

DWH outline

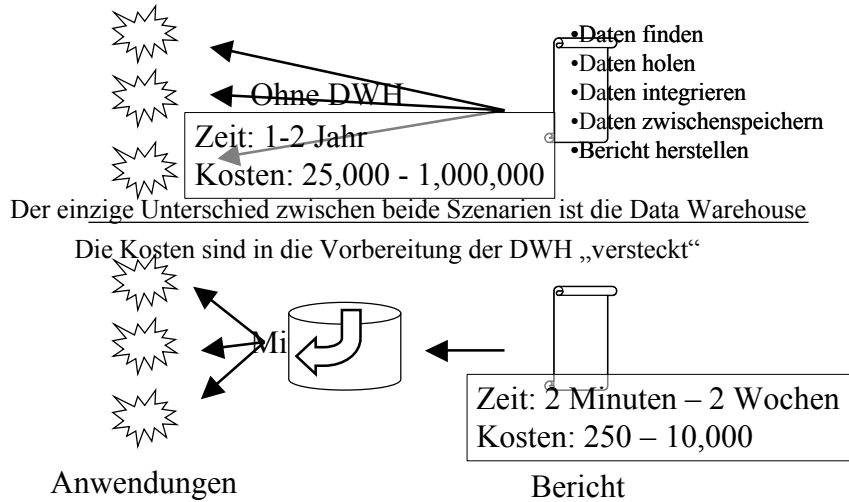
- Einführung: Datenintegration, OLAP vs. OLTP
- Gründe für die Benutzung einer Data Warehouse (DWH):
 - - Kosten des Zugriffs zur Information
 - - Vereinheitlichung der Information
 - - Geschichtliche Information
- Drei komplementäre Trends:
 - - Data Warehousing
 - - Data Mining
 - - OLAP: On-line Analytical Processing
- Data Warehousing Merkmale:
 - - Multidimensionales Datenmodell
 - - Implementierungen: MOLAP und ROLAP
 - - Dimensionshierarchien

DWH outline

- Merkmale zu OLAP Abfragen:
 - - Star Schema
 - - Aggregatfunktionen, wie z.B. Aufsummierung
 - - Roll-up, drill-down, pivoting, slicing, dicing
 - - OLAP Abfragen mit SQL: CUBE, Windowing
- Indizes und Views
 - - Bitmap index
 - - View materialization
- Interaktive Abfragen
 - - Top N
 - - Online Aggregation

Wieso eine DWH?

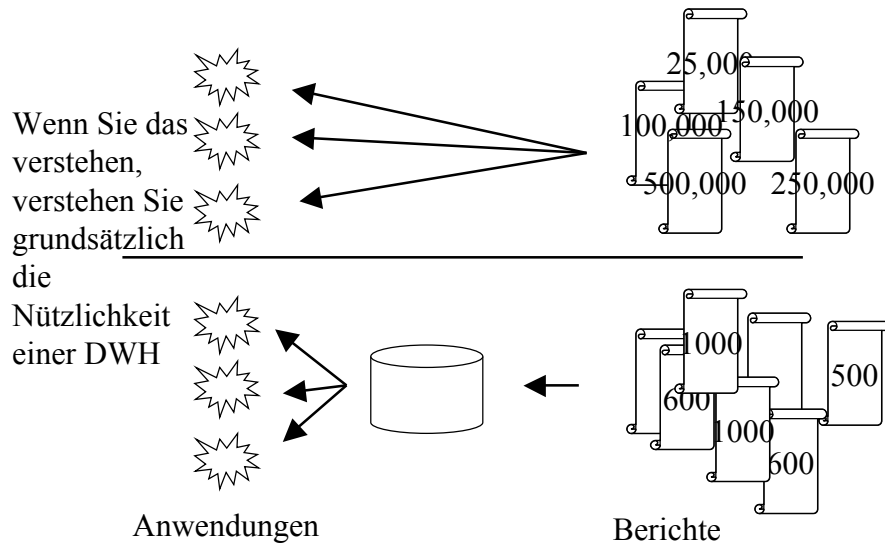
Um den Bericht zu erstellen müssen die Anwendungen
Betrachten wir eine Firma, die Information braucht
zugegriffen werden



