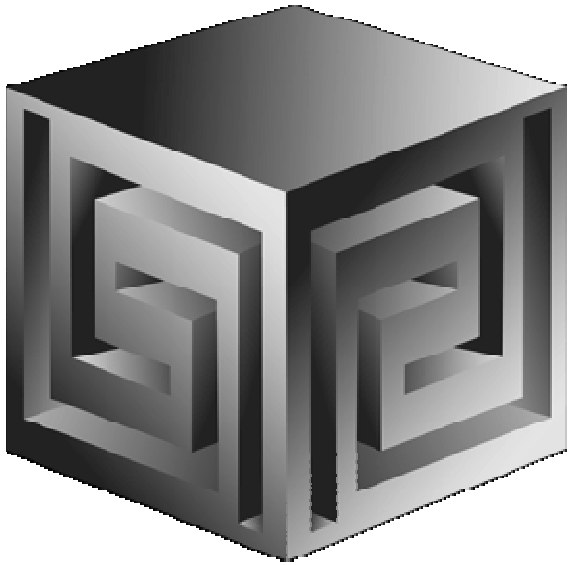
Accelerate Your Oracle DW with OLAP 11g

Collaborate '08

Session 211



Chris Claterbos

claterbos@vlamis.com

816-729-1034

www.vlamis.com

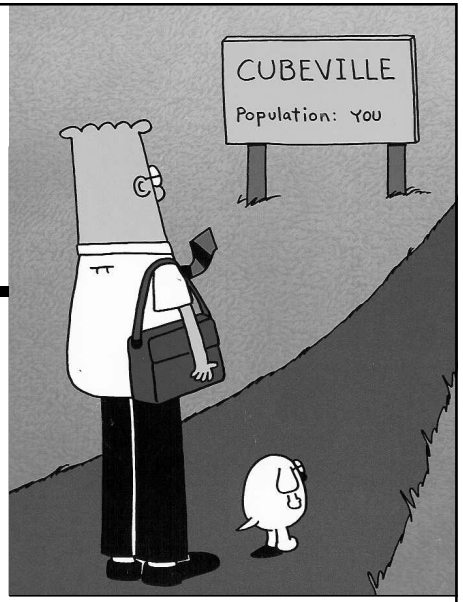
Vlamis Software Solutions, Inc.

- **Founded in 1992 in Kansas City, Missouri**
- **Oracle Partner and reseller since 1995**
- **Specializes in ORACLE-based:**
  - **Data Warehousing**
  - **Business Intelligence**
  - **Data Transformation (ETL)**
  - **Web development and portals**
  - **Express-based applications**
- **Delivers**
  - **Design and integrate BI and DW solutions**
  - **Training and mentoring**
- **Expert presenter at major Oracle conferences**

Who I Am

- **Chris Claterbos, Consulting Manager**
  - **Consulting and Development Manager for Vlamis Software Solutions, Inc.**
  - **DBA and applications developer for Oracle products, since 1981.**
  - **Beta tester and early adopter of - including Oracle 8i, 9i, 10g and 11g, JDeveloper and BIBeans, Oracle AS, Portal , and Reports.**
  - **Speaker and author.**
  - **Previous IOUG Focus Area Manager for Data Warehousing and BI**

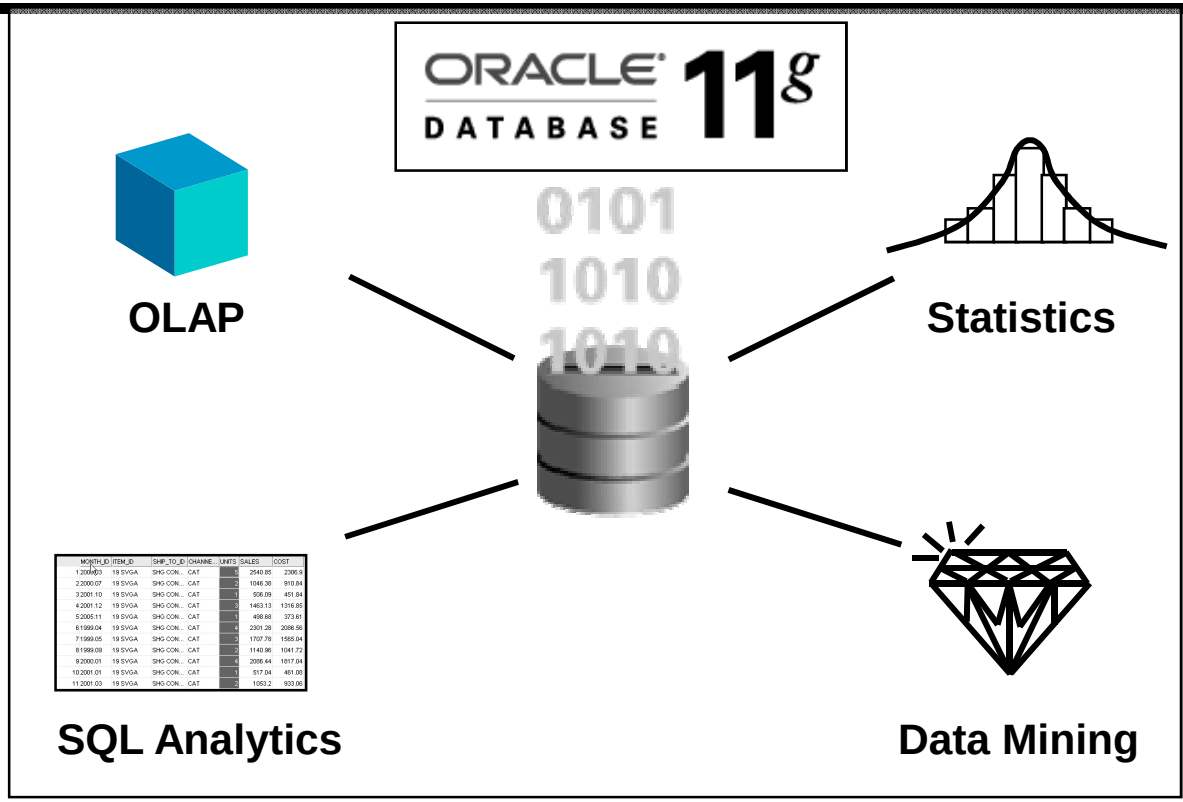
Presentation Agenda



- Oracle OLAP Overview
- Enhancing BI Solutions Transparently
- Delivering Rich Analytics Easily
- Positioning

