Übung

Data Warehouses

Wintersemester 2000/2001

Datenanalyse

Übungsfälle:

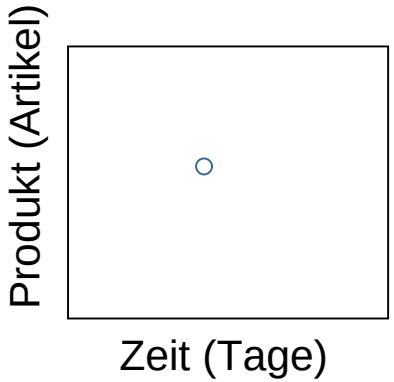
Multidimensionale Anfragen:

- Punktanfrage: Saldo der Sparbücher in der 40. Kalenderwoche des Jahres 2000
- Partial-Match-Anfrage: Gesamtsaldo in der 40. Kalenderwoche des Jahres 2000
- Partielle Bereichsanfrage: Umsätze der 41. und 42. Woche des Jahres 2000
- Bereichsanfrage: Umsätze der Sparprodukte in der 41. und 42. Woche des Jahres 2000
- Slicing: Umsätze aller Produkte in der 41. und 42. Woche des Jahres 2000 der Individualkunden der Direktion 1
- Dicing: Umsätze aller Sparprodukte in der 41. und 42. Woche des Jahres 2000 der Individualkunden der Direktion 1

Vergleich der Anfragepläne der Dicing-Anfrage:

- Ohne weitere Optimierung
- Mit B-Baum-Indizes auf den Join-Attributen
- Mit Bitmap-Indizes auf den Join-Attributen
- Bei Vorhandensein einer Materialized View
    - View über die aggregierten benötigten Kennzahlen und die gruppierten Schlüssel anlegen
- Bei Partitionierung der Faktentabelle
    - Neue (partitionierte) Tabelle mit den benötigten Attributen anlegen (Nachträgliche Partitionierung ist nicht möglich)
    - Werte der bestehenden Tabelle kopieren

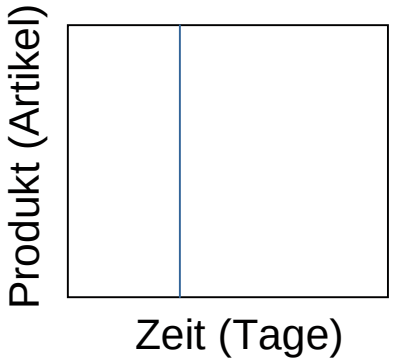
Punktanfrage



Saldo der Sparbücher in der 40. Kalenderwoche des Jahres 2000

```
SELECT SUM(saldo)
FROM fakten_kundengeschaeft, zeit, produkt, produktklasse
WHERE fakten_kundengeschaeft.tag = zeit.tag AND
    fakten_kundengeschaeft.produkt_id = produkt.produkt_id AND
    produkt.kennzeichen_produktgruppe
       = produktklasse.kennzeichen_produktgruppe AND
    produktklasse.bezeichnung_produktgruppe = 'Sparbuch' AND
    zeit.woche = 200040;
```

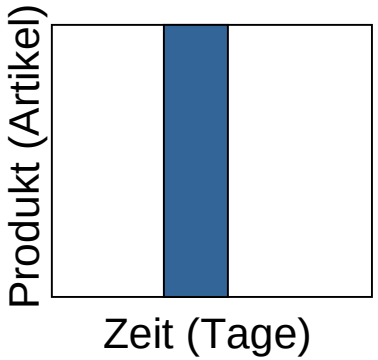
Partial-Match-Anfrage



Gesamtsaldo in der 40. Kalenderwoche des Jahres 2000

```
SELECT SUM(saldo)
FROM fakten_kundengeschaeft, zeit
WHERE
    fakten_kundengeschaeft.tag = zeit.tag AND
    zeit.woche = 200040;
```

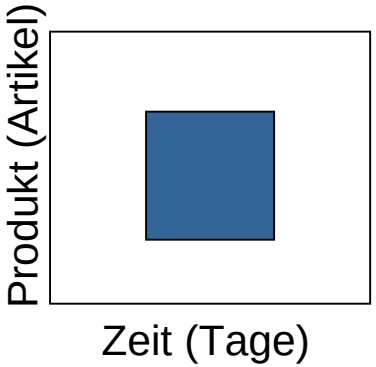
Partielle Bereichsanfrage (*partial range query*)



