**LECTURE 9**

**Relational OLAP (ROLAP)**

**Learning Goals**

- Why ROLAP?
- How to create a "Cube" in ROLAP
- Problem with simple approach
- ROLAP Issues

**Why ROLAP?**

Issue of scalability i.e. curse of dimensionality for MOLAP

- Deployment of significantly large dimension tables as compared to MOLAP using secondary storage.
- Aggregate awareness allows using pre-built summary tables by some front-end tools.
- Star schema designs usually used to facilitate ROLAP querying (in next lecture).

The biggest problem with MOLAP is the requirement of large main memory as the cube size increases. There may be many reasons for the increase in cube size, such as increase in the number of dimensions, or increase in the cardinality of the dimensions, or increase in the amount of detail data or a combination of some or all these aspects. Thus there is an issue of scalability which limits its applications to large data sets.

Despite advances in MOLAP technologies, high-end OLAP implementations will normally require assistance from a relational database. Hence a ROLAP or Relational OLAP. ROLAP tools will query the relational database using SQL generated to conform to a framework using the facts and dimensions paradigm using the star schema.

The other approach is "aggregate awareness" i.e. the environment is smart enough to develop or compute higher level aggregates using lower level or more detailed aggregates.

**ROLAP as a "Cube"**

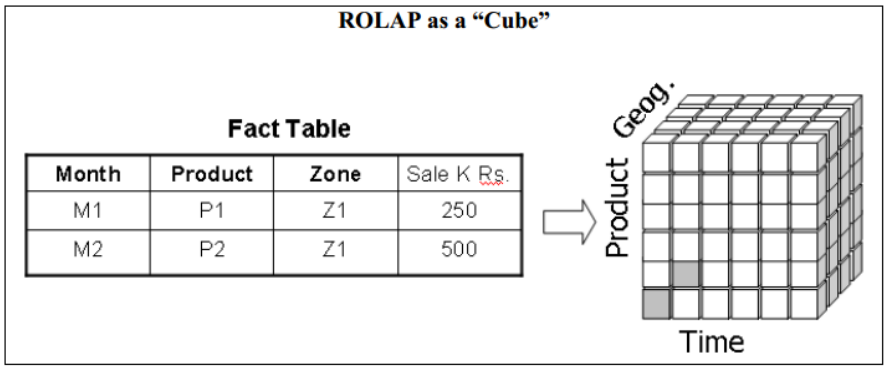
OLAP data is stored in a relational database (e.g. a star schema)

The fact table is a way of *visualizing as* an "un-rolled" cube.

So where is the cube?

