Insight
Americas 07

TALES FROM A UNIVERSE NINJA: PART ONE

David G. Rathbun, Integra Solutions

Business Objects™

BREAKOUT INFORMATION

- Tales from a Universe Ninja, Part One
  - Training classes and product manuals can take you only so far. What do you do when the manual stops, but the project requirements do not? In this two-part session, find out how a team of experienced universe designers solved real-world universe challenges. Get tips, many of which are applicable to versions prior to BusinessObjects XI Release 2. Part one focuses primarily on the Designer application. See creative uses of standard Designer features, new wrinkles on some standard solutions, and in-depth coverage of specific universe parameters. Download sample files after the session. Add to your arsenal of universe practices, and be at peace with the universe.

Print Information (please leave for Business Objects use)

Print Code

AGENDA

1. Introduction
2. In depth index awareness
3. Leveraging shortcut joins
4. Using JOIN_BY_SQL
5. Conclusion
6. Q&A

INTRODUCTION

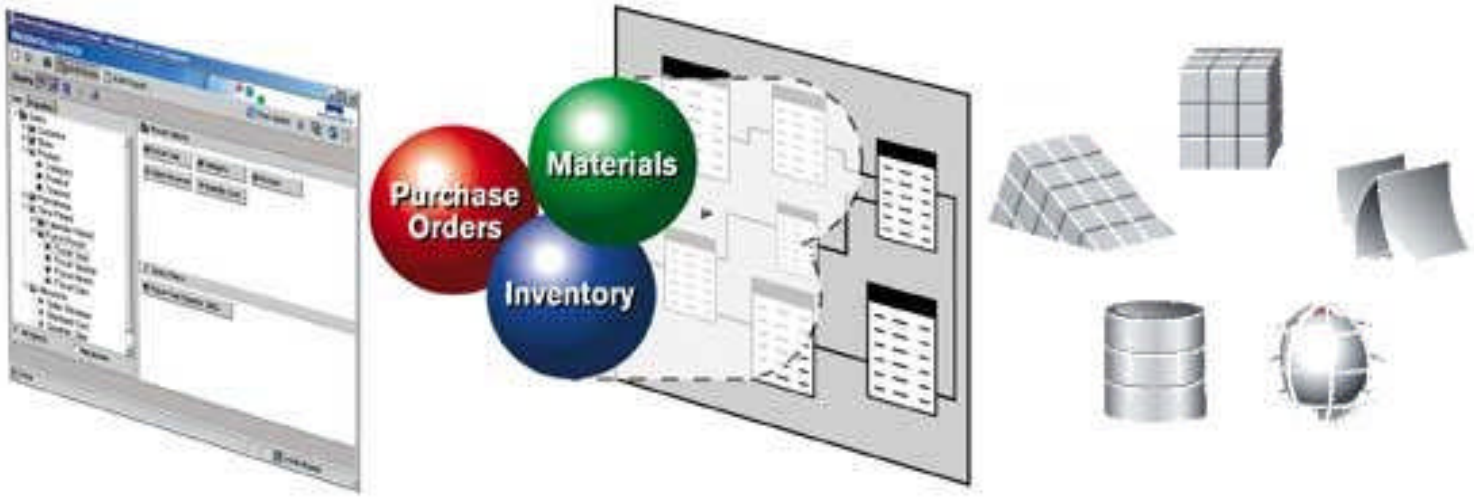
WHO IS DAVE?

- Dedicated to BusinessObjects solutions since 1995
  - Consulting / mentoring / troubleshooting
  - Primary focus on knowledge transfer and client education
- Selected to present at 1996 - 2007 user conference
  - 12 consecutive years as a presenter
- Charter member of BOB
  - http://busobj.forumtopics.com

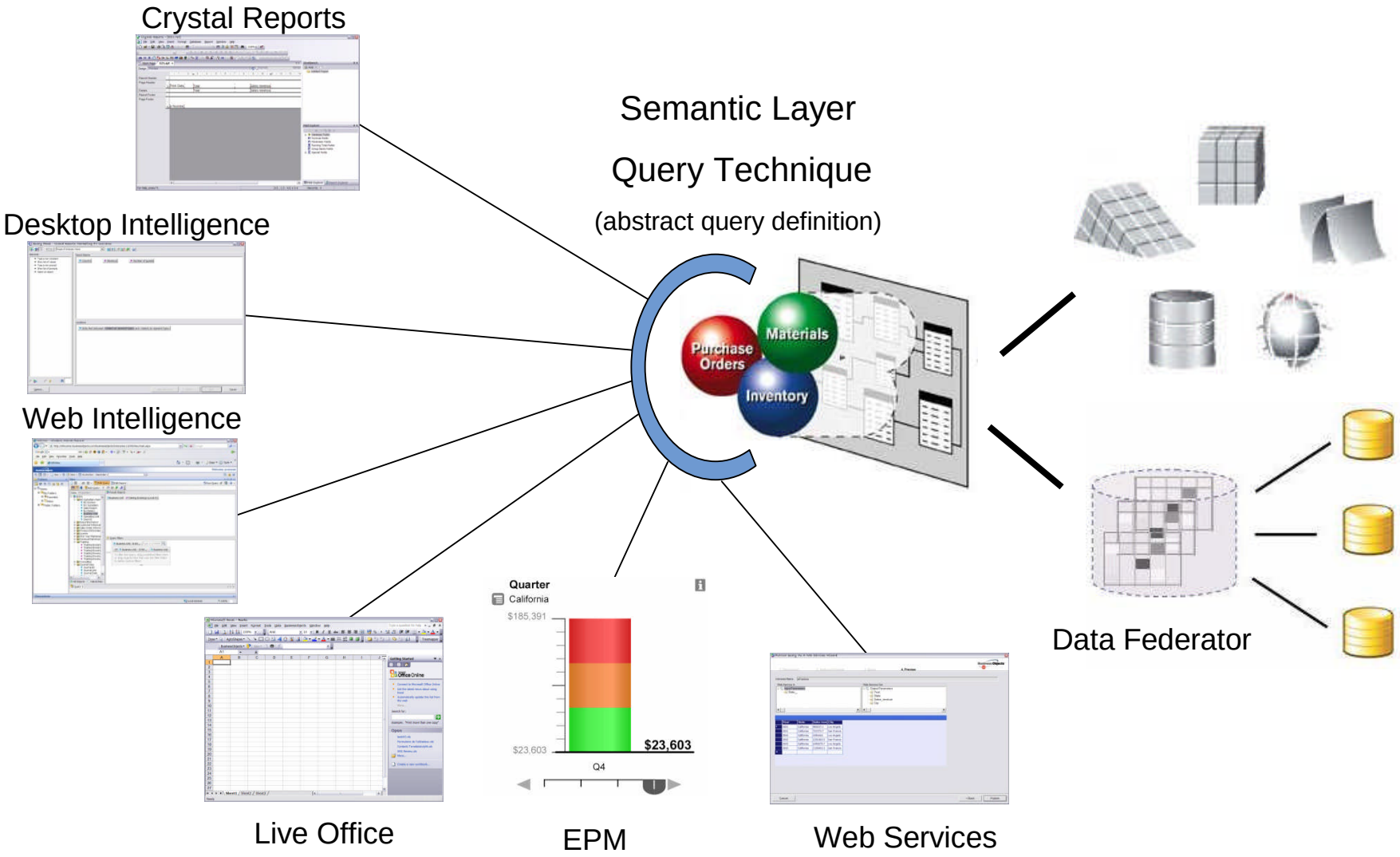
- I Blog! Dave's Adventures in Business Objects
  - http://www.dagira.com

WHAT IS UNIVERSE DESIGNER?

- Provides a business representation of corporate data for end users
- They can access data autonomously using their everyday vocabulary
- Allows IT to keep control of the data access with fine-grained security



WHERE CAN YOU USE UNIVERSE DESIGNER?



DEMONSTRATIONS TODAY

- Demonstration universes
  - Island Resorts Marketing
  - eFashion
- Both demonstration databases were converted to Oracle
  - Required in order to demonstrate advanced features
  - JOIN_BY_SQL
- Full software configuration
  - BusinessObjects Enterprise XI Release 2 Service Pack 2
  - Microsoft SQL Server is used for the repository
  - Oracle 10g is used for both databases

IN DEPTH INDEX AWARENESS

Strategy without tactics is the slowest route to victory. Tactics without strategy is the noise before defeat.