Umsätze der 41. und 42. Woche des Jahres 2000

```
SELECT SUM(umsaetze)
FROM fakten_kundengeschaeft, zeit
WHERE
    fakten_kundengeschaeft.tag = zeit.tag AND
    zeit.woche BETWEEN 200041 AND 200042;
```

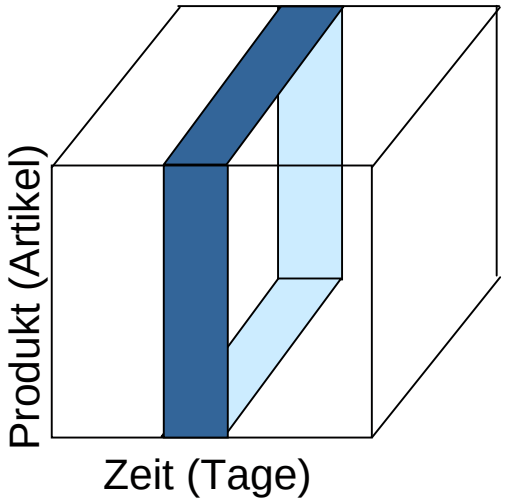
Bereichsanfrage (*range query*)



Umsätze der Sparprodukte in der 41. und 42. Woche des Jahres 2000

```
SELECT SUM(umsaetze)
FROM fakten_kundengeschaeft, zeit, produkt, produktklasse
WHERE fakten_kundengeschaeft.tag = zeit.tag AND
    fakten_kundengeschaeft.produkt_id = produkt.produkt_id AND
    produkt.kennzeichen_produktgruppe =
       produktklasse.kennzeichen_produktgruppe AND
    produktklasse.bezeichnung_produktgruppe = 'Sparbuch' AND
    zeit.woche BETWEEN 200041 AND 200042;
```

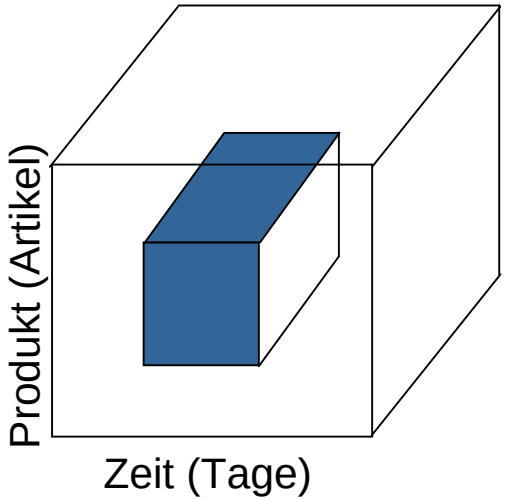
Mehrdimensionale partielle Bereichsanfrage (*slicing*)



Umsätze aller Produkte in der 41. und 42. Woche des Jahres 2000 der Individualkunden der Direktion 1

```
SELECT SUM(umsaetze)
FROM fakten_kundengeschaeft, zeit, kundenart,
organisation
WHERE
    fakten_kundengeschaeft.tag = zeit.tag AND
    fakten_kundengeschaeft.kundenart_id =
         kundenart.kundenart_id AND
    fakten_kundengeschaeft.beraterschluessel =
         organisation.beraterschluessel AND
    zeit.woche BETWEEN 200041 AND 200042 AND
    kundenart.kundengruppe = 'Individualkunde' AND
    organisation.direktion = 'Direktion 1';
```

Mehrdimensionale Bereichsanfrage (*dicing*)



Umsätze aller Sparprodukte in der 41. und 42. Woche des Jahres 2000 der Individualkunden der Direktion 1

```
SELECT SUM(umsaetze)
FROM fakten_kundengeschaeft, zeit, produkt,   produktklasse,
kundenart, organisation
WHERE fakten_kundengeschaeft.tag = zeit.tag AND
    fakten_kundengeschaeft.produkt_id =
        produkt.produkt_id AND
    fakten_kundengeschaeft.kundenart_id =
        kundenart.kundenart_id AND
    fakten_kundengeschaeft.beraterschluessel =
        organisation.beraterschluessel AND
    produkt.kennzeichen_produktgruppe =
        produktklasse.kennzeichen_produktgruppe AND
    produktklasse.bezeichnung_produktgruppe = 'Sparbuch' AND
    zeit.woche BETWEEN 200041 AND 200042 AND
    kundenart.kundengruppe = 'Individualkunde' AND
    organisation.direktion = 'Direktion 1';
```