It's a matter of perception

Visualize the fact table as an elementary cube



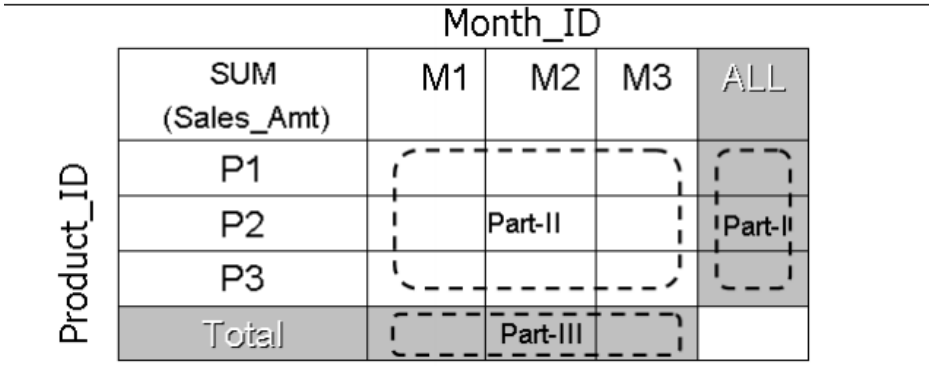

**ROLAP as a cube**

Recall from the last lecture that an implementation of the OLAP framework will be accepted as OLAP after it passes the FAS**M**I test i.e. it has to be **M**ulti-dimensional. The question is "how to convert something into a cube" which is intrinsically not a cube. And more importantly, if MOLAP has the space requirement limitations, and to overcome those limitations we are using a different implementation, then wouldn't the conversion back to a "cube" defeat the purpose? This is actually a very good question, and needs some detailed explanation. Fig-12.1 shows two rows of a fact table, and the aggregates corresponding to the cells of the fact table are shown correspondingly in a three dimensional cube. Extending this argument, each and every cell of the fact table can have a corresponding mapping. To clarify this further, consider the analogy of the map of the world. Our planet earth is NOT flat, it is spherical in nature i.e. three dimensional, yet the map of the earth is flat i.e. two dimensional in nature i.e. there is a mapping from 3D space to 2D space. If you look closely at the map, you will see that the map is divided into a grid based on longitude and latitude, and the corresponding cells or rectangles are not of the same size.

Similarly a 2D table can be considered to be a mapping or representation of a multi-dimensional cube or vice-a-versa.

**How to create a "Cube" in ROLAP**

- **Cube is a logical entity containing values of a certain fact at a certain aggregation level at an intersection of a combination of dimensions.**
- **The following table can be created using 3 queries**




**Creating tables from queries**

When we talked of a cube for a MOLAP, it was not actually a physical cube, but was a logical entity. We continue with that concept, and assume that what was stored in a cube at a certain combination of indexes, corresponding to such a group of indices, we store the aggregates in a two dimensional table, and we use such groups of tables to store the same data that was stored in a MOLAP. The table shown in fig-12.2 is divided into three parts shown shaded and also shown by dotted lines. Corresponding to each part of the table, there is a query and consequently the table can actually be filled using three SQL queries as follows:

- **For the table entries, without the totals**

```
SELECT     S.Month_Id, S.Product_Id, SUM(S.Sales_Amt)
FROM       Sales
GROUP BY S.Month_Id, S.Product_Id;
```

- **For the row totals**

```
SELECT      S.Product_Id, SUM (Sales_Amt)
FROM        Sales
GROUP BY S.Product_Id;
```

- **For the column totals**

```
SELECT     S.Month_Id, SUM (Sales)
FROM       Sales
GROUP BY   S.Month_Id;
```

The first query can be used to fill Part-II of the table, the second query used to fill Part-I of the table, and the third query used to fill Part-III of the table, thus using these three queries, we create a ROALP structure.

**Problem with simple approach**

Number of required queries increases exponentially with the increase in number of dimensions.

It's wasteful to compute all queries.

In the example, the first query can do most of the work of the other two queries

If we could save that result and aggregate over Month_Id and Product_Id, we could compute the other queries more efficiently

Using typical SQL to fill-up the tables quickly runs into a problem, as the number of dimensions increases, the number of aggregates also increases, and the number of queries required to calculate those aggregates also increases. Actually it becomes extremely wasteful to compute all queries, wasteful, because if we are smart, we can use the results of the queries already computed to get the answers to new queries. How to do this? it is not very difficult. For example for the column total queries, we could just add the aggregates over the results of the months. So the moral of the story is "Work smart not hard".

**Cube clause**

The CUBE clause is part of SQL:1999

GROUP BY CUBE (v1, v2, …, vn)

Equivalent to a collection of GROUP BYs, one for each of the subsets of v1, v2, …, vn

The other problem with using standard SQL is that one has to write too many statements and that could lead to mistakes. Therefore, back in 1999 a CUBE clause was made part of SQL, and that clause is equivalent to a collection of GROUP BY clauses.

Some students who did a BS final year project with me of an HOLAP implementation, used dynamic web page generation to dynamically generate SQL instead of hard-coding the queries to generate the aggregates. Meaning, they used SQL to generate aggregates to fill a MOLAP cube. The project was a success, all got jobs based on this work; the first prize in the 13th Annual National Software Competition along with a cash prize of Rs. 30,000 was a bonus.

**ROLAP and Space Requirement**

If one is not careful, with the increase in number of dimensions, the number of summary tables gets very large

Consider the example discussed earlier with the following two dimensions on the fact table...
Time: Day, Week, Month, Quarter, Year, All Days

Product: Item, Sub-Category, Category, All Products

OK so we worked smart and got around the problem of aggregate generation. But the aggregates once generated have to be stored somewhere too i.e. in tables as this is a ROLAP environment.