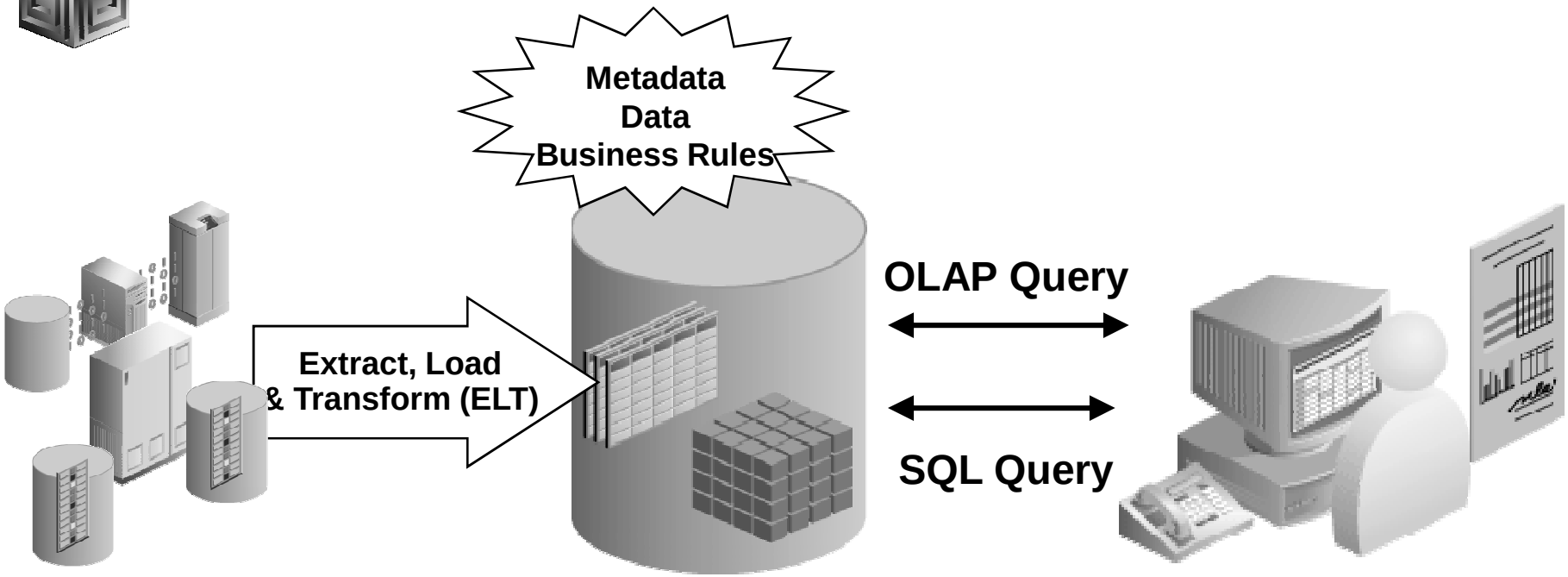
Oracle Database Strategy for DW

Embedded Analytics



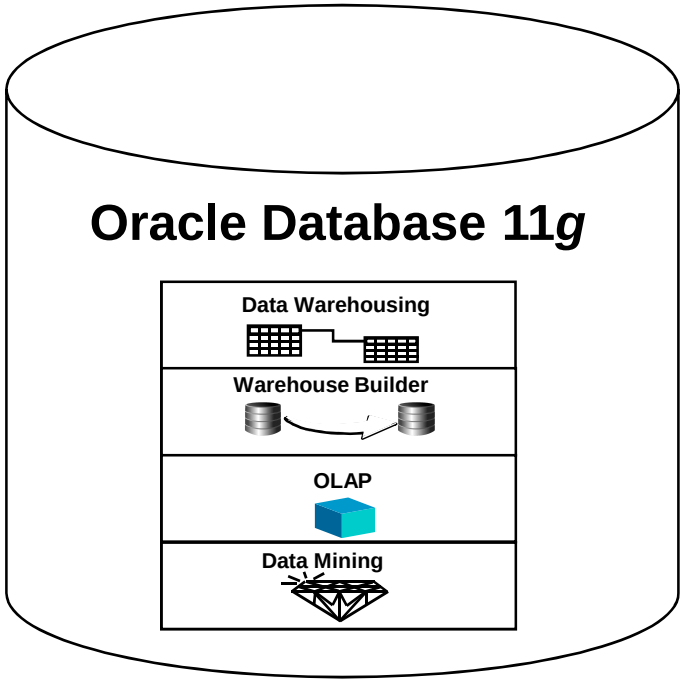
- Bring the analytics to the data
- Leverage core database infrastructure

One Cube, Dimensional or SQL Tools
Single version of the truth



Centrally managed data, meta data and business rules

Oracle OLAP
Leveraging Core Database Infrastructure



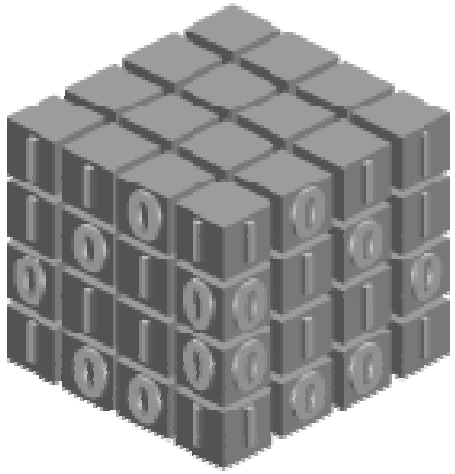
- Single RDBMS-MDDS process
- Single data storage
- Single security model
- Single administration facility
- Grid-enabled
- Accessible by any SQL-based tool
- Embedded BI metadata
- Connects to all related Oracle data

Oracle OLAP Goals

- **Improve the delivery of information rich queries by SQL-based business intelligence tools and applications**
  - **Embedded business rules**
  - **Fast query performance**
  - **Simplified access to analytic calculations**
  - **Fast incremental update**
  - **Leverage existing Oracle Database expertise**

OLAP Option

- **A full featured multidimensional OLAP server**
  - **Excellent query performance for ad-hoc / unpredictable query**
  - **Enhances the analytic content of Business intelligence application**
  - **Fast, incremental updates of data sets**
  - **Fully Integrated into RDBMS kernel**
- **A summary management solution for SQL based business intelligence applications**
  - **An alternative to table-based materialized views, offering improved query performance and fast, incremental update**



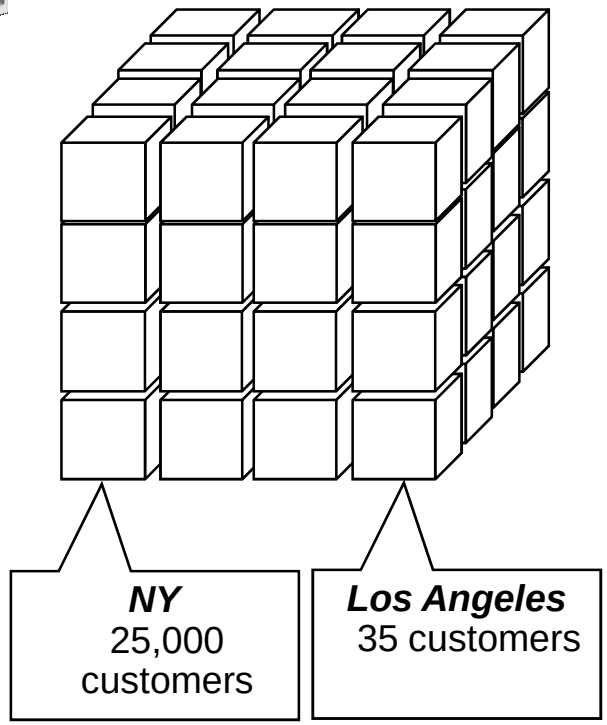
Top OLAP 11g New OLAP Features

- SQL Query
  - SQL cube scan
  - SQL cube join
  - CUBE_TABLE
  - Optimized looping
  - System maintained dimension and fact views
- Cube based Materialized Views
- SQL-like calculation expressions
- Cost-based aggregation
- Security
  - SQL Grant / Revoke
  - Permit with Extensible Data Security and AWM

Top 11g New OLAP Features

- Cube and maintenance scripts
  - Declarative calculation rules
  - Based on logical model
- All meta data in the Oracle Data Dictionary
  - Dimensional Model
  - Calculation definitions
  - Security policies
  - Data source mappings
  - SQL representation of model

Cost Based Aggregation
Pinpoint Summary Management



Precomputed

Computed when queried

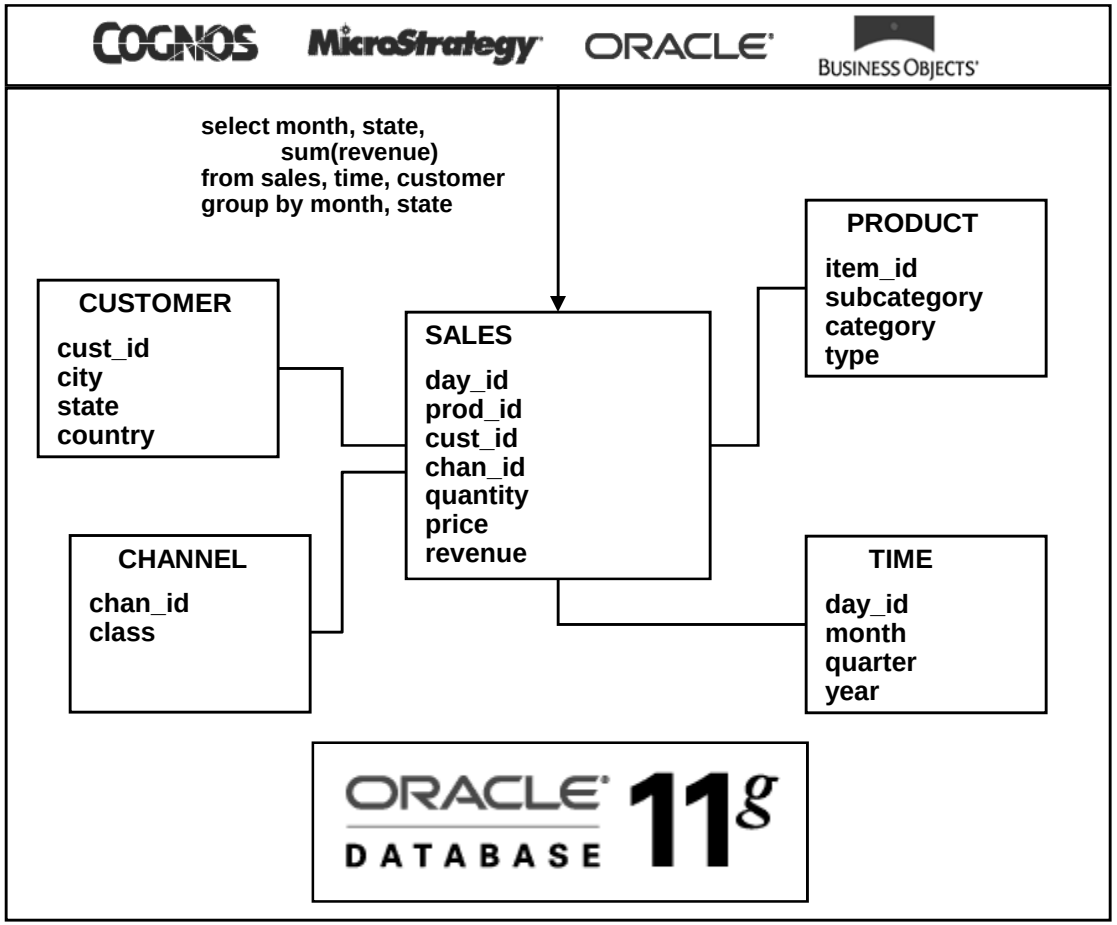
- **Improves aggregation speed and storage consumption by pre-computing *cells* that are most expense to calculate**
- **Easy to administer**
- **Simplifies SQL queries by presenting data as fully calculated**

Demonstration
BI Tools – how to optimize?

