B17 Enhanced OLAP integration with Rollup and Cube, as well as latest AST enhancements in DB2 UDB

William O'Connell, Senior Technical Staff member, IBM Toronto Lab

This session will discuss the standardized support in DB2 Universal Database on OLAP and statistical functionality, and more importantly, its integration with the Rollup and Cube operators, as well as the Materialized Query Tables (MQT) and Automatic Summary Table (AST) technology. The latest enhancements in MQT and AST technology will also be discussed. This session will briefly explain the emerging role of the database in business intelligence development and how DB2 helps OLAP and mining tools, and ERP applications. In doing so, it will mainly focus on overviewing and explaining these concepts through SQL examples based on DB2 for UNIX, Windows. Lastly, future directions will be briefly mentioned.

Session B17

Enhanced OLAP integration with Rollup and Cube, as well as associated latest AST enhancements in DB2 UDB

William O'Connell, IBM Toronto Lab, boconnel@ca.ibm.com

IBM Data Management Technical Conference

Anaheim, CA **Sept 9 - 13, 2002**

Focus and Terminology

- We will focus on DB2 UDB for Linux, UNIX, and Windows.
- V7 uses the term AST, whileas V8 also uses the term MQT (Materialized Query Table)
  - ASTs are still referenced in V8 when explicitly referring to aggregated materialized views.
  - This presentation is focusing on OLAP interaction with ASTs.
  - However, all discussions apply to MQTs too.

Outline

- BI and OLAP Support (overview strategy)
- Analytic: basics of statistics
- OLAP: basics of scalar aggregate functions
- Interactions with Rollup, Cubes, and ASTs
- Advanced Cubes and ASTs dealing with High Dimensionality

Note:

See sessions U09 and U10 on additional AST discussions

U09 - The new and improved Automatic Summary Table feature

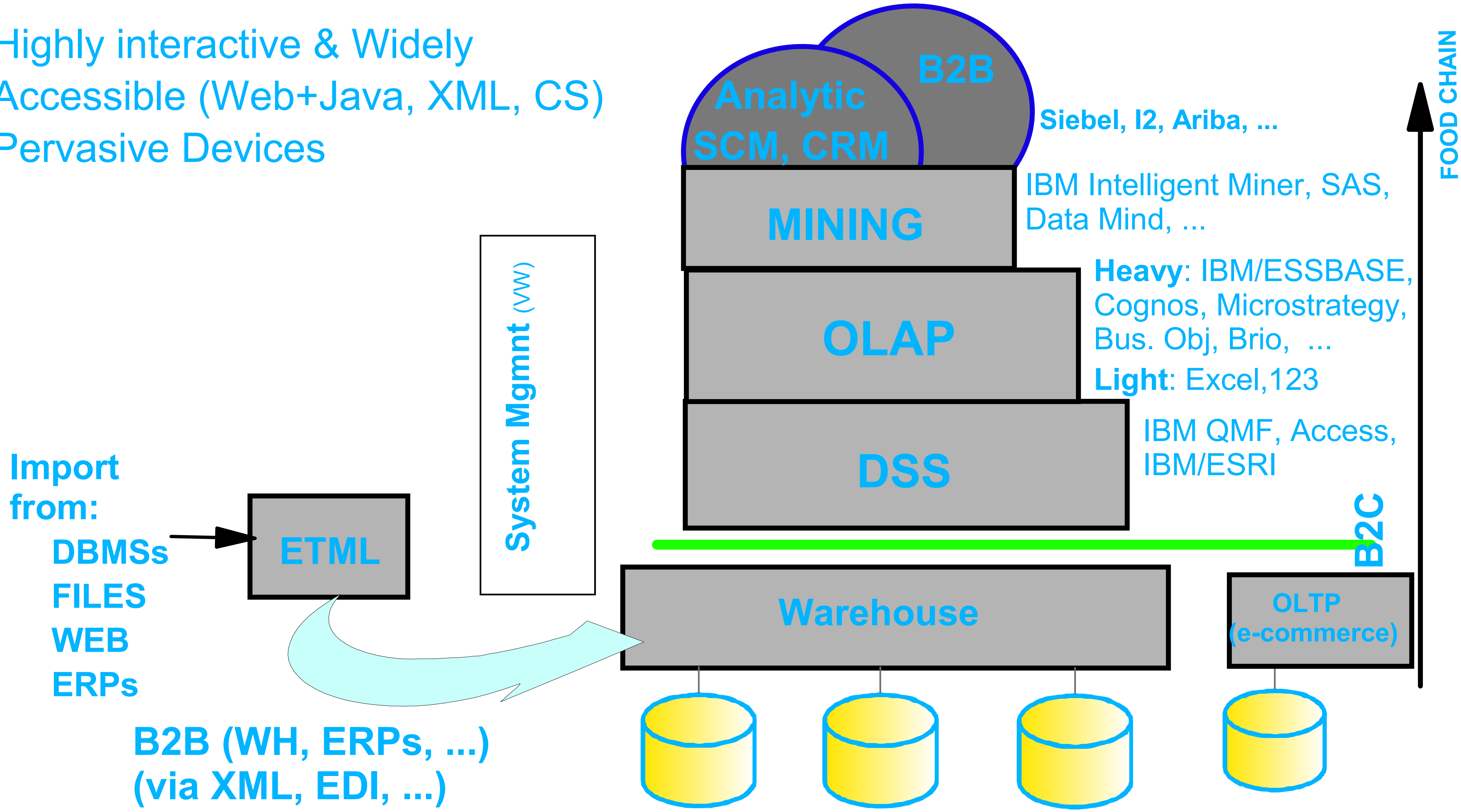
U10 - Matching Queries to Automatic Summary Tables

See session B15 on Advanced analytics for business intelligence

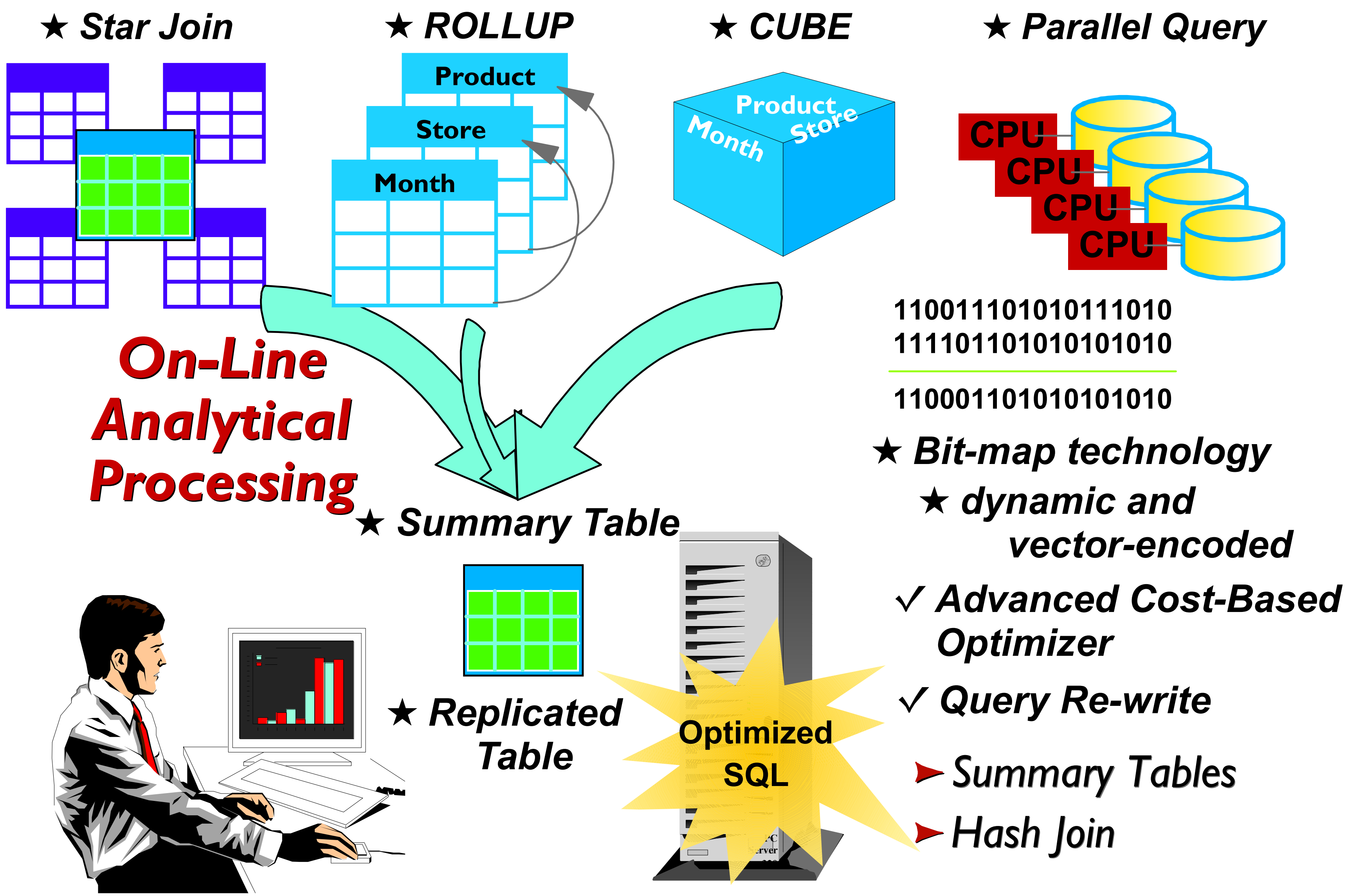
See session B16 on Sampling used with analytical processing

E-Business: Moving Up The Food Chain

Rich functionality with High performance



Business Intelligence Technology



Some basics of advanced statistics and OLAP Queries first

Statistics: Transform Data to Knowledge

- The problem: extracting useful business information from data

- Why push computation into database?
  - ✓ processing occurs close to data
  - ✓ automatically exploits parallelism
  - ✓ exploit other DB features: incremental maintenance, OLAP capabilities, etc.

- The DB2 toolkit
  - ✓ statistical functions: aggregates, correlation, regression suite
  - ✓ OLAP functions
  - ✓ synergy: can combine these tools with MQTs, ASTs, and other capabilities

Analytics: advanced statistics

- Find sales areas where individual income and sales are not aligned

```
select country,state,correlation(sumsales,income_range),
                     covariance (sumsales,income_range)
from ...  where ...GROUP BY COUNTRY,STATE
```

COUNTRY	STATE	CORRELATION	COVARIANCE
USA	AK	-0.13	-145217876
USA	AL	0.29	104791704
USA	DE	0.20	223579152
USA	GA	0.28	239422676
USA	IL	0.16	87015909
USA	KS	-0.47	-20807683
USA	LA	0.15	16366277

Analytics: statistics

- avg, stddev, max, min, ...
- Advanced functions:
  - Correlation
  - Covariance
  - Family of linear regression functions

    fitting of an ordinary-least-squares regression line of the form y = a * x + b

    to a set of number pairs

    `REGR_SLOPE, REGR_INTERCEPT, REGR_ICPT, REGR_COUNT, REGR_R2, REGR_AVGX, REGR_AVGX, REGR_AVGY, REGR_AVGY, REGR_SXX, REGR_SYY, REGR_SXY`

Analytics: advanced statistics

- Get the linear regression slope of sales as a function of income. Also get the correlation between the two.

```
select country,state,
       correlation(sumsales,income_range),
       REGR_SLOPE(income_range, sumsales)
from ..  where ...
```

COUNTRY	STATE	CORRELATION	SLOPE
USA	AK	-0.13	-0.28
USA	AL	0.29	0.62
USA	DE	0.20	0.35
USA	GA	0.28	0.73
USA	IL	0.16	0.49
USA	KS	-0.47	-5.84
USA	LA	0.15	0.86

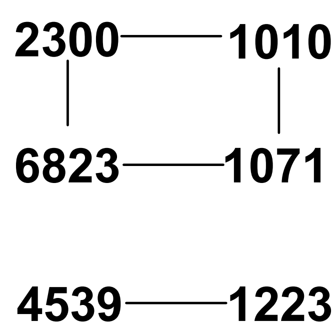
Another Use for Correlation

- Customers with similar buying habits:

```
VIEW transvw3(custid, prodid, amount)

SELECT a.custid as custid1, b.custid as custid2,
       corr(a.amount, b.amount)as corr
FROM transvw3 a, transvw3 b
WHERE a.prodid = b.prodid and a.custid < b.custid
GROUP BY a.custid, b.custid
HAVING corr(a.amount, b.amount) >= 0.5 and count(*) > 100
ORDER BY corr desc;
```

