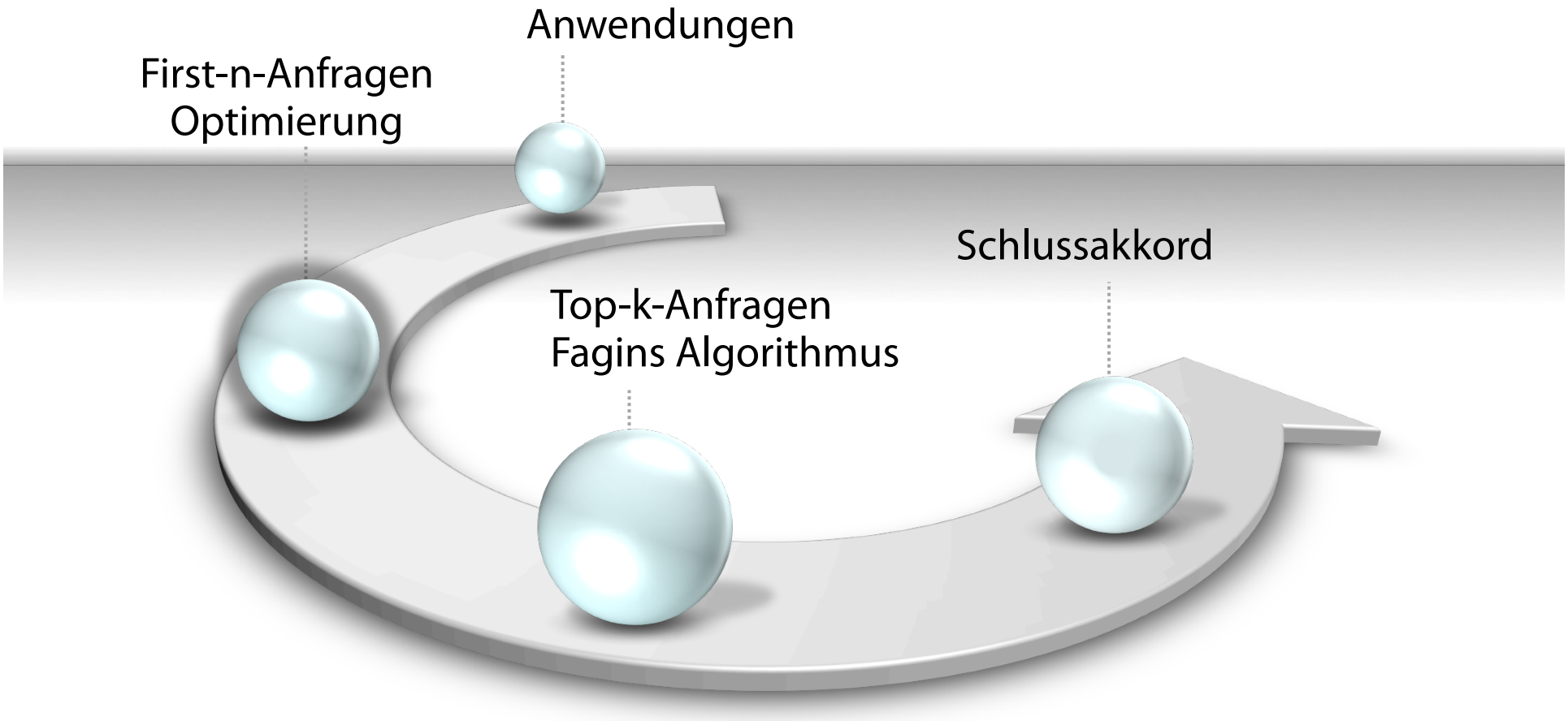
```
SELECT ...
FROM ...
...
LIMIT { count | ALL }
OFFSET start
```

- Verwendung in Kombination mit `ORDER-BY` möglich und sinnvoll

- Falls der Ausdruck ***count*** sich zu `NULL` evaluiert, wird `ALL` angenommen.
- Falls der Ausdruck ***start*** sich zu `NULL` evaluiert, wird 0 angenommen.

Non-Standard-Datenbanken

First-n und Top-k-Anfragen



Danksagung an Felix Naumann für Material
aus VL Informationsintegration, WS 05/06

Optimierung mit Stop-Operator

- Platzierung des Stop Operators im Anfrageplan
- Fundamentales Problem: Frühe Platzierung vorteilhaft aber risikoreich
  - Vorteil: Kleine Zwischenergebnisse ⇒ geringe Kosten
  - Risiko: Endergebnis nicht groß genug ⇒ Erneute Ausführung
- Zwei Strategien
  - „Konservativ“ und „aggressiv“

Optimierung mit Stop-Operator

- Konservative Strategie
  - Kostenminimal:
    Platziere Stop so früh wie möglich in Plan.
  - Korrekt: Platziere Stop nie so, dass Tupel entfernt werden, die später eventuell gebraucht werden.
    - Operatoren, die Tupel filtern, müssen also früher ausgeführt werden.

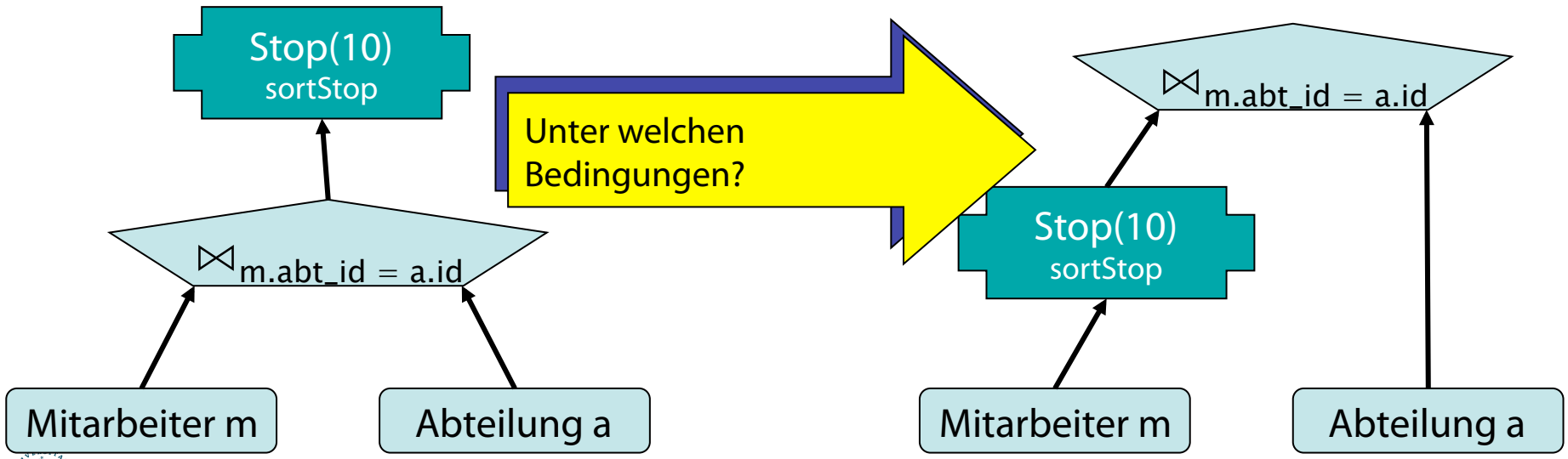
STOP AFTER: Optimierung

```
SELECT *
FROM   mitarbeiter m, abteilung a
WHERE  m.abt_id = a.id
ORDER BY m.gehalt DESC
STOP AFTER 10
```