Cube Organized Materialized Views

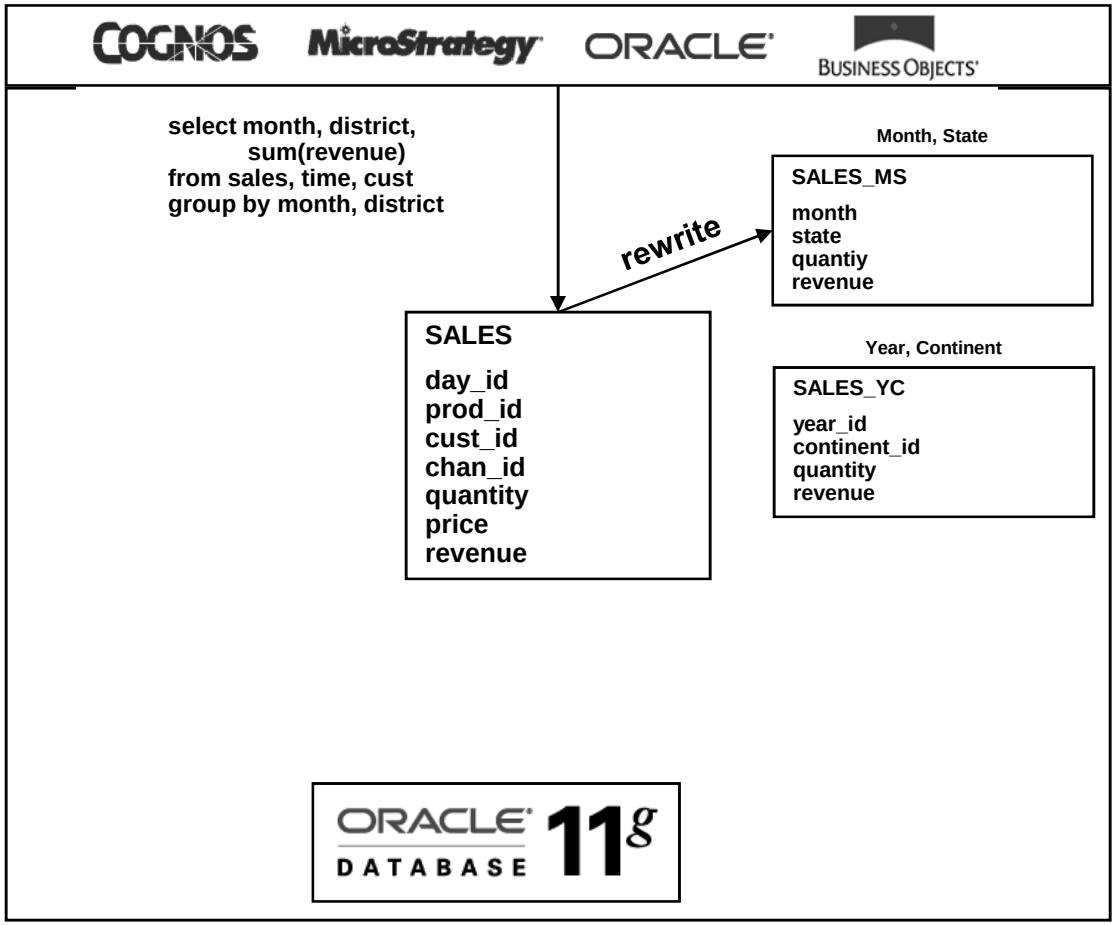
- **Transparently enhance the query performance of BI applications**
  - **Data is managed in an Oracle cube**
    - **Fast query**
    - **Fast refresh**
    - **Manage a single cube instead of 10's, 100's or 1,000's of table-based materialized views**
  - **Applications query base / detail relational tables**
    - **Oracle automatically rewrites SQL queries to OLAP cubes**
    - **Access to summary data in the cube is fully transparent**

Materialized Views
Typical MV Architecture Today



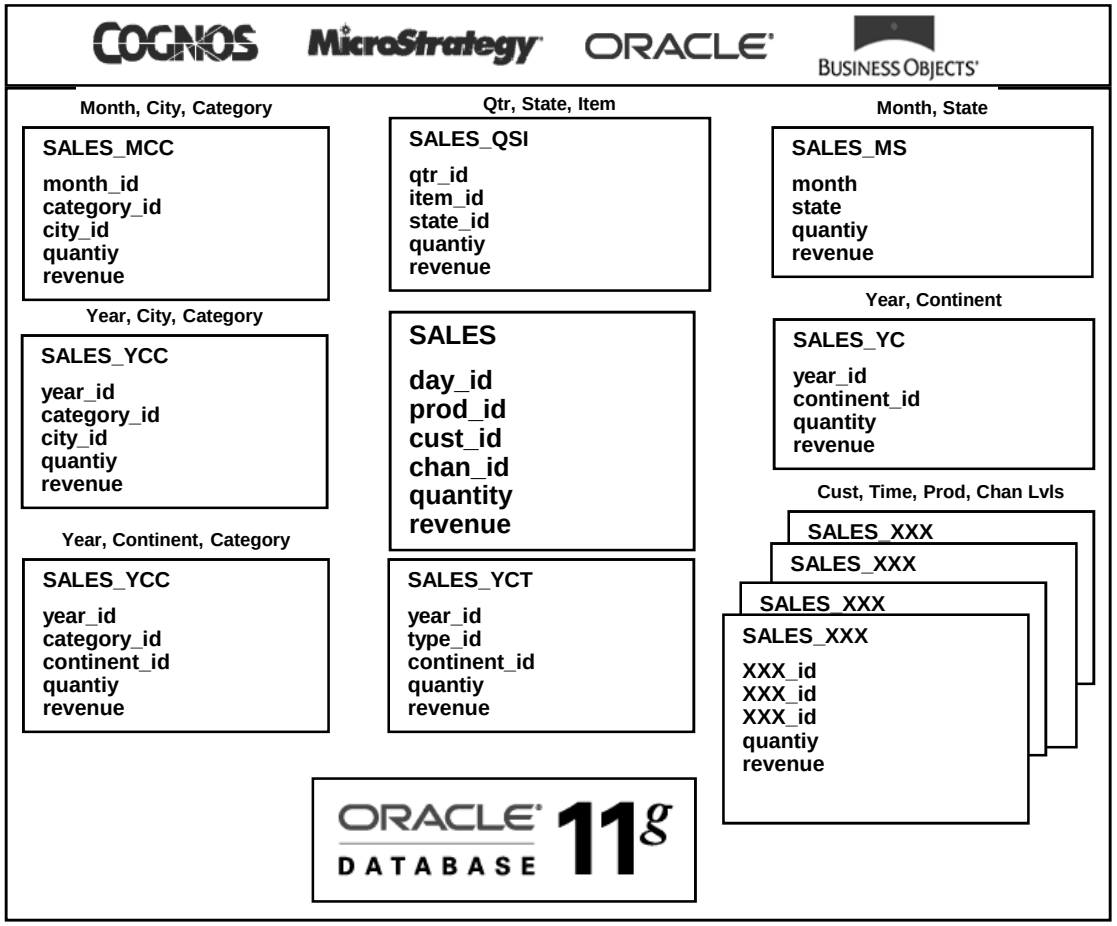
- Query tools access star schema stored in Oracle data warehouse
- Most queries at a summary level
- Summary queries against star schemas can be expensive to process