– Sun Tzu

WHAT IS INDEX AWARENESS?

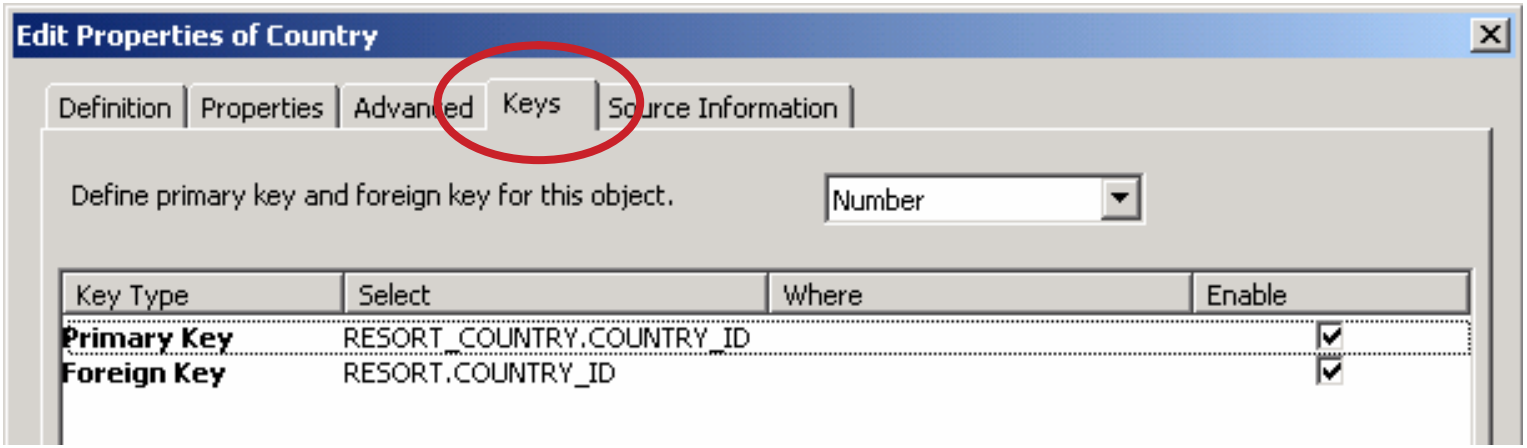
- What does the Universe Designer help say?

  Object keys allow Universe Designer to generate more efficient SQL by filtering on primary key values and eliminating unnecessary joins.

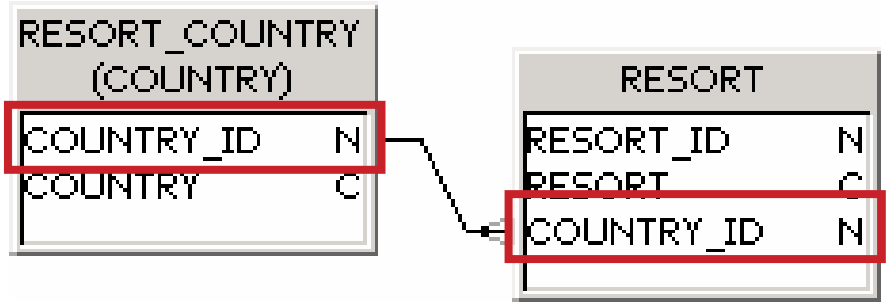
- That's great, but what does it mean?
- How do I set it up?
- When does it work?
- When does it not work?

SETTING UP INDEX AWARENESS

- Select an object and navigate to the keys tab



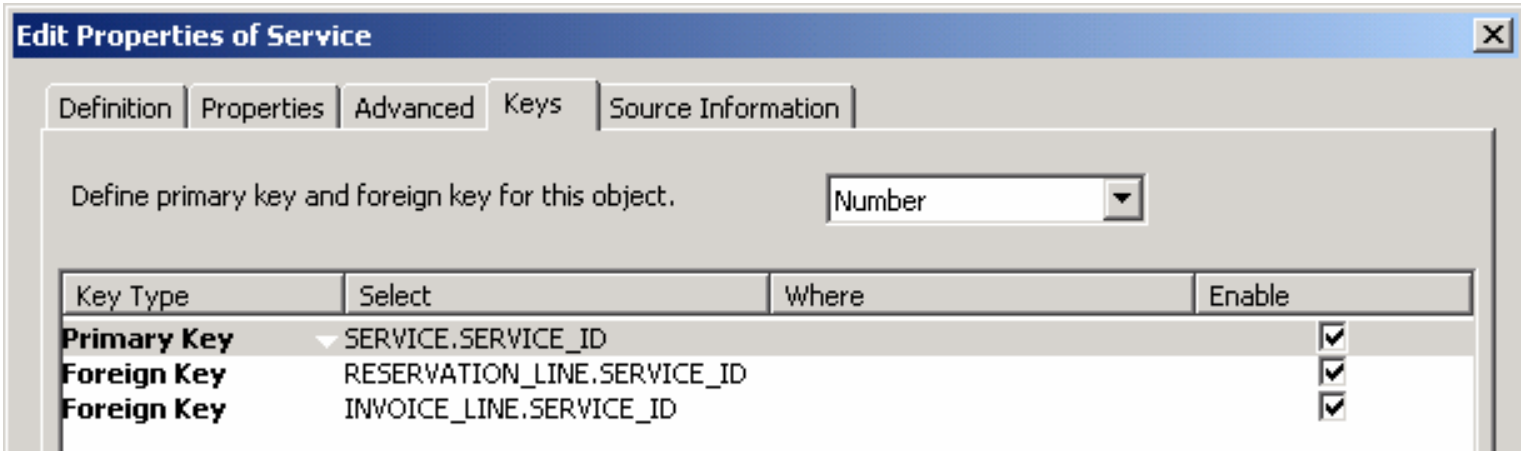
- Key info is driven by the table relationships

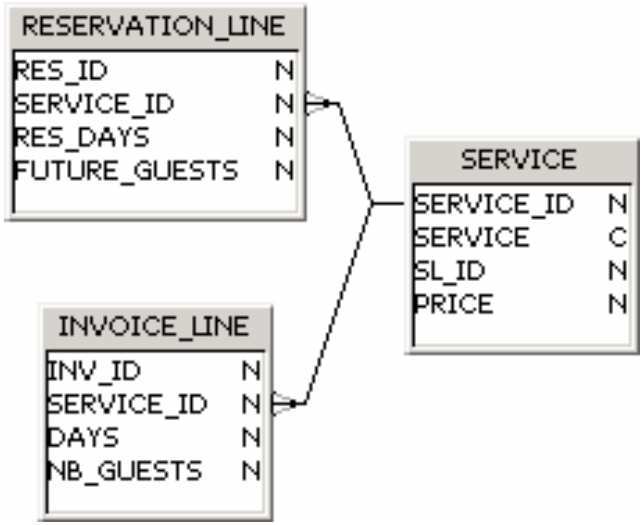


- Repeat for each object from that table

USING MULTIPLE KEYS

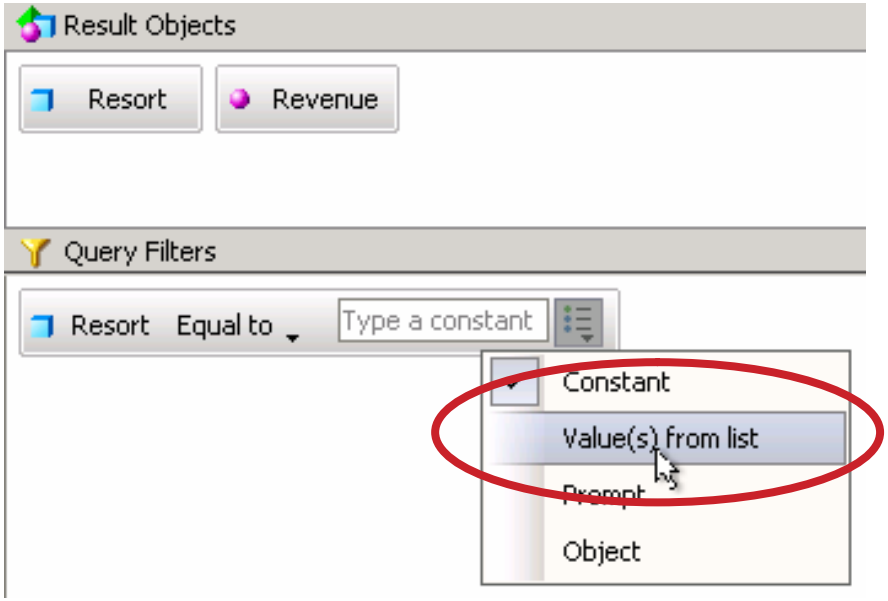
- Index aware can go in more than one direction





WHAT DOES IT DO FOR ME?