DM outline

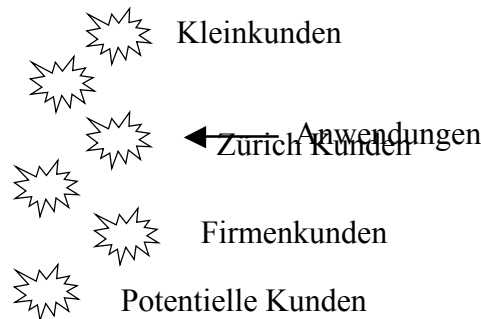
- Häufig vorkommende Objektmenngen
- Assoziationsregeln
- Decision Trees
- Clustering

Der Gewinn wird durch die Wiederverkauf der Berichte realisiert



Aber es gibt noch weitere Gründe für eine DWH

- Betrachten wir die Anwendungen, die eine Firma über Zeit kreiert oder gekauft hat:

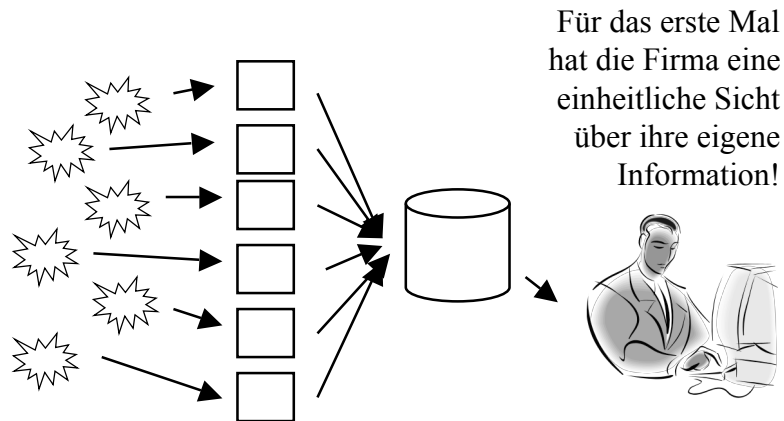


Für die gesamte Firma:

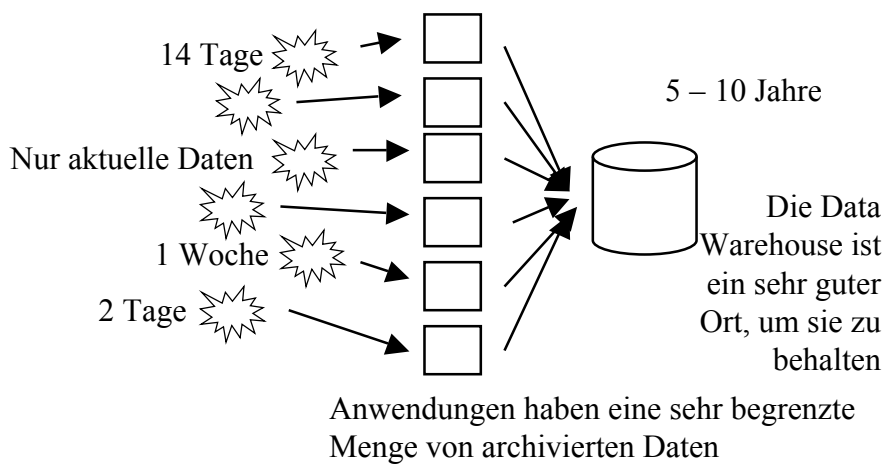
Was ist ein Kunde?