Materialized Views
Automatic Query Rewrite



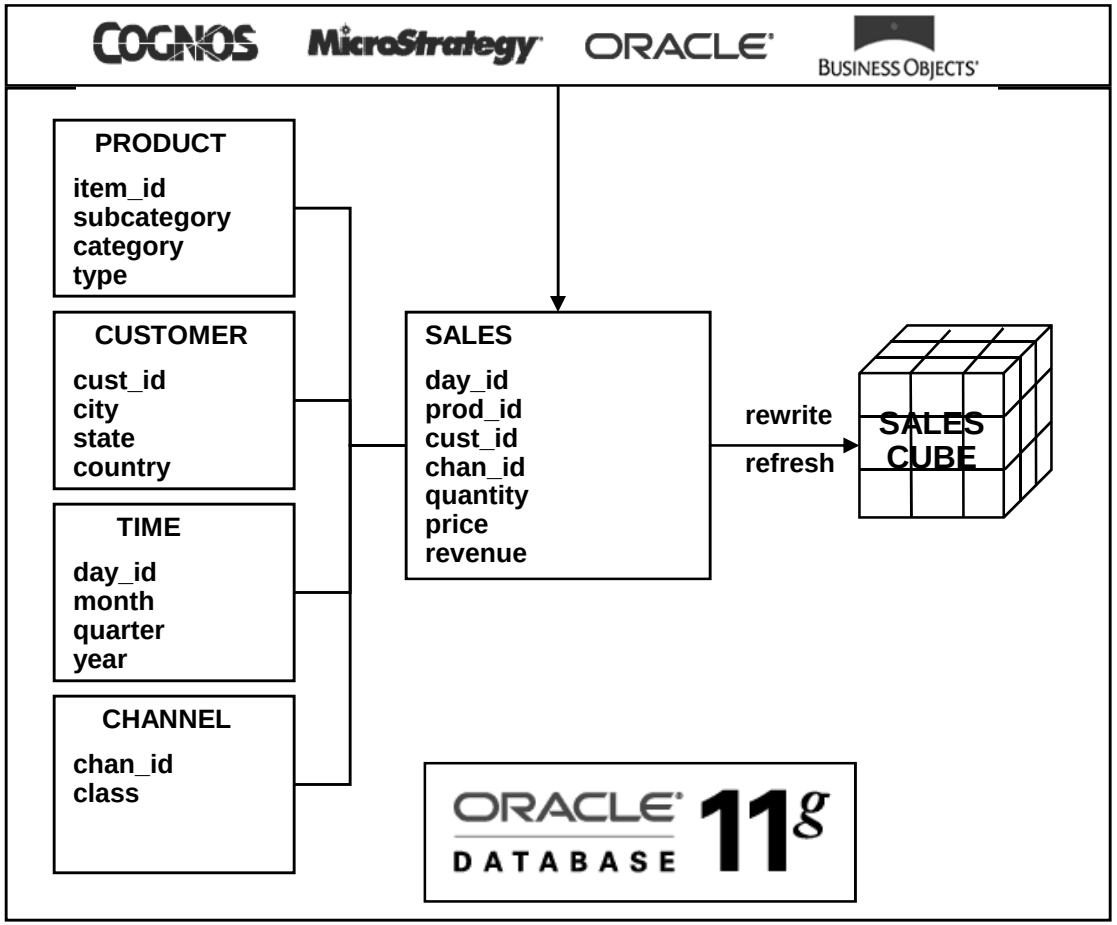
- Most DW/BI customers use Materialized Views (MV) today to improve summary query performance
- Define appropriate summaries based on query patterns
- Each summary is typically defined at a particular grain
  - Month, State
  - Qtr, State, Item
  - Month, Continent, Class
  - etc.
- The SQL Optimizer automatically rewrites queries to access MV's whenever possible

Materialized Views
Challenges in Ad Hoc Query Environments



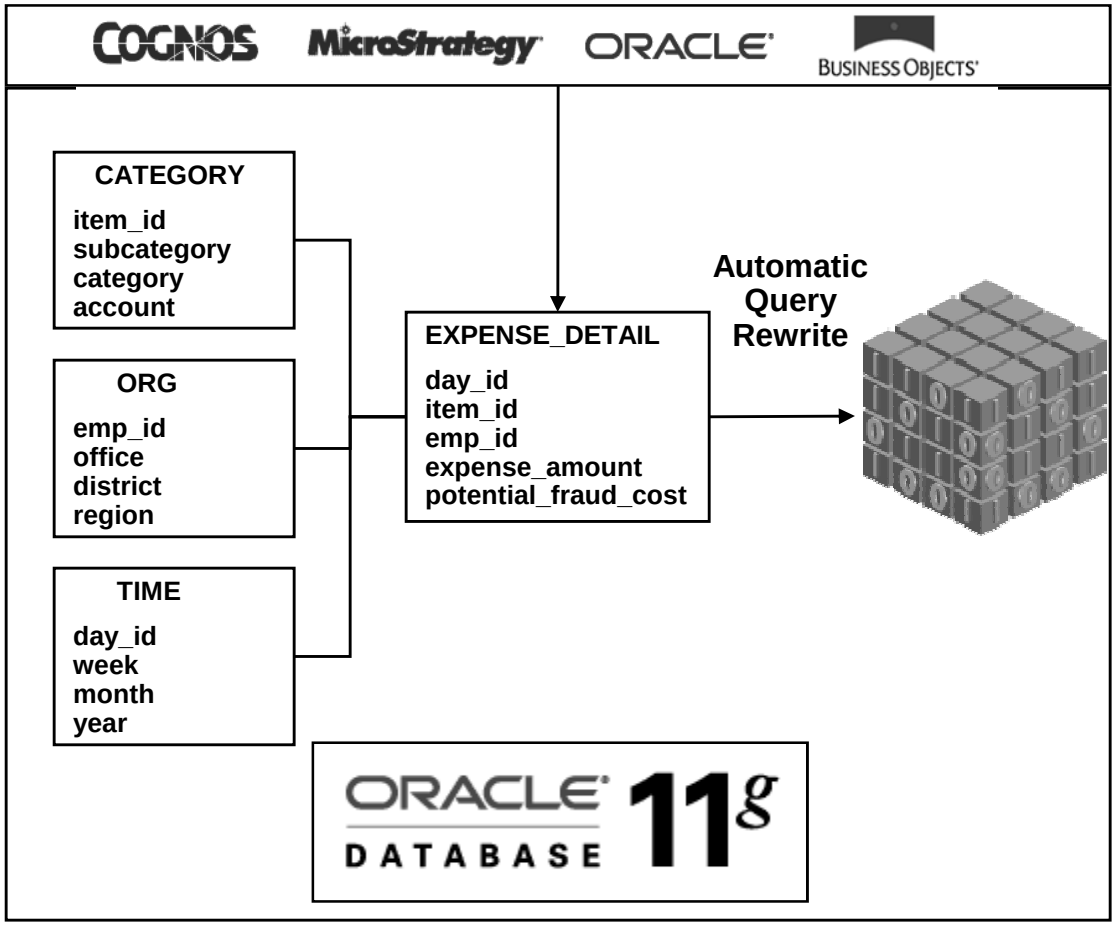
- Creating MVs to support ad hoc query patterns is challenging
- Users expect excellent query response time across any summary
- Potentially many MVs to manage
- Practical limitations on size and manageability constrain the number of materialized views

Cube-based Materialized Views
Breakthrough Manageability & Performance



- **A single cube provides the equivalent** *of thousands of summary combinations*
- **The 11g SQL Query Optimizer treats OLAP cubes as MV's and rewrites queries to access cubes** *transparently*
- **Cube refreshed using standard MV procedures**

Cube Organized Materialized Views
Breakthrough Performance



- A single cube manages summaries for all groupings in the model
- A cube can be represented as a cube-organized materialized view
- Oracle automatically rewrites summary queries to the cube
- A singe cube can replace 10's, 100's or 1,000's of materialized views

Cube Organized Materialized Views
Breakthrough Manageability

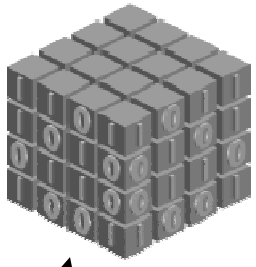
COGNOS MicroStrategy ORACLE BUSINESS OBJECTS