Total amount purchased overall transactions

CUSTID1	CUSTID2	CORR
2300	6823	0.99
1071	2300	0.85
1223	4539	0.83
1010	1071	0.78
1010	2300	0.72
1071	6823	0.65



OLAP Functions

- Enriching SQL in the OLAP domain
- Rank, Denserank, Rownumber, Moving aggregates, ...
- A major extension to SQL, which was adopted by ANSI
- These functions are in addition to the Cube functionality in the SQL standard
  (DB2 UDB supports multidimensional hierarchical cubes
  with extensions: cube, multiple rollup's, grouping sets)

Ranking

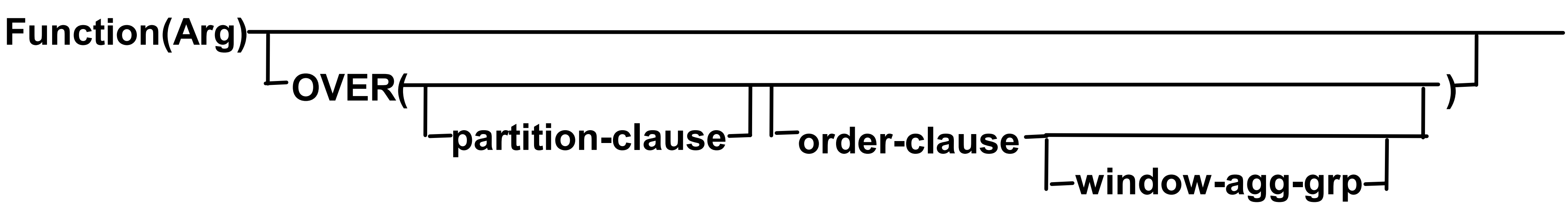
- Rank annual sales and annual count of sales.

```
select rank() over (order by sum(ti.amount) desc ) as rank_for_sum,
       sum(ti.amount) as sum, year(pdate) as year,
       rank() over (order by count(*) desc ) as rank_for_count,
       count(*) as count
from trans t, transitem ti where t.transid = ti.transid
group by year(pdate)
```

RANK FOR SUM	SUM	YEAR	RANK FOR COUNT	COUNT
1	4854484.01	1996	4	940
2	4822312.32	1989	2	947
3	4775518.17	1991	3	945
4	4605738.00	1988	5	918
5	4565246.21	1987	1	954
6	4551154.94	1995	6	894
7	4322151.92	1993	9	837
8	4269707.26	1992	7	852
9	4108654.03	1994	8	844
10	3962436.22	1990	10	814

Scalar Aggregate Functions

- Scalar Aggregate Functions operate on values from a set of rows, and return a single result per row.
  - We'll refer to these generically as OLAP
- The set of rows is defined using the *window-clause*
- This set has three primary attributes
  - An Ordering
  - A Partitioning
  - A Window Aggregation Group
- This set is defined with the OVER clause



Ranking Within Partitions

- Rank annual sales by country -- Each country has its own rank

```
select loc.country,
    rank() over (partition by loc.country order by sum(ti.amount)
    desc ) as  rank_for_sum, year(t.pdate) as year,  sum(ti.amount) as
    sum, rank() over(order by sum(ti.amount) desc ) as global_rank
from trans t, transitem ti, loc loc
where t.transid = ti.transid and loc.locid = t.locid
group by year(pdate), loc.country
```

COUNTRY	RANK_FOR_SUM	YEAR	SUM	GLOBAL_RANK
USA	1	1998	1679467.97	1
USA	2	1996	1620410.14	2
USA	3	1997	1408984.07	3
UK	1	1997	609344.48	10
UK	2	1996	535244.11	13
UK	3	1998	426842.79	15
Canada	1	1998	1224256.25	6
Canada	2	1997	1081640.89	8
Canada	3	1996	973548.88	9

Rownumber(): Unique sequential numbering

- Very useful in select list or insert with subquery
- Can reset the numbers per partition

```
Insert into mytable
select rownumber() over(order by loc.country desc, year(pdate)),
       loc.country,  year(t.pdate) as year,  sum(ti.amount) as sum
from  trans t, transitem ti, loc loc
where t.transid = ti.transid and loc.locid = t.locid
group by year(pdate), loc.country
```

```
 ROWNUMBER  COUNTRY     YEAR          SUM
 ---------- ----------- ---- ------------
          1 USA         1995   1930395.45
          2 USA         1996   1620410.14
          3 USA         1997   1408984.07
          4 USA         1998   1679467.97
          5 UK          1995    682856.41
          6 UK          1996    535244.11
          7 UK          1997    609344.48
          8 UK          1998    426842.79
          9 Germany     1995    888913.51
```

Cume Window Aggregate Functions

- Show monthly sales, running sum of sales, and running count of sales

```
SELECT year(t.pdate) as year, sum(ti.amount) as sum,
       sum(sum(ti.amount)) over (order by year(t.pdate)) as cumesum,
       count(*) as count,
       sum(count(*)) over (order by year(t.pdate)) as cumecount
FROM  trans t, transitem ti
WHERE t.transid = ti.transid
GROUP BY year(t.pdate)
```

```
YEAR     SUM            CUMESUM           COUNT  CUMECOUNT
---- ----------- ------------- ----------- ---------
1990  3765738.79    3765738.79         783       783
1991  4372445.01    8138183.80         870      1653
1992  4165324.25   12303508.05         821      2474
1993  4158406.91   16461914.96         861      3335
1994  4130432.59   20592347.55         833      4168
1995  4940724.03   25533071.58         993      5161
1996  4055131.32   29588202.90         817      5978
1997  3958294.95   33546497.85         784      6762
1998  4276335.41   37822833.26         798      7560
1999  4117996.32   41940829.58         840      8400
```

Cume Window Aggregate Functions

- Show monthly sales, running sum of sales, running count of sales

```
SELECT ti.pgid as pgroup, year(t.pdate) as year,
       sum(ti.amount) as sum_per_prod_year,
       sum( sum(amount)) over( partition by ti.pgid order by
              year(t.pdate) rows unbounded preceding ) as cume_prod,
       sum( sum(amount)) over( order by year(t.pdate), ti.pgid )
              as cume_all
FROM   trans t, transitem ti
WHERE  t.transid = ti.transid
GROUP BY ti.pgid, year(t.pdate)
```

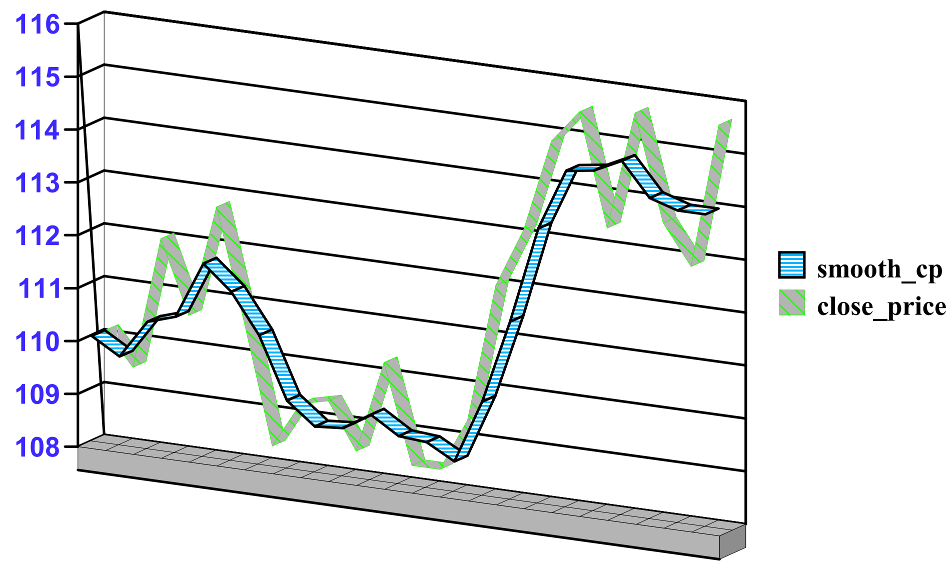
```
PGROUP YEAR   SUM_PER_PROD_YEAR    CUME_PROD    CUME_ALL
------ ---- ------------------- ----------- -----------
1      1995          1907763.47  1907763.47  1907763.47
1      1996          1671601.49  3579364.96  3579364.96
1      1997          1590642.59  5170007.55  5170007.55
1      1998          1834192.15  7004199.70  7004199.70
1      1999          1563596.19  8567795.89  8567795.89
NEW PGID  <==Look!                  RESET       NO RESET
4      1995            76630.07    76630.07  8644425.96
4      1996           102487.46   179117.53  8746913.42
4      1997            55114.54   234232.07  8802027.96
4      1998           122088.31   356320.38  8924116.27
4      1999            73078.32   429398.70  8997194.59
```

Curve Smoothing

Find the three day historical average of IBM stock for each day it traded

```
select date,symbol, close_price,
   avg(close_price) over (order by
      date rows 2 preceding)
   as smooth_cp
from stocktab
where symbol = 'IBM' and date between
     '1999-08-01' and '1999-09-01';
```

DATE	SYMBOL	CLOSE_PRICE	SMOOTH_CP
08/02/1999	IBM	110.125	110.1250
08/03/1999	IBM	109.500	109.8125
08/04/1999	IBM	112.000	110.5416
08/05/1999	IBM	110.625	110.7083
08/06/1999	IBM	112.750	111.7916
08/09/1999	IBM	110.625	111.3333
08/10/1999	IBM	108.375	110.5833
08/11/1999	IBM	109.250	109.4166
08/12/1999	IBM	109.375	109.0000
08/13/1999	IBM	108.500	109.0416
08/16/1999	IBM	110.250	109.3750
08/17/1999	IBM	108.375	109.0416
08/18/1999	IBM	108.375	109.0000
08/19/1999	IBM	109.375	108.7083
08/20/1999	IBM	112.000	109.9166
08/23/1999	IBM	113.125	111.5000
08/24/1999	IBM	114.875	113.3333
08/25/1999	IBM	115.500	114.5000
08/26/1999	IBM	113.375	114.5833
08/27/1999	IBM	115.625	114.8333
08/30/1999	IBM	113.625	114.2083
08/31/1999	IBM	112.875	114.0416
09/01/1999	IBM	115.625	114.0416b



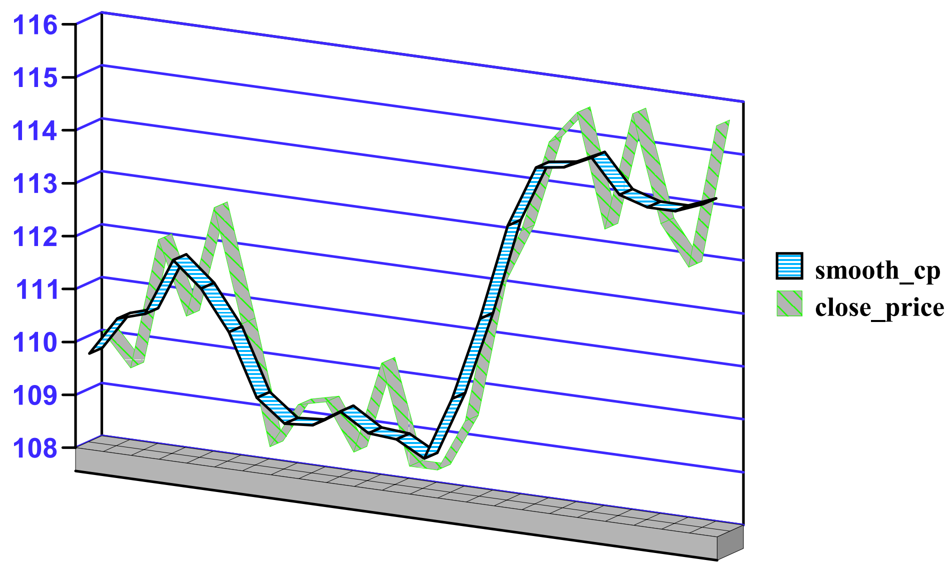
Three day historical average

Curve Smoothing

Find the three day centered average of IBM stock for each day it traded

```
 select date,symbol, close_price,
   avg(close_price) over(order by date
        rows between 1 preceding and
                     1 following)
   as smooth_cp
 from stocktab ...
```

DATE	SYMBOL	CLOSE_PRICE	SMOOTH_CP
08/02/1999	IBM	110.125	109.8125
08/03/1999	IBM	109.500	110.5416
08/04/1999	IBM	112.000	110.7083
08/05/1999	IBM	110.625	111.7916
08/06/1999	IBM	112.750	111.3333
08/09/1999	IBM	110.625	110.5833
08/10/1999	IBM	108.375	109.4166
08/11/1999	IBM	109.250	109.0000
08/12/1999	IBM	109.375	109.0416
08/13/1999	IBM	108.500	109.3750
08/16/1999	IBM	110.250	109.0416
08/17/1999	IBM	108.375	109.0000
08/18/1999	IBM	108.375	108.7083
08/19/1999	IBM	109.375	109.9166
08/20/1999	IBM	112.000	111.5000
08/23/1999	IBM	113.125	113.3333
08/24/1999	IBM	114.875	114.5000
08/25/1999	IBM	115.500	114.5833
08/26/1999	IBM	113.375	114.8333
08/27/1999	IBM	115.625	114.2083
08/30/1999	IBM	113.625	114.0416
08/31/1999	IBM	112.875	114.0416
09/01/1999	IBM	115.625	114.2500