- Wie würden Sie den Anfragebeantwortungsplan gestalten?
- Unter welchen Bedinungen ist eine Optimierung möglich?

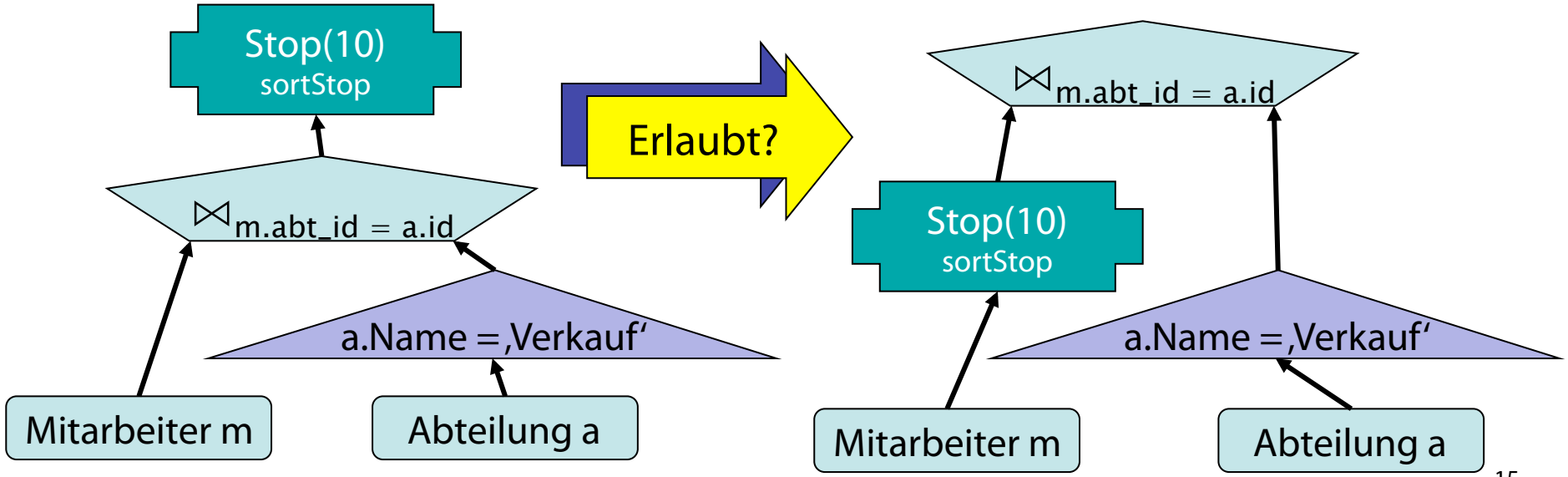
Optimierung mit Stop-Operator

```
SELECT *
FROM   mitarbeiter m, abteilung a
WHERE  m.abt_id = a.id
ORDER BY m.gehalt DESC
STOP AFTER 10
```



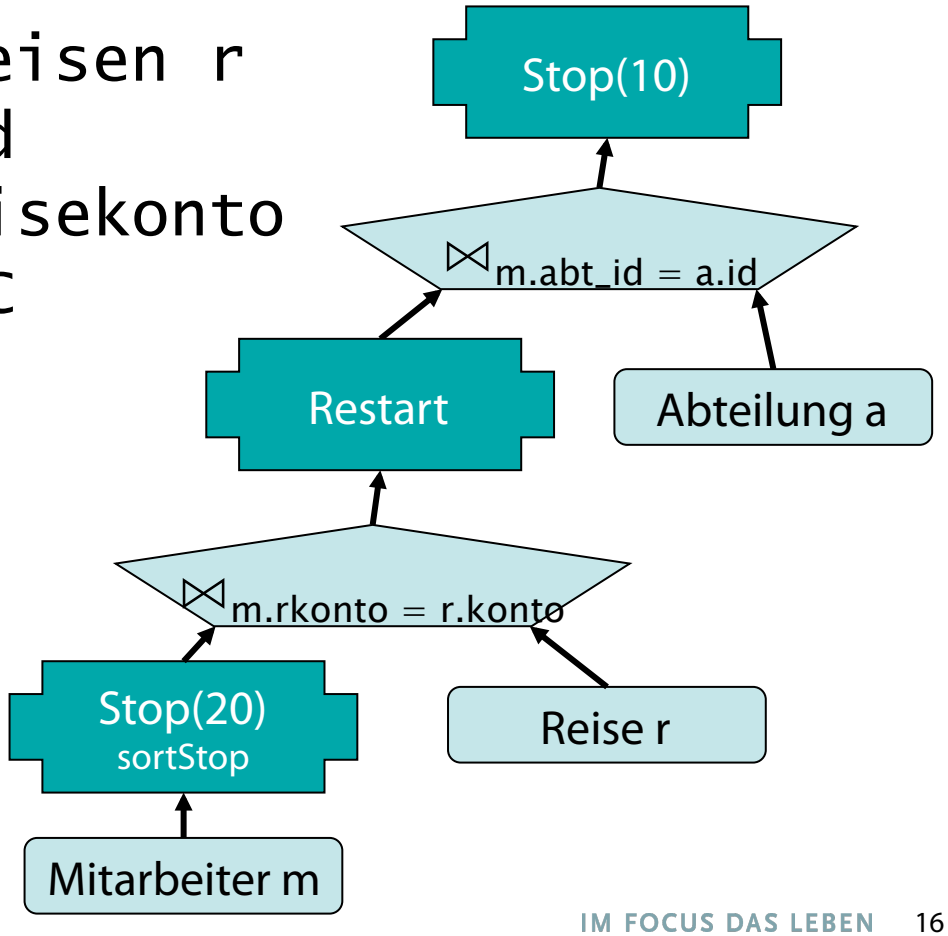
Optimierung mit Stop-Operator

```
SELECT *
FROM   mitarbeiter m, abteilung a
WHERE  m.abt_id = a.id
AND    a.name = ‚Verkauf‘
ORDER BY m.gehalt DESC
STOP AFTER 10
```