Be warned: Pre-aggregates can very quickly get out of control in a ROLAP environment. Do not try to pre-aggregate all combinations of all dimensions at all levels of the hierarchies. If you do, the storage and maintenance costs will quickly overwhelm your implementation.

For example, consider the combinatorial explosion with just two dimensions as shown above...

**EXAMPLE: ROLAP & Space Requirement**

**A naïve implementation will require all combinations of summary tables at each and every aggregation level.**

❷

| | 2001 | | | | 2002 | | | |
|---|---|---|---|---|---|---|---|---|
| | Q1 | Q2 | Q3 | Q4 | Q1 | Q2 | Q3 | Q4 |
| Orange juice | 232 | 2,432 | 4,353 | 354 | 535 | 345 | 7,897 | 789 |
| Rola-Kola | 2,342 | 243 | 353 | 4,535 | 5,655 | 4,424 | 789 | 798 |
| 8-UP | 2,424 | 3,131 | 1,313 | 5,675 | 567 | 5,675 | 789 | 9,797 |
| Pola-Kola | 242 | 3,112 | 567 | 646 | 567 | 567 | 789 | 798 |
| Mango juice | 2,342 | 243 | 243 | 4,564 | 564 | 1,232 | 242 | 4,553 |
| Bubbly-UP | 3,453 | 3,453 | 535 | 2,422 | 2,131 | 242 | 1,321 | 245 |
| Apple juice | 253 | 456 | 2,433 | 567 | 2,442 | 5,453 | 4,566 | 345 |

❸

| | 2001 | | | | 2002 | | | |
|---|---|---|---|---|---|---|---|---|
| | Q1 | Q2 | Q3 | Q4 | Q1 | Q2 | Q3 | Q4 |
| Soda Drinks | 8,461 | 9,939 | 2,768 | 13,278 | 8,920 | 10,908 | 3,688 | 11,638 |
| Juices | 2,827 | 3,131 | 7,029 | 5,485 | 3,541 | 7,030 | 12,705 | 5,687 |

❹

| | 2001 | 2002 |
|---|---|---|
| Orange juice | 7,371 | 9,566 |
| Mango juice | 7,392 | 6,591 |
| Apple juice | 3,709 | 12,806 |
| Rola-Kola | 7,473 | 11,666 |
| 8-UP | 12,543 | 16,828 |
| Bubbly-UP | 9,863 | 3,939 |
| Pola-Kola | 4,567 | 2,721 |

❿

| | 2001 | 2002 |
|---|---|---|
| Juices | 18,472 | 28,963 |
| Soda Drinks | 36,447 | 37,156 |

24 summary tables, add in geography, results in 120 tables

There are 24 possible pre-aggregates just with the product and Time dimensions! Add in the geography dimension and we are quickly up to 120 pre-aggregates (and the number of levels in the hierarchies that we are assuming is very conservative). The largest aggregate will be the summarization by day and product because this is the most detailed. Clearly, we do not want to pre-aggregate all of these combinations.

Smart tools will allow less detailed aggregates to be constructed from more detailed aggregates (full aggregate awareness) at run-time so that we do not go all the way down to the detail for every aggregation. However, for this to work, the metrics must be additive (e.g., no ratios, averages, etc.). More detailed pre-aggregates are larger, but can also be used to build less detailed aggregates on-the-go.

**ROLAP Issues**

- Maintenance.
- Non standard hierarchy of dimensions.

- Non standard conventions.
- Explosion of storage space requirement.
- Aggregation pit-falls.

Creating an aggregate is relatively easy as compared to keeping it updated, the maintenance issue will eat you alive, if you (say) you get data late, and it is also to be reflected in the aggregate archives.

Dimensions are not just simply atomic item, then SKU, then product category. They can cross boundaries too and can become expensive to create.

The conventions for (say) week may be absolute within an organization, but differ across organizations, worst they could differ across the organization too e.g. marketing people looking differently at the week as compared to the accounts people.

**ROLAP Issue: Maintenance**

Summary tables are mostly a maintenance issue (similar to MOLAP) than a storage issue.

