Gliederung

DHBW
Duale Hochschule
Baden-Württemberg
Lörrach

1. Die Bedeutung des Datenmanagements
2. Datenbank-Architektur
3. Modellierung und Entwurf von DB-Systemen
4. Relationale Algebra und Normalisierung
5. Definition und Abfrage von Datenbank-Systemen
6. Dateiorganisation und Zugriffsstrukturen
7. **Optimierung von Anfragen**
8. Transaktionen
9. weitere Aspekte der Datenbanken

DHBW
Duale Hochschule
Baden-Württemberg
Lörrach

7.1 Grundprinzipien der Optimierung

2. Logische/Algebraische Optimierung

3. Beispiel einer Optimierung

7.1 Grundprinzipien der Optimierung

Ziel der Optimierung

- Verarbeitung von **möglichst wenigen Seiten** (Datensätze)
- **Selektion möglichst früh** → kleines Zwischenergebnis
- Basisoperationen (wie Selektion, Projektion) **ohne Zwischenspeicherung von Zwischenergebnissen**
- nur „sinnvolle“ Berechnungen: z.B. keine Vereinigung mit sich selbst, keine leeren Zwischenrelationen
- **Zusammenfassen gleicher Teilausdrücke** → Wiederverwendung von Zwischenergebnissen

Phasen einer Optimierung

1. Umformung der Anfrage → **logische Optimierung**
   (wird näher betrachtet)
2. Erzeugung von **Zugriffsplänen** (konkrete Algorithmen) → **physische Optimierung**
   (wird *nicht* näher betrachtet)
3. **Kostenbasierte Auswahl** eines Zugriffsplans aus **statistischen Infos**
   (wird *nicht* näher betrachtet)

7.2. Logische/Algebraische Optimierung

Beispieltabellen

- **Produkt**(ProdNr, Preis, Bezeichnung)
- **Kunde**(KNr, Nachname)
- **Bestellung**(BestNr, ProdNr, KNr, Datum)

Logische (= Algebraische) Optimierung

- basiert auf der **Ersetzung von Termen der relationalen Algebra** durch Algebraäquivalenzen

→ **Ersetzungsregeln:** Umformung von Anfragen

  - Verschiebung von **Operationen**, um **kleinere Zwischenergebnisse** zu erhalten
  - Erkennung von **Redundanzen**

Regel: Entfernen redundanter Operationen (1)

- Beispiel
  - Gegeben sei folgende „Zwischenrelation“ oder Sicht (`current_date` steht hierbei für „heute“):

$$AKT\_PRODUKTE = PRODUKT * \pi_{ProdNr}(\sigma_{Datum=current\_date}(BESTELLUNG))$$

  - Anfrage: die Sicht AKT_PRODUKTE wird mit der Basisrelation PRODUKTE verbunden:

$$\pi_{Preis}(PRODUKT * AKT\_PRODUKTE)$$

Regel: Entfernen redundanter Operationen (2)

- Ersetzung der Sicht: durch Sichtexpansion entsteht

$$\Pi_{Preis}(PRODUKT * AKT\_PRODUKTE) =$$

$$\Pi_{Preis}(PRODUKT * PRODUKT * \Pi_{ProdNr}(\sigma_{Datum=current\_date}(BESTELLUNG)))$$

- Die natürliche Verbundoperation ist idempotent ($R * R = R$):
  → die doppelte Relation PRODUKTkann weggelassen werden

DHBW
Duale Hochschule
Baden-Württemberg
Lörrach

Regel: Verschieben von Selektionen

- Beispiel: Selektion über einen natürlichen Verbund

$$\sigma\sigma_{Preis>100}(BESTELLUNG * PRODUKT)$$

zunächst Berechnung des Verbunds, anschließend Selektion
→ eventuell sehr großes Zwischenergebnis

- günstiger ist folgende Berechnung

$$BESTELLUNG * \sigma\sigma_{Preis>100}(PRODUKT)$$

DHBW
Duale Hochschule
Baden-Württemberg
Lörrach

Regel: Reihenfolge von Verbunden

- Mehrfachverbunde: Reihenfolge der Berechnung beeinflusst Größe der Zwischenergebnisse
- Keine eindeutig Regel anwendbar → Statistikinformationen des Data Dictionary verwenden
- Beispiel: $(KUNDE * PRODUKT) * BESTELLUNG$
- Umformung anhand der Regel: $*$ ist assoziativ und kommutativ, jedoch keine Vorzugsrichtung gegeben:

$$KUNDE * (PRODUKT * BESTELLUNG)$$

DHBW
Duale Hochschule
Baden-Württemberg
Lörrach

Optimierungsregeln (1)