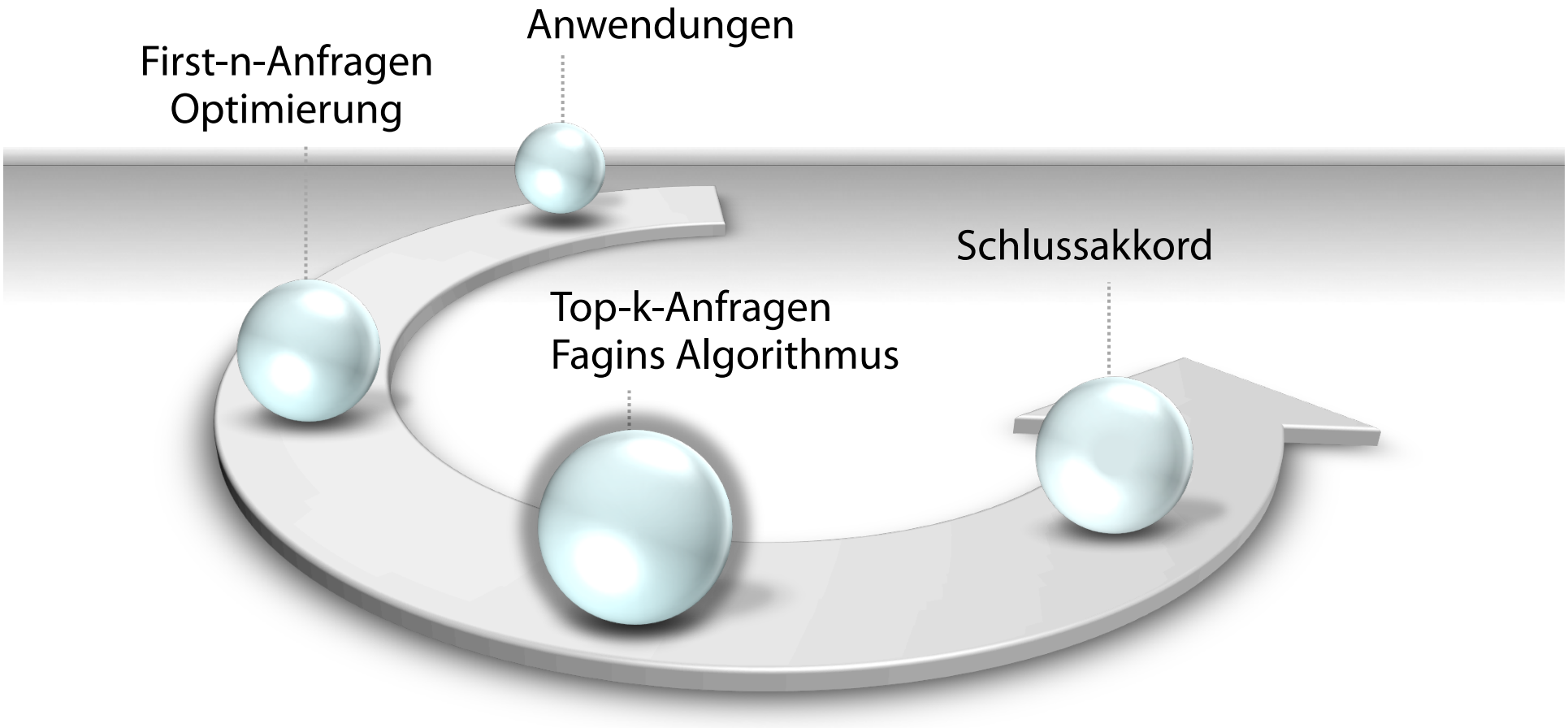
Optimierung mit Stop-Operator (aggressiv)

```
SELECT *
FROM   mitarbeiter m,
       abteilung a, reisen r
WHERE  m.abt_id = a.id
AND    r.konto = m.reisekonto
ORDER BY m.gehalt DESC
STOP AFTER 10
```



Non-Standard-Datenbanken

First-n und Top-k-Anfragen



Danksagung an Felix Naumann für Material
aus VL Informationsintegration, WS 05/06

Top-k Ergebnisse: Motivation

- First-n beschränkt Ergebnismenge bzgl. eines Attributs
  - Bei Verwendung mehrerer Attribute (Sekundärschlüssel) bleibt erstes Attribut dominant
- Top-k beschränkt Ergebnismenge bzgl. mehrerer Attribute
  - Sortierung nach einem (komplexen) Maß
  - Maße sind oft *unscharf*
  - Maße haben oft mehrere gleichberechtigte Attribute als Input

Top-k in Multimedia DBMS

- Beatles „Red Album“
- Anfrage:
  - Farbe = ‚rot‘ ∧ Name = ‚Beatles‘

Als Standard-Prädikat
Antwort ist (unsortierte) Menge

Als unscharfes Prädikat:
Antwort ist sortierte Liste

Top-k: Benotete Mengen

- Benotete Menge:
  - Menge aus Paaren *(x,g)*
  - *x* ist ein Objekt
  - $g \in [0,1]$ ist eine Note (*grade*)
- Anfrage: Name = ‚Beatles'
  - Antwort: benotete Menge mit $g \in \{0,1\}$
- Anfrage: Farbe = ‚rot'
  - Antwort: benotete Menge mit $g \in [0,1]$

Top-k: Benotete Mengen

- Anfrage:
  - Name = ‚Beatles' ∧ Farbe = ‚rot'
  - Name = ‚Beatles' ∨ Farbe = ‚rot'
- Problem:
  - Maß: Benotung der Objekte in Antwort
- Sei $g_A(x)$ die Note von Objekt x unter Anfrage A.
- Erwünschte Eigenschaften
  - Falls $g \in \{0,1\}$ sollte Standard-Logik gelten.
  - Bewahrung der logischen Äquivalenz
    - $g_{A \wedge A}(x) = g_A(x)$
    - $g_{A \wedge (B \vee C)}(x) = g_{(A \wedge B) \vee (A \wedge C)}(x)$
  - Monotonie: $g_A(x) \leq g_A(y), g_B(x) \leq g_B(y) \Rightarrow g_{A \wedge B}(x) \leq g_{A \wedge B}(y)$

