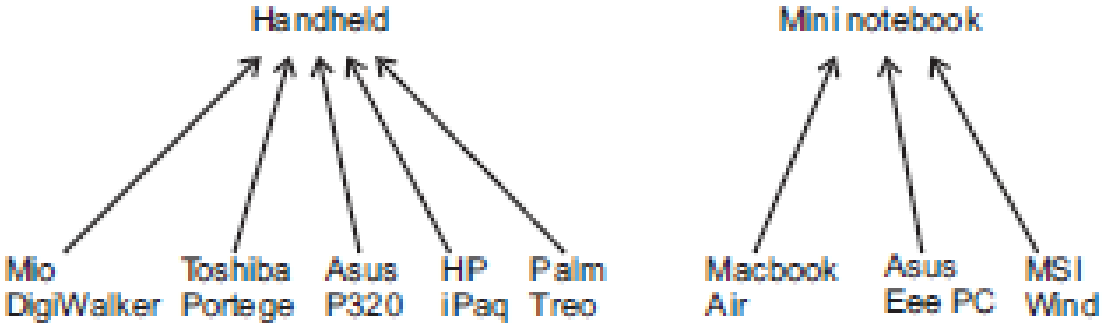
HOBI

Join index: values of attribute index the fact table

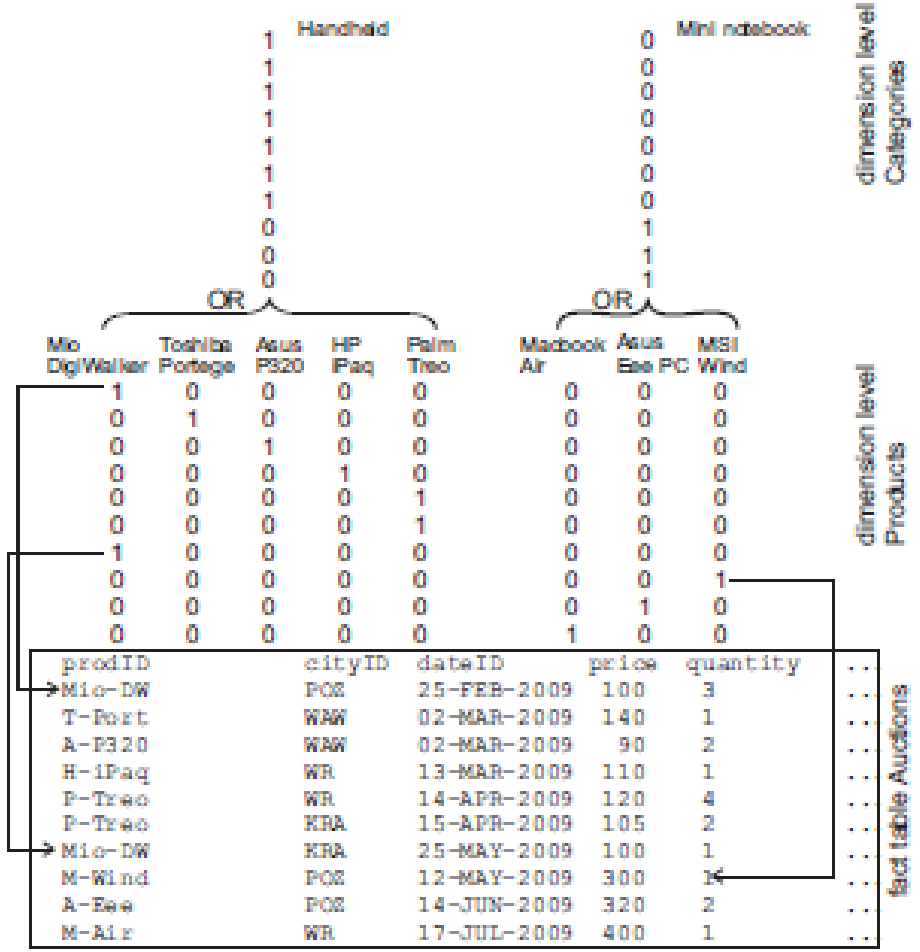
Hierarchically structured bit map index

One level per attribute level

e.g. product → categories



Example HOBI for products



Query 1

```
/* query Q1 */
SELECT COUNT(1)
FROM Auctions, Products, Categories, Days
WHERE Auctions.prodID = Products.prodID
AND Products.categID = Categories.categID
AND Auctions.dateID = Days.dateID
AND Categories.categName = 'Handheld'
AND Days.dateID
  BETWEEN to_date('1-04-2007','dd-mm-yyyy')
  AND to_date('31-07-2007','dd-mm-yyyy');
```

Query 1

```
/* query Q2 */
SELECT Cities.cityName, SUM(Auctions.price)
FROM Auctions, Cities
WHERE Auctions.cityID = Cities.cityID
GROUP BY Cities.cityName;
```

Query 1

```
/* query Q3 */
SELECT Regions.regionName, SUM(Auctions.price)
FROM Auctions, Cities, Regions