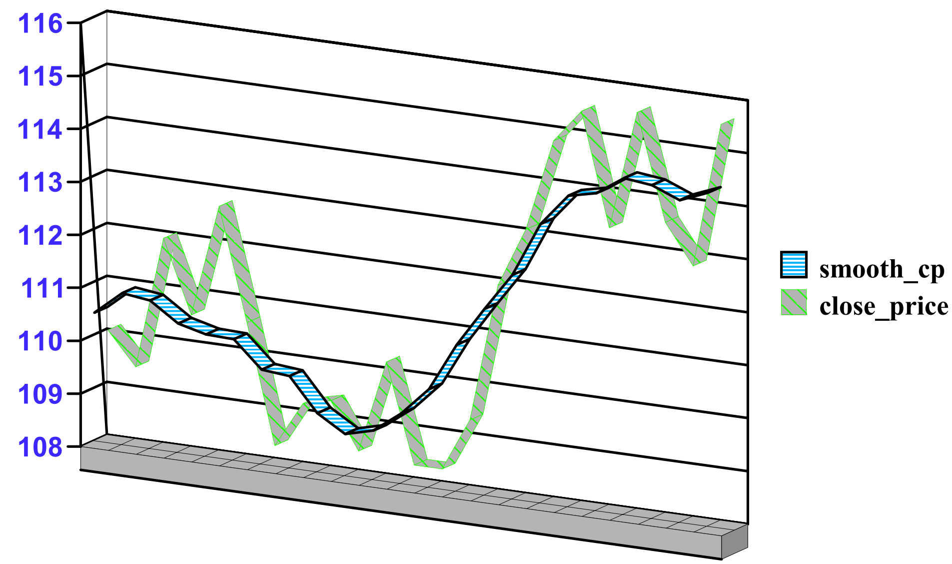
Three day centered average

Curve Smoothing

Find the seven day centered average of IBM stock for each day the stock traded

```
select date,symbol, close_price,
   avg(close_price) over (order by
    date
    rows between 3 preceding and
                 3 following)
   as smooth_cp
from stocktab
where symbol = 'IBM' and date between
      '1999-08-01' and '1999-09-01';
```

DATE	SYMBOL	CLOSE_PRICE	SMOOTH_CP
08/02/1999	IBM	110.125	110.5625
08/03/1999	IBM	109.500	111.0000
08/04/1999	IBM	112.000	110.9375
08/05/1999	IBM	110.625	110.5714
08/06/1999	IBM	112.750	110.4464
08/09/1999	IBM	110.625	110.4285
08/10/1999	IBM	108.375	109.9285
08/11/1999	IBM	109.250	109.8750
08/12/1999	IBM	109.375	109.2500
08/13/1999	IBM	108.500	108.9285
08/16/1999	IBM	110.250	109.0714
08/17/1999	IBM	108.375	109.4642
08/18/1999	IBM	108.375	110.0000
08/19/1999	IBM	109.375	110.9107
08/20/1999	IBM	112.000	111.6607
08/23/1999	IBM	113.125	112.3750
08/24/1999	IBM	114.875	113.4107
08/25/1999	IBM	115.500	114.0178
08/26/1999	IBM	113.375	114.1428
08/27/1999	IBM	115.625	114.5000
08/30/1999	IBM	113.625	114.4375
08/31/1999	IBM	112.875	114.2250
09/01/1999	IBM	115.625	114.4375



7 day centered average

Histograms - Equi-width

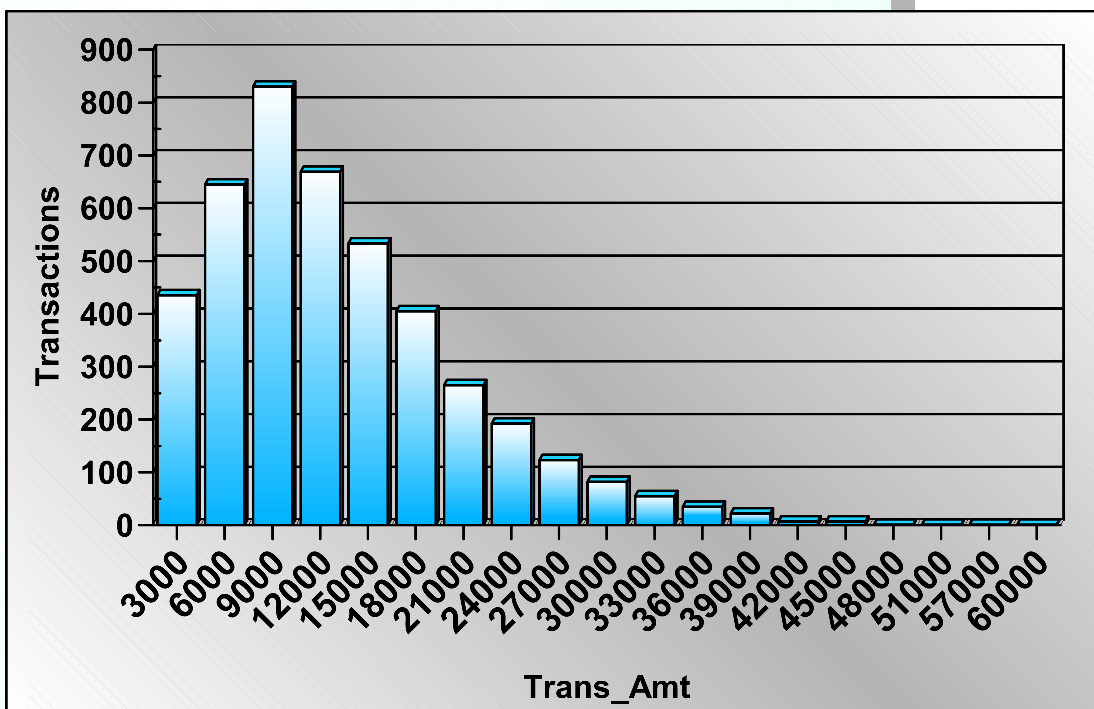
Plot an equi-width histogram with 20 buckets for the distribution of transaction amounts

```
with dt as (select t.transid, sum(amount) as trans_amt,
       case
         when     (sum(amount)-0)/((60000-0)/20) < 0 then 0
         when     (sum(amount)-0)/((60000-0)/20) > 19 then 19
         else int((sum(amount)-0)/((60000-0)/20))
       end as bucket
     from stars.trans t, stars.transitem ti
     where t.transid=ti.transid
     group by t.transid)
select bucket, count(bucket) as height, (bucket+1) * (60000-0)/20
as max_amt
from dt
group by bucket;
```

This is a simple scalar calculation

BUCKET	HEIGHT	MAX_AMT
0	435	3000
1	645	6000
2	830	9000
3	669	12000
4	533	15000
5	405	18000
6	265	21000
7	192	24000
8	123	27000
9	82	30000
10	55	33000
11	35	36000
12	22	39000
13	7	42000
14	7	45000
15	1	48000

...



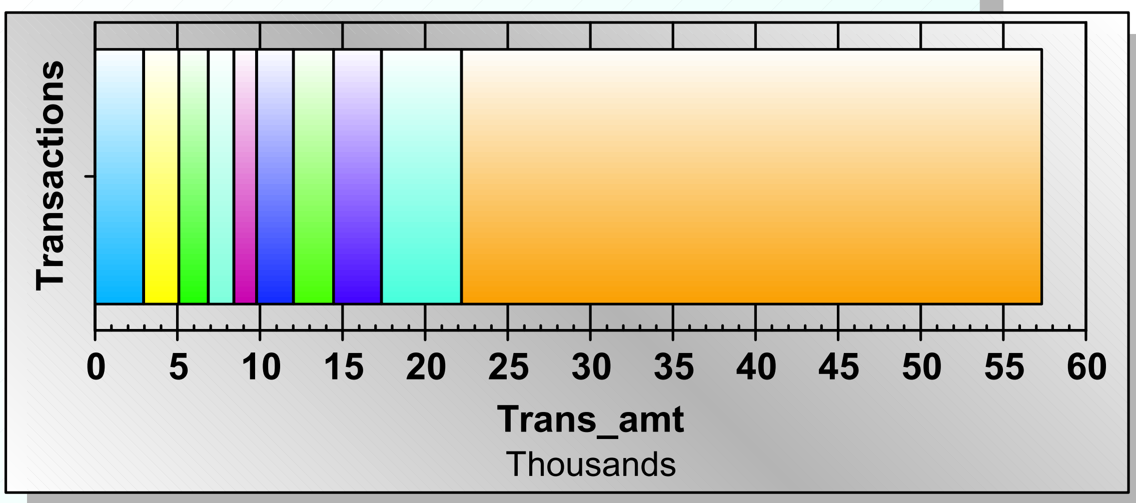
ference

Histograms - Equi-height

Plot an equi-height histogram with 10 buckets for the distribution of transaction amounts

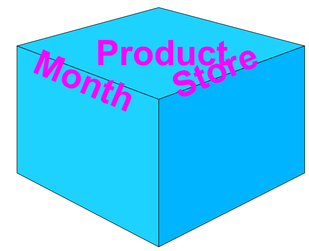
```
with dt as
  (select t.transid, sum(amount) as trans_amt,
          rownumber() over (order by sum(amount)) * 10 /
           (select count(distinct transid)+1
            from stars.transitem) as bucket
   from stars.trans t, stars.transitem ti
   where t.transid=ti.transid
   group by t.transid)
select bucket, count(bucket) as b_count,
       max(trans_amt) as part_value
from dt
group by bucket;
```

BUCKET	B_COUNT	PART_VALUE
0	430	2957.54
1	431	5094.14
2	431	6873.05
3	431	8429.81
4	431	9793.69
5	431	12019.40
6	431	14468.20
7	431	17355.26
8	431	22215.92
9	431	57360.41

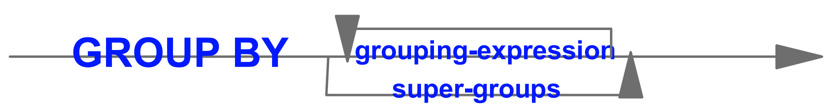


ROLLUP and CUBE - OLAP SQL Extensions

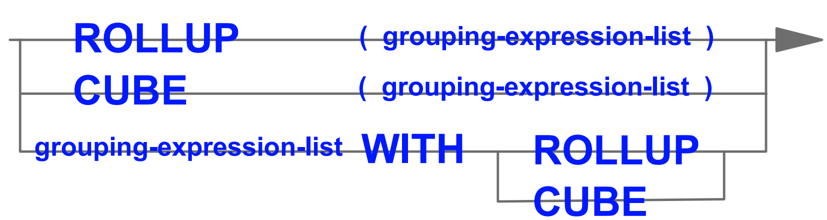
- Multiple ROLLUPs for multidimensional hierarchies
- ROLLUP is aggregation along a dimension hierarchy
- Extension to GROUP BY clause
- Produces "super aggregate" rows
- CUBE equivalent to "cross tabulation", equivalent to multiple rollups of height one
- May use star join for performance