- **KommJoin**: $*$ist kommutativ: $r_1 * r_2 \Leftrightarrow r_2 * r_1$
- **AssozJoin**: $*$ist assoziativ: $(r_1 * r_2) * r_3 \Leftrightarrow r_1 * (r_2 * r_3)$
- **ProjProj**: die äußere Projektion dominiert die innere, falls die äußere nur Spalten der inneren enthält:

$$\pi_{X}(\pi_{Y}(r_1)) \Leftrightarrow \pi_{X}(r_1) \text{ falls } X \subseteq Y$$

- **SelSel**: Selektionen können vertauscht werden:

$$\sigma_{F_1}(\sigma_{F_2}(r_1)) \Leftrightarrow \sigma_{F_1 \wedge F_2}(r_1) \Leftrightarrow \sigma_{F_2}(\sigma_{F_1}(r_1))$$

Optimierungsregeln (2)

- **SelProj:** Selektion und Projektion kommutieren

$$\sigma_F(\pi_X(r_1)) \Leftrightarrow \pi_X(\sigma_F(r_1)) \text{ falls } attr(F) \subseteq X$$

- **SelJoin:** Selektion und Verbund kommutieren

$$\sigma_F(r_1 * r_2) \Leftrightarrow \sigma_F(r_1) * r_2 \text{ falls } attr(F) \subseteq R_1$$

$$\sigma_F(r_1 * r_2) \Leftrightarrow \sigma_{F_1}(r_1) * \sigma_{F_2} r_2 \text{ falls } attr(F_1) \subseteq R_1 \text{ und } attr(F_2) \subseteq R_2$$

$$\sigma_F(r_1 * r_2) \Leftrightarrow \sigma_{F_2}(\sigma_{F_1}(r_1) * r_2) \text{ falls } attr(F_1) \subseteq R_1 \text{ und } attr(F_2) \subseteq R_1 \cup R_2$$

Optimierungsregeln (3)

- **SelUnion**: Kommutieren von Selektion und Vereinigung

$$\sigma_{F}(r_1 \cup r_2) \Leftrightarrow \sigma_{F}(r_1) \cup \sigma_{F}(r_2)$$

- **SelDiff**: Kommutieren von Selektion und Differenzmenge

$$\sigma_{F}(r_1 - r_2) \Leftrightarrow (\sigma_{F} r_1) - \sigma_{F}(r_2)$$

$$\sigma_{F}(r_1 - r_2) \Leftrightarrow \sigma_{F}(r_1) - r_2$$

- **ProjJoin**: Kommutieren von Projektion und Verbund

$$\pi_{XX}(r_1 * r_2) \Leftrightarrow \pi_{XX}\left(\pi_{YY_1}(r_1) * \pi_{YY_2}(r_2)\right) \quad \text{mit } YY_1 = (XX \cap RR_1) \cup (RR_1 \cap RR_2)$$

$$\text{und } YY_2 = (XX \cap RR_2) \cup (RR_1 \cap RR_2)$$

Optimierungsregeln (4)

- **ProjUnion:** Kommutieren von Projektion + Vereinigung

$$\pi_{XX}(r_1 \cup r_2) \Leftrightarrow \pi_{XX}(r_1) \cup \pi_{XX}(r_2)$$

- weitere Regeln wie z.B.

$$r_1 \cup r_1 \Leftrightarrow r_1$$
$$r_1 \cap r_1 \Leftrightarrow r_1$$
$$r_1 * r_1 \Leftrightarrow r_1$$
$$r_1 - r_1 \Leftrightarrow \emptyset$$

$$r_1 \cup \emptyset \Leftrightarrow r_1$$
$$r_1 \cap \emptyset \Leftrightarrow \emptyset$$
$$r_1 * \emptyset \Leftrightarrow \emptyset$$
$$r_1 - \emptyset \Leftrightarrow r_1$$
$$\emptyset - r_1 \Leftrightarrow \emptyset$$

Ein einfacher Optimierungsalgorithmus

- liefert bereits brauchbare Ergebnisse
- Schritte
  - Auflösung komplexer Selektionsprädikate: Regel **SelSel**
  - Selektionen möglichst früh anwenden: Regeln **SelJoin**, **SelProj**, **SelUnion** und **SelDiff**
  - Projektionen möglichst früh anwenden: Regeln **ProjProj**, **ProjJoin** und **ProjUnion**

Beispiel (1)

- Gegeben sei folgende unoptimierte Anfrage

$$\Pi_{BestNr,KNr}(\sigma\sigma_{Datum>`18.2.15` \wedge Bezeichnung=`ArabicaBlack`}$$
$$\left(\Pi_{BestNr,KNr,Datum,Bezeichnung} \quad ((PRODUKT) * (BESTELLUNG) * (KUNDE))\right)$$

es gilt: Datum ist Attribut von Bestellung
Bezeichnung ist Attribut von Produkt

Beispiel (2)