CATEGORY
item_id
subcategory
category
account

ORG
emp_id
office
district
region

TIME
day_id
week
month
year

EXPENSE_DETAIL
day_id
item_id
emp_id
expense_amount
potential_fraud_cost



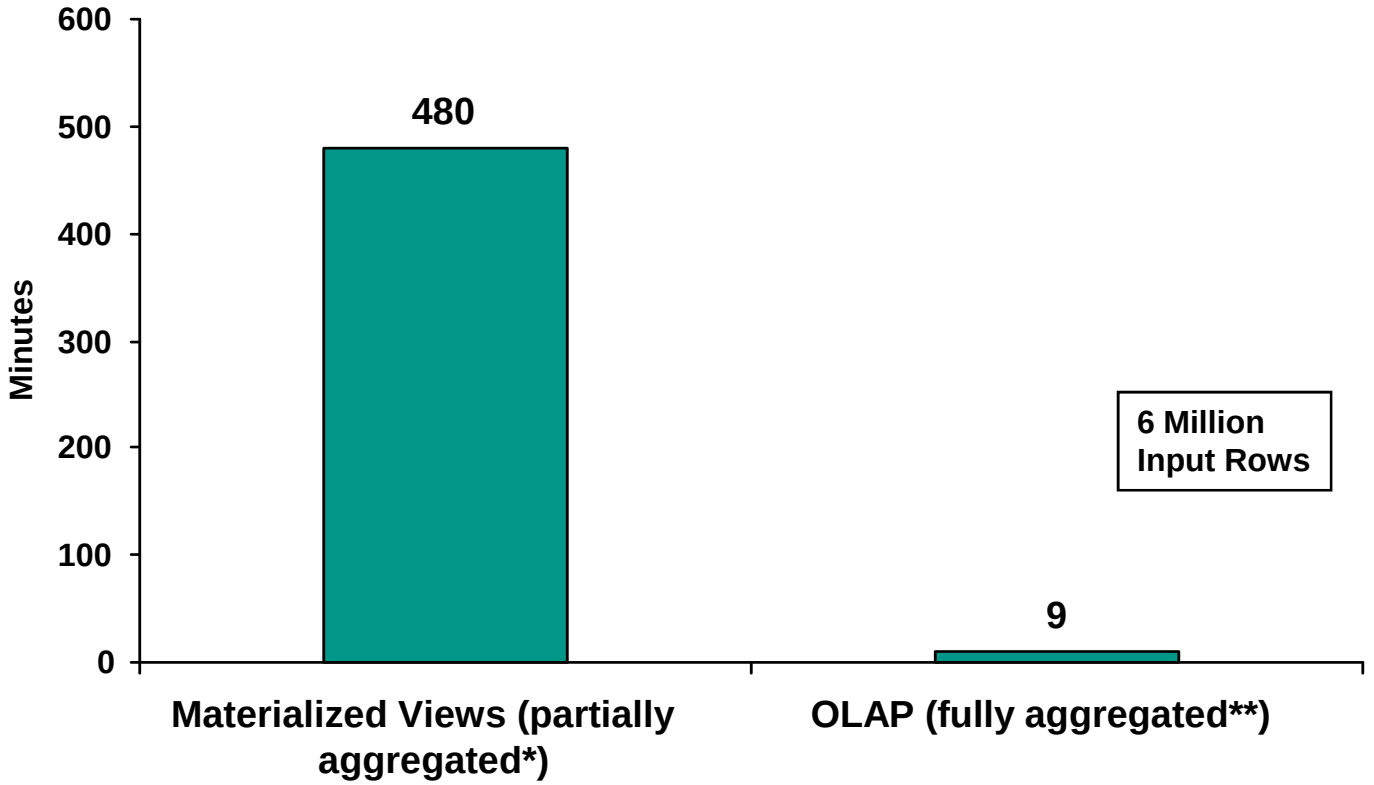
Materialized View Refresh

```
dbms_mview.refresh('CB$SALES_CUBE','F')
```

- Like 10g MV's, provides fast incremental refresh of the cube as underlying data changes
- Simple - Cube refresh syntax is identical to MV Refresh syntax

ORACLE DATABASE 11g

Performance Case Study
Oracle Applications: Finance DBI



* MV aggregated 1 dimension and 1 measure

** OLAP aggregated 7 dimensions and 11 measures

Cube-Organized Materialized Views

- A SQL object, just like table-based (heap organized) materialized views
- The cube-organized MV is similar to a MV on pre-built table
  - Summary data is managed by the cube
  - The cube MV is meta data only
    - Data is not materialized (replicated) into the cube-organized MV

Implementing Cube MVs
Process

1. Design dimensions and cubes
2. Enable dimension and cube MVs
3. Prepare relational schema for query rewrite
4. Build/maintain dimensions and cubes

Implementing Cube MVs
Requirements - Privileges

- **Cube owner must have CREATE MATERIALIZED VIEW privilege**

Implementing Cube MVs
Requirements – Design

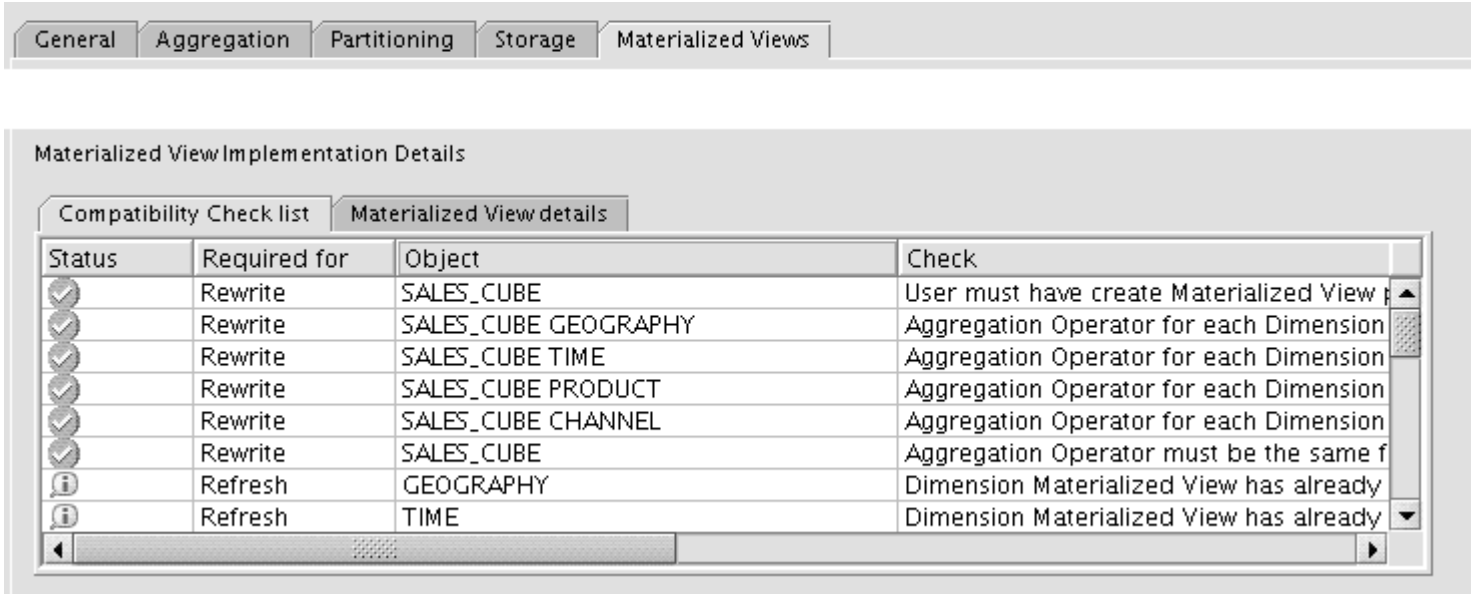
- **Dimensions**
  - **Dimensions must have hierarchies**
    - **Hierarchies must be level based**
    - **Hierarchies must not be skip level or ragged**
  - **Dimensions must be fully mapped to star / snowflake relational tables**
    - **Star tends to be a better choice at the moment**
  - **Dimensions should not be mapped to constants**

Implementing Cube MVs
Requirements – Design

- **Cubes**
  - **Cube must be Compressed**
  - **Aggregation methods must be**
    - **SUM, MIN or MAX**
    - **The same for all dimensions**
  - **The cube must be fully mapped to sources**
    - **Tables only**
    - **All stored measures**
    - **All dimensions**
  - **Cube is solved by aggregation only**
    - **No models, OLAP DML assignments**

Implementing Cube MVs
Requirements – Design

- **Compatibility checks lists identify many cube and dimension design issues**



Status	Required for	Object	Check
	Rewrite	SALES_CUBE	User must have create Materialized View p
	Rewrite	SALES_CUBE GEOGRAPHY	Aggregation Operator for each Dimension
	Rewrite	SALES_CUBE TIME	Aggregation Operator for each Dimension
	Rewrite	SALES_CUBE PRODUCT	Aggregation Operator for each Dimension
	Rewrite	SALES_CUBE CHANNEL	Aggregation Operator for each Dimension
	Rewrite	SALES_CUBE	Aggregation Operator must be the same f
	Refresh	GEOGRAPHY	Dimension Materialized View has already
	Refresh	TIME	Dimension Materialized View has already

Implementing Cube MVs
Requirements - Query Rewrite

- **Cube MV must be enabled**

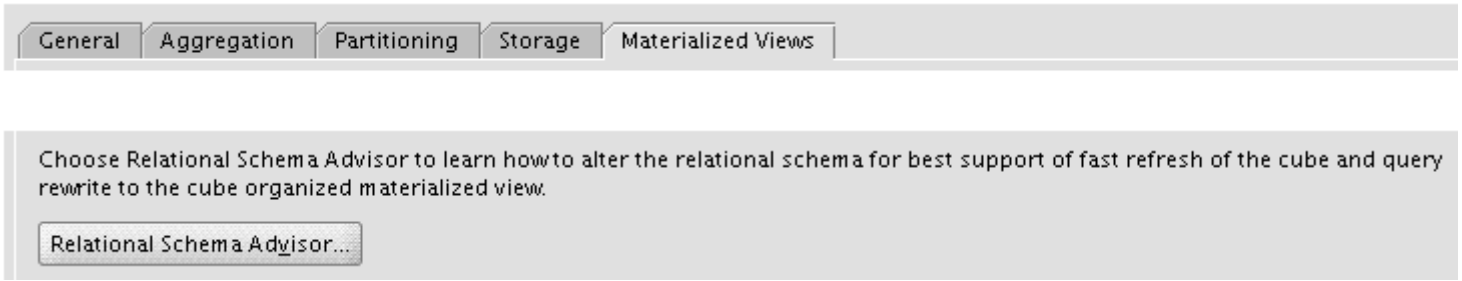
```
alter materialized view CB$SALES_CUBE" enable query
   rewrite;
```