WHERE Auctions.cityID = Cities.cityID
AND Cities.regionID = Regions.regionID
group by Regions.regionName;
```

Time-HOBI

All dimensional models include Day, or a rollup of Day

Time-HOBI can eliminate a join to the Day dimension

Assumption: the fact table is sorted by date
-reasonable and easily achieved
It is typical for a fact table to stored in physical disk partitions based on date

Time-HOBI is HOBI augmented with a time index, TI

Time Index (TI)

Only one TI is created
- It will work with any/every HOBI index

The TI stores ranges of bit numbers belonging to a specified time interval
- Each entry of a TI specifies a time interval and a range of bit numbers (bit positions in a bit map referencing the fact table)

Time Index (TI)

Implementation of TI (Section 5.3)

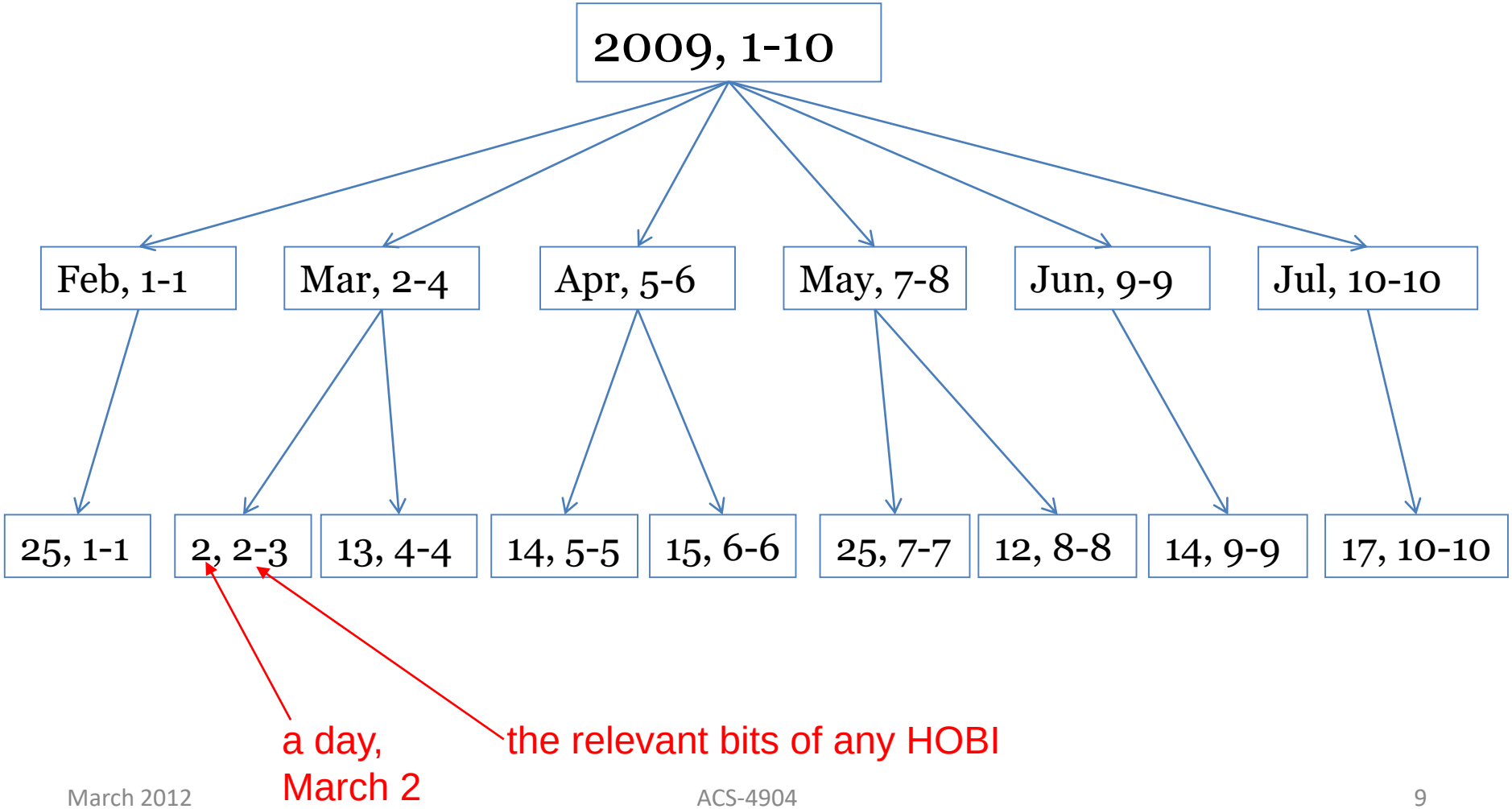
Tree structure where each level corresponds to a level of the day hierarchy:

Days → Months → Y ears

One tree per year - Each root is a year
Next level is month
Leaf level is day

Time Index (TI)

E.g. (based on figure 5)



Conceptual execution of query

Conceptually for any query that references time and attributes indexed by HOBI:

Using TI the system determines the fragments of the HOBI bit maps that need to be accessed.

Then, from the system retrieves HOBI fragments

The bit map fragments are ‘combined’ as required by the *where* clause

Lastly, the system retrieves rows of the fact table

Q1

Query Q1:
the number of auctions between April 1st and July 31st

SELECT count(1) from Auctions inner join Products ON …
inner join Categories ON … inner join Days ON …
WHERE
    category name = “Handheld”
AND
    date between Apr 1, 2007 and July 31, 2007

To answer the above … next slide

Q1

To answer the above:

1. the system determines the fragments to access that are relevant to "*between Apr 1, 2007 and July 31, 2007*"

2. the system retrieves the fragment for the Handheld category.

3. Because of "count(1)" there is no need to access Auctions … the system can count the 1 bits

Q1-1 (related to Figure 5)

Query Q1 **Modified**: Consider Figure 5 and the number of auctions between April 1st and June 30th 2009

SELECT count(1) from Auctions inner join Products ON … inner join Categories ON … inner join Days ON …
WHERE
    category name = "Handheld"
AND
    date between Apr 1, 2009 and Jun 30, 2009

1. To answer the above: the system determines the fragments to access that are relevant to "*between Apr 1, 2009 and June 30, 2009*" which are: _____
2. The system retrieves the fragment for the Handheld category: _____
3. The system counts the 1 bits: _____

Q1-2 (related to Figure 5)

Query Q1 **Modified**: Consider Figure 5 and the number of auctions in April and May of 2009

SELECT count(1) from Auctions inner join Products ON … inner join Categories ON … inner join Days ON …
WHERE
    category name = “Handheld”
AND (
        month = Apr
    OR
        month = May
    AND
        year = 2009
)

Processing required to answer the query ……