- algebraischen Umformungsregeln
  1. Regel **SelSel:** → Aufteilung in zwei Bedingungen
  2. Selektionen nach innen verschieben:
     - Regel **SelProj**: $\sigma\sigma_{Datum>`18.2.15`}$ unter $\pi\pi_{BestNr,KNr,Datum,Bezeichnung}$
     - Regel **SelProj**: $\sigma\sigma_{Bezeichnung=`AAAAAAAAAAAAAAAAAAAA`}$ unter $\pi\pi_{BestNr,KNr,Datum,Bezeichnung}$
     - Regel **SelJoin**: $\sigma\sigma_{Datum>`18.2.15`}$ vor BESTELLUNG
     - Regel **SelJoin**: $\sigma\sigma_{Bezeichnung=`AAAAAAAAAAAAAAAAAAAA`}$ vor PRODUKT
  3. Regel **ProjProj** erlaubt die Zusammenfassung der beiden Projektionen

Beispiel (3)

- Ergebnis der Umformungen

$$\Pi_{BestNr,KNr}\left(\sigma_{Bezeichnung=`ArabicaBlack`}(Produkt)\right)$$

7.3 Beispiel einer Optimierung

DHBW
Duale Hochschule
Baden-Württemberg
Lörrach

Beispiel (1)

- Natürlicher Verbund über das Schlüsselattribut `KNr`, sowie Projektion und Selektion

```
SELECT KUNDE.KNr, KUNDE.Nachname

FROM KUNDE, BESTELLUNG

WHERE KUNDE.KNr = BESTELLUNG.KNr AND

      BESTELLUNG.Datum = date('22-NOV-15')
```

DHBW
Duale Hochschule
Baden-Württemberg
Lörrach

Beispiel (2)

- in der Relation `KUNDE` seien 100 Datensätze gespeichert, auf eine Seite bzw. einen Block passen dabei 5 Datensätze
- in der Relation `BESTELLUNG` seien 10.000 Datensätze gespeichert, auf eine Seite passen jeweils 10 Datensätze
- 3 Datensätze des Kreuzprodukts (KUNDE) × (BESTELLUNG) passen auf eine Seite.
- 50 Datensätze von `(KUNDE.KNr,KUNDE.Nachname)` passen auf 1 Seite
- Ca. 50 Bestellungen pro Tag werden aufgegeben

Beispiel (3)

- Vereinfachungen:
  - für jeden Ausführungsschritt wird eine Zwischenrelation mit dem Ergebnis angelegt
  - der Puffer für jede Relation hat die Größe 1 (also eine Seite bzw. ein Block)
  - auf Seiten werden nur ganze Datensätze abgespeichert
  - Kosten für die Berechnungen in der CPU können ignoriert werden.
  - l / s: Kosten für lesende / schreibende Seitenzugriffe

DHBW
Duale Hochschule
Baden-Württemberg
Lörrach

1. Variante: Direkte Auswertung (1)

- Vorgehen
  - **Kombination aller Datensätze** der ersten Relation mit allen Datensätzen der zweiten Relation
  - Test auf **Zugehörigkeit zum natürlichen Verbund**
  - Test der **Selektionsbedingung**
  - Berechnung der **Ergebnisprojektion**
- Interne Realisierung: **geschachtelte Schleifen**

1. Variante: Direkte Auswertung (2)

```
FROM KUNDE, BESTELLUNG
```

- *l*: (100/5 • 10.000/10) = 20.000 Seitenzugriffe
  Datensätze beider Relationen werden seitenweise eingelesen;
  aufgrund der Puffergröße muss für jede Seite der `KUNDE`-Relation
  jeweils jede Seite der `BESTELLUNG`-Relation erneut eingelesen werden

- *s*: (100 • 10.000)/3 = 333.333 (ca.) Seitenzugriffe
  alle Seiten des Zwischenergebnisses werden gespeichert
  (3 Datensätze des Kreuzprodukts (KUNDE) × (BESTELLUNG) passen auf
  eine Seite)

DHBW
Duale Hochschule
Baden-Württemberg
Lörrach

1. Variante: Direkte Auswertung (3)

```
WHERE KUNDE.KNr = BESTELLUNG.KNr AND
BESTELLUNG.Datum = date('22-NOV-15')
```

- bei der Selektion muss erst das Zwischenergebnis des vorigen Schrittes gelesen werden; geschrieben werden alle Datensätze, die das Prädikat erfüllen, also insgesamt 50, die auf 17 Seiten passen:

  I: 1000000/3 = 333.333 (ca.) Seitenzugriffe

  s: 50/3 = 17 (ca.) Seitenzugriffe

1. Variante: Direkte Auswertung (4)

```
SELECT KUNDE.KNr, KUNDE.Nachname
```

- bei der abschließenden Projektion müssen alle 17 Zwischenergebnisseiten gelesen und die eine Seite mit dem Endergebnis zurückgeschrieben werden

  l: 50/3 ≈ 17 Seitenzugriffe

  s: 50/50 = 1 Seitenzugriff

- Insgesamt werden ca. 353.350 Seitenzugriffe lesend und ca. 333.018 Seiten zur Zwischenspeicherung benötigt

2. Variante: Optimierte Auswertung (1)

- Direkte Auswertung: **sinnlose Berechnung des vollen Kreuzprodukts** beider Relationen
- Alternativer Ansatz mit einer Unterabfrage für die Kunden, die eine Bestellung haben, welche die **Selektionsbedingung** `Datum = date('22-NOV-15')` erfüllt.

```
SELECT KNr, Nachname
FROM KUNDE
WHERE KNr IN (SELECT KNr FROM BESTELLUNG
                    WHERE Datum = date('22-NOV-15'))
```

2. Variante: Optimierte Auswertung (2)

```
WHERE Datum = date('22-NOV-15')
```

- hier erfolgt eine **Vorselektion**, um die Mengenzugehörigkeit (**IN**…) mit einer kleineren Relation durchführen zu können; alle Seiten werden gelesen, und die 50 Bestelleinträge des Tages geschrieben

  I: 10.000/10 = 1.000 Seitenzugriffe

  s: 50/10 = 5 Seitenzugriffe

- (die Kosten für das Schreiben des Ergebnisses der Unterabfrage sind aufgrund der Projektion eigentlich geringer)

2. Variante: Optimierte Auswertung (3)

```
WHERE KNr IN (SELECT KNr FROM BESTELLUNG …
```

– die Ermittlung der Kunden, die zu der Menge aus der Unterabfrage gehören, ergibt:

I: 100/5 * 50/10 = 100 Seitenzugriffe

s: 50/3 ≈ 17 Seitenzugriffe

2. Variante: Optimierte Auswertung (4)

```
SELECT KNr, Nachname
```

- abschließend wird die Ergebnisprojektion berechnet:
  I: 17 Seitenzugriffe
  s: 1 Seitenzugriffe

Insgesamt werden ca. 1.140 Seitenzugriffe benötigt

DHBW
Duale Hochschule
Baden-Württemberg
Lörrach

3. Variante: Indexausnutzung (1)

- **interne Zugriffsstrukturen: Senkung** der Ausführungszeit um eine weitere Größenordnung
- Gegeben sind **zwei Indexe**: `I(BESTELLUNG(Datum))` und `I(KUNDE(KNr))`, die **baumstrukturiert** sind, so dass eine **Sortierreihenfolge** ermöglicht wird

3. Variante: Indexausnutzung (2)

```
WHERE Datum = date('22-NOV-15')
```

– die Vorselektion für die Verbundberechnung erfolgt über den Index *I*(BESTELLUNG(Datum)); der tatsächliche Aufwand hängt von der Organisation des Indexes ab (Primär- versus Sekundärindex)

l: minimal 5, falls alle Bestellungen des Tages optimal zusammen liegen; maximal 50, wenn beim Zugriff über Sekundärindex alle 50 Bestellungen auf verschiedenen Seiten gespeichert sind

s: 50/10 = 5 Seitenzugriffe

3. Variante: Indexausnutzung (3)

- Sortierung nach `KNr`
  - als Vorbereitung wird das Zwischenergebnis sortiert

    $l + s$: 5 • log5 = 15 (ca.) Seitenzugriffe

    (Hinweis: log5 zur Basis 2 ≈ 2,3; aufgerundet 3)

3. Variante: Indexausnutzung (3)

WHERE KNr **IN** (**SELECT** KNr **FROM** BESTELLUNG …

- der Verbund wird als Join über den Index I(KUNDE(KNr)) und der sortierten Relation r_2 realisiert; der Index I(KUNDE(KNr)) ist ein baumstrukturierter Primärindex über KNr und erlaubt daher ein effizientes Auslesen in Sortierreihenfolge

  l: 100/5 + 5 = 25 Seitenzugriffe

  s: 50/3 = 17 Seitenzugriffe

3. Variante: Indexausnutzung (4)

```
SELECT KNr, Nachname
```

- Berechnung der Ergebnisprojektion ergibt:
  I: 17 Seitenzugriffe
  s: 1 Seitenzugriffe

Insgesamt werden maximal ca. 130 und minimal ca. 85 Seitenzugriffe benötigt

Durch weitere Optimierungsverfahren, kann man noch weitere Zugriffe sparen