- **Cube and dimension MVs must be fresh**
  - **Stale tolerated is not supported with the cube**
- **If the cube's detail is a summary of the table**

```
alter session set query rewrite integrity = trusted;
```

Implementing Cube MVs
Requirements - Query Rewrite

- Relational tables must be prepared for query rewrite with
  - NOT NULL constraints on “ID” columns
  - Primary key constraints
  - Foreign key constraints
  - SQL dimension objects
- Relational Schema Advisor can provide sample SQL script



Implementing Cube MVs
Requirements – MV Refresh

- Same design requirements as MV rewrite
- For Fast refresh (incremental load from fact tables), MV log tables are required

Implementing Cube MVs
Notes

- Custom measures may be included in a cube used as an MV
  - They will not be included in the MV (because they are not in the source tables)
- Any change to the model will cause the MV to become UNUSABLE (not fresh) for MV refresh
  - A complete refresh will be required after any change in the model
- The database will lock AW objects from OLAP DML assignments

Implementing Cube MVs
Database-Generated Objects

- When a cube is enabled as an MV
  - Dimensions are automatically enabled as MVs
  - The Database generates
    - One CR$ table for the cube
      - A cube organized table that allows MV refresh to SQL insert into the cube
    - One CB$ materialized view
      - Contains MV meta data (e.g., defining query, Fresh/Stale, etc.)
- Do not delete or modify generated objects
- Do not attempt to create these objects outside of the OLAP API (that is, with SQL commands)

Using Cube MVs
MV Refresh of Cube

- Cubes enabled as MVs may be refreshed using
  - dbms_mview.refresh
  - dbms_cube.build
  - OLAP API
  - AWM
- All methods accomplish the same thing – a MV compatible refresh of the cube

Using Cube MVs
MV Refresh of Cube

- **dbms_mview.refresh**
  - **The standard MV refresh program**
    - **Use cube or dimension MV as the refresh object**
    - **Refresh dimensions and cube separately (dimensions first)**

```
dbms_mview.refresh('CB$TIME','C')
dbms_mview.refresh('CB$PRODUCT','C')
dbms_mview.refresh('CB$GEOGRAHY','C')
dbms_mview.refresh('CB$CHANNEL','C')
dbms_mview.refresh('CB$SALES_CUBE','F')
```

Using Cube MVs
MV Refresh of Cube

- dbms_cube.build
  - Cube-specific program that uses the MV refresh system
    - Use logical object names as the build / refresh object
    - Build / refresh cubes or dimensions
    - Will automatically build / refresh dimensions of a cube

```
dbms_cube.build('SALES_CUBE','F')
```

Using Cube MVs
Query Rewrite

- The defining query of the cube MV determines what queries can be satisfied by the cube
- The defining query includes the member columns for all levels, in all dimensions
  - The cube MV can satisfy queries for any level of summarization in the cube when selecting member columns
    - Member columns are the columns dimension members are mapped to (e.g., in AWM)

Using Cube MVs
Query Rewrite

Member Columns are included in the defining query of the the Cube MV

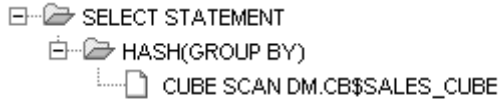
PRODUCT	Source Column
⊟HIERARCHIES	
⊟STANDARD	
⊟ALL_PRODUCTS	
Member	DM.PRODUCTS.ALL_PRODUCTS_ID
LONG_DESCRIPTION	DM.PRODUCTS.ALL_PRODUCTS_DESC
SHORT_DESCRIPTION	DM.PRODUCTS.ALL_PRODUCTS_DESC
PRODUCT_ALL_PROD...	DM.PRODUCTS.ALL_PRODUCTS_ID
⊟DEPARTMENT	
Member	DM.PRODUCTS.DEPARTMENT_ID
LONG_DESCRIPTION	DM.PRODUCTS.DEPARTMENT_LONG_DESC
SHORT_DESCRIPTION	DM.PRODUCTS.DEPARTMENT_SHORT_DESC
PRODUCT_DEPARTME...	DM.PRODUCTS.DEPARTMENT_ID
⊟CATEGORY	
Member	DM.PRODUCTS.CATEGORY_ID
LONG_DESCRIPTION	DM.PRODUCTS.CATEGORY_LONG_DESC
SHORT_DESCRIPTION	DM.PRODUCTS.CATEGORY_SHORT_DESC
PRODUCT_CATEGORY...	DM.PRODUCTS.CATEGORY_ID
⊟TYPE	
Member	DM.PRODUCTS.TYPE_ID
LONG_DESCRIPTION	DM.PRODUCTS.TYPE_LONG_DESC
SHORT_DESCRIPTION	DM.PRODUCTS.TYPE_SHORT_DESC
PRODUCT_TYPE_ID	DM.PRODUCTS.TYPE_ID
⊟SUBTYPE	
Member	DM.PRODUCTS.SUB_TYPE_ID
LONG_DESCRIPTION	DM.PRODUCTS.SUB_TYPE_LONG_DESC
SHORT_DESCRIPTION	DM.PRODUCTS.SUB_TYPE_SHORT_DESC
PRODUCT_SUBTYPE_ID	DM.PRODUCTS.SUB_TYPE_ID

Using Cube MVs
Query Rewrite

- Queries that include only those columns that are included in the defining query of the cube MV are resolved entirely by the cube
  - Aggregates
  - Joins

Using Cube MVs
Query Rewrite - Example

```
SELECT t.calendar_year_id,
  p.department_id,
  cu.region_id,
  ch.class_id,
  SUM(f.sales) sales
FROM times t,
  products p,
  customers cu,
  channels ch,
  sales_fact f
WHERE t.day_id = f.day_id
 AND p.item_id = f.item_id
 AND cu.customer_id = f.customer_id
 AND ch.channel_id = f.channel_id
GROUP BY t.calendar_year_id,
  p.department_id,
  cu.region_id,
  ch.class_id;
```

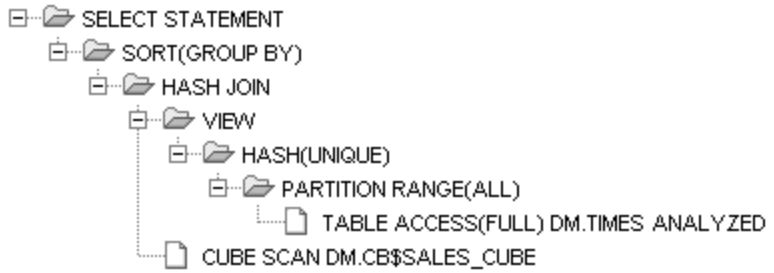


Using Cube MVs
Query Rewrite

- **Queries that include other (non-member) columns are solved in part in the SQL engine**
  - **Queries that have relatively few SQL joins after the CUBE SCAN are generally good**
  - **Queries that have many SQL joins after the CUBE SCAN might not be as good**

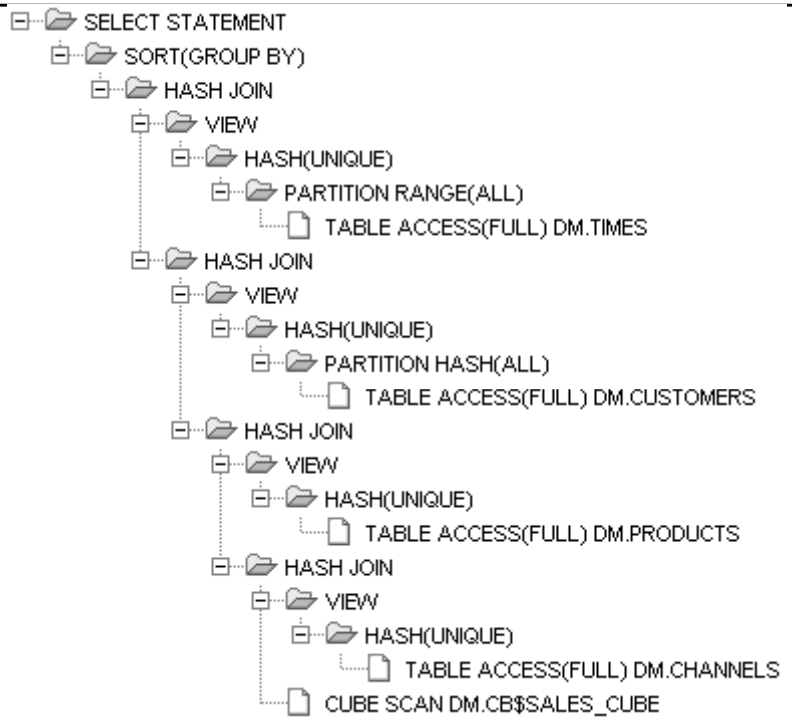
Using Cube MVs
Query Rewrite - Example

```
SELECT t.calendar_year_id,
  t.calendar_year_end_date,
  p.department_id,
  cu.region_id,
  ch.class_id,
  SUM(f.sales) sales
FROM times t,
  products p,
  customers cu,
  channels ch,
  sales_fact f
WHERE t.day_id = f.day_id
 AND p.item_id = f.item_id
 AND cu.customer_id = f.customer_id
 AND ch.channel_id = f.channel_id
GROUP BY t.calendar_year_id,
  t.calendar_year_end_date,
  p.department_id,
  cu.region_id,
  ch.class_id
ORDER BY t.calendar_year_end_date;
```