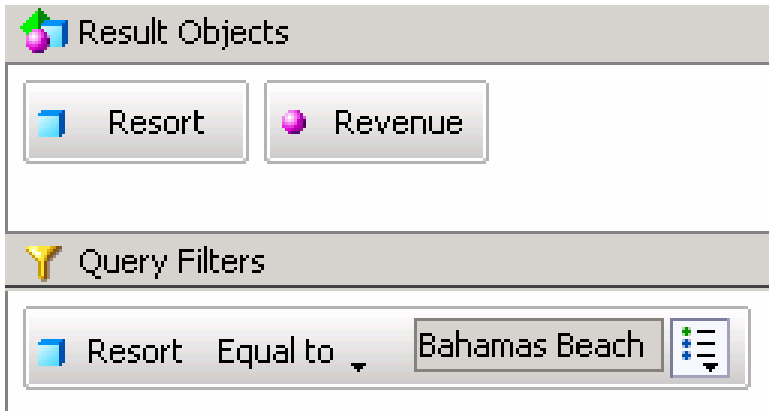
- Index awareness is not an “always on” feature
  - In some cases it works
  - In others it does not
- When it works…
  - Keys are defined
  - User selects from the list of values
- When it does not work
  - User types the value manually
  - User responds to a prompt



EFFICIENT SQL

- When a user selects from an index aware list of values…

```
SELECT
  max( RESORT.resort  ),
  sum(INVOICE_LINE.DAYS * INVOICE_LINE.NB_GUESTS *
SERVICE.PRICE)
FROM  RESORT,  INVOICE_LINE,  SERVICE,  SERVICE_LINE
WHERE  ( RESORT.RESORT_ID=SERVICE_LINE.RESORT_ID  )
  AND  ( SERVICE_LINE.SL_ID=SERVICE.SL_ID )
  AND  ( SERVICE.SERVICE_ID=INVOICE_LINE.SERVICE_ID  )
  AND    RESORT.RESORT_ID  =  2
GROUP BY
  RESORT.RESORT_ID
```



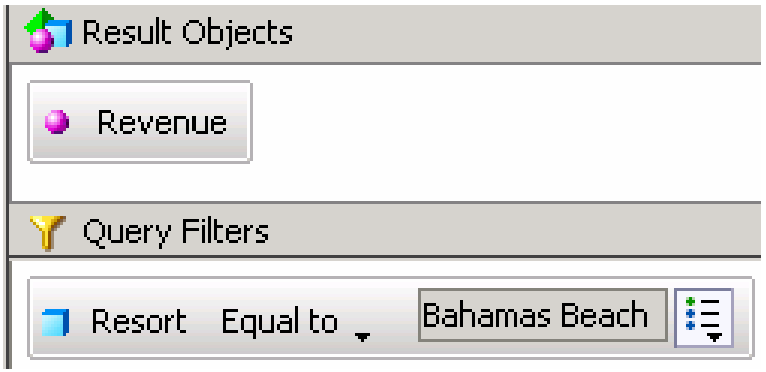
Demonstration 1 – Index aware queries

EVEN MORE EFFICIENT SQL

- If the index aware object is not in the output set

```
SELECT
  sum(INVOICE_LINE.DAYS * INVOICE_LINE.NB_GUESTS *
SERVICE.PRICE)
FROM  INVOICE_LINE,  SERVICE,  SERVICE_LINE
WHERE  ( SERVICE_LINE.SL_ID=SERVICE.SL_ID  )
  AND  ( SERVICE.SERVICE_ID=INVOICE_LINE.SERVICE_ID  )
  AND    SERVICE_LINE.RESORT_ID  =  2
```

- Query changes
  - Condition moves to the foreign key
  - Resort table is eliminated

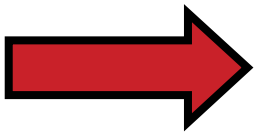


Demonstration 2 – Index aware query with no result dimension

FACT-ONLY QUERIES ARE POSSIBLE

- The query is extremely efficient…
- … but the report is not very useful

```
SELECT
  sum(SHOP_FACTS.Amount_sold)
FROM
  SHOP_FACTS
WHERE
  (SHOP_FACTS.WEEK_KEY  =  233
   AND
   SHOP_FACTS.ARTICLE_CODE  =  177264
   AND
   SHOP_FACTS.SHOP_CODE  =  203)
```