Notice that summary tables get much smaller as dimensions get less detailed (e.g.,year vs. day).

Should plan for twice the size of the un-summarized data for ROLAP summaries in most environments.

Assuming "to-date" summaries, every detail record that is received into warehouse must aggregate into EVERY summary table.

It is not unusual for the collection of pre-aggregate summary tables to take up significantly more space (e.g., double or higher) than the raw data tables in a deployment that is focused heavily on query performance delivery. However, the maintenance cost for keeping all of the pre-aggregates up-to-date with the detailed data (very important for consistent query results) is very high. Pre-aggregate summary tables in the relational environment have the same “update problem” as cubes in a MOLAP environment.

**ROLAP Issue: Hierarchies**

Dimensions are NOT always simple hierarchies

Dimensions can be more than simple hierarchies i.e. item, subcategory, category, etc

The product dimension might also branch off by trade style that cross simple hierarchy boundaries such as:

Looking at sales of air conditioners that cross manufacturer boundaries, such as COY1, COY2, COY3 etc.

Looking at sales of all "green colored" items that even cross product categories (washing machine, refrigerator, split-AC, etc.)

Looking at a combination of both.

It is quite common that dimensions are more than simple hierarchies. A simple product hierarchy might include item, subcategory, category, etc.However; the product dimension might also branch off by trade style in ways that cross boundaries of the simple hierarchy. We may want to look at sales of air conditioners that cross manufacturer boundaries, such as COY1, COY2, COY3 etc. Looking at all "green colored" items will cross product categories (washing machine, refrigerator, split-AC, etc.). Looking at a combination of both, or other will result in a combinatorial explosion, as these combinations get very large - such that brute-force pre-aggregation is not practical.

Providing analytic capability along multiple calendar hierarchies is even more unmanageable in terms of pre-aggregate construction.

**ROLAP Issue: Semantics**

Conventions are NOT absolute

**Example:** What is calendar year? What is a week?

- Calendar:

01 Jan. to 31 Dec or 01 Jul. to 30 Jun. or 01 Sep to 30 Aug.

- Week:

Mon. to Sat. or Thu. to Wed.

Conventions may vary across organizations, and even within an organization. Consider the apparently simple case of calendar year. There is no absolute standard definition of calendar year i.e. is it Jan to Dec. or Jun to Jul or any other time period covering 12 months? Similarly what is meant by week i.e. it starts from which day and ends at which day? Even within an organization the nomenclature may vary, for example finance people may consider a Mon. to Sun. week, while the marketing people may consider a Wed. to Tue. Week. Consequently the results of aggregation will vary across the organization for the same data, thus creating lot of confusion. To bring all the departments on the same grid may require lot of arm twisting from the highest level and will involve office politics too.

**ROLAP Issue: Storage Space Explosion**

Summary tables required for non-standard grouping

Summary tables required along different definitions of year, week etc.
Brute force approach would quickly overwhelm the system storage capacity due to a combinatorial explosion.

We just saw the problem of semantics and thought it was complex, we were wrong! There are many more and other non-standard grouping too. For example during the month of August to capitalize on the patriotism because of the Independence Day, different manufacturers print the national flag on the packages. This adds a new grouping, as the decision maker would be interested in knowing the sales of item with flag as compared to items without a flag printed on them.

In some cases bringing all the departments on the same grid for agreeing on the same definition of year or week may not be advisable, in such a case the number of summary tables increases, as tables are required for each definition of the week and the year. One may naively follow the brute force path i.e. creating all possible summary tables with all possible definitions, groupings and nomenclatures. This may work for small databases, but for VLDB (Very Large Data Bases) very soon the memory requirements of the system will be chocked. Thus this is not a viable option for large data sets.

**ROLAP Issue: Aggregation Pitfalls**

1. Coarser granularity correspondingly decreases potential cardinality.
2. Aggregating whatever that can be aggregated.
3. Throwing away the detail data after aggregation.

As data is aggregated to higher levels in the dimensional hierarchy the number of rows retained will obviously reduce. However, this reduction is not usually directly proportional to the ratio of entries at each level in the hierarchy. For example, one might (incorrectly) think that the weekly summary table will be about 14% or 1/7th of the daily summary table as there are seven days in a week. Therefore, keeping the data at a daily level of granularity should be seven times the size of the weekly level, right? Wrong. The reason being all products that sell in a week definitely do not sell every day of the week.