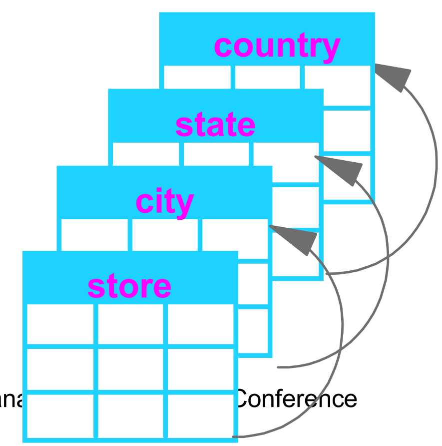
GROUP-BY Clause



super-groups

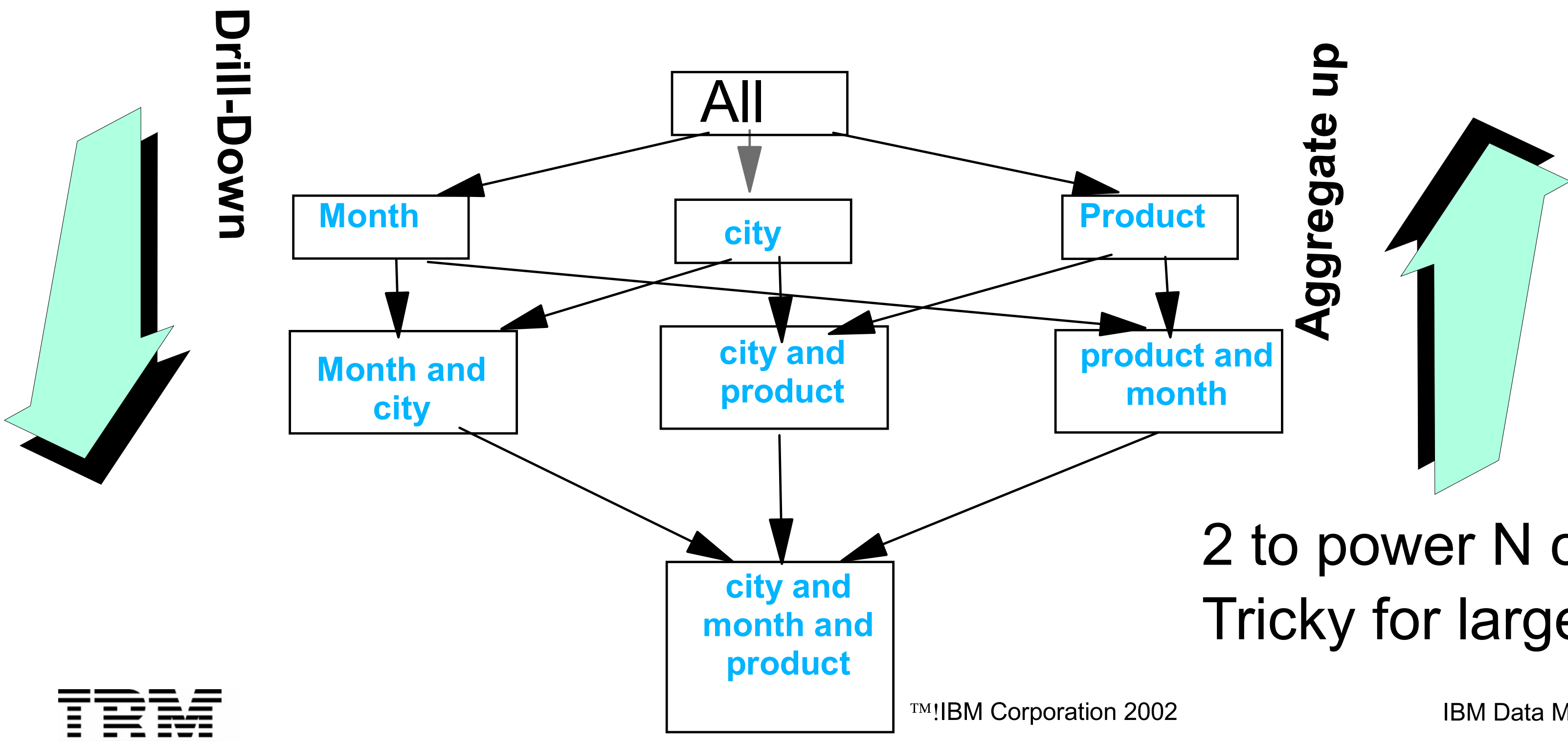


Hierarchical dimension



CUBE Example Query

```
Select pe.month,st.city, sa.product_id, sum(sa.units)
From sales sa, period pe. store st
Where pr.product_id = sa.product_id and
      st.store_id = sa.store_id and
      pe.year between 1995 and 1996
Group by CUBE (pe.month, st.city, sa.product.id)
```



2 to power N combinations
Tricky for large N

Hierarchical Cubes and Rolling OLAP Functions

- Hierarchical Cubes and Rolling OLAP functions: how do they interact?
- For example: rank of sales:

  We should not rank annual sales against monthly sales.

  Annual sales tend to be larger than monthly sales, and they win over monthly sales.

- We must rank at the peer level: rank month versus month, year versus year.
- ***Two kinds of ranking***:
  - Rank within all peers: Rank all monthly sales together
  - Rank within parent: Rank monthly sales within their year

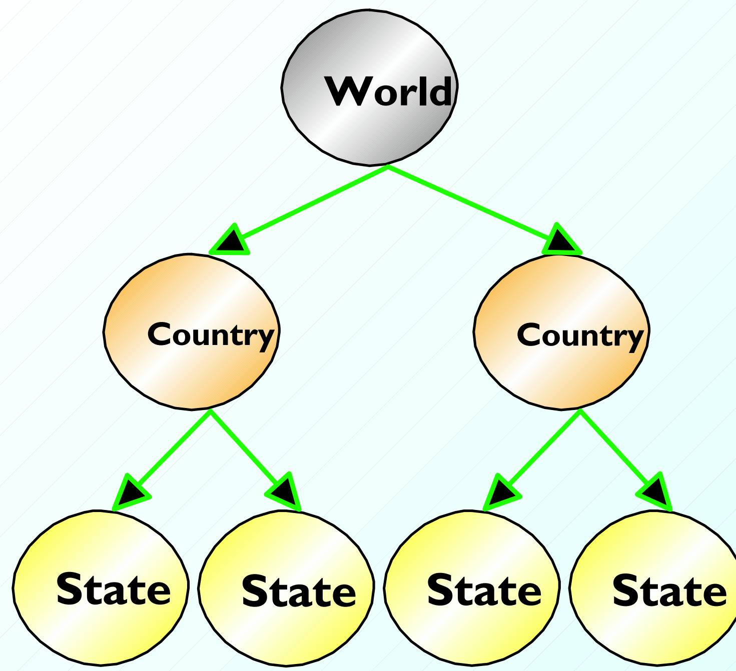
- DB2 handles all combinations.

ROLLUP - Ranking against Peers

Rollup the 1998 sales by country and state, and rank the sales among peers

```
select sum(ti.amount) as sum, loc.country, loc.state,
      grouping(loc.country) + grouping(loc.state) as lochierarchy,
      rank() over (partition by grouping(loc.country)+grouping(loc.state)
                   order by sum(ti.amount) desc ) as rank_within_peers
from stars.trans t, stars.transitem ti, stars.loc loc
where t.transid = ti.transid
  and loc.locid = t.locid
  and year(pdate) = 1998
group by rollup(loc.country, loc.state)
order by lochierarchy desc,
         rank_within_peers;
```

SUM	COUNTRY	STATE	LOCHIERARCHY	RANK_WITHIN_PEERS
4276335.41	-	-	2	1
1455411.12	USA	-	1	1
1061704.19	Canada	-	1	2
1019150.00	Germany	-	1	3
415764.83	Australia	-	1	4
324305.27	UK	-	1	5
363391.77	Germany	AB	0	1
349848.89	Germany	BC	0	2
312429.28	Canada	NB	0	3
287915.32	USA	CO	0	4
228375.32	Germany	EF	0	5
217056.57	Australia	BC	0	6
183276.29	USA	DE	0	7
...	...	...	...	...
9754.92	USA	ID	0	33
4910.12	USA	AZ	0	34



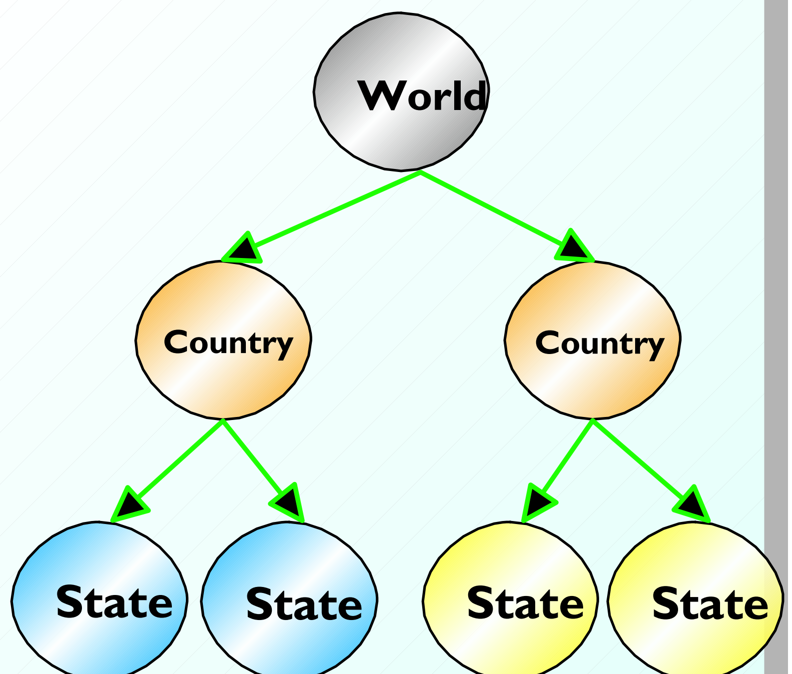
Hierarchical Cube
Ranking Within Parent

Rollup the 1998 sales by country and state, and rank the sales among peers within parent

```
select sum(ti.amount) as sum, loc.country, loc.state,
        grouping(loc.country) + grouping(loc.state) as lochierarchy,
  rank() over (partition by grouping(loc.country)+grouping(loc.state),
                   case when grouping(loc.state) = 0 then loc.country end
               order by sum(ti.amount) desc ) as rank_within_parent
from stars.trans t, stars.transitem ti, stars.loc loc
where t.transid = ti.transid
  and loc.locid = t.locid
  and year(pdate) = 1998
group by rollup(loc.country, loc.state)
order by lochierarchy,
         case when lochierarchy = 0 then loc.country end,
         rank_within_parent;
```

If we haven't rolled up the states, then partition by the country

SUM	COUNTRY	STATE	LOCHIERARCHY	RANK_WITHIN_PARENT
4276335.41	-	-	2	1
1455411.12	USA	-	1	1
1061704.19	Canada	-	1	2
1019150.00	Germany	-	1	3
415764.83	Australia	-	1	4
324305.27	UK	-	1	5
217056.57	Australia	BC	0	1
112367.50	Australia	AB	0	2
59732.92	Australia	CD	0	3
26607.84	Australia	EF	0	4
312429.28	Canada	NB	0	1
176149.63	Canada	AB	0	2
167361.54	Canada	BC	0	3
136346.53	Canada	NF	0	4
101362.88	Canada	NS	0	5
89847.04	Canada	MB	0	6
63707.96	Canada	NWT	0	7
14499.33	Canada	ON	0	8

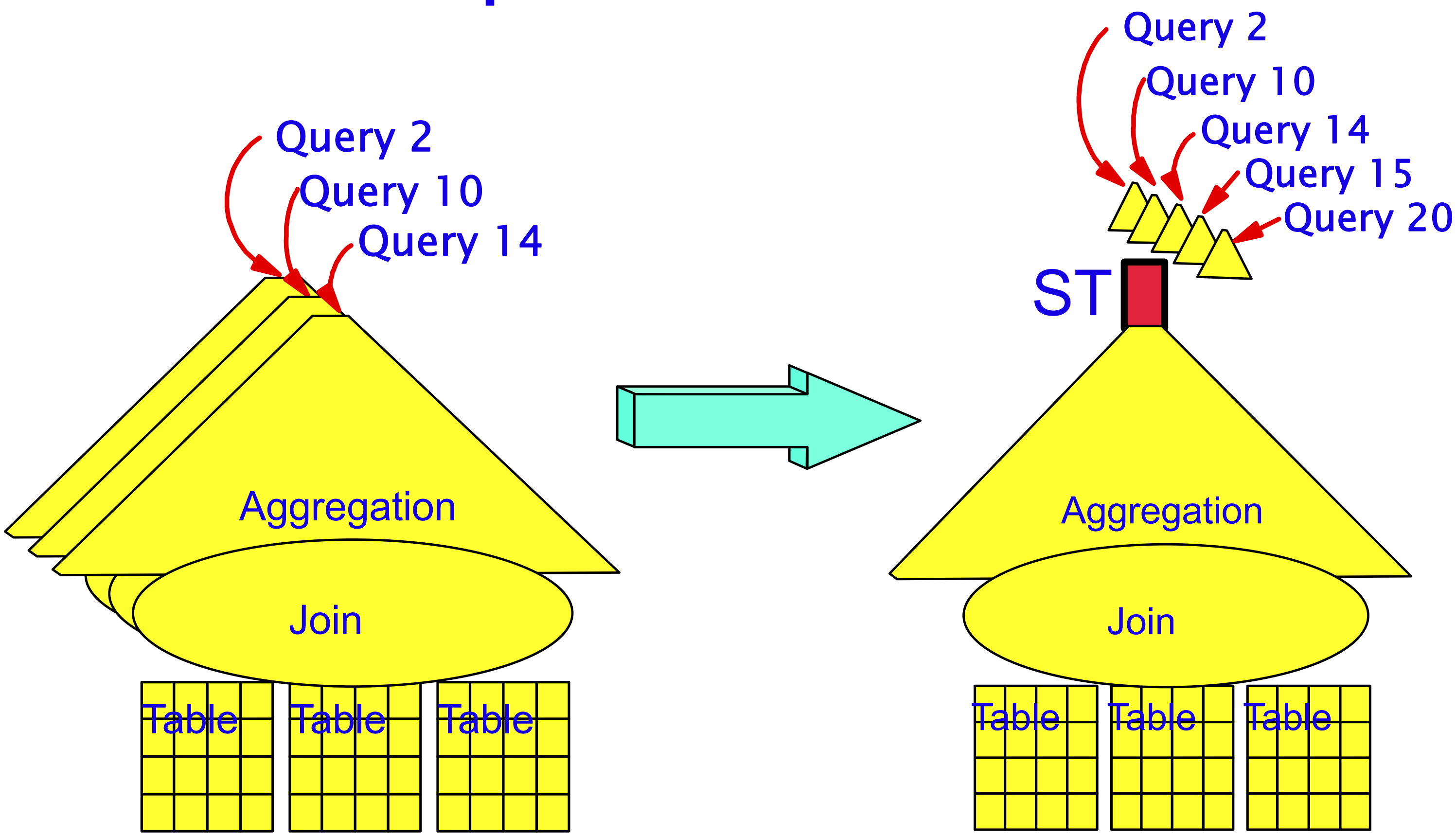


Optimization and parallelization

- Complex queries are heavily Optimized and parallelized automatically
- *Major* Optimization Feature for Business Intelligence
- Join and Aggregate Indexes
- Most queries do similar aggregations
- Usually on few dimension tables
- Precomputation is very attractive