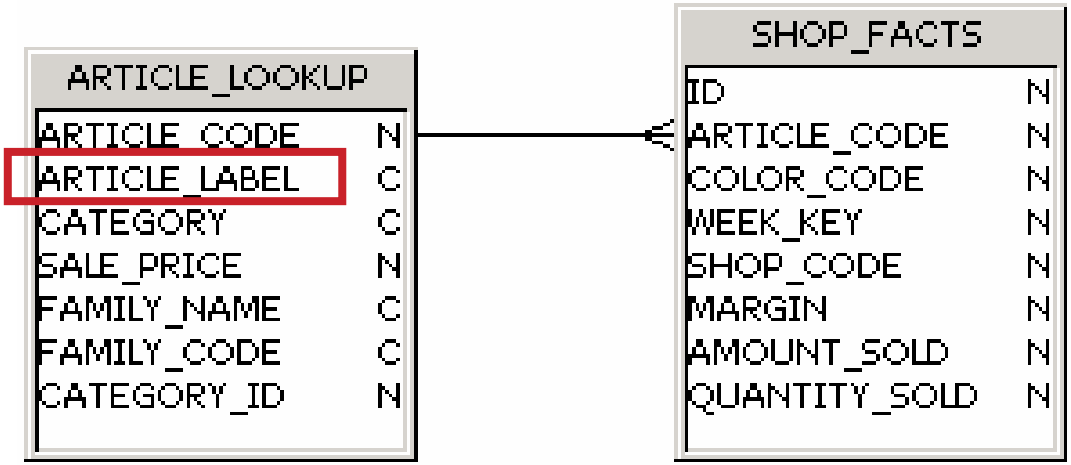


Sales revenue
$4,309

Demonstration 3 – Index aware fact table query

INDEX AWARE LIMITATIONS

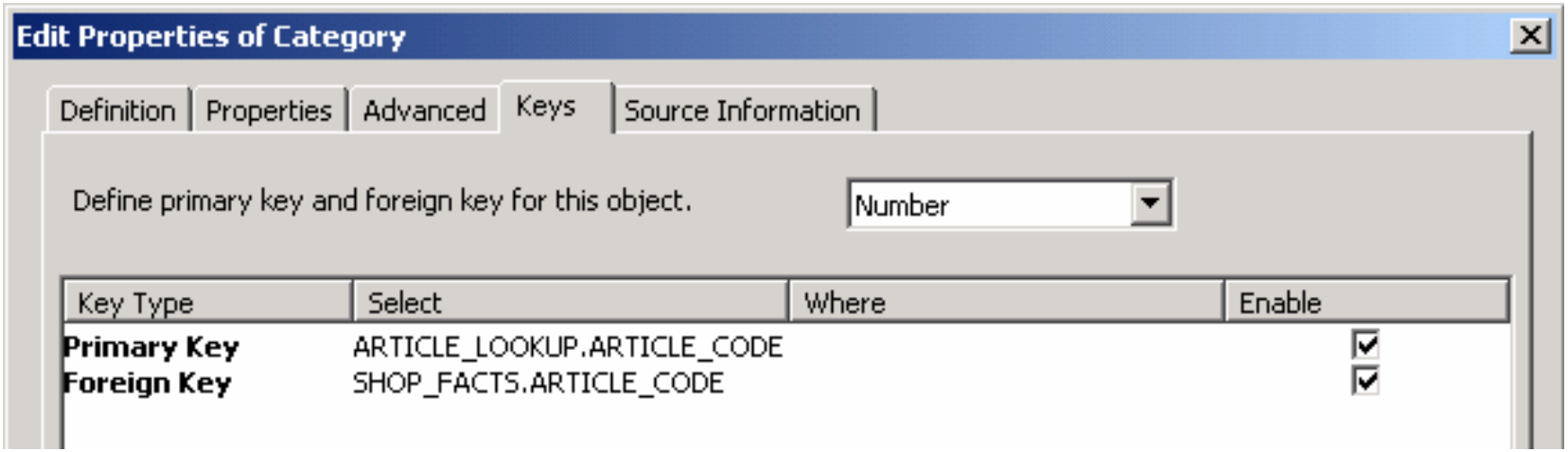
- Keys are defined for individual objects rather than tables
  - Object LOV is altered automatically
  - That is why the LOV must be invoked
- Only unique values are candidates
  - Snow-flaking may be required to fully leverage this feature
- Let's review a simple case in Universe Designer



💻 Demonstration 4 – Universe Designer (covers the next several slides)

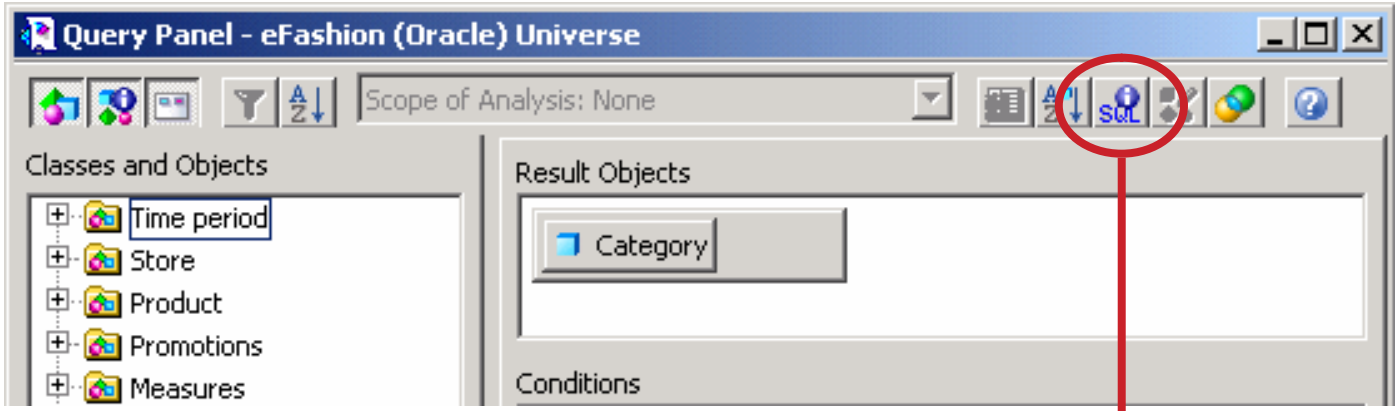
CATEGORY KEYS DEFINED

- The category object comes from the ARTICLE_LOOKUP table
- Foreign key is in the SHOP_FACTS table



LOV UPDATES AUTOMATICALLY

- LOV contains an invisible reference to the KEY column

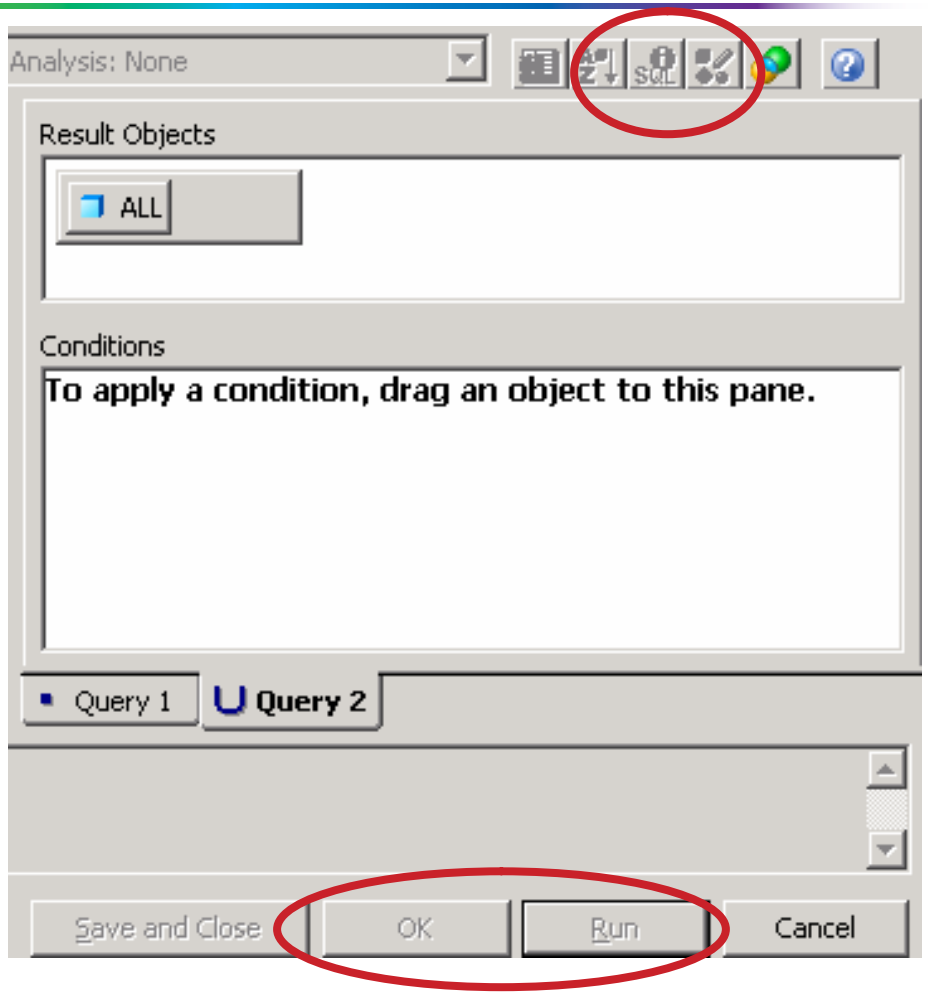



```
SELECT DISTINCT
  ARTICLE_LOOKUP.CATEGORY,
  ARTICLE_LOOKUP.ARTICLE_CODE
FROM
  ARTICLE_LOOKUP
```