As a result, the factor is 2 rather than 7. For example rental of movies usually goes up during the weekends. Most heart patients suffer a stroke on Monday, so are most of stock sales. A general rule of thumb is that aggregation from one level of detail to another should only be undertaken if there is a 10x (or more) reduction in table size from the more detailed level to the coarser level of aggregation. Of course, this is only a guideline and will be influenced by the frequency of query activity at each level of potential aggregation.

Keeping the detail is important because it is inevitable that advanced analysis will require drill down to the most granular level of detail available. Remember Murphy's Law, the day you throw away the detail, is the day it would be required.

**How to reduce summary tables?**

Many ROLAP products have developed means to reduce the number of summary tables by:

1. Building summaries on-the-fly as required by end-user applications.
2. Enhancing performance on common queries at coarser granularities.
3. Providing smart tools to assist DBAs in selecting the "best" aggregations to build i.e. trade-off between speed and space.

Tools from vendors such as Microsoft and Micro Strategy are “fully aggregate aware” with the ability to take summaries from one (more detailed) level of aggregation and roll them up into a

less detailed summary at run-time. In this way, aggregate metrics can be delivered without forcing run-time aggregation from the most detailed data in the warehouse or mart.

Wizards have come into the marketplace with the ability to make suggestions as to optimal aggregates that should be built for maximizing performance for a defined workload.

**Performance vs. Space Trade-off**

1. Maximum performance boost implies using lots of disk space for storing every pre-calculation.
2. Minimum performance boost implies no disk space with zero pre-calculation.
3. Using Meta data to determine best level of pre-aggregation from which all other aggregates can be computed.

Theoretically there can be two extremes i.e. free space and free performance. If storage is not an issue, then just pre-compute every cube at every unique combination of dimensions at every level as it does not cost anything. This will result in maximum query performance. But in reality, this implies huge cost in disk space and the time for constructing the pre-aggregates. In the other case where performance is free i.e. infinitely fast machines and infinite number of them, then there is not need to build any summaries. Meaning zero cube space and zero pre-calculations, and in reality this would result in minimum performance boost, in the presence of infinite performance.

What is meant by Meta data? Meta data is data about data. So what is the data about data one would be interested in? For example, how many entries are there at each level of the hierarchy? How many stores are there? How many stores per zone? How many zones per district? Etc. When the density of each dimension is known, it gives a fair idea where the aggregation is going to have the biggest bang for the buck. Because if I have a dimension that has for example on average two UPCs (Universal Product Code) per SKU (Stock Keeping Unit), its really not very interesting to build a summary of UPCs in SKUs because all it saves is adding two records together.

