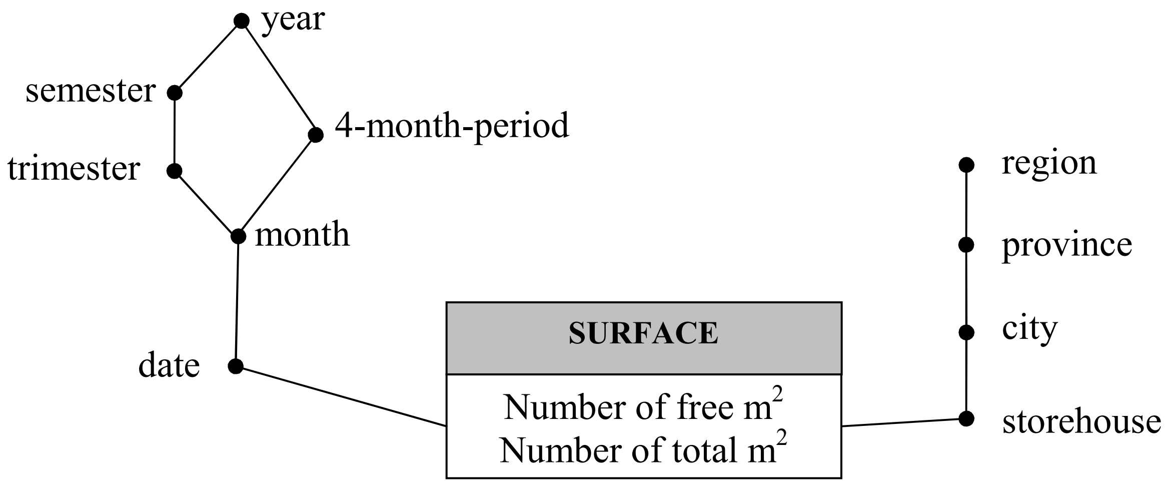
Data Base and Data Mining Group of Politecnico di Torino

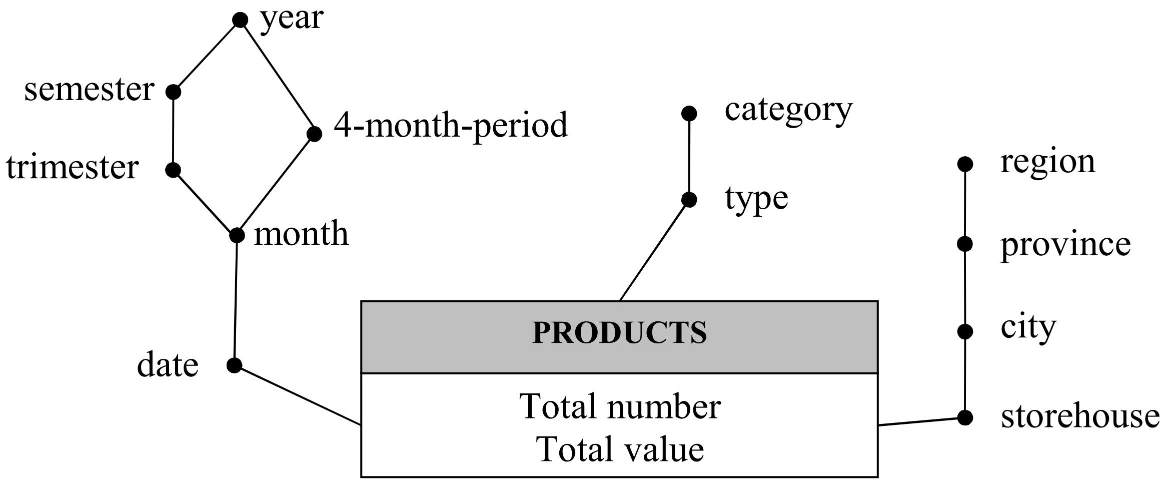
Analisi di basi di dati

Politecnico di Torino
III Facoltà di Ingegneria
Laurea Specialistica in Ingegneria Informatica

ESAME DEL 31-01-2007 – Soluzione DRAFT

Modello Concettuale





Modello Logico

Primary keys are underlined.