- Key column is what drives index aware

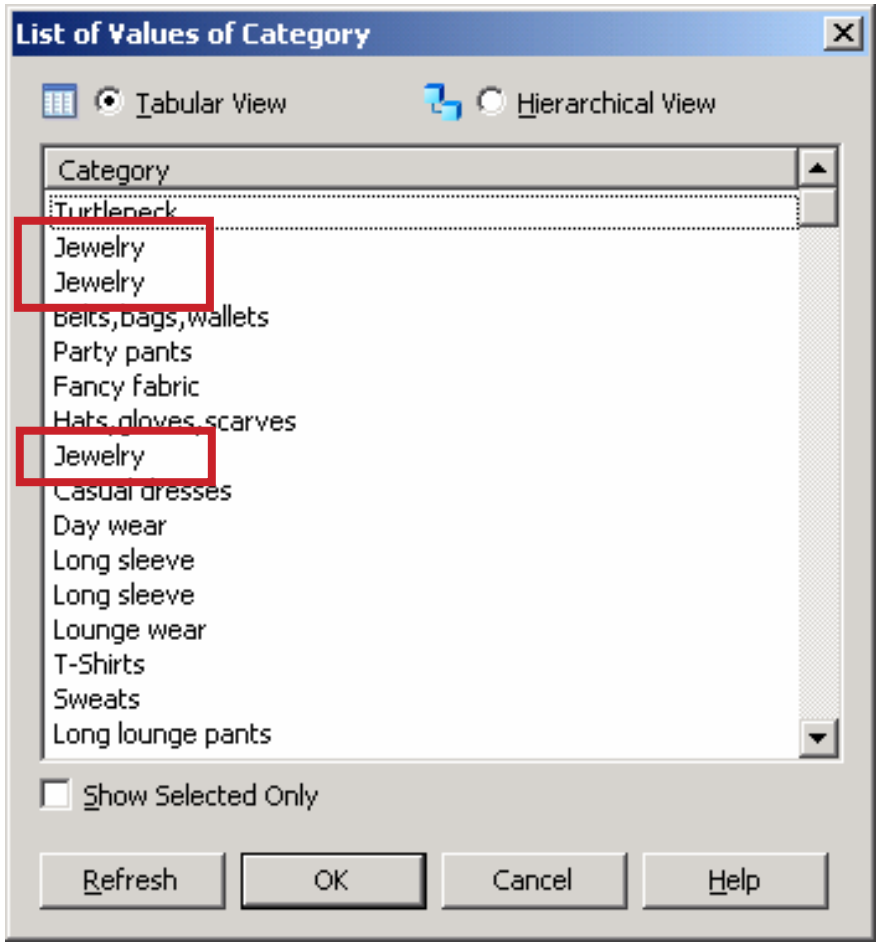
CANNOT ADD “ALL” TO THE LOV

- Because the LOV contains two columns
  - One dimension value
  - One invisible key value
- SQL cannot be generated if “ALL” is added to the LOV



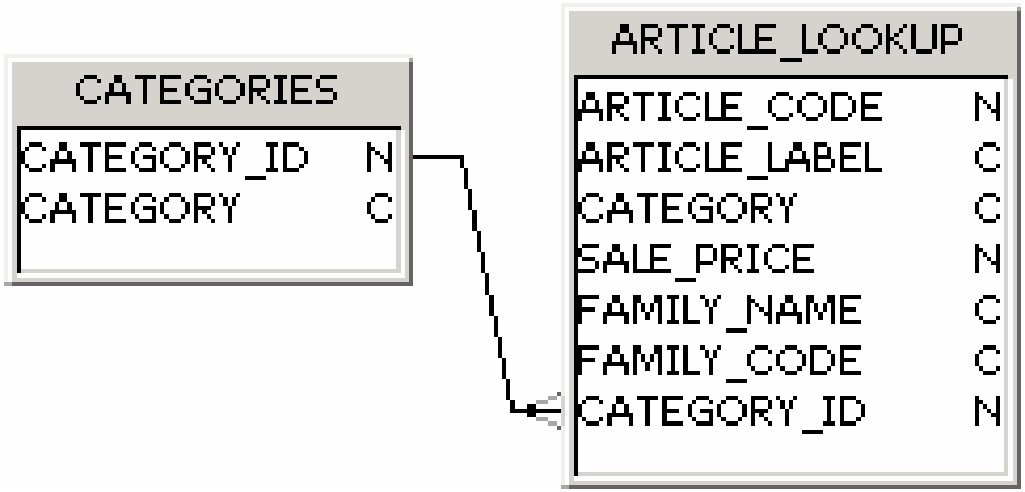
LOV VALUES LOOK STRANGE…

- Article key is at a lower level than category
- Category values duplicate because of the addition of this column



FIXING THE CATEGORY OBJECT

- A snowflake dimensional model works better than a star



FIXING THE CATEGORY OBJECT

- More opportunities to define keys
- More opportunities for table elimination
- Process
  - Create and populate the snowflake table (not via Universe Designer)
  - Include the table, add joins, update contexts
  - Set up the keys and test the results
- Do not use a derived table as you won't have a key

💻 Demonstration 5 – Universe Designer

THINGS THAT DO NOT WORK

- Manual LOV selections are required
- Prompt LOV selections do not invoke this feature
- Index awareness and aggregate awareness do not mix
  - Aggregate aware objects can reference more than one table
  - Index awareness requires a unique primary key definition
- May require snowflake tables
  - Only unique element from a dimension table can be index aware
  - Non-unique values will have improper LOV contents

💻 Demonstration 6 – Prompts with index aware

IS IT WORTH SETTING UP?

- Yes, if you have a lot of ad-hoc users
- Yes, if you are working with a normalized or snowflake structure with lots of key opportunities
- Yes, if you have extra time during your implementation

- Probably not, if you have a star schema
  - Not enough options to define key values

- No, if you deliver primarily canned reports with prompts
- No, if you want to include “ALL” as an option in your LOV
- No, if you use `@Aggregate_Aware()`
  - In my opinion aggregate tables provide more benefit

BONUS MATERIAL

- The primary_key undocumented option enables index awareness on Designer prompts
  - Prompt on key column
  - List of values from related attribute

```
RESORT.RESORT_ID IN @Prompt('Please select
Resort','A','Resort\Resort',multi,primary_key)
```

- The primary_key attribute is not documented…
- … but does appear to work
- Use at your own risk

💻 Demonstration 6a – Prompts with index aware and primary_key

WHAT IS INDEX AWARENESS?

- What does the Universe Designer help say?

  Object keys allow Universe Designer to generate more efficient SQL by filtering on primary key values and eliminating unnecessary joins.

- … but only when the user manually selects a value from a list of values on the query panel
- No prompts
- No aggregate tables
- No “ALL” option
- Limited to key (unique) dimensions

LEVERAGING SHORTCUT JOINS

If ignorant both of your enemy and yourself, you are certain to be in peril.

– Sun Tzu

WHAT ARE SHORTCUT JOINS?

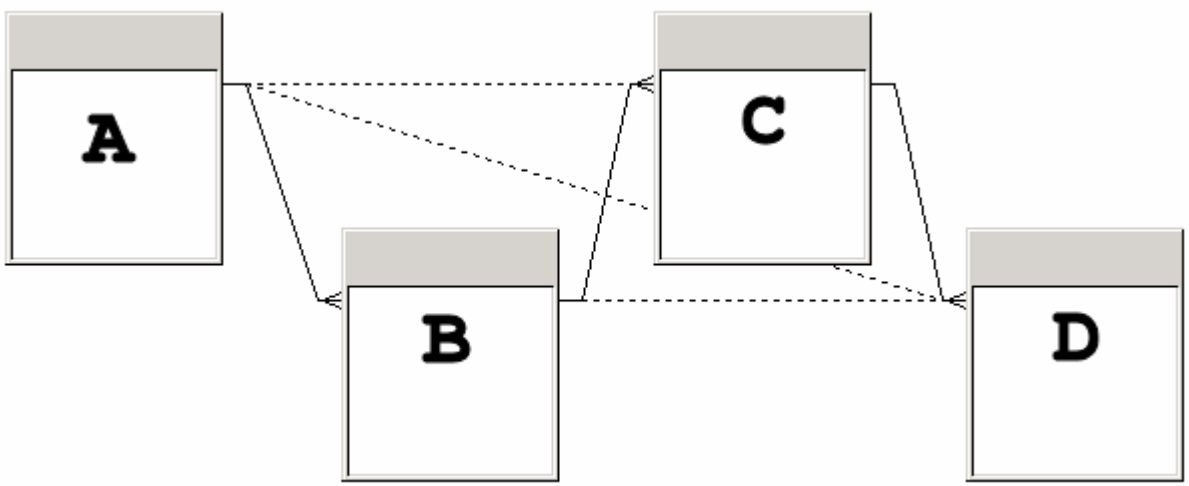
- What does the Universe Designer help say?

  A shortcut join is a join that provides an alternative path between two tables. Shortcut joins improve the performance of a query by not taking into account intermediate tables, and so shortening a normally longer join path.

- This is the **only** purpose of a shortcut
- Do not use shortcuts to resolve loops
  - Use the appropriate method (alias or context) instead
  - Loops will not be discussed today

HOW MANY SHORTCUTS CAN I TAKE?