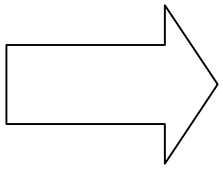
Anfrageplan der Bereichsanfrage ohne Optimierung

```
SELECT STATEMENT    Cost = 7
  SORT AGGREGATE
    NESTED LOOPS
      MERGE JOIN CARTESIAN
        MERGE JOIN CARTESIAN
          MERGE JOIN CARTESIAN
            NESTED LOOPS
              TABLE ACCESS FULL PRODUKTKLASSE
              TABLE ACCESS FULL PRODUKT
            SORT JOIN
              TABLE ACCESS FULL ZEIT
          SORT JOIN
            TABLE ACCESS FULL KUNDENART
        SORT JOIN
          TABLE ACCESS FULL ORGANISATION
      TABLE ACCESS BY INDEX ROWID FAKTEN_KUNDENGESCHAEFT
        INDEX RANGE SCAN TAG_KUNDENGESCHAEFT_FK
```

Anfrageplan der Bereichsanfrage mit Indizes auf den Join-Attributen

Anlegen der Indizes

```
CREATE INDEX TAG_KG on FAKTEN_KUNDENGESCHAEFT (TAG)
CREATE INDEX OE_KG on FAKTEN_KUNDENGESCHAEFT
(BERATERSCHLUESSEL)
CREATE INDEX PRODUKT_KG on FAKTEN_KUNDENGESCHAEFT
(PRODUKT_ID)
CREATE INDEX KUNDENART_KG on FAKTEN_KUNDENGESCHAEFT
(KUNDENART_ID)
```



Hat keine Auswirkungen!
Die Indizes werden nicht benutzt.

Bitmap-Indizes

Anlegen der Indizes:

```
CREATE BITMAP INDEX BMP1 on FAKTEN_KUNDENGESCHAEFT (TAG)
CREATE BITMAP INDEX BMP2 on FAKTEN_KUNDENGESCHAEFT (BERATERSCHLUESSEL)
CREATE BITMAP INDEX BMP3 on FAKTEN_KUNDENGESCHAEFT (PRODUKT_ID)
CREATE BITMAP INDEX BMP4 on FAKTEN_KUNDENGESCHAEFT (KUNDENART_ID)
```

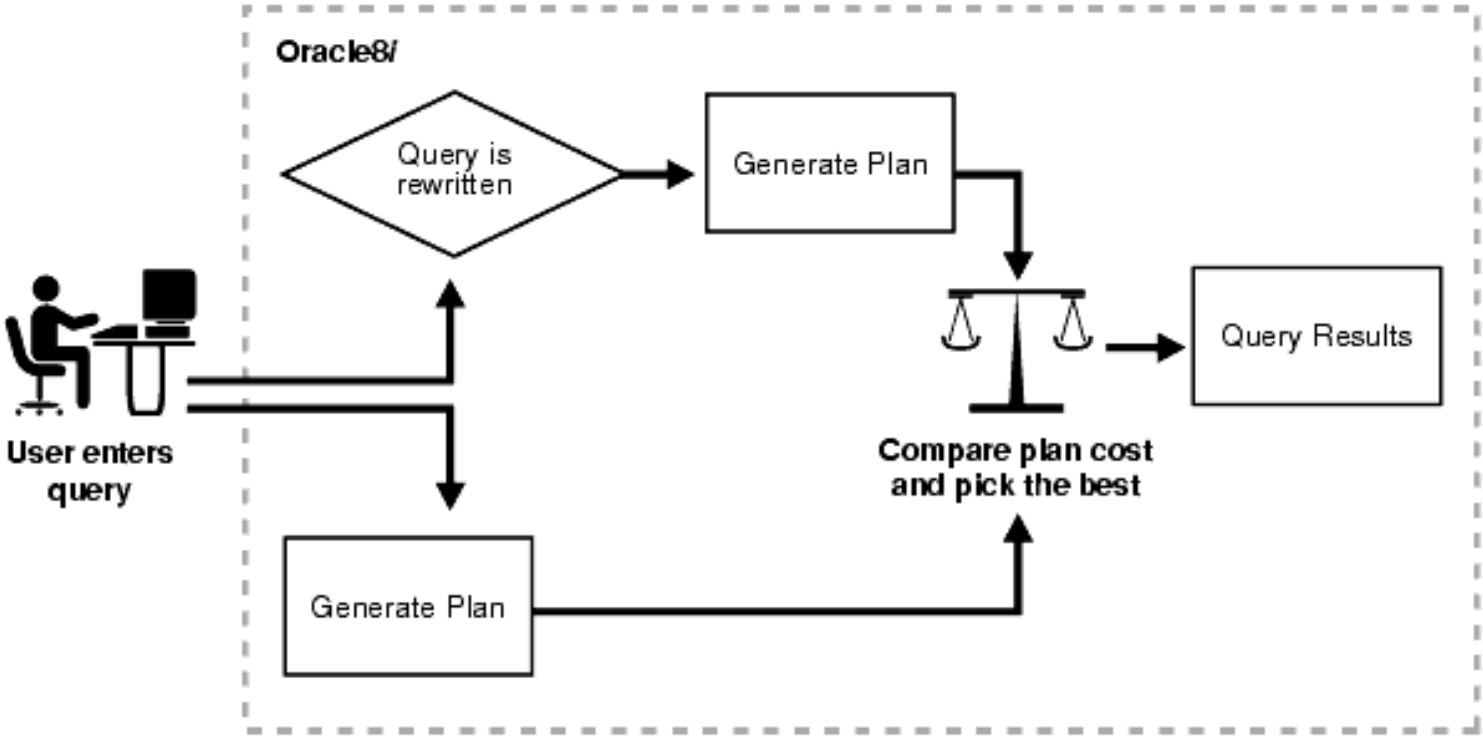
Hinweis auf die Benutzung der Indizes:

```
/*+INDEX_COMBINE(fakten_kundengeschaeft,BMP1,BMP2,BMP3,BMP4)*/
```