Top-k: Benotete Mengen

- Vorschlag
  - Konjunktionsregel:
    - $g_{A \wedge B}(x) = \min\{g_A(x), g_B(x)\}$
  - Disjunktionsregel:
    - $g_{A \vee B}(x) = \max\{g_A(x), g_B(x)\}$

Lotfi A. Zadeh: Fuzzy Sets. Information and Control 8(3): 338-353, **1965**

Top-k: Benotete Mengen

- $g_{A \wedge B}(x) = \min\{g_A(x), g_B(x)\}$, $g_{A \vee B}(x) = \max\{g_A(x), g_B(x)\}$
- Standardlogik ($g \in \{0,1\}$)
  - $0 \wedge 1 = \min\{0,1\} = 0$
  - $0 \vee 1 = \max\{0,1\} = 1$
- Äquivalenz
  - $g_{A \wedge A}(x) = \min\{g_A(x), g_A(x)\} = g_A(x)$
  - $g_{A \wedge (B \vee C)}(x) = \min\{g_A(x), max\{g_B(x), g_C(x)\}\} = \max\{\min\{g_A(x), g_B(x)\}, \min\{g_A(x), g_C(x)\}\} = g_{(A \wedge B) \vee (A \wedge C)}(x)$
- Monotonie
  - $g_A(x) \leq g_A(y), g_B(x) \leq g_B(y) \Rightarrow g_{A \wedge B}(x) \leq g_{A \wedge B}(y)$
  - $g_A(x) \leq g_A(y), g_B(x) \leq g_B(y) \Rightarrow \min\{g_A(x), g_B(x)\} \leq \min\{g_A(y), g_B(y)\}$

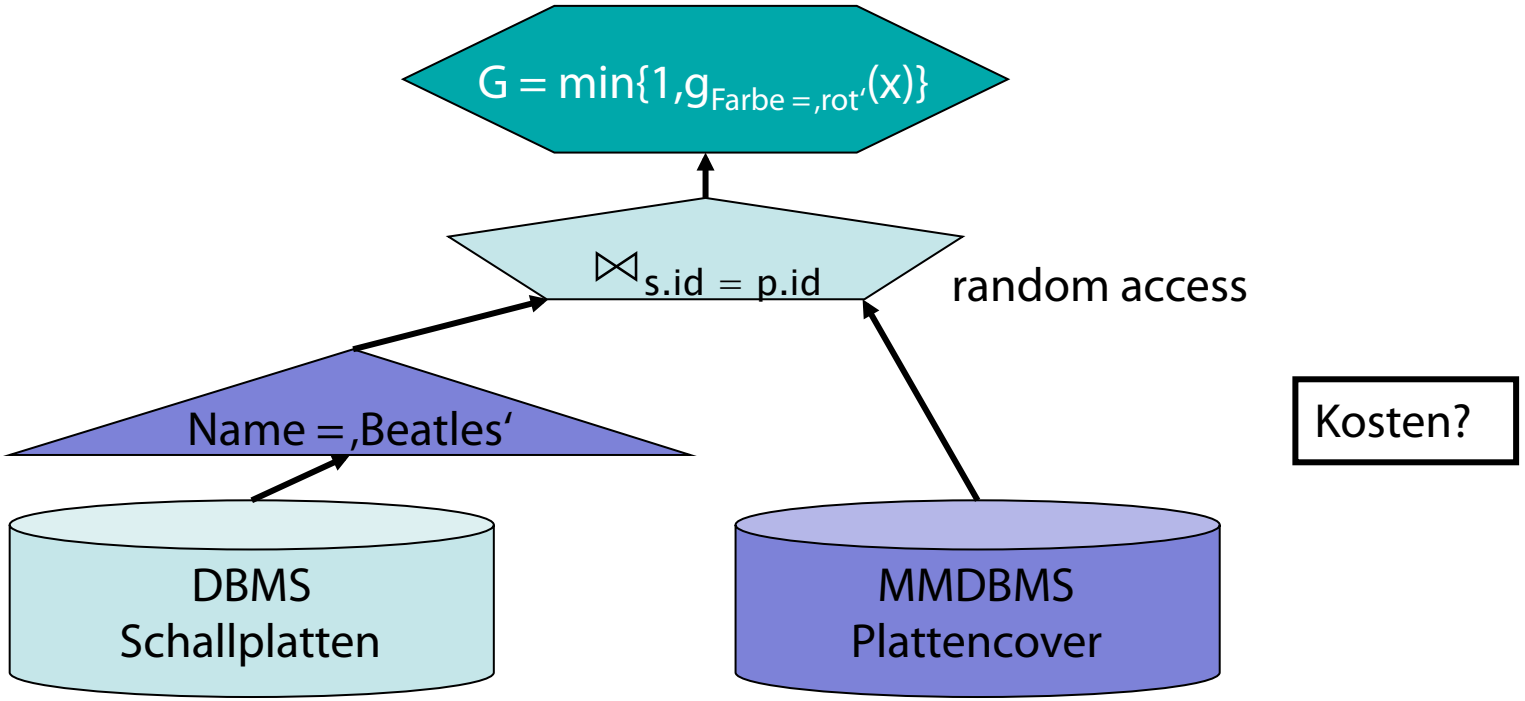
Anfragebeantwortung

- Gegeben: Konjunktive Anfrage
  mit teilweise unscharfen Prädikaten.
- Gesucht: Benotete Menge, so dass k beste Elemente identifiziert werden können

- Zugriffsmodell auf MMDBMS
  - *Sorted access*: Cursor auf sortierte Liste
  - *Random access*: Note eines bestimmten Objekts
- Kostenmodell:
  - Jedes angefragte Objekt kostet 1
- Optimierung:
  - Minimiere Kosten

Top-k: Beispiel

- Anfrage:
  - Name = ‚Beatles' ∧ Farbe = ‚rot'