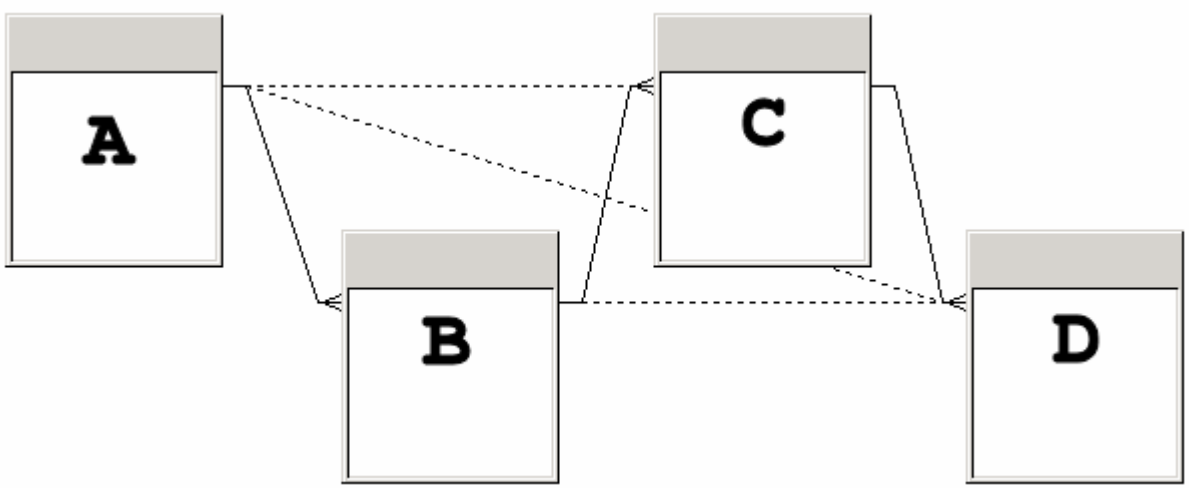
- Multiple shortcut joins can be used at the same time
- … but only in the right situation
- For example consider these four tables



- Shortcuts are applied only if they remove a table
- Shortcuts do not provide an alternate route, only a shorter path

HUH?

- Repeated from prior slide: Shortcuts do not provide an alternate route, only a shorter path
- Consider a query with objects from table A, C, and D



- The only valid path is A – C – D, using shortcut A – C
- Why not A – D – C instead?

TABLE REMOVAL ALGORITHM

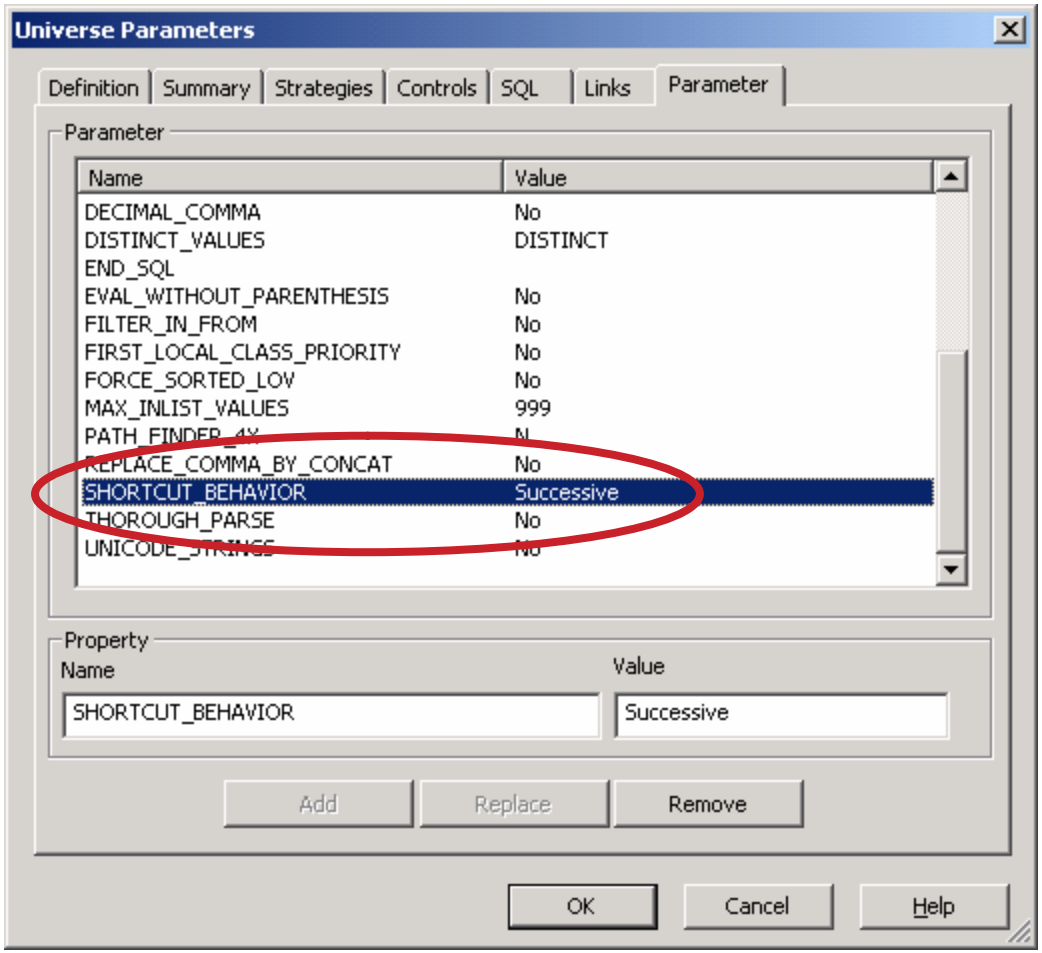
- Query includes objects from table A, C, and D
- Shortcut A – C is considered as it drops table B, which is not needed
- Shortcut A – D is considered but cannot be used as it drops table C which is needed
- Therefore the only valid query path is A – C – D
- This avoids a “backwards” join which could result in a Cartesian product

SHORTCUT_BEHAVIOR

- There is a parameter in the .PRM file (6.0) or the universe parameters (6.5 and higher) that configures the shortcut process
- 6.0 PRM file entry is GLOBAL_SHORTCUTS
  - N or Missing = standard shortcut behavior
  - Y = try to use every shortcut that is available
- 6.5 + universe parameter SHORTCUT_BEHAVIOR
  - Successive = standard shortcut behavior
  - Global = try to use every shortcut available
  - Note: Universe Designer help (XI R2) says global is the default which does not appear to be the case
- When should you change this?