Anfrageplan der Bereichsanfrage mit Bitmap Indizes

```
SELECT STATEMENT   Cost = 68 ???
  SORT AGGREGATE
    NESTED LOOPS
      NESTED LOOPS
        NESTED LOOPS
          NESTED LOOPS
            HASH JOIN
              TABLE ACCESS FULL KUNDENART
              NESTED LOOPS
                TABLE ACCESS FULL ZEIT
                TABLE ACCESS BY INDEX ROWID FAKTEN_KUNDENGESCHAEFT
                  BITMAP CONVERSION TO ROWIDS
                    BITMAP INDEX SINGLE VALUE BMP1
            TABLE ACCESS BY INDEX ROWID ORGANISATION
              INDEX UNIQUE SCAN PK_ORGANISATION
          TABLE ACCESS BY INDEX ROWID PRODUKT
            INDEX UNIQUE SCAN PK_PRODUKT
        TABLE ACCESS BY INDEX ROWID PRODUKTKLASSE
          INDEX UNIQUE SCAN PK_PRODUKTKLASSE
```

Query Rewriting



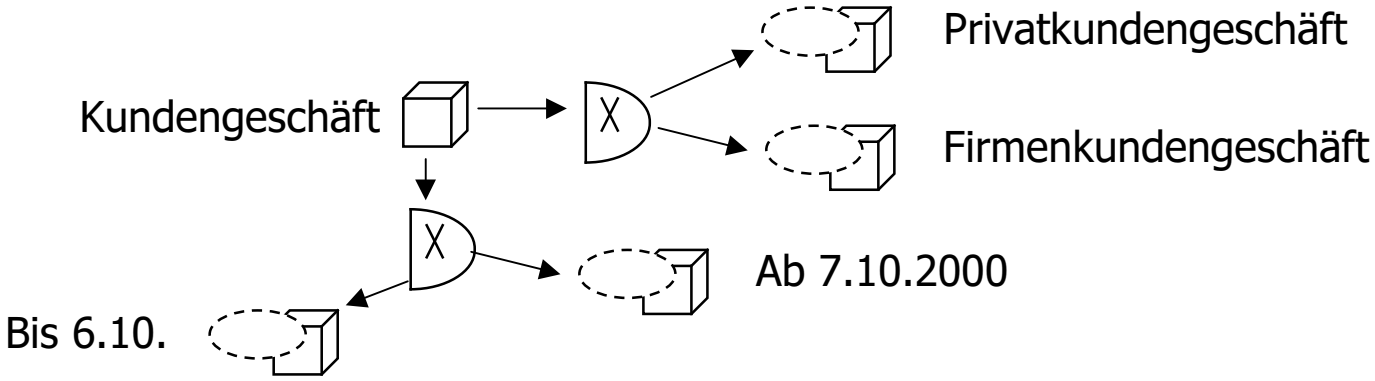
Anlegen einer Materialized View

```
CREATE MATERIALIZED VIEW umsatz_mv
PCTFREE 0 TABLESPACE datawarehouse
STORAGE (INITIAL 100k NEXT 100k PCTINCREASE 0)
PARALLEL
BUILD (IMMEDIATE |  DEFERRED)
REFRESH (COMPLETE | FAST | FORCE | NEVER) ON (COMMIT | DEMAND)
ENABLE QUERY REWRITE
AS SELECT
zeit.woche,produktklasse.bezeichnung_produktgruppe,kundenart.kundengruppe,organisation.direktion,
    SUM(umsaetze) AS umsaetze
FROM fakten_kundengeschaeft, zeit, produkt, produktklasse, kundenart, organisation
WHERE
    fakten_kundengeschaeft.produkt_id = produkt.produkt_id AND
    fakten_kundengeschaeft.tag = zeit.tag AND
    fakten_kundengeschaeft.kundenart_id = kundenart.kundenart_id AND
    fakten_kundengeschaeft.beraterschluessel = organisation.beraterschluessel AND
    produkt.kennzeichen_produktgruppe = produktklasse.kennzeichen_produktgruppe
GROUP BY
    zeit.woche, produktklasse.bezeichnung_produktgruppe,
    kundenart.kundengruppe, organisation.direktion;
```