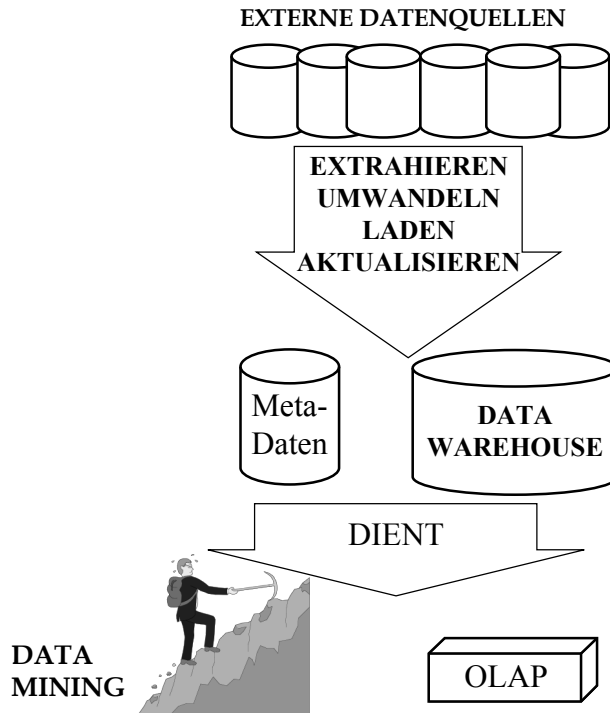
Jede Anwendung hat ihre eigene Interpretation von was ein Kunde ist. Es gibt kein einheitliches Konzept vom Kunde

- Daten werden aus der einzelnen Anwendungen geholt und in der Data Warehouse integriert
- Mit der Integration werden Daten vereinheitlicht und stellen dann eine einheitliche Sicht über dem Begriff „Kunden“



Aber es gibt noch ein wichtiger Grund, wieso eine Firma eine Data Warehouse braucht ...





Multidimensionales Datenmodell

Sammlung von numerische Messungen, die von einer Menge Dimensionen abhängen.

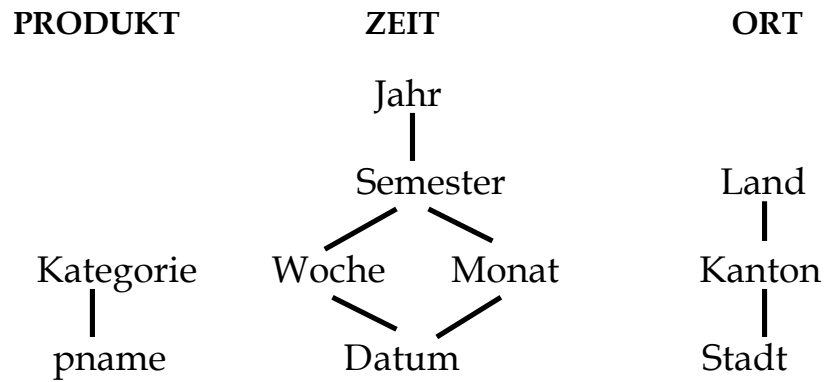
- Beispiel: Messung Verkauf, Dimensionen Produkt (key: pid), Ort (ortid), und Zeit (zeitid).

Hier sehen wir die Tranche [slice] ortid=1

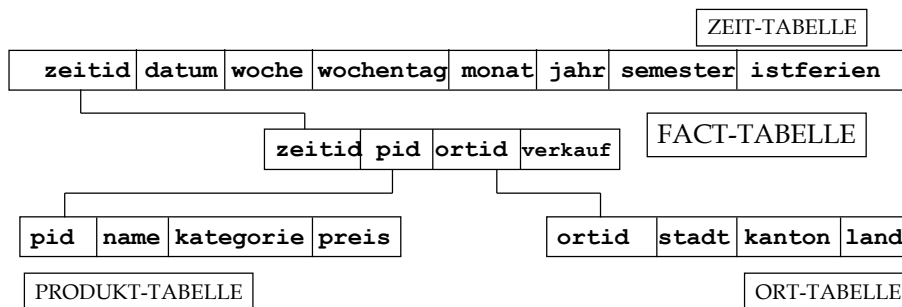
pid	11	12	13
	8	10	10
	30	20	50
	1	2	3
	25	8	15
	zeitid		

pid	zeitid	ortid	verkauf
11	1	1	8
11	2	1	10
11	3	1	10
12	1	1	30
12	2	1	20
12	3	1	50
13	1	1	25
13	2	1	8
13	3	1	15
	0	0	0