Facts

SURFACE (storehouseID, timeID, m2free, m2tot)
PRODUCTS (storehouseID, timeID, typeID, totNumber, totValue)

Dimensions

TIME (timeID, date, month, trimester, 4month-period, semester, year) → shared both facts
TYPES (typeID, type, category) → only for Products fact
STOREHOUSES (storehouseID, storehouse, city, province, region) → shared both facts

Query A

```
select
    storehouse, date, sum(totValue),
    avg( sum(totValue) ) over (partition by storehouse order by date range between interval '6' day preceding and current row)
from
    products p, storehouses sh, time t
where
    p.storehouseID=sh.storehouseID and p.timeID=t.timeID and
    t.year=2003 and t.trimester=1 and sh.city='Turin'
group by
    storehouseID, storehouse, date;
```

Card: 5 x (30 x 3) = 450 << 7300k → a materialized view on this query is convenient.
Removing the constraints on trimester and city, the view would be useful to answer query **d** and **e** too.

NB: averaging the daily total value over the last week could be done using the *sum(sum(totValue)/7)* expression, which handles missing days as if their *totValue* were 0, while the proposed solution fills missing values with the week average; furthermore note that *totValue* is a level measure, thus there should be no missing values in the data warehouse.

Query B

```
select
    city, date,
    sum(m2free)/sum(m2tot)*100,
    rank() over (order by sum(m2free)/sum(m2tot) asc)
from
    surface s, storehouses sh, time t
where
    s.storehouseID=sh.storehouseID and s.timeID=t.timeID and t.year=2004
group by
    city, date;
```

Card: 90 x 365 = 32850 ≈ 73000 → a materialized view on this query is NOT convenient.

Query C

```
select
    storehouse, date, m2free/m2tot,
from
    products p, storehouses sh, time t
where
    p.storehouseID=sh.storehouseID and p.timeID=t.timeID and
    t.year=2004 and t.month>=1 and t.month<=6
group by
    storehouseID, storehouse, date;
```

Card: 100 x (30 x 6) = 18000 ≈ 73000 → a materialized view on this query is NOT convenient.

Query D

```
select
    storehouse, month,
    sum(totValue)/count(distinct date)
from
    products p, storehouses sh, time t
where
    p.storehouseID=sh.storehouseID and p.timeID=t.timeID and t.year=2003
group by
    storehouseID, storehouse, month;
```

```
select distinct
    storehouse, month,
    avg( sum(totValue) ) over (partition by storehouse, month)
from
    products p, storehouses sh, time t
where
    p.storehouseID=sh.storehouseID and p.timeID=t.timeID and t.year=2003
group by
    storehouseID, storehouse, date, month;
```

Card: 100 x 12 = 1200 << 7300k → a materialized view on this query is convenient and it helps to answer query **e** too.

NB: the DISTINCT command does **not** remove rows with the same storehouse; it removes duplicate rows considering all attribute values of each row.

Query E

```
select
    region, sum(totValue)/count(distinct date)
from
    products p, storehouses sh, time t
where
    p.storehouseID=sh.storehouseID and p.timeID=t.timeID and t.year=2003
group by
    region;
```

```
select distinct
    region, avg(sum(totValue)) over (partition by region)
from
    products p, storehouses sh, time t
where
    p.storehouseID=sh.storehouseID and p.timeID=t.timeID and t.year=2003
group by
    region, date;
```

Card: 40 << 7300k → a materialized view on this query is convenient.

Query F

```
select distinct
    region, month,
    avg(sum(m2free)/sum(m2tot)*100) over (partition by region, month)
from
    surface s, storehouses sh, time t
where
    s.storehouseID=sh.storehouseID and s.timeID=t.timeID and t.year=2004
group by
    region, month, date;
```

Card: 40 x 12 = 480 << 7300k → a materialized view on this query is convenient.