Anfrageplan nach Query Rewriting

```
SELECT STATEMENT    Cost = 3
  SORT AGGREGATE
    TABLE ACCESS BY INDEX ROWID UMSATZ_MV
      INDEX RANGE SCAN I_SNAP$_UMSATZ_MV
```

Partitionierung der Faktentabelle




```
CREATE TABLE FAKTEN_KUNDENGESCHAEFT
[...]
  PARTITION BY RANGE(tag)
  SUBPARTITION BY HASH(kundenart) SUBPARTITIONS 2
  STORE IN(DATA_DW1, DATA_DW2)
   ( PARTITION p1 VALUES LESS THAN(TO_DATE('07-10-2000','DD-MM-YYYY')),
     PARTITION p2 VALUES LESS THAN(MAXVALUE));
```

Anfrageplan der Bereichsanfrage bei Partitionierung

```
SELECT STATEMENT    Cost = 6
  SORT AGGREGATE
    NESTED LOOPS
      NESTED LOOPS
        NESTED LOOPS
          NESTED LOOPS
            NESTED LOOPS
              PARTITION RANGE ALL
                PARTITION HASH ALL
                  TABLE ACCESS FULL P_FAKTEN_KUNDENGESCHAEFT
              TABLE ACCESS BY INDEX ROWID ZEIT
                INDEX UNIQUE SCAN PK_ZEIT
            TABLE ACCESS BY INDEX ROWID KUNDENART
              INDEX UNIQUE SCAN PK_KUNDENART
          TABLE ACCESS BY INDEX ROWID ORGANISATION
            INDEX UNIQUE SCAN PK_ORGANISATION
        TABLE ACCESS BY INDEX ROWID PRODUKT
          INDEX UNIQUE SCAN PK_PRODUKT
      TABLE ACCESS BY INDEX ROWID PRODUKTKLASSE
        INDEX UNIQUE SCAN PK_PRODUKTKLASSE
```

Bei Einschränkung des Kundentyps

```
SELECT STATEMENT    Cost = 6
  SORT AGGREGATE
    NESTED LOOPS
      NESTED LOOPS
        NESTED LOOPS
          NESTED LOOPS
            NESTED LOOPS
              PARTITION RANGE ALL
                TABLE ACCESS FULL P_FAKTEN_KUNDENGESCHAEFT
              TABLE ACCESS BY INDEX ROWID ZEIT
                INDEX UNIQUE SCAN PK_ZEIT
            TABLE ACCESS BY INDEX ROWID KUNDENART
              INDEX UNIQUE SCAN PK_KUNDENART
          TABLE ACCESS BY INDEX ROWID ORGANISATION
            INDEX UNIQUE SCAN PK_ORGANISATION
        TABLE ACCESS BY INDEX ROWID PRODUKT
          INDEX UNIQUE SCAN PK_PRODUKT
      TABLE ACCESS BY INDEX ROWID PRODUKTKLASSE
        INDEX UNIQUE SCAN PK_PRODUKTKLASSE
```