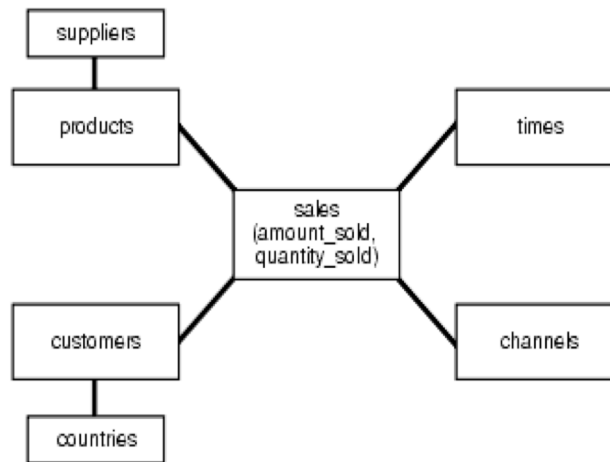
Hierarchien von Dimensionen: Für jede Dimension kann die Menge von Werte als Hierarchie organisiert werden:



STAR SCHEMA

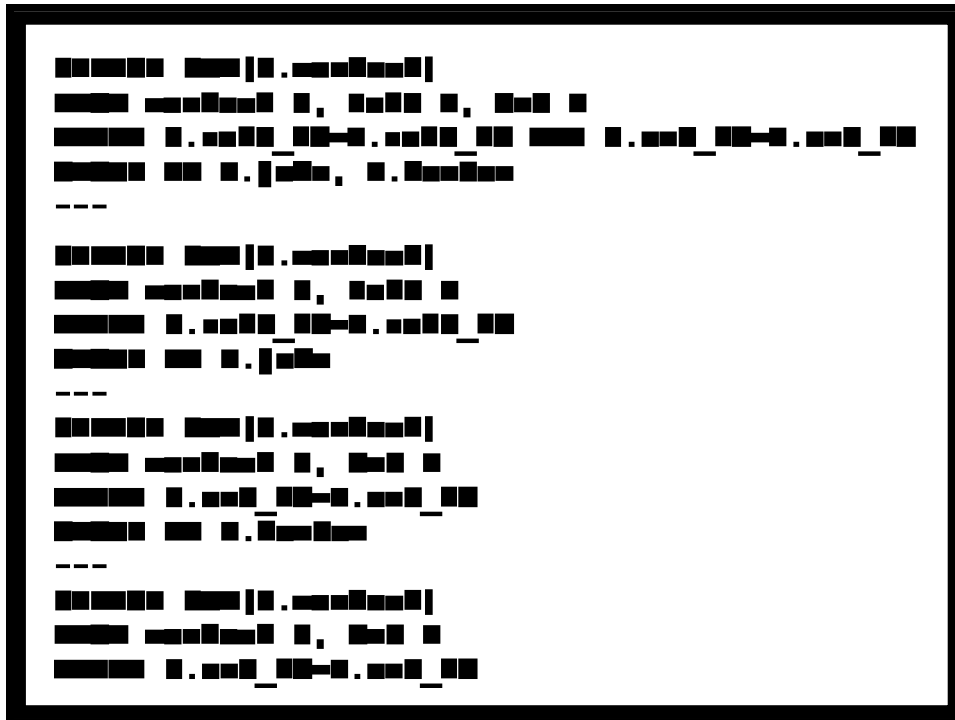


SNOWFLAKE SCHEMA



PIVOTING

	AG	ZH	Total
1995	63	81	144
1996	38	107	145
1997	75	35	110
Total	176	223	339



```

SELECT channels.channel_desc, countries.country_iso_code,
TO_CHAR(SUM(amount_sold), '9,999,999,999') SALES$
FROM sales, customers, times, channels, countries
WHERE sales.time_id=times.time_id AN
      sales.cust_id=customers.cust_id AND
      sales.channel_id= channels.channel_id AND
      channels.channel_desc IN ('Direct Sales', 'Internet') AND
      times.calendar_month_desc='2000-09' AND
      customers.country_id=countries.country_id AND
      countries.country_iso_code IN ('US','FR')
GROUP BY CUBE(channels.channel_desc, countries.country_iso_code);

```

CHANNEL_DESC	CO	SALES\$
		833,224
	FR	70,799
	US	762,425
Internet		133,821
Internet	FR	9,597
Internet	US	124,224
Direct Sales		699,403
Direct Sales	FR	61,202
Direct Sales	US	638,201

```

SELECT channel_desc, calendar_month_desc, countries.country_iso_code,
       TO_CHAR(SUM(amount_sold), '9,999,999,999') SALES$
FROM sales, customers, times, channels, countries