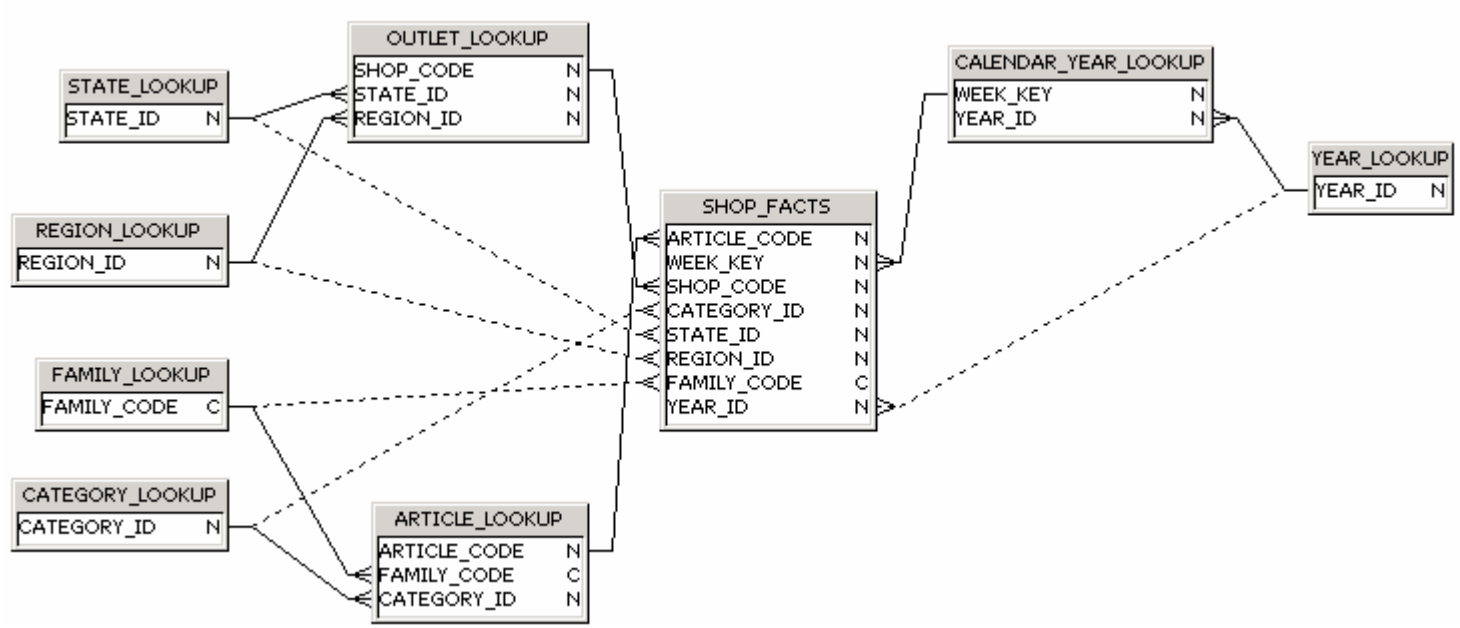
UNIVERSE PARAMETER SETTING

- You may have to add the setting if it does not already exist…



SHORTCUT_BEHAVIOR BEHAVIOR

- Consider a modified version of eFashion



- Category, family (product line), region, state, and year dimensions have been snowflaked
- Shortcut joins exist between all snowflake tables and the fact table

SHORTCUT_BEHAVIOR SUCCESSIVE

```
SELECT
  max( REGION_LOOKUP.REGION_NAME  ),
  max( STATE_LOOKUP.STATE  ),
  sum(SHOP_FACTS.Amount_sold)
FROM
  REGION_LOOKUP,
  STATE_LOOKUP,
  SHOP_FACTS,
  OUTLET_LOOKUP
WHERE
  ( OUTLET_LOOKUP.SHOP_CODE=SHOP_FACTS.SHOP_CODE  )
  AND  ( STATE_LOOKUP.STATE_ID=OUTLET_LOOKUP.STATE_ID  )
  AND  ( REGION_LOOKUP.REGION_ID=OUTLET_LOOKUP.REGION_ID
)
GROUP BY
  REGION_LOOKUP.REGION_ID,
  STATE_LOOKUP.STATE_ID
```

- Demonstration 7 – Shortcut joins in a snowflake schema

SHORTCUT_BEHAVIOR GLOBAL

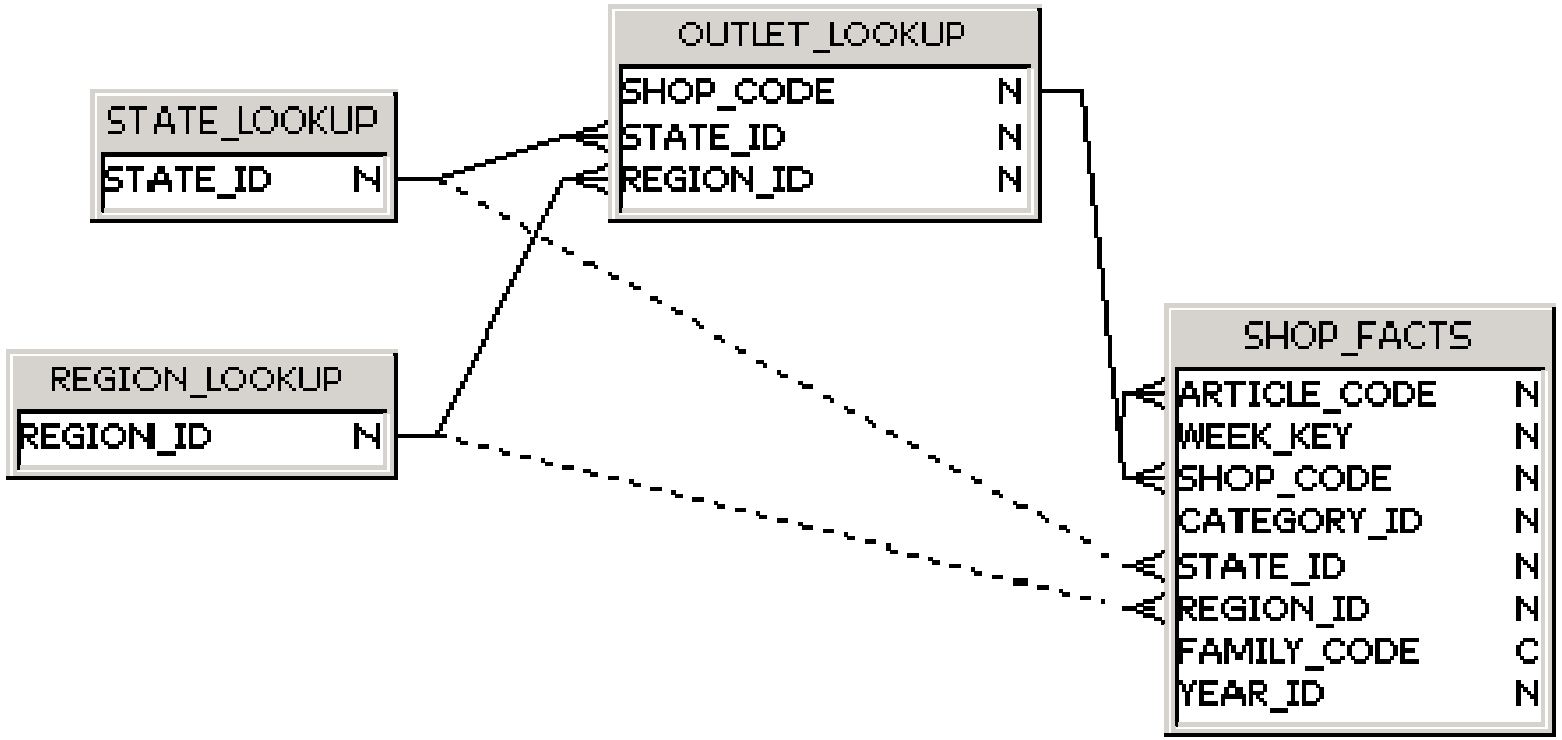
- Removing the OUTLET_LOOKUP table makes a more efficient query in the snowflake universe

```
SELECT
  max( REGION_LOOKUP.REGION_NAME  ),
  max( STATE_LOOKUP.STATE  ),
  sum(SHOP_FACTS.Amount_sold)
FROM
  STATE_LOOKUP,
  SHOP_FACTS,
  REGION_LOOKUP
WHERE
  ( STATE_LOOKUP.STATE_ID=SHOP_FACTS.STATE_ID )
  AND  ( REGION_LOOKUP.REGION_ID=SHOP_FACTS.REGION_ID )
GROUP BY
  REGION_LOOKUP.REGION_ID,
  STATE_LOOKUP.STATE_ID
```

💻 Demonstration 7 (Repeat) – Shortcut joins in a snowflake schema

SHORTCUT_BEHAVIOR GLOBAL

- A global setting changes the shortcut algorithm



SHORTCUT_BEHAVIOR GLOBAL

- Two potential shortcuts
  - STATE_LOOKUP to SHOP_FACTS
  - REGION_LOOKUP to SHOP_FACTS
- Either would drop the OUTLET_LOOKUP table
- Successive processes joins one at a time
  - OUTLET_LOOKUP won’t be dropped because the “other” join requires that table
- Global processes all at once and recognizes that it can drop the table

SHORTCUT JOINS + INDEX AWARENESS

- The SQL on the prior page showed this

  `max( REGION_LOOKUP.REGION_NAME )`

- This is because index keys have been set up in the snowflake universe
- How does index aware interact with shortcut joins?