Automatic Summary Tables (ASTs)

- ASTs are a sub-class of MQTs, due to aggregation
- Optimizer <u>automatically</u> exploits Summary Tables
- Save on huge repeat work across queries
  - **Without ST: Complete computation for each query**
  - **With ST: Precompute once and then reuse**



Aggregate Aware Optimization In DB2 UDB

- ***No change to User queries required***
- DBA predefines and pre-aggregates a set of joins/aggregates in indexes called ASTs (Automatic Summary Tables)
- Optimizer automatically/transparently exploits ASTs
- Drastic Impact: over-night queries become interactive
- Sharing the cost of join/aggregations across many queries
- DBA controls what should be precomputed and when
- Independent Partitioning/Indexing:
  - ✓ ASTs can be partitioned independent of the base data
  - ✓ ASTs can be indexed as well. This is like index on index
  - ✓ Enables optimizer to choose from many possible indexes, and possible collocated processing alternatives

Automatic Summary Table Creation

-- Aggregate Automatic Summary Table (AST)

-- Precompute popular aggregates along different dimensions.

```
CREATE TABLE dba.PG_SALESSUM_IMJ  AS (
  SELECT loc.country, loc.state,
         YEAR(pdate) AS year, MONTH(pdate) AS month,
         l.lineid AS prodline, pg.pgid AS pgroup,
         SUM(ti.amount) AS amount,  COUNT(*) AS count
  FROM   stars.transitem AS ti, stars.trans AS t,
         stars.loc AS loc, stars.pgroup AS pg, stars.prodline AS l
  WHERE  ti.transid = t.transid AND ti.pgid = pg.pgid
         AND pg.lineid = l.lineid AND t.locid = loc.locid
  GROUP BY loc.country, loc.state,              <<< region dimension
           year(pdate),month(pdate),            <<< time dimension
           l.lineid, pg.pgid                    <<< product dimension
) DATA INITIALLY DEFERRED  REFRESH IMMEDIATE;
```

-- Later, when you are ready to populate the AST issue:

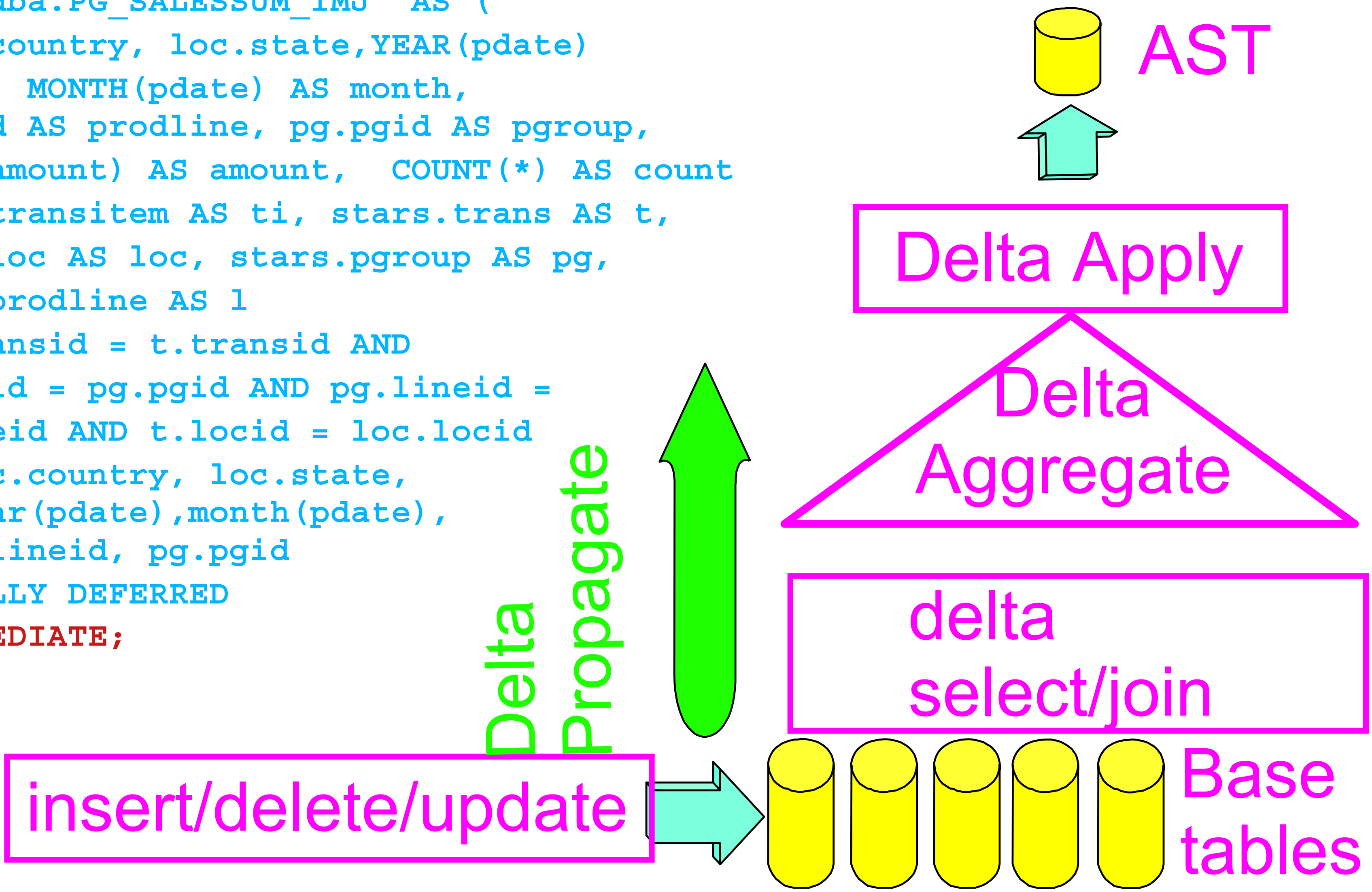
```
refresh table dba.pg_salessum;                 <<< Build

create index pg_salessumxy_pgid on dba.pg_salessum(year,month);
runstats on table dba.pg_salessum and indexes all;
```

Incremental Maintenance of AST with aggregation over join of tables

- Refresh command populates the AST initially
- DB2 automatically and efficiently synchronizes the AST with changes to the base table
  - collects the inserted/deleted/updated records (delta) for all base tables involved in AST
  - delta joins the deltas and base tables, and deltas with deltas,
  - reduces the resulting delta by aggregating the records on group by columns
  - applies the summarized delta to the AST

Incremental Maintenance of Immediate AST with aggregation over join of table

```
CREATE TABLE dba.PG_SALESSUM_IMJ  AS (
  SELECT loc.country, loc.state,YEAR(pdate)
      AS year, MONTH(pdate) AS month,
      l.lineid AS prodline, pg.pgid AS pgroup,
      SUM(ti.amount) AS amount,  COUNT(*) AS count
  FROM stars.transitem AS ti, stars.trans AS t,
       stars.loc AS loc, stars.pgroup AS pg,
       stars.prodline AS l
  WHERE ti.transid = t.transid AND
        ti.pgid = pg.pgid AND pg.lineid =
        l.lineid AND t.locid = loc.locid
  GROUP BY loc.country, loc.state,
           year(pdate),month(pdate),
           l.lineid, pg.pgid
) DATA INITIALLY DEFERRED
  REFRESH IMMEDIATE;
```



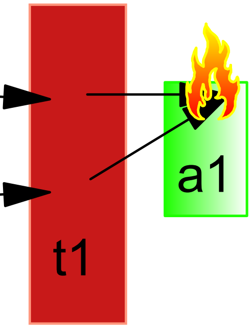
Incremental Maintenance of Deferred ASTs

- **Incremental Maintenance for Deferred ASTs**
  - I/U/D operations immediately 'propagated' to staging table
  - When AST is refreshed, they are done so incrementally
- **Avoids lock contention that may result with Immediate ASTs when multiple transactions are updating the base table simultaneously**
  - "Hot spot" could exist at aggregation points

V7

```
CREATE TABLE t1 (c1 INT, c2 INT)
CREATE TABLE a1 AS
  (SELECT c1, COUNT(*) as count
   FROM t1 GROUP BY c1)
   DATA INITIALLY DEFERRED REFRESH IMMEDIATE
SET INTEGRITY FOR a1 IMMEDIATE CHECKED
```

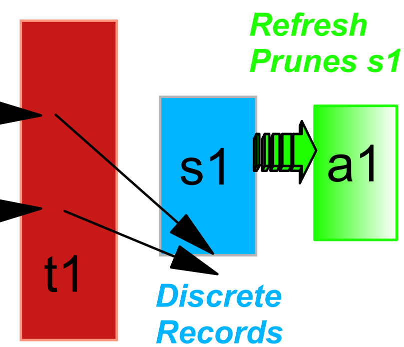
```
TRAN 1:
   INSERT INTO t1
   VALUES (1,2)

TRAN n:
   INSERT INTO t1
   VALUES (1,100)
```



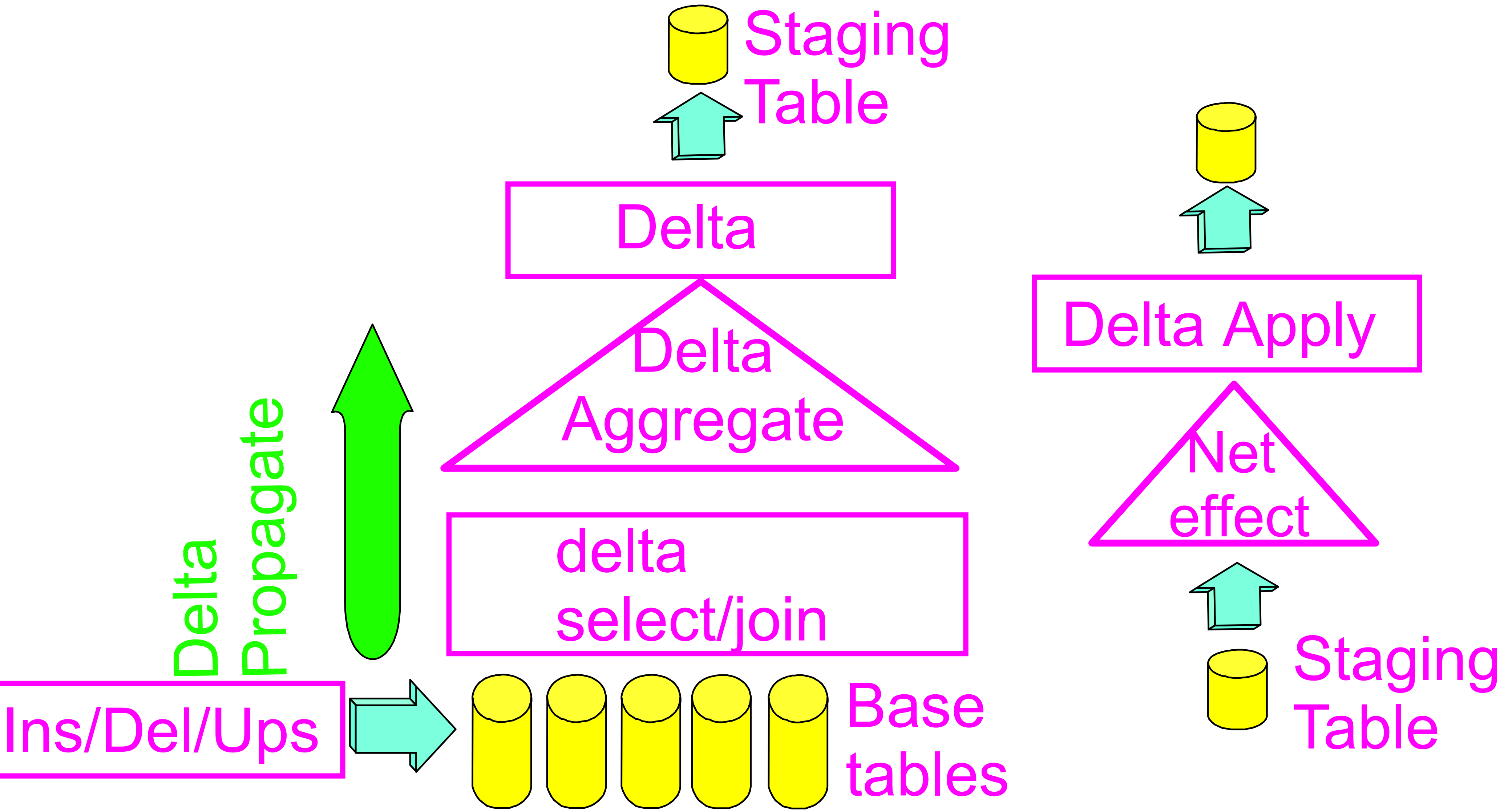
New! V8

```
CREATE TABLE t1 ...
CREATE TABLE a1 AS ...
    DATA INITIALLY DEFERRED REFRESH DEFERRED
SET INTEGRITY FOR a1 ...
CREATE TABLE s1 FOR a1 PROPAGATE IMMEDIATE
SET INTEGRITY FOR s1 IMMEDIATE CHECKED
...
REFRESH TABLE a1    // prunes s1
```