WHERE sales.time_id=times.time_id AND sales.cust_id=customers.cust_id AND
      sales.channel_id= channels.channel_id AND channels.channel_desc IN
      ('Direct Sales', 'Internet') AND times.calendar_month_desc IN
      ('2000-09', '2000-10') AND countries.country_iso_code IN ('GB', 'US')
GROUP BY channel_desc, ROLLUP(calendar_month_desc,
                              countries.country_iso_code);

```

CHANNEL_DESC	CALENDAR	CO	SALES\$
Internet	2000-09	GB	228,241
Internet	2000-09	US	228,241
Internet	2000-09		456,482
Internet	2000-10	GB	239,236
Internet	2000-10	US	239,236
Internet	2000-10		478,473
Internet			934,955
Direct Sales	2000-09	GB	1,217,808
Direct Sales	2000-09	US	1,217,808
Direct Sales	2000-09		2,435,616
Direct Sales	2000-10	GB	1,225,584
Direct Sales	2000-10	US	1,225,584
Direct Sales	2000-10		2,451,169
Direct Sales			4,886,784

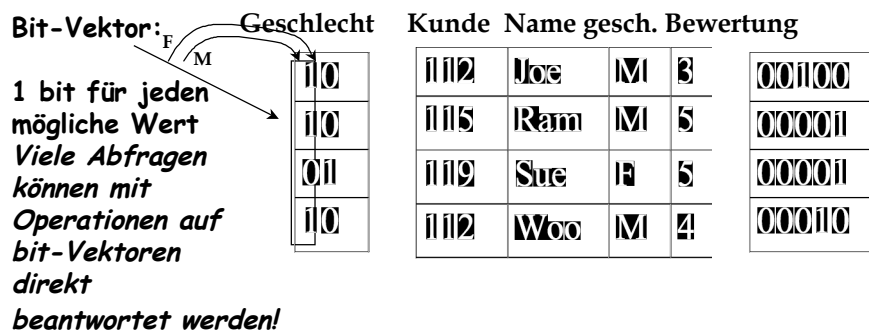
```

SELECT t.time_id, TO_CHAR (SUM(amount_sold), '9,999,999,999')
AS SALES, TO_CHAR(AVG(SUM(amount_sold)) OVER
 (ORDER BY t.time_id
  RANGE BETWEEN INTERVAL '1' DAY PRECEDING AND
  INTERVAL '1' DAY FOLLOWING), '9,999,999,999') AS
CENTERED_3_DAY_AVG
FROM sales s, times t
WHERE s.time_id=t.time_id AND t.calendar_week_number IN (51)
AND calendar_year=1999
GROUP BY t.time_id
ORDER BY t.time_id;