Demonstration 8 – Shortcut joins and index aware objects

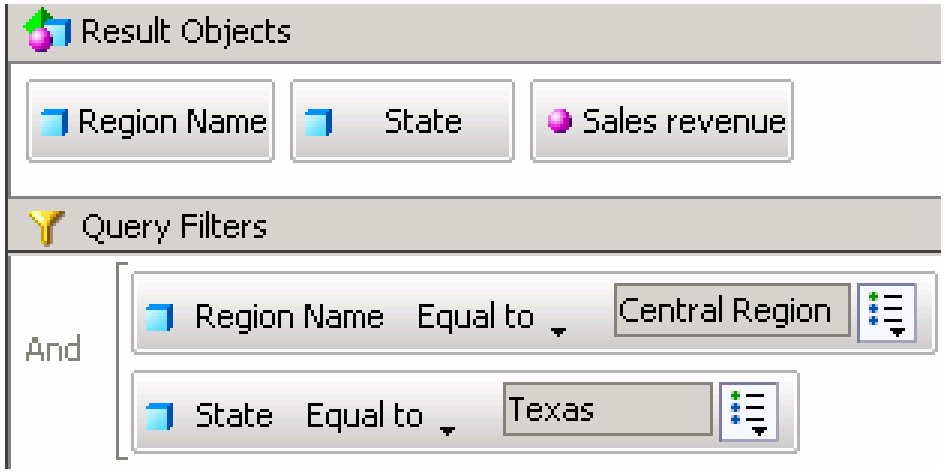
SQL SHORTCUTS ALL THE WAY AROUND

- Shortcut joins are used
- Index keys are used

```
SELECT
  max( REGION_LOOKUP.REGION_NAME  ),
  max( YEAR_LOOKUP.YEAR  ),
  sum(SHOP_FACTS.Amount_sold)
FROM  REGION_LOOKUP,  SHOP_FACTS,  YEAR_LOOKUP
WHERE
  ( REGION_LOOKUP.REGION_ID=SHOP_FACTS.REGION_ID  )
  AND  ( SHOP_FACTS.YEAR_ID=YEAR_LOOKUP.YEAR_ID  )
  AND  (REGION_LOOKUP.REGION_ID  =  2
      AND YEAR_LOOKUP.YEAR_ID  =  3  )
GROUP BY
  REGION_LOOKUP.REGION_ID,
  YEAR_LOOKUP.YEAR_ID
```

SHORTCUTS, INDEXES, AND GLOBAL PARAMETERS, OH MY!

- The last example did not require global shortcut behavior
- What happens when index aware objects are used in combination with global shortcut paths?
- This query uses region name and state
  - Both reference the OUTLET_LOOKUP table for index keys
  - Both have shortcuts to the SHOP_FACTS table



Demonstration 9 – Shortcut joins, index aware objects, and global algorithm

SQL SHORTCUTS ALL THE WAY AROUND

- Shortcut joins are used
- Index keys are used

```
SELECT
  max( REGION_LOOKUP.REGION_NAME  ),
  max( STATE_LOOKUP.STATE  ),
  sum(SHOP_FACTS.Amount_sold)
FROM  STATE_LOOKUP,  SHOP_FACTS,  REGION_LOOKUP
WHERE
  ( STATE_LOOKUP.STATE_ID=SHOP_FACTS.STATE_ID  )
  AND
(REGION_LOOKUP.REGION_ID=SHOP_FACTS.REGION_ID  )
  AND  ( REGION_LOOKUP.REGION_ID  =  2
   AND  STATE_LOOKUP.STATE_ID  =  1 )
GROUP BY  REGION_LOOKUP.REGION_ID,
  STATE_LOOKUP.STATE_ID
```

WHAT ARE SHORTCUT JOINS?

- What does the Universe Designer help say?

  A shortcut join is a join that provides an alternative path between two tables. Shortcut joins improve the performance of a query by not taking into account intermediate tables, and so shortening a normally longer join path.

- Shortcuts are used to eliminate tables not provide alternate paths
- There are two different behaviors for the shortcut algorithm, use the one that fits your needs

USING JOIN_BY_SQL

For to win one hundred victories in one hundred battles is not the acme of skill. To subdue the enemy without fighting is the acme of skill.

– Sun Tzu

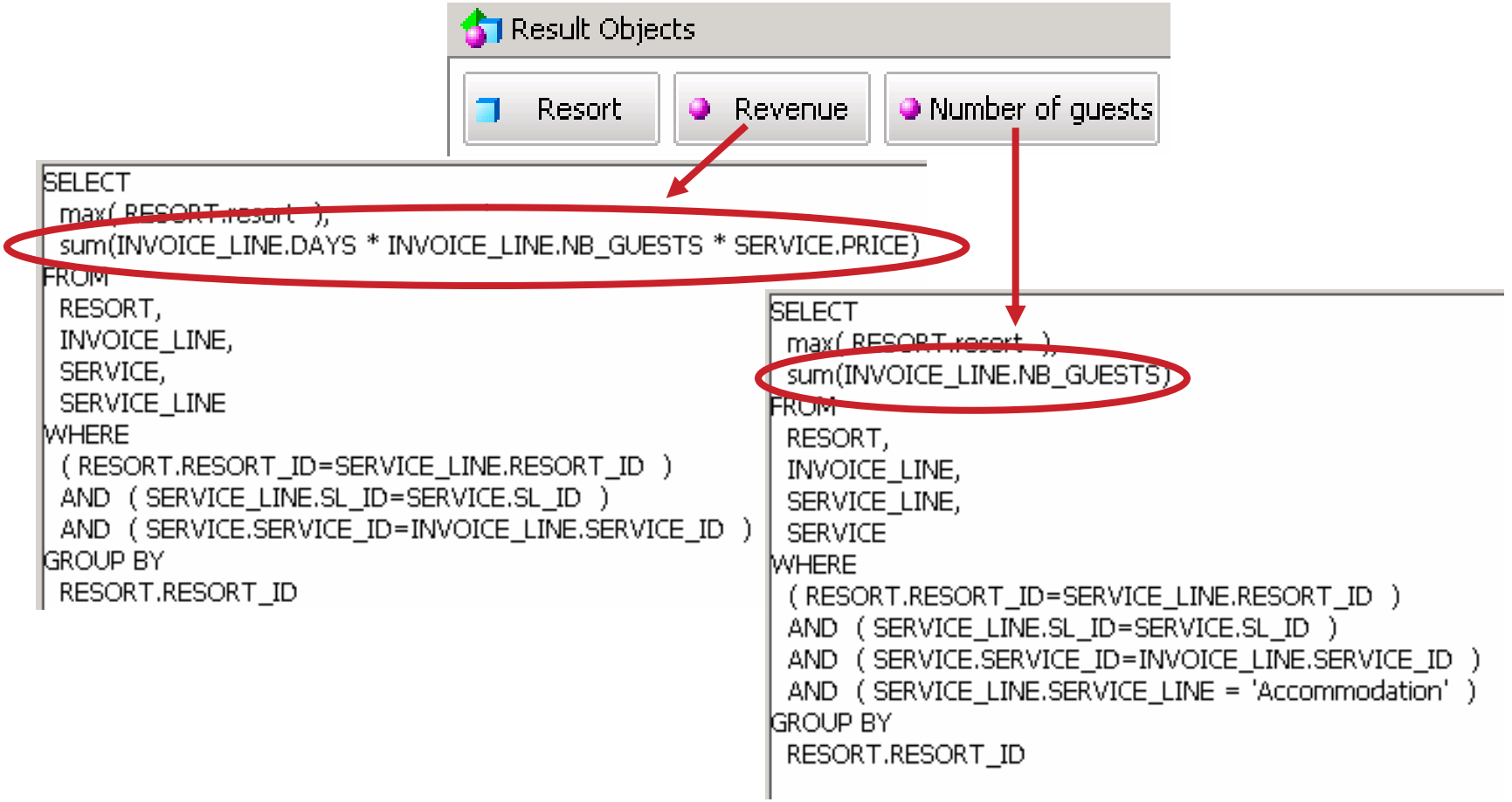
WHAT IS JOIN_BY_SQL?

- What does the Universe Designer help say?

- Despite the lack of information this is a very interesting feature
- It can reduce or eliminate the need for multiple data providers
- This feature is not supported by Desktop Intelligence
- This feature did not work reliably until SP2

BEFORE JOIN_BY_SQL

- The following query would normally generate two select statements