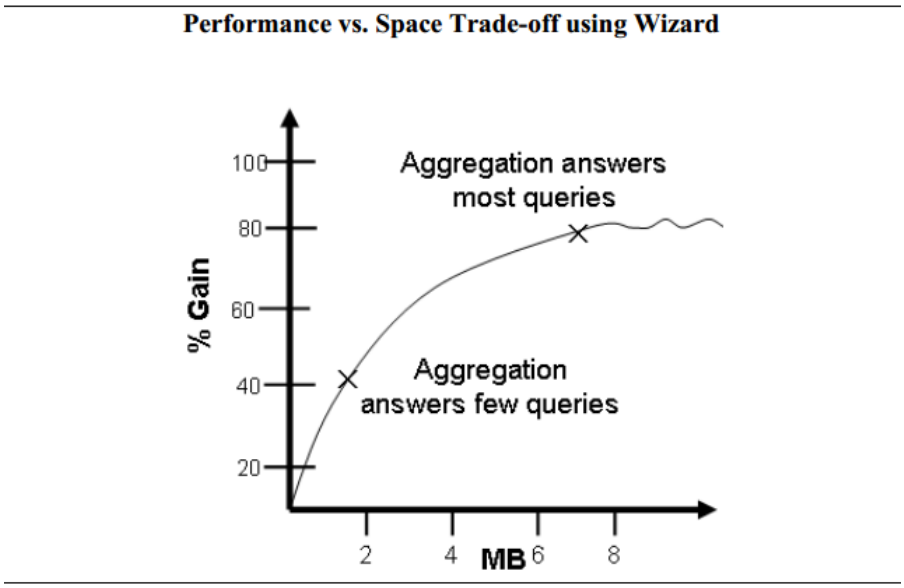



**Figure-12.4: Aggregation vs. Performance**

Aggregation design wizards allow a cube or pre-aggregate designer to specify the tradeoff between disk storage and performance to determine the maximum volume of pre- calculated aggregates, as shown in Figure-12.4. Suggestions about amount of aggregation are based on the amount of data reduction along different dimensions and the size of each aggregate. Optimizing the amount of aggregation is a very hard problem, thus heuristics are used. Heuristics in the aggregate design wizard can be improved by Usage-Based Optimization Wizards via examination of query logs to determine which pre-aggregate will deliver best bang for your storage buck.

Microsoft's Usage-Based Optimization Wizard (shown here) also allows the DBA to tell Analytic (OLAP) Services to create a new set of aggregations for all queries exceeding a defined response time threshold.

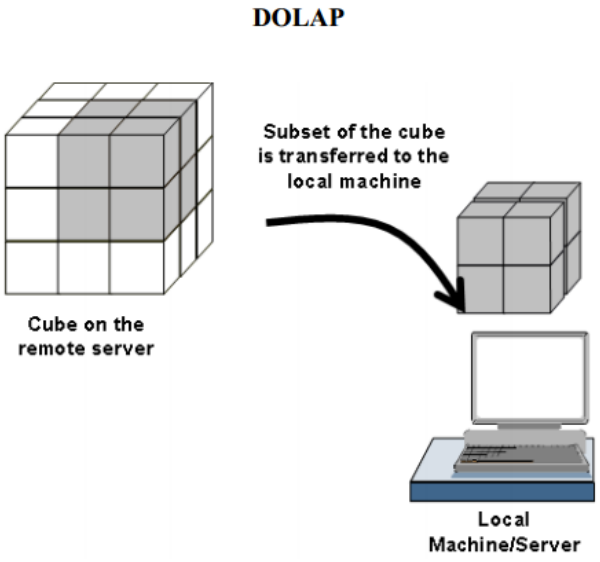
**HOLAP**

Target is to get the best of both worlds.

HOLAP (Hybrid OLAP) allow co-existence of pre-built MOLAP cubes alongside relational OLAP or ROLAP structures.

How much to pre-build?

The hybrid OLAP (HOLAP) solution is a mix of MOLAP and relational ROLAP architectures that supports queries against summary and transaction data in an integrated fashion. HOLAP environments use MOLAP cubes to support common access paths with reasonably small dimensional cardinality and number of dimensions and relational structures when greater scalability for OLAP queries is required. This coexistence strategy allows exploiting the best of both worlds. Microsoft OLAP Services supports a HOLAP environment, as do tools such as HOLOS. The HOLAP approach enables a user to perform multidimensional analysis on data in the MDDB along with query based probing. However, if the user reaches the bottom of the multidimensional hierarchy and requires further detail data, the smart HOLAP engine automatically generates SQL to retrieve the detail data from the source RDBMS and returns it to the end user. This is done transparently to the user. Several MOLAP vendors, such as Arbor and Oracle, have transitioned to HOLAP architectures that include a ROLAP component. However, these HOLAP architectures are typically more complex to implement and administer than ROLAP or MOLAP architectures individually.




DOLAP typically is the simplified version of MOLAP or ROLAP. DOLAP is inexpensive, it is fast and easy to setup on small data sets comprising of thousands of rows instead of millions of rows. It provides specific cube for the analysis. The DOLAP systems developed are extensions of production system report writers, while the systems developed in the early days of client /server computing aimed to take advantage of the power of the emerging PC desktop machine. DOLAP

also provides the mobile operations of OLAP for the people who travel and move extensively, such as sales people. The one obvious disadvantage of DOLAP is that it lacks the ability to manage large data sets. But this is just another technique to suit the business requirement.