Top-k: Fagins Algorithmus

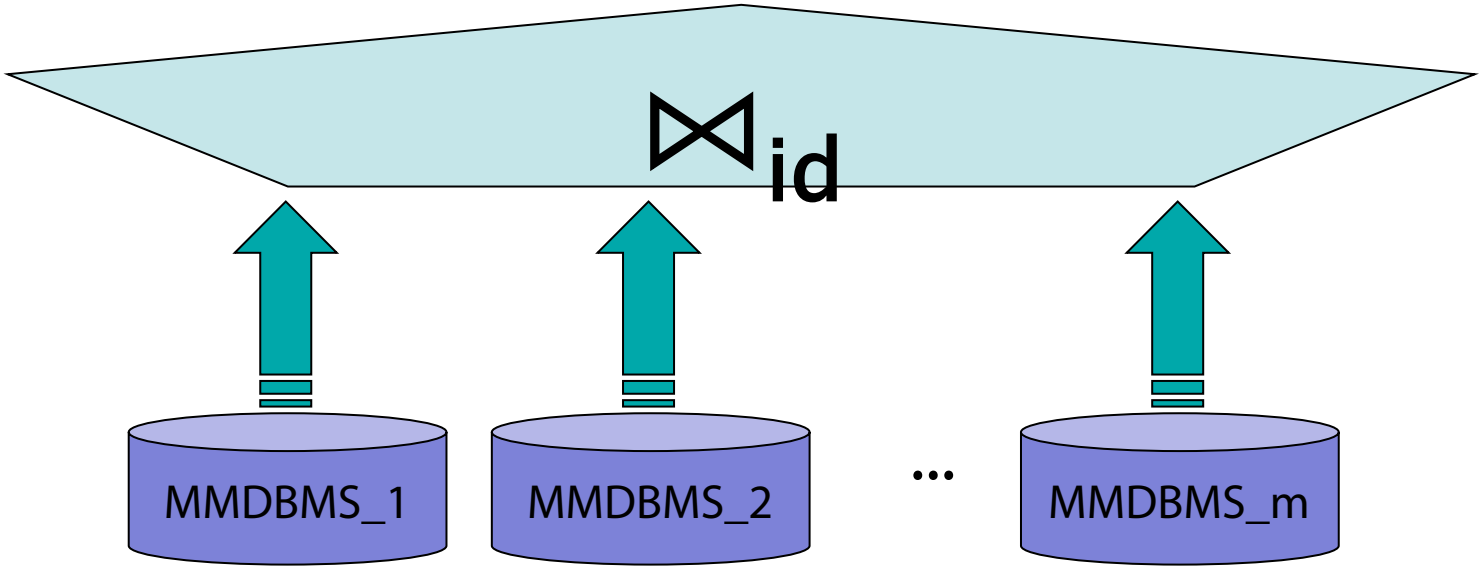
- Allgemeineres Problem:
  - Anfrage statt $A \wedge B$ nun
    $A_1 \wedge A_2 \wedge \ldots \wedge A_m$
  - Für jedes Prädikat eine Quelle.
    - bzw. Zugriffsmöglichkeit durch sorted und random access
- Phase 1: Sorted access
- Phase 2: Random access
- Phase 3: Berechnung und Sortierung

Ronald Fagin. Combining Fuzzy Information from Multiple Systems. PODS-96, 216-226., **1996**

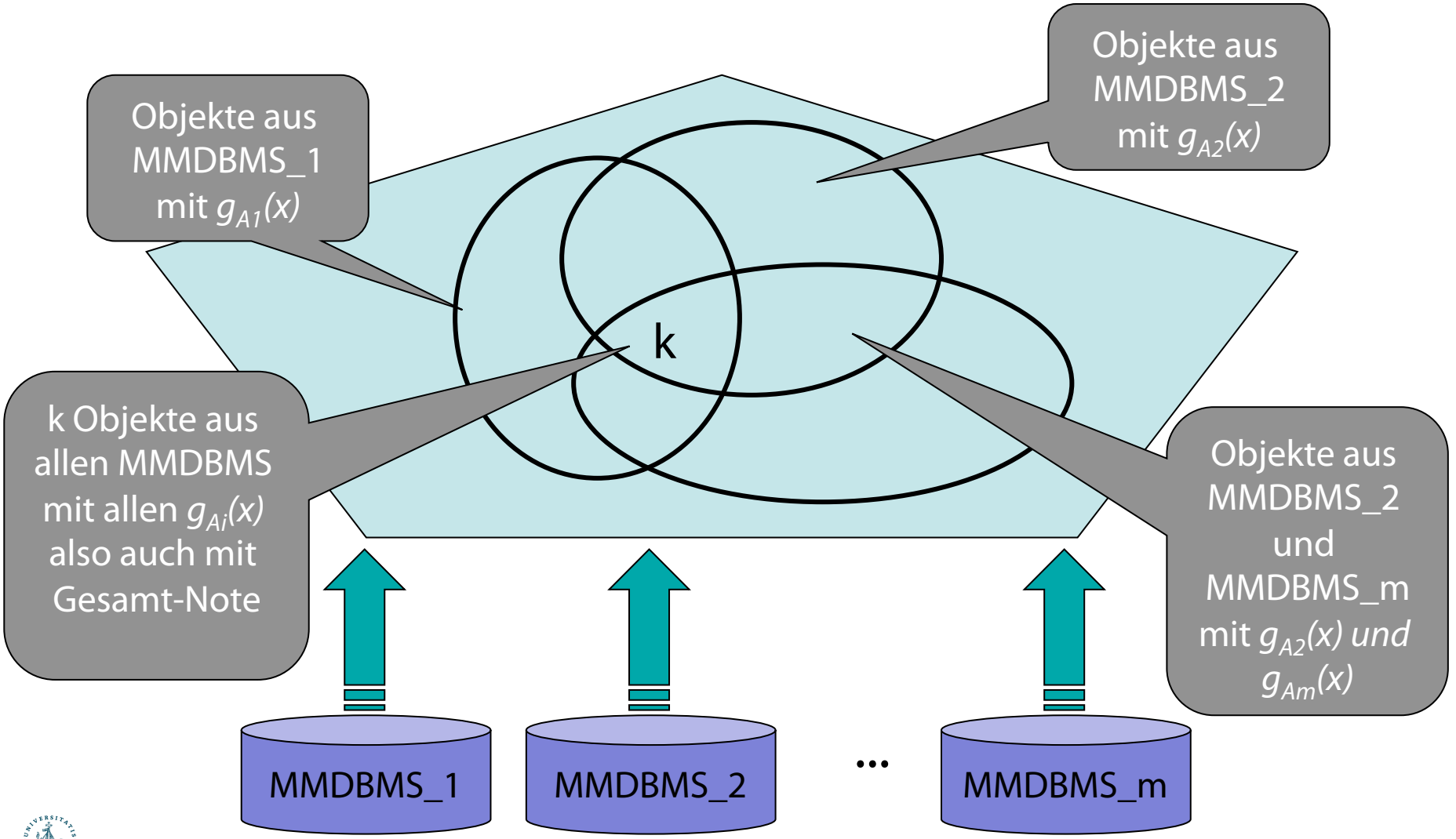
Ronald Fagin: Fuzzy Queries in Multimedia Database Systems. Proc. PODS-98, 1-10, **1998**