Using Cube MVs
Query Rewrite - Example

```
SELECT t.calendar_year_desc,
  t.calendar_year_end_date,
  p.department_long_desc,
  cu.region_desc,
  ch.class_desc,
  SUM(f.sales) sales
FROM times t,
  products p,
  customers cu,
  channels ch,
  sales_fact f
WHERE t.day_id = f.day_id
 AND p.item_id = f.item_id
 AND cu.customer_id = f.customer_id
 AND ch.channel_id = f.channel_id
GROUP BY t.calendar_year_desc,
  t.calendar_year_end_date,
  p.department_long_desc,
  cu.region_desc,
  ch.class_desc
ORDER BY t.calendar_year_end_date;
```



Using Cube MVs
Query Rewrite

- **Query rewrite will use the highest level in the cube possible to satisfy GROUP BY on attribute columns**
  - **If a column representing a level is in a query, that level is selected**
  - **If dimensions are omitted from the a query, the highest level in the cube is selected (and SQL aggregates over those values)**
  - **If attributes of a level are selected for GROUP BY, that level is selected**

Using Cube MVs
Query Rewrite - Example

```
SELECT t.calendar_year_id,
  p.item_marketing_manager,
  p.item_buyer,
  SUM(f.sales) sales
FROM time_dim t,
  product_dim p,
  units_fact f
WHERE t.month_id = f.month_id
 AND p.item_id = f.item_id
GROUP BY t.calendar_year_id,
  p.item_marketing_manager,
  p.item_buyer;
```

- This query
  - Accesses Calendar Year level for time
  - The top levels for dimensions not in the query (Customer and Channel)
  - Detail (Item) level data for product

Operation	...	...	...	...	...	...	...	...	FILTER PREDICATES
SELECT STATEMENT	...	15	9	...					
HASH(GROUP BY)		15	9	...					
HASH JOIN		14	...	...				"...	
TABLE ACCESS(FULL) GLOBAL.PRODUCT_DIM	...	2	36	...					
CUBE SCAN(PARTIAL OUTER) GLOBAL.CB$UNITS_CUBE		11	...	...					SYS_OP_ATG(VALUE(KOKBF$),27,28,2)=61565

Using Cube MVs
Query Rewrite - Example

F

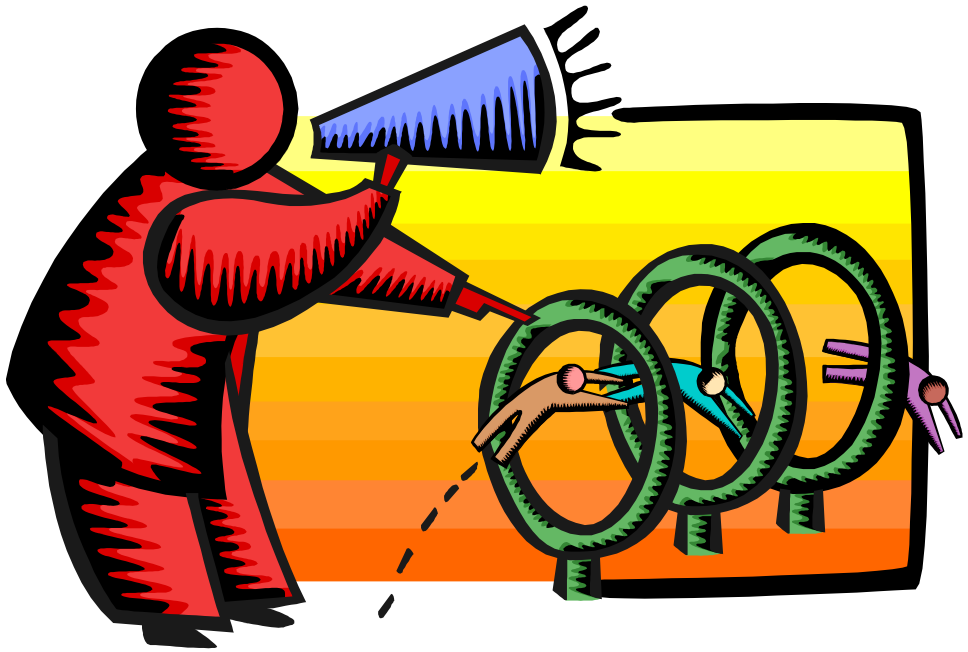
- This query
  - Accesses Calendar Year level for time
  - The top levels for dimensions not in the query (Customer and Channel)
  - Detail (Item) level data for product

Operation	...	...	...	...	...	...	...	...	FILTER PREDICATES
SELECT STATEMENT	...	15	9	...					
HASH(GROUP BY)		15	9	...					
HASH JOIN		14	...	...			"...		
TABLE ACCESS(FULL) GLOBAL.PRODUCT_DIM	...	2	36	...					
CUBE SCAN(PARTIAL OUTER) GLOBAL.CB$UNITS_CUBE		11	...	...					SYS_OP_ATG(VALUE(KOKBF$),27,28,2)=61565

Using Cube MVs
Query Rewrite - Example

Operation	...	...	...	...	...	...	...	...	FILTER PREDICATES
SELECT STATEMENT	...	15	9	...					
HASH(GROUP BY)		15	9	...					
HASH JOIN		14	...	...				"...	
TABLE ACCESS(FULL) GLOBAL.PRODUCT_DIM	...	2	36	...					
CUBE SCAN(PARTIAL OUTER) GLOBAL.CB$UNITS_CUBE		11	...	...					SYS_OP_ATG(VALUE(KOKBF$),27,28,2)=61565

GID identifying levels access from cube



Demonstration Transparently Improving Performance of BI Solutions

BNP Paribas
Advanced Time-Series Analyses in Real-Time

- **Large European financial institution**
- **Used by traders to help decrease susceptibility to market volatility**
- **Replacing FAME Time Series Database**
  - **F**orecasting, **A**nalysis and **M**odeling **E**nvironment
- **Three billion stored facts on RAC**
- **Data updated every 2 seconds – processing approximately 1m records daily**
- **SQL-based custom application used by 1500 concurrent users**

Parkinson

$$P_N = \frac{1}{N\sqrt{4\ln 2}} \sum_N \left(\ln(High / Low)\right)^2$$

Garman-Klass

$$G_N = \frac{1}{N} \sum_N \left[\left(\ln\left(\frac{High}{Low}\right) \right)^2 - (2\ln 2 - 1)\left(\ln\left(\frac{Close}{Open}\right) \right)^2 \right]$$

Rogers

$$R_N = \frac{1}{N} \sum_N \left[\left(\ln\left(\frac{High}{Low}\right) \right)^2 + \left(\ln\left(\frac{Close}{Open}\right) \right)^2 \right]$$