```

TIME_ID	SALES	CENTERED_3_DAY
20-DEC-99	134,337	106,676
21-DEC-99	79,015	102,539
22-DEC-99	94,264	85,342
23-DEC-99	82,746	93,322
24-DEC-99	102,957	82,937
25-DEC-99	63,107	87,062
26-DEC-99	95,123	79,115

Bitmap Index



View Modification (Evaluate On Demand)

View

```
CREATE VIEW RegionalSales(category,sales,state)
AS SELECT P.category, S.sales, L.state
FROM Products P, Sales S, Locations L
WHERE P.pid=S.pid AND S.locid=L.locid
```

Query

```
SELECT R.category, R.state, SUM(R.sales)
FROM RegionalSales AS R GROUP BY R.category, R.state
```

Modified
Query

```
SELECT R.category, R.state, SUM(R.sales)
FROM (SELECT P.category, S.sales, L.state
FROM Products P, Sales S, Locations L
WHERE P.pid=S.pid AND S.locid=L.locid) AS R
GROUP BY R.category, R.state
```

Materialized View

```
CREATE MATERIALIZED VIEW
  RegionalSales(category,sales,state)
AS SELECT P.category, S.sales, L.state
   FROM Products P, Sales S, Locations L
   WHERE P.pid=S.pid AND S.locid=L.locid
```

View Materialisation (Vorberechnung)

```
SELECT R.kanton,
       SUM(R.verkauf)
FROM Regionalverkauf R
WHERE R.kategorie="Laptop"
GROUP BY R.kanton
```

Index auf vorgerechnete
Sicht ist gut!

```
SELECT R.kanton,
       SUM(R.verkauf)
FROM Regionalverkauf R
WHERE R.kanton="Aargau"
GROUP BY R.kategorie
```

Index ist weniger
nützlich (alle Blätter müssen
durchgescannt werden)

Top N Abfragen

```
SELECT P.p_id, P.pname, S.verkauf
FROM verkauf S, Products P
WHERE S.p_id=P.p_id AND S.ort_id=1 AND S.zeit_id=3
ORDER BY S.verkauf DESC
OPTIMIZE FOR 10 ROWS
```

```
SELECT P.p_id, P.pname, S.verkauf
FROM verkauf S, Products P
WHERE S.p_id=P.p_id AND S.ort_id=1 AND S.zeit_id=3
      AND S.verkauf > c
ORDER BY S.verkauf DESC
```

- **OPTIMIZE FOR** Klausel ist nicht in SQL:1999!
- Cut-off Wert c ist vom Optimizer gewählt

```
SELECT channel_desc, TO_CHAR(SUM(amount_sold), '9,999,999,999') SALE$$,
      RANK() OVER (ORDER BY SUM(amount_sold)) AS default_rank,
      RANK() OVER (ORDER BY SUM(amount_sold) DESC NULLS LAST) AS custom_rank
FROM sales, products, customers, times, channels, countries
WHERE sales.prod_id=products.prod_id AND sales.cust_id=customers.cust_id AND
      sales.time_id=times.time_id AND sales.channel_id=channels.channel_id AND
      times.calendar_month_desc IN ('2000-09', '2000-10') AND country_iso_code='US'
GROUP BY channel_desc;
```

CHANNEL_DESC	SALE\$\$	DEFAULT_RANK	CUSTOM_RANK
Direct Sales	2,443,392	3	1
Partners	1,365,963	2	2
Internet	467,478	1	3

```
CREATE MATERIALIZED VIEW cust_sales_mv
PCTFREE 0 TABLESPACE demo
STORAGE (INITIAL 16k NEXT 16k PCTINCREASE 0)
PARALLEL
BUILD IMMEDIATE
REFRESH COMPLETE
ENABLE QUERY REWRITE AS
SELECT c.cust_last_name, SUM(amount_sold) AS sum_amount_sold
FROM customers c, sales s WHERE s.cust_id = c.cust_id
GROUP BY c.cust_last_name;
```