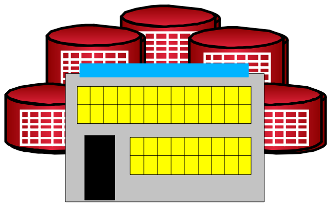
```
TRAN 1:
   INSERT INTO t1
   VALUES (1,2)

TRAN n:
   INSERT INTO t1
   VALUES (1,100)
```



Incremental Maintenance of Deferred AST with aggregation over join of table

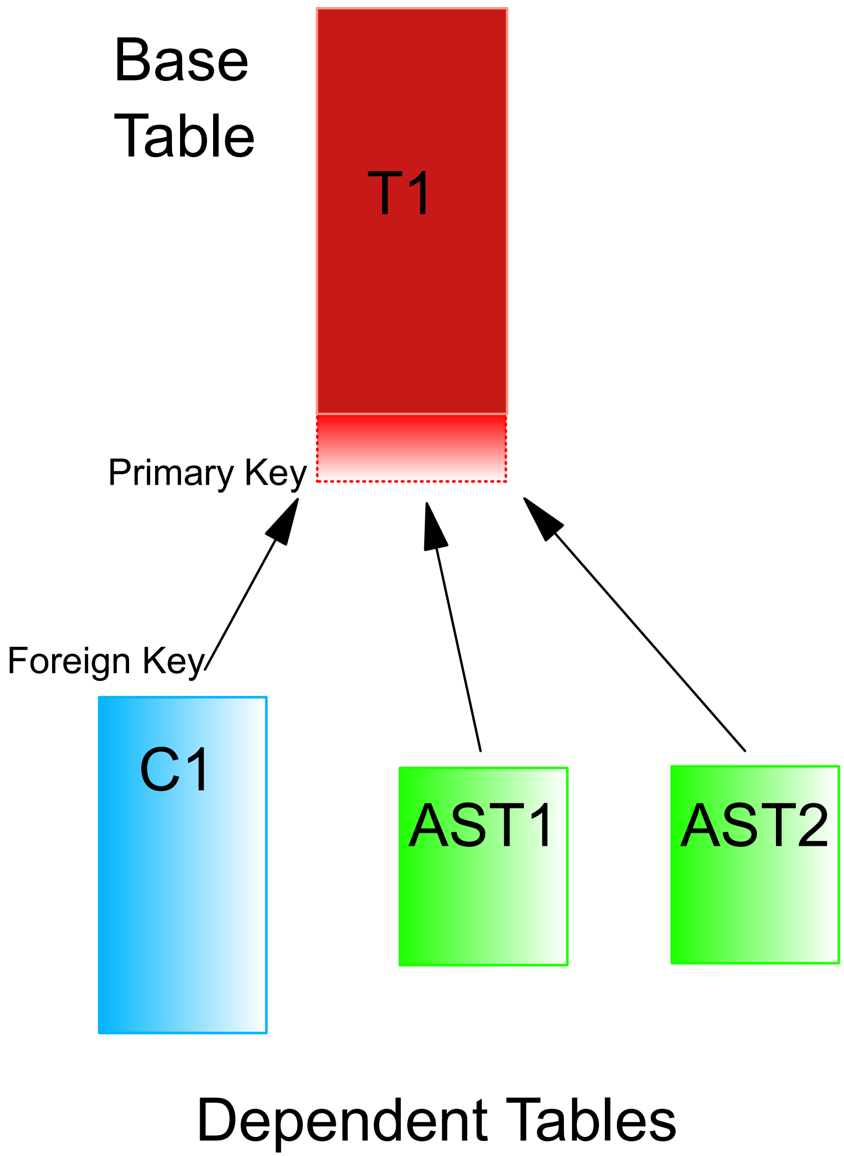


Materialized View Maintenance Interaction with Data Loading



Data Warehouse

Online Load and AST Maintenance interaction



On-line Load new in V8

```
LOAD INSERT INTO T1 ...
     ...
REFRESH TABLE AST1
     ...
REFRESH TABLE AST2
```

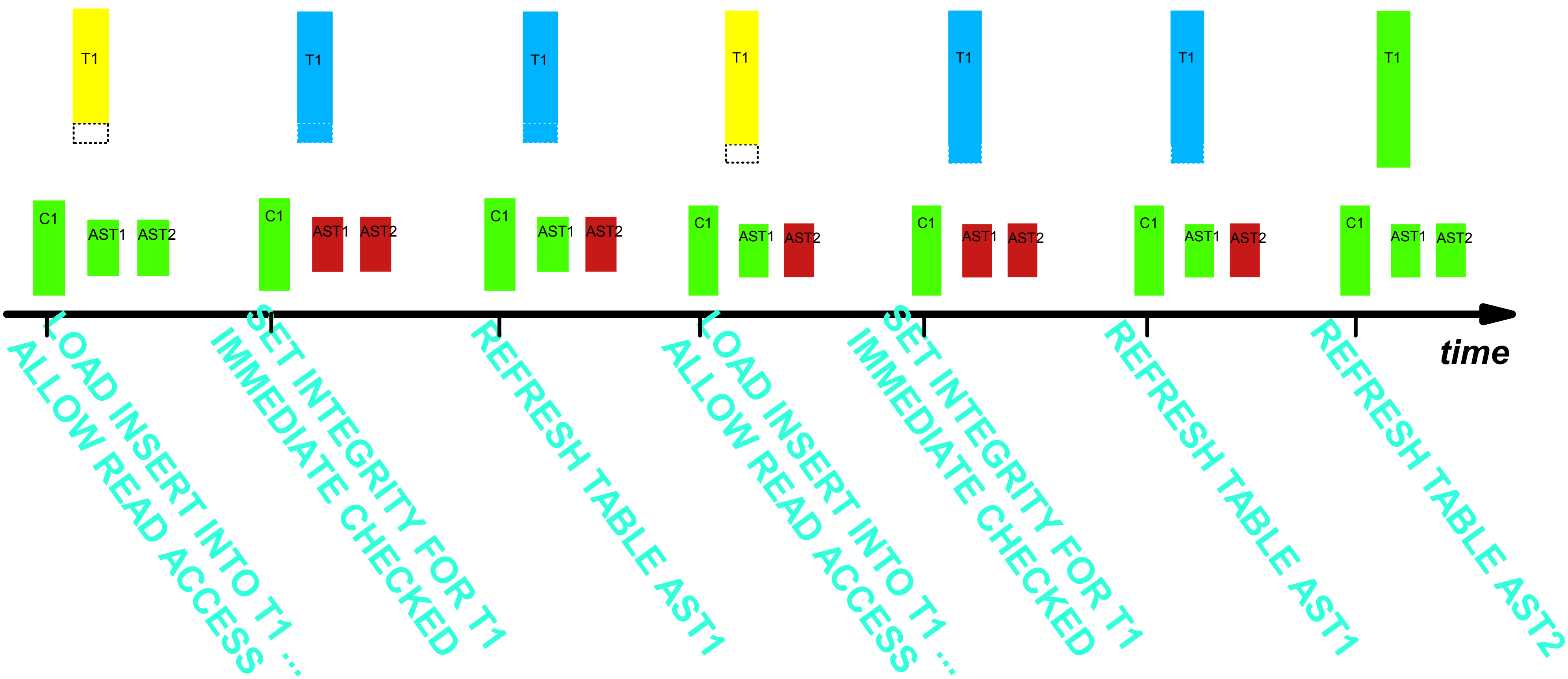
Existing Behavior:

- **T1, C1, AST1 an AST1 put into "CHECK PENDING" state**
  - No Access Allowed
  - "CASCADE IMMEDIATE"
- **AST1 and AST2 Fully Refreshed**
  - Full Scan of T1 Required

New! V8 Features:

- **"CASCADE DEFERRED"**
  - Check pending on dependent tables eliminated or minimized
- **INCREMENTAL AST REFRESH**
  - Delta of underlying table used to incrementally refresh ASTs

Online Load and AST Maintenance Example



- Full Read/Write Access
- Check Pending - Read Only
- No Data Movement - I/U/D Allowed if doesn't affect ASTs
- No Access

Online Load and AST Maintenance Example

Operation	T1	C1	AST1	AST2
LOAD INSERT INTO T1 ... ALLOW READ ACCESS ...	"Check-Pending / Read Access" (Existing portion of table can be read)	Full Access	Full Access	Full Access
SET INTEGRITY FOR T1 IMMEDIATE CHECKED	Constraints incrementally checked Enters "No Data Movement" state (Full Access except those that can move RIDs (eg REORG; update partition key)	Full Access	No Access	No Access
REFRESH TABLE AST1	"No Data Movement"	Full Access	Incrementally Refreshed Full Access	No Access
LOAD INSERT INTO T1 ... ALLOW READ ACCESS ...	"Check-Pending / Read Access" (Existing and data from first load is visible)	Full Access	Full Access (Existing and data from first load is visible)	No Access
SET INTEGRITY FOR T1 IMMEDIATE CHECKED	Constraints incrementally checked (Only data from 2nd load) Enters "No Data Movement" state	Full Access	No Access	No Access
REFRESH TABLE AST1	"No Data Movement"	Full Access	Incrementally Refreshed (W.r.t. 2nd load) Full Access	No Access
REFRESH TABLE AST2	Full Access	Full Access	Full Access	Incrementally Refreshed (W.r.t. both loads) Full Access