Top-k: Fagins Algorithmus

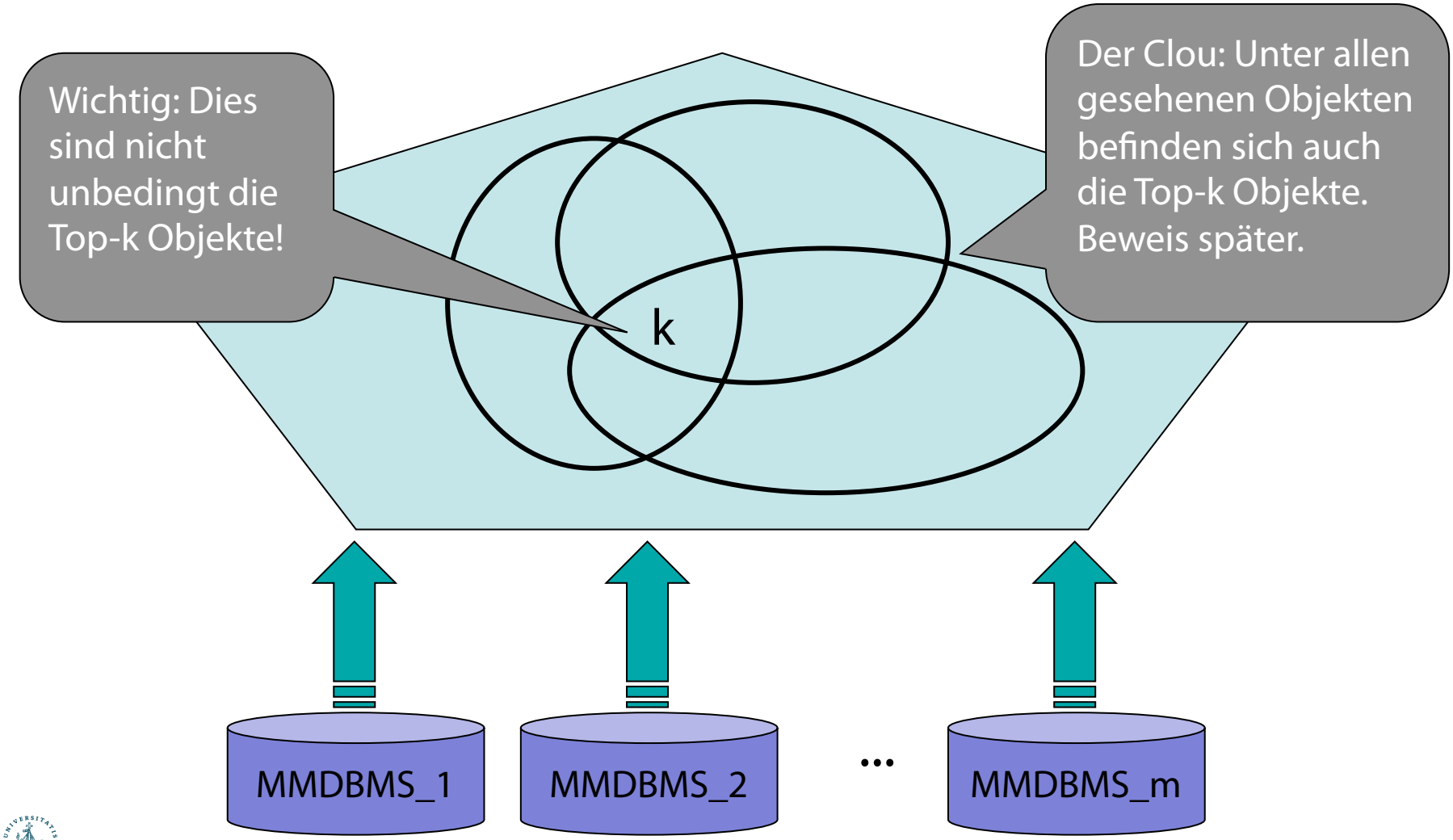
- $A_1 \wedge A_2 \wedge \ldots \wedge A_m$
- Phase 1: Sorted access
  - Für jedes i: Schicke A_i an Quelle i
  - Schreite sukzessive voran, bis Join über alle Teilergebnisse die Größe N hat.