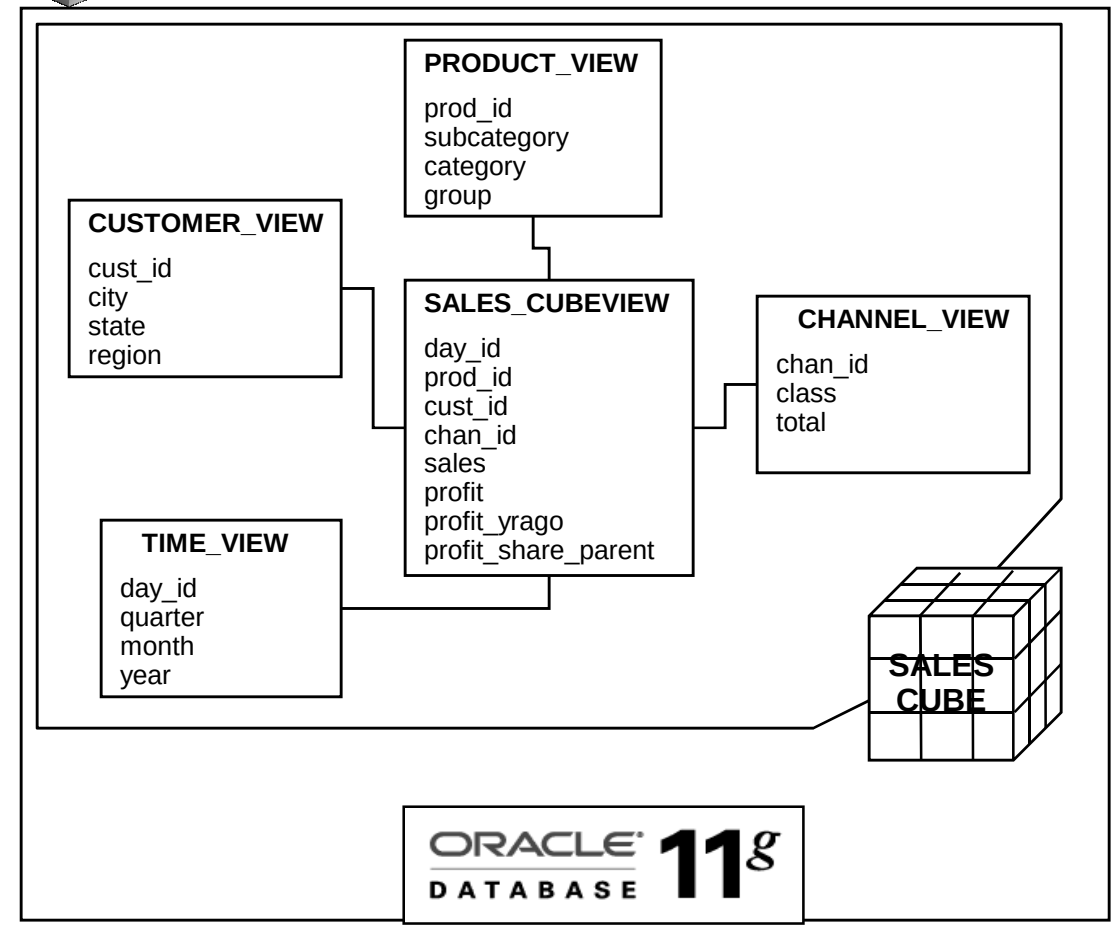
BLUE

$$B_N = \frac{1}{7N} \sum_N \left[\left(\ln\left(\frac{Close}{Open}\right) \right)^2 + 6\left(\ln\left(\frac{Low}{Close}\right) \right)^2 \right]$$

One Cube Accessed Many Ways...

- One cube can be used as
  - A summary management solution to SQL-based business intelligence applications as cube-organized materialized views
  - A analytically rich data source to SQL-based business intelligence applications as SQL cube-views
  - A full-featured multidimensional cube, servicing dimensionally oriented business intelligence applications

Cube Represented as Star Model
Simplifies Access to Analytic Calculations



- Cube represented as a star schema
- Single cube view presents data as completely calculated
  - Analytic calculations presented as columns
  - Includes all summaries
- Automatically managed by OLAP

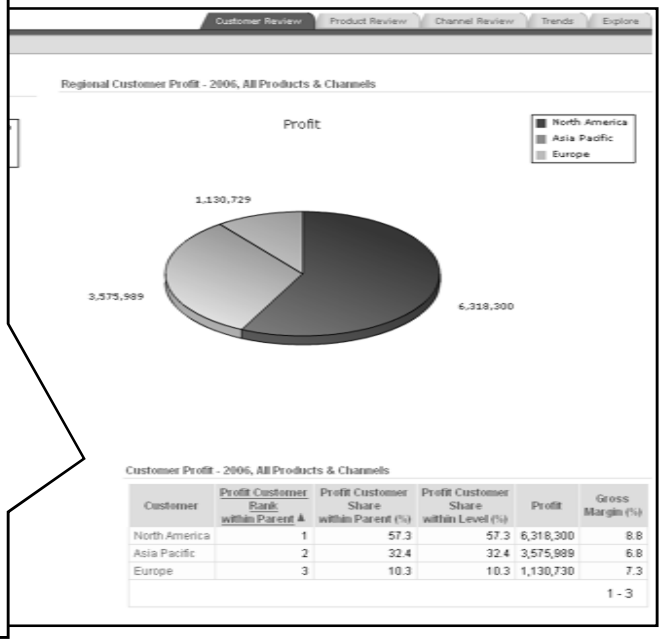
Empowering Any SQL-Based Tool
Leveraging the OLAP Calculation Engine

Application Express on Oracle OLAP

```
SELECT cu.long_description customer,
  f.profit_rank_cust_sh_parent,
  f.profit_share_cust_sh_parent,
  f.profit_rank_cust_sh_level,
  f.profit,
  f.gross_margin

FROM time_calendar_view t,
     product_primary_view p,
     customer_shipments_view cu,
     channel_primary_view ch,
     units_cube_view f

WHERE t.level_name = 'CALENDAR_YEAR'
   AND t.calendar_year = 'CY2006'
   AND p.dim_key = 'TOTAL'
   AND cu.parent = 'TOTAL'
   AND ch.dim_key = 'TOTAL'
   AND t.dim_key = f.TIME
   AND p.dim_key = f.product
   AND cu.dim_key = f.customer
   AND ch.dim_key = f.channel;
```



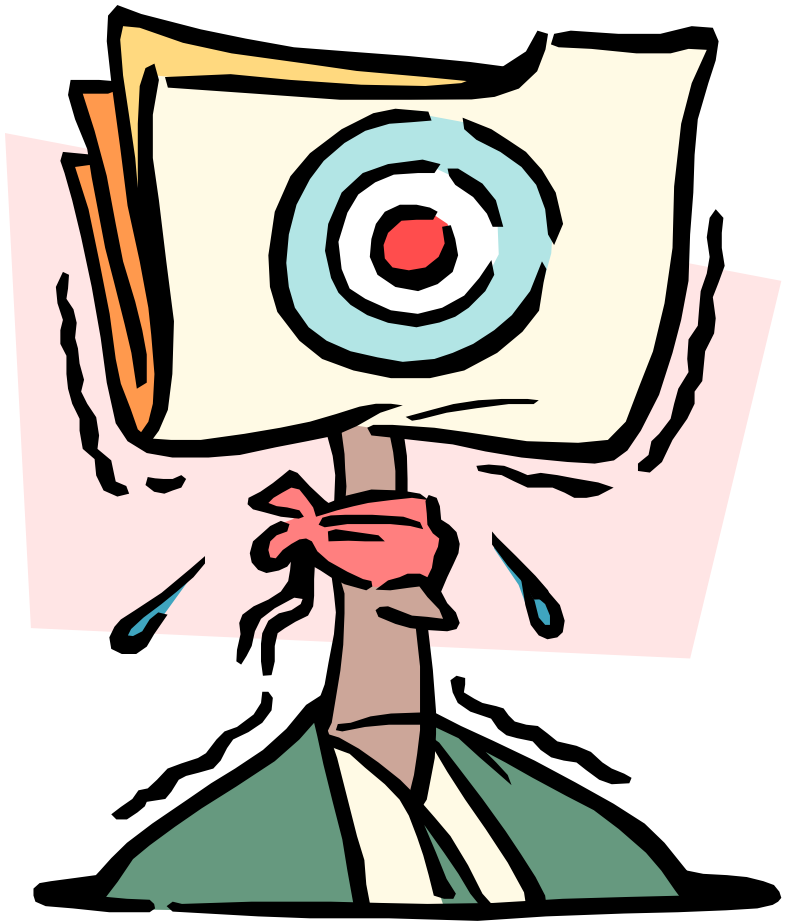
Oracle OLAP 11g Summary

- **Improve the delivery of information rich queries by SQL-based business intelligence tools and applications**
  - **Fast query performance**
  - **Simplified access to analytic calculations**
  - **Fast incremental update**
  - **Centrally managed by the Oracle Database**

For More Information

- www.vlamis.com
- Oracle Technology Network:
  - http://www.oracle.com/technology/products/bi/olap/index.html
- Product Discussion Forum:
  - http://forums.oracle.com/forums/forum.jspa?forumID=16

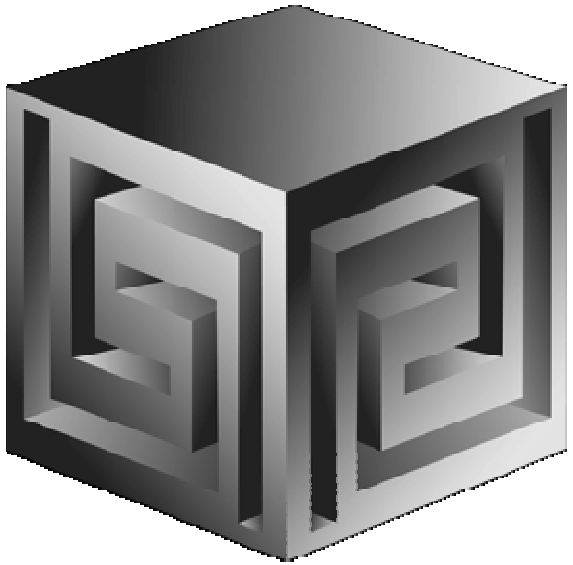
QUESTIONS?



Accelerate Your Oracle DW with OLAP 11g

Collaborate '08

Session 211



Chris Claterbos

claterbos@vlamis.com

816-729-1034

www.vlamis.com