New ! V8

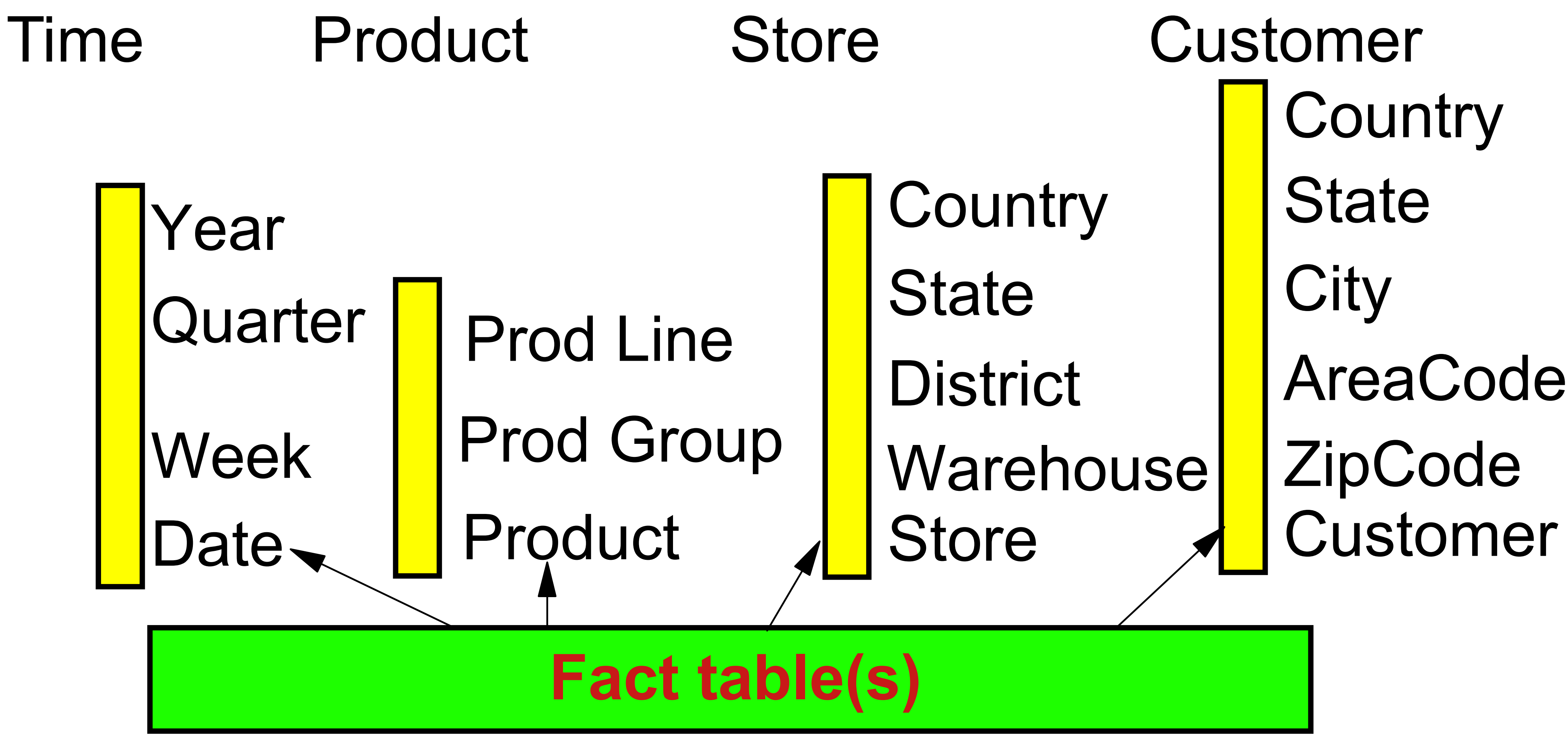
Advanced ASTs And High Dimensionality

Solving the problem of AST proliferation

Exponential Explosion of ASTs

Aggregate BY Time X Prod X Store X Cust

Large number of combinations



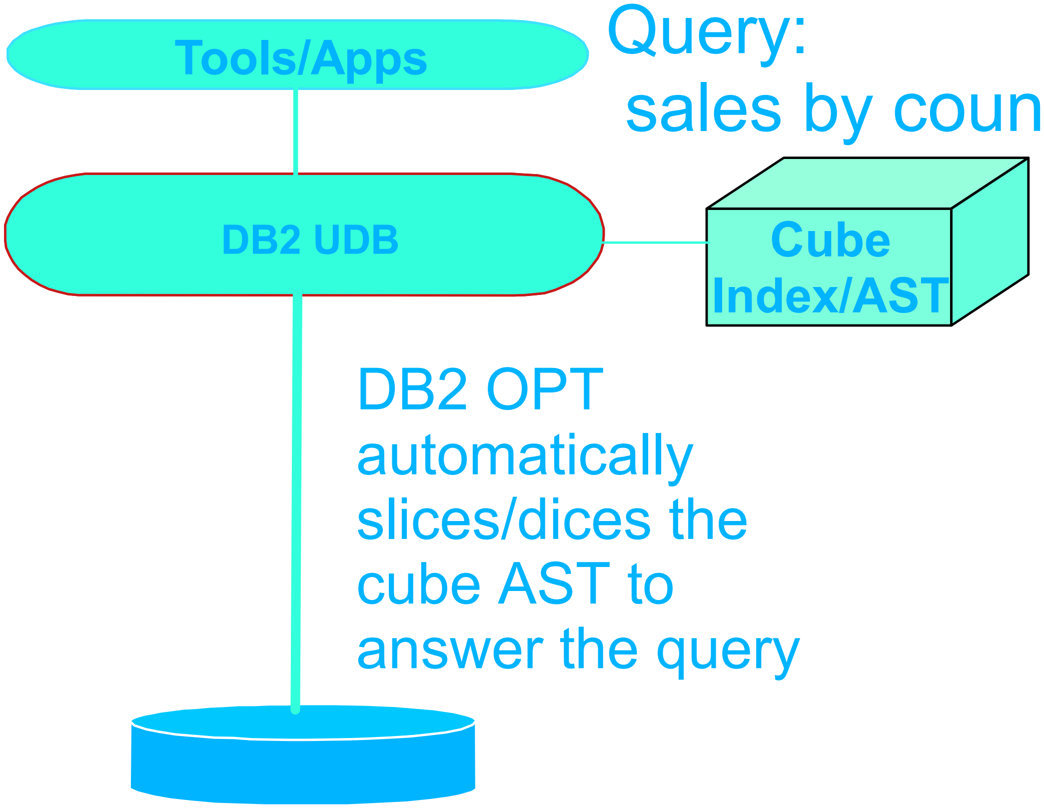
Advanced ASTs

- Cube ASTs
- Swiss cheese cube ASTs
- AST Consolidation
  Reducing Impact On Batch Windows

Cube Asts with Cube Slicing&Dicing

Pushing OLAP Cubes into DB2 UDB

Query:
sales by country for (USA, Canada)



DB2 OPT
automatically
slices/dices the
cube AST to
answer the query

DBA defines
ONE Hierarchical Cube AST:

```
sum(sales), count(sales),
sum(profit)
BY: ROLLUP(year,month)
    ROLLUP(country,state)
    ROLLUP(prodline,
           prodgroup)
```

Equivalent to 27 regular ASTs.

Universal use:
Transparent Exploitation by BI tools (Microstrategy, Cognos, Essbase,Brio, ...)

Cube Asts with Cube Slicing&Dicing

Pushing OLAP Cubes into DB2 UDB

- One hierarchical Cube AST can do the work of 100's of regular ASTs
- Huge reduction in time/resources needed to populate cube ASTs due to computation sharing
- Advanced optimization technology allows automatic slicing&dicing of AST cubes to answer queries (e.g., sales by country for (USA, Canada))

AST Cube interaction with *Multi-Dimensional Clustering (MDC)*

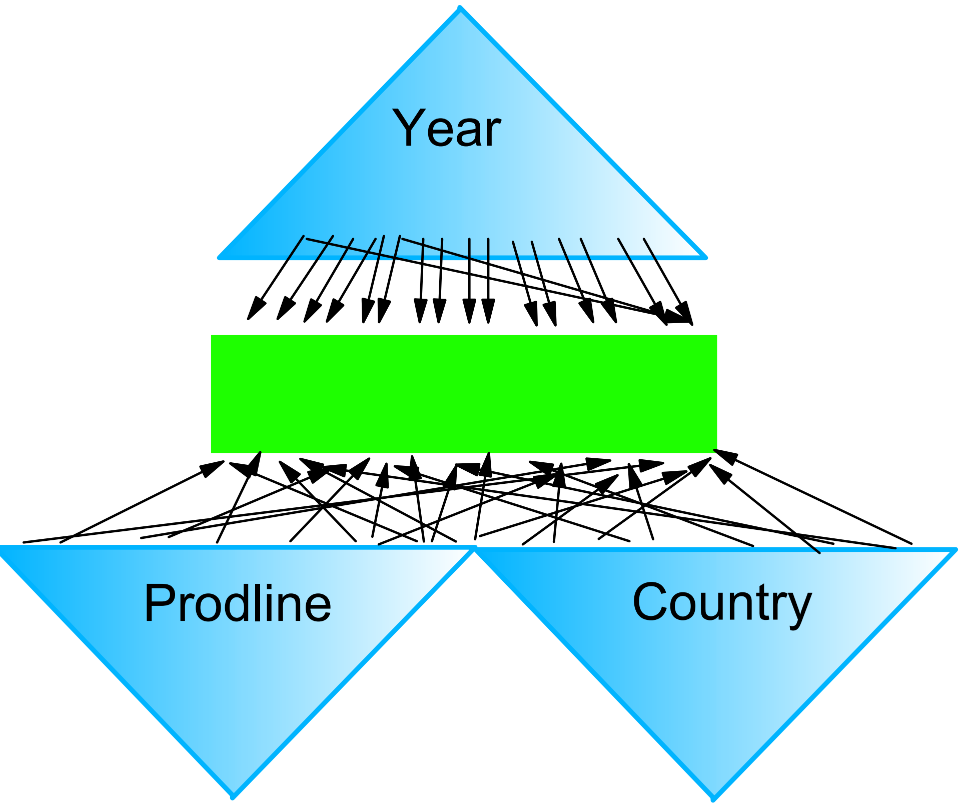
```
CREATE TABLE cube AS
   (SELECT SUM(amount) as sum, COUNT(*) as cnt,
               country, state, year(pdate) as year, month(pdate) as month,
               day(pdate) as day, prodline, prodgroup
    FROM   transitem, trans, loc, pgroup
    WHERE ...
    GROUP BY ROLLUP(year, month, day)
                  ROLLUP(country, state),
                  ROLLUP(prodline, prodgroup)
    ORGANIZE BY (year, country, prodline)
  ) DATA INITIALLY DEFERRED REFRESH DEFERRED;
```

New! V8

In addition to run-time slicing & dicing of cube, this also exploits multiple clustered index and storage layout on disk too.

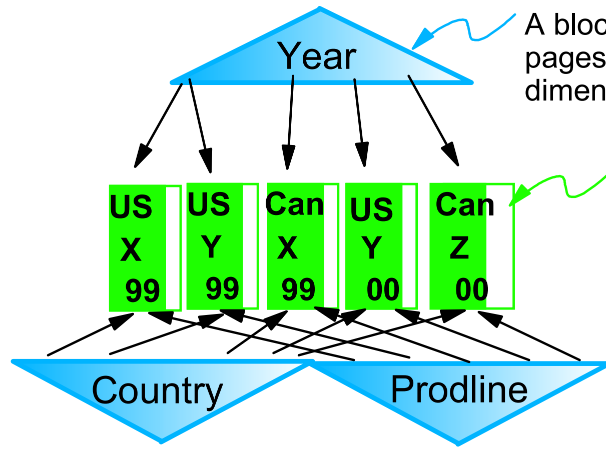
Allows for further exploitation of multidimensional clustering due to disk layout

AST Cube interaction with *Multi-Dimensional Clustering (MDC)*



Prior to MDC

- All indexes RECORD-based
- Clustering in one dimension only
- Clustering NOT guaranteed (degrades once page free space is exhausted)



"Block indexes" are just like normal indexes, except they have pointers to blocks instead of individual records. A block is a group of consecutive pages with the same key values in all dimensions.

All records in this block are from country **Canada,** product line **Z,** and the year **2000**

With MDC

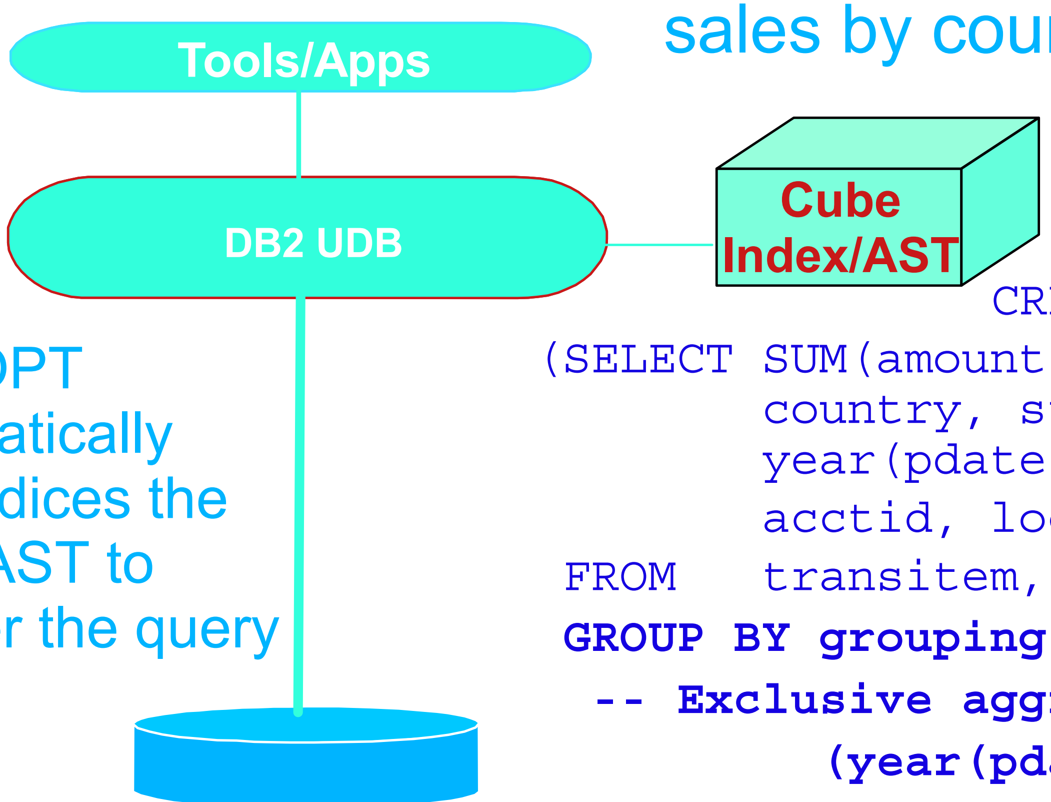
- Tables managed by BLOCK according to defined clustering dimensions
- ***Clustering guaranteed !***
  - each insert transparently places a row in an existing block which satisfies all dimensions, or creates a new block
- Dimension indexes are BLOCK-based
  - results in much smaller indexes
  - RECORD-based indexes also supported
- ***Queries in clustering dimensions only do I/Os absolutely necessary for selected data***