Top-k: Fagins Algorithmus

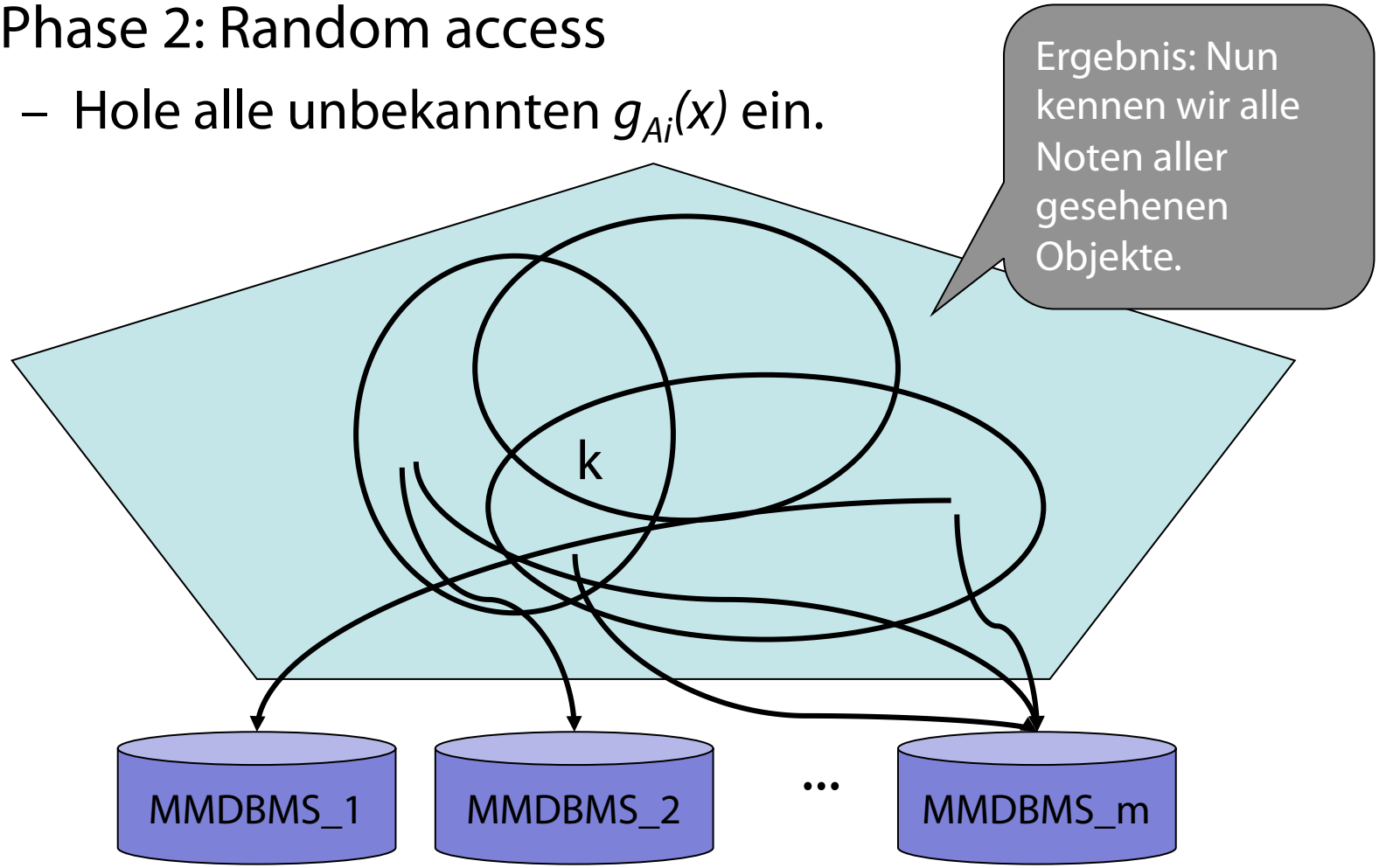


Top-k: Fagins Algorithmus



Top-k: Fagins Algorithmus

- Phase 2: Random access
  - Hole alle unbekannten $g_{Ai}(x)$ ein.



Top-k: Fagins Algorithmus

- Phase 3: Berechnung und Sortierung
  - Berechne für jedes Objekt
    $g_{A1 \wedge A2 \wedge \ldots \wedge Am}(x)$
    $= \min\{g_{A1}(x), g_{A2}(x), \ldots, g_{Am}(x)\}$
  - Sortiere alle Objekte nach $g_{A1 \wedge A2 \wedge \ldots \wedge Am}(x)$
  - Selektierte die höchsten k Objekte.
  - Ausgabe dieser Top-k Objekte.

Fagins Algorithmus – Beispiel

- Anfrage:
  - Form = ‚rund' ∧ Farbe = ‚rot' ∧ Stil = ‚Modern'
  - k = 2

MMDBMS_1

ID	Form	Rundheit
1	oval	0.8
2	achteck	0.6
3	viereck	0.15
4	dreieck	0.1
5	strich	0

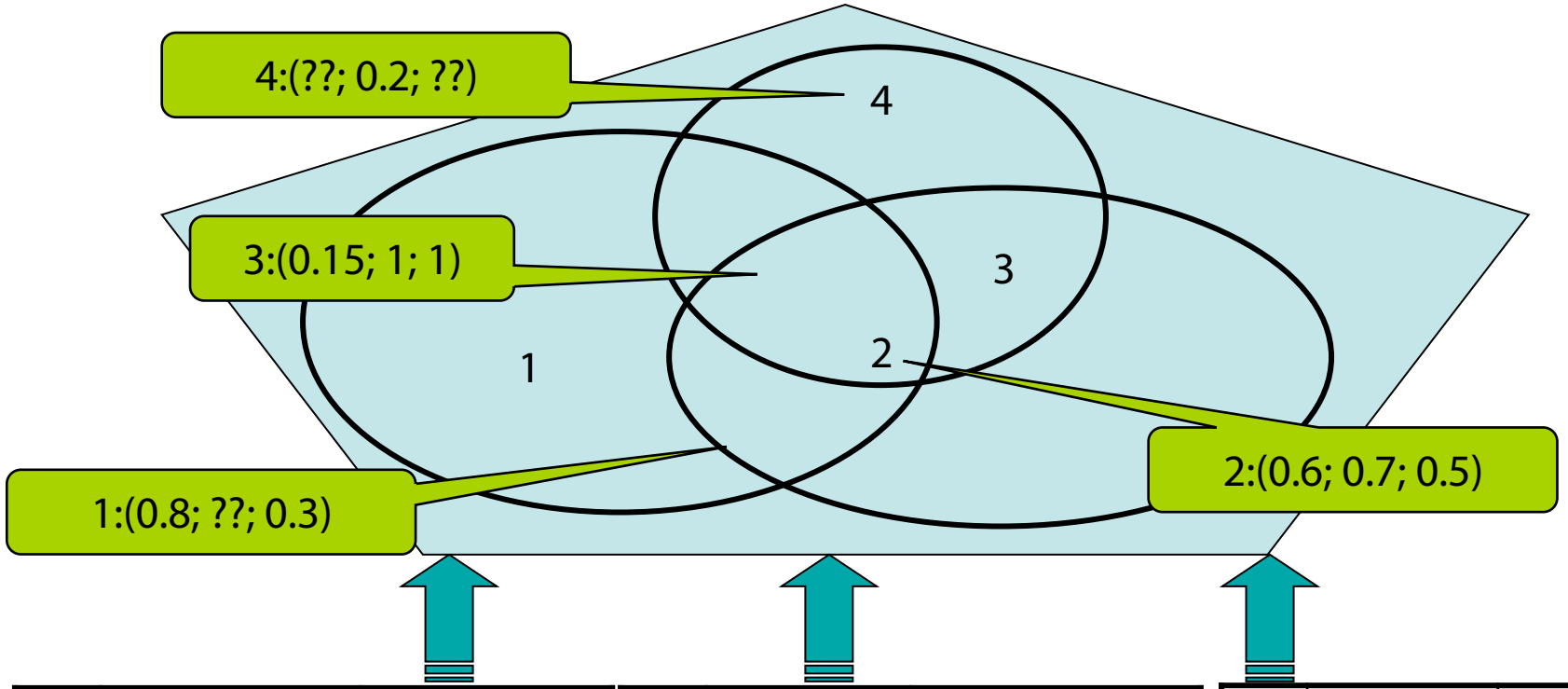
MMDBMS_2

ID	Stil	Modernität
3	modern	1
2	rock	0.7
4	barock	0.2
1	keltisch	0.1
5	uralt	0.01

MMDBMS_3

ID	Farbe	Rotheit
3	rot	1
2	orange	0.5
1	gelb	0.3
4	blau	0.01
5	grün	0

Fagins Algorithmus – Beispiel



ID	Form	Rundheit
1	oval	0.8
2	achteck	0.6
3	viereck	0.15
4	dreieck	0.1
5	strich	0

ID	Stil	Modernität
3	modern	1
2	rock	0.7
4	barock	0.3
1	keltisch	0.2
5	uralt	0.01

ID	Farbe	Rotheit
3	rot	1
2	orange	0.5
1	gelb	0.3
4	blau	0.01
5	grün	0

Top-k: Fagins Algorithmus

- Korrektheit:
  - Fagins Algorithmus findet die Top-N Objekte gemäß $g_A(x)$.
- Beweis:
  - Idee: Wir zeigen für jedes ungesehene Objekt *y*, dass es nicht unter den Top-k sein kann:
  - Notation
    - *x*: gesehene Objekte
    - *y*: ungesehene Objekte
  - Für jedes *x* der Joinmenge nach Phase 1 und jedes Prädikat *Ai* gilt: $g_{Ai}(y) \leq g_{Ai}(x)$.
  - Wegen Monotonie von min{} gilt: $g_{A1 \wedge A2 \wedge \ldots \wedge Am}(y) \leq g_{A1 \wedge A2 \wedge \ldots \wedge Am}(x)$.
  - Es gibt mindestens k solcher Objekte *x* (Abbruch-Kriterium Phase 1).
  - Schlussfolgerung: Es gibt kein *y*, das besser ist als die besten N *x*.

Wichtig: Wir können dies nicht für andere gesehene Objekte zeigen.

Top-k: Fagins Algorithmus

- Aufwand: $O(n^{(m-1)/m}k^{1/m})$ (Beweis: siehe [Fa96])
  - n = DB-Größe; m = Anzahl der DBs
    - Beispiel: 10000 Objekte, 3 Prädikate, Top 10
    - $10.000^{2/3} \times 10^{1/3} = 1.000$
  - Gilt falls A_i unabhängig.
  - Gilt mit beliebig hoher Wahrscheinlichkeit.
    - D.h.: Für jedes $\varepsilon>0$ $\exists$ c, so dass die Wahrscheinlichkeit dass der Aufwand höher ist als angegeben $< \varepsilon$ ist.
- Zum Vergleich: Naiver Algorithmus in O(nm)
  - Im Beispiel: 10.000 x 3 = 30.000

Ronald Fagin. Combining Fuzzy Information from Multiple Systems. PODS-96, 216-226., **1996**

Top-k-Anfragen: Herausforderungen

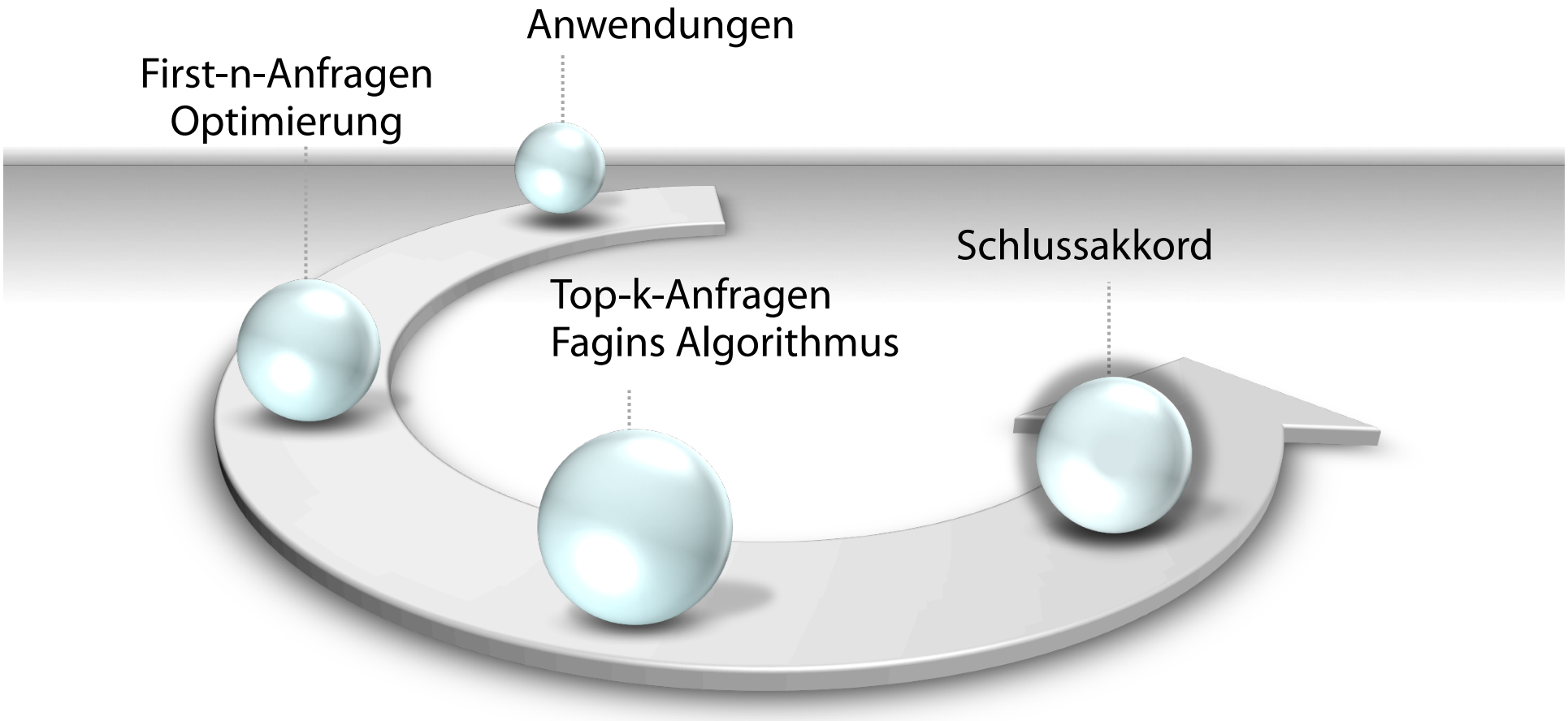
- Beliebige Maße
  - Je nach Nutzer bzw. Anwendung
- Effiziente Ausführung in bestehenden DBMS
  - Unter Ausnutzung vorhandener Datenstrukturen und Metadaten
- Korrektheit und Vollständigkeit

- In der Literatur verfolgter Ansatz:
  Wandele Top-k-Anfragen
  in herkömmliche Anfragen um

Surajit Chaudhuri, Luis Gravano: Evaluating Top-k Selection Queries.
VLDB 1999: 397-410, **1999**

Non-Standard-Datenbanken

First-n und Top-k-Anfragen



Danksagung an Felix Naumann für Material
aus VL Informationsintegration, WS 05/06