Swiss Cheese Cube Asts
With Slicing/Dicing Optimization

- Detailed cubes can get very large
  (even larger than fact table)
- Solution: precompute a subset of full cubes
  (precomputed cubes with holes
  ==> Swiss Cheese Cubes)
- DB2 UDB supports Swiss Cheese Cube ASTs
- To answer queries, DB2 optimizer automatically exploits
  Swiss Cheese Cubes
  By Slicing and Dicing
- Huge reduction in time/resources needed
  to populate due to computation sharing

Swiss Cheese Cube Asts With Slicing/Dicing Optimization

Query:
sales by country for (USA, Canada)



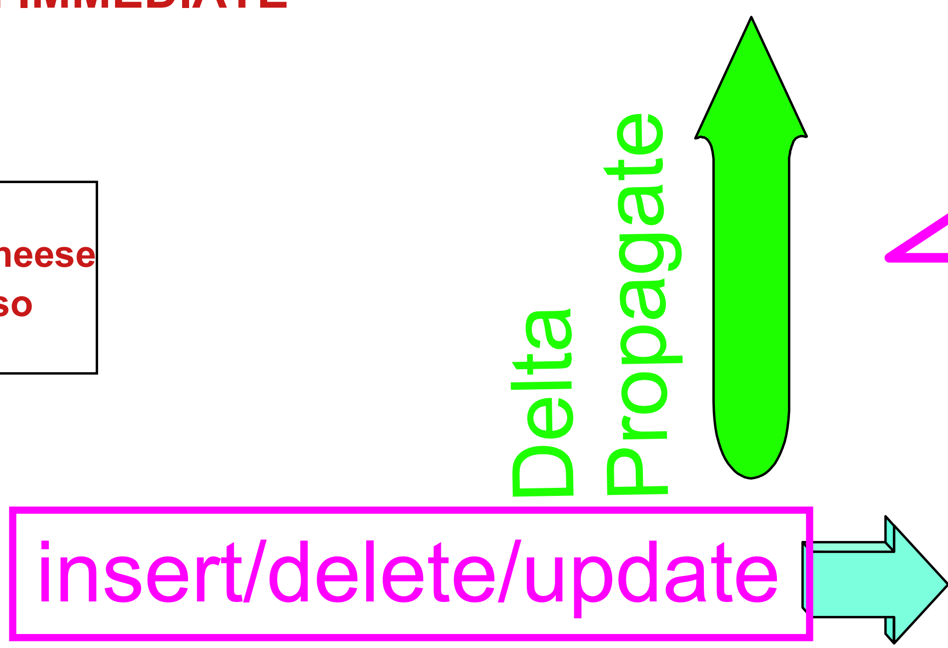
DB2 OPT automatically slices/dices the cube AST to answer the query

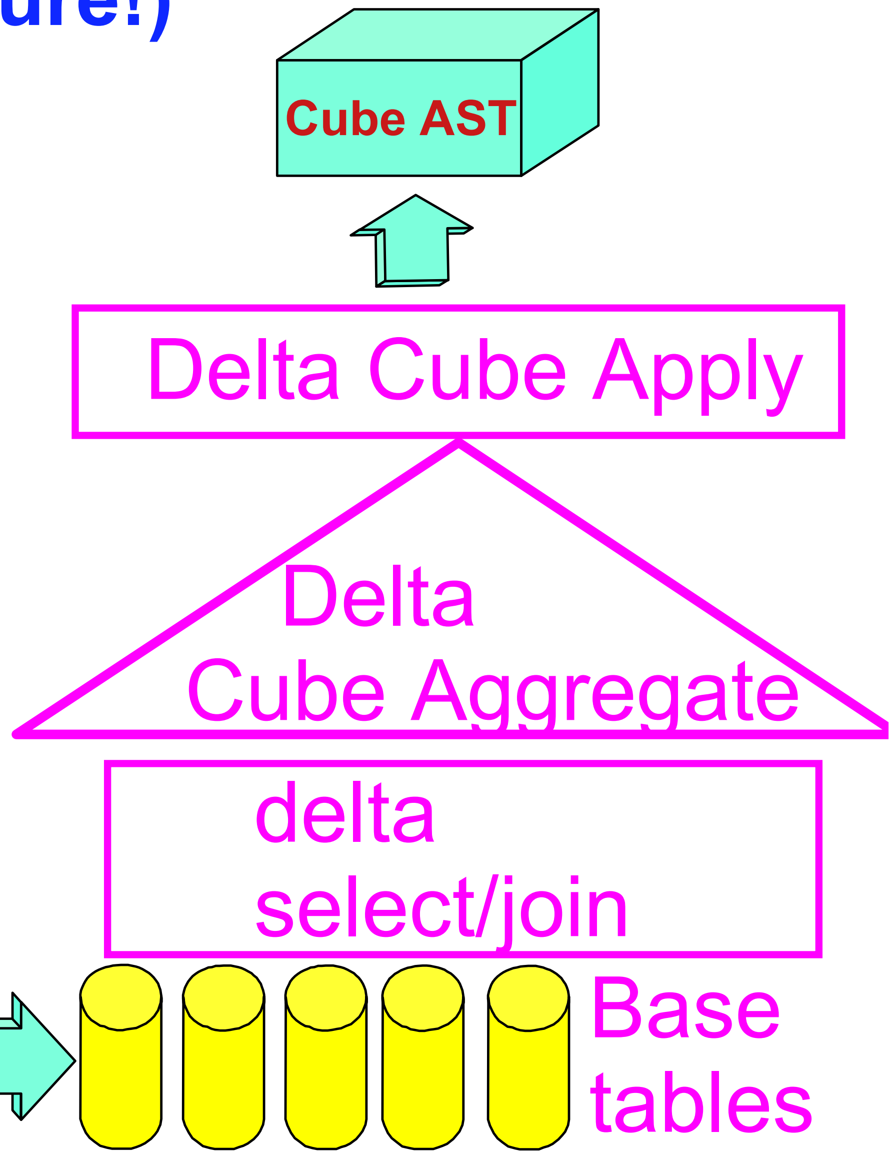
```
                    CREATE TABLE dba.swisscubeast  AS
(SELECT SUM(amount), COUNT(*)
        country, state, city,
        year(pdate), month(pdate), day(pdate),
        acctid, locid, status,
 FROM   transitem, trans, loc, pgroup WHERE ...
 GROUP BY grouping sets(
  -- Exclusive aggregation for time dimension
         (year(pdate), month(pdate), day(pdate)),
  -- Exclusive aggregation for region dimension
         (country, state, city),
  -- Cross dimensional aggregation
         (year(pdate), month(pdate), locid,
          acctid, pgid)
 ) ) DATA INITIALLY DEFERRED REFRESH DEFERRED;
```

Incremental Maintenance of Cube ASTs and Swiss Cheese (sparse) cube ASTs (Advance Feature!)

CREATE TABLE dba.cubeast AS (
... cube ast, swiss cheese (sparse) cube ast
) DATA INITIALLY DEFERRED
REFRESH IMMEDIATE

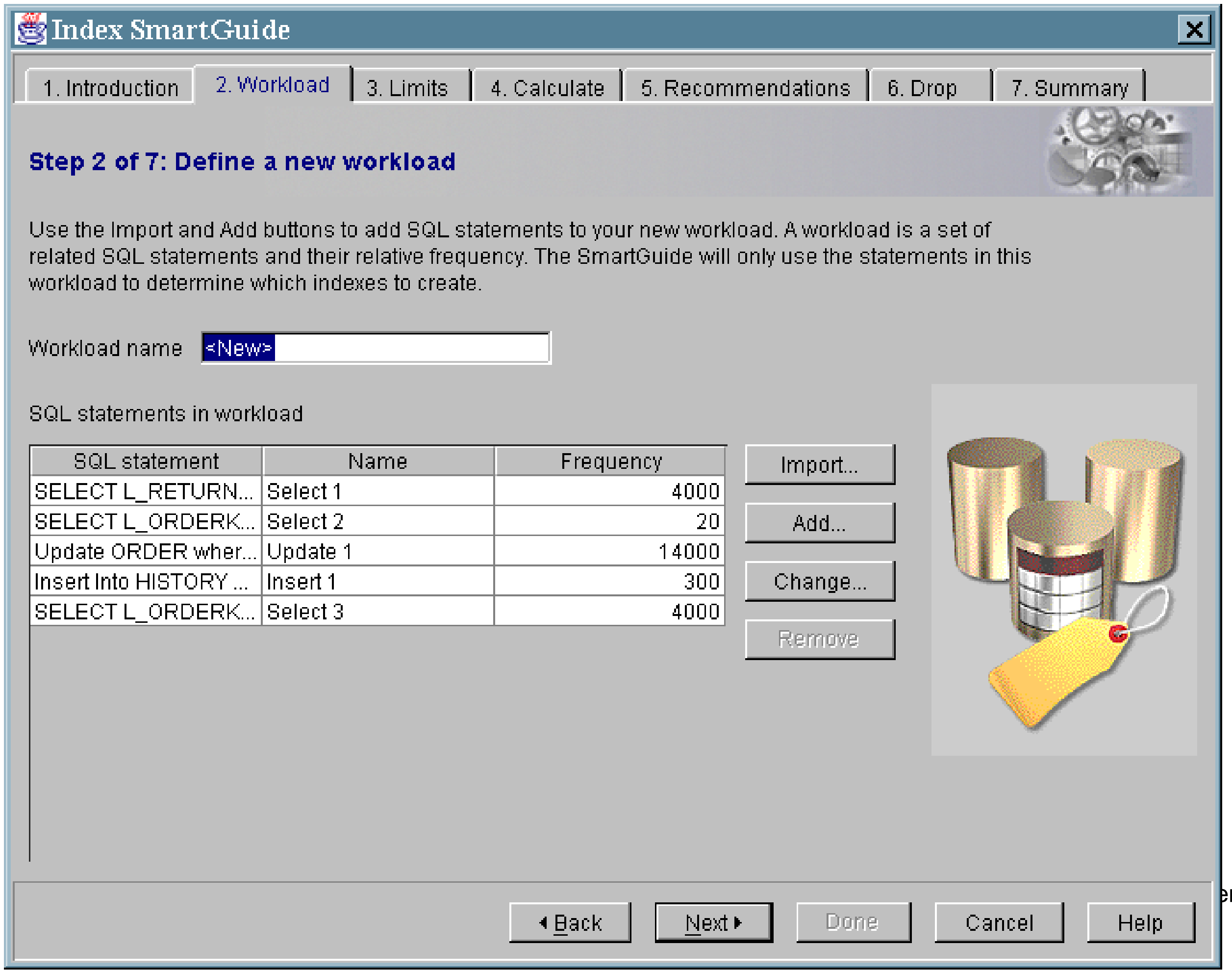
Note:
Deferred swiss cheese Cube ASTs also Supported!





DBA - Index Wizard

- Index wizard/SmartGuide
  - Given a workload of one or more SQL statements and some constraints (e.g. index space, computation time limit), find a set of indexes designed to maximize performance
    - Make it easy for the DBA to find the "right" set of indexes
    - Reduce complexity of performance analysis and tuning



DBA - MQT (AST) Wizard

- MQT (AST) wizard/SmartGuide
  - Given a workload of one or more SQL statements and some constraints (e.g. storage space, computation time limit), find a set of materialized views designed to maximize performance
    - Make it easy for the DBA to find the "right" set of materialized views
    - Reduce complexity of performance analysis and tuning
- Not in V8 GA, but coming soon

Summary

- Heavily engaged in e-Business, including B2C, B2B
- A mainstream e-Business DBMS relied upon by leading e-Business partners (Ariba, I2, Siebel, SAP, Net.Commerce, Broadvision, ...)
- Provides Powerful Query Capabilities and Strong Standard Compliance
- Provides advanced BI features with full optimization and parallelism
  - ✓ used by applications or BI tools (a strong BI ISV program)
  - ✓ Advanced cube, OLAP capabilities, strong support for statistical functions
- Strong optimization, including support for ASTs, and cube slicing/dicing
  - ✓ Dramatic reduction in resource consumption,
  - ✓ Dramatic increase in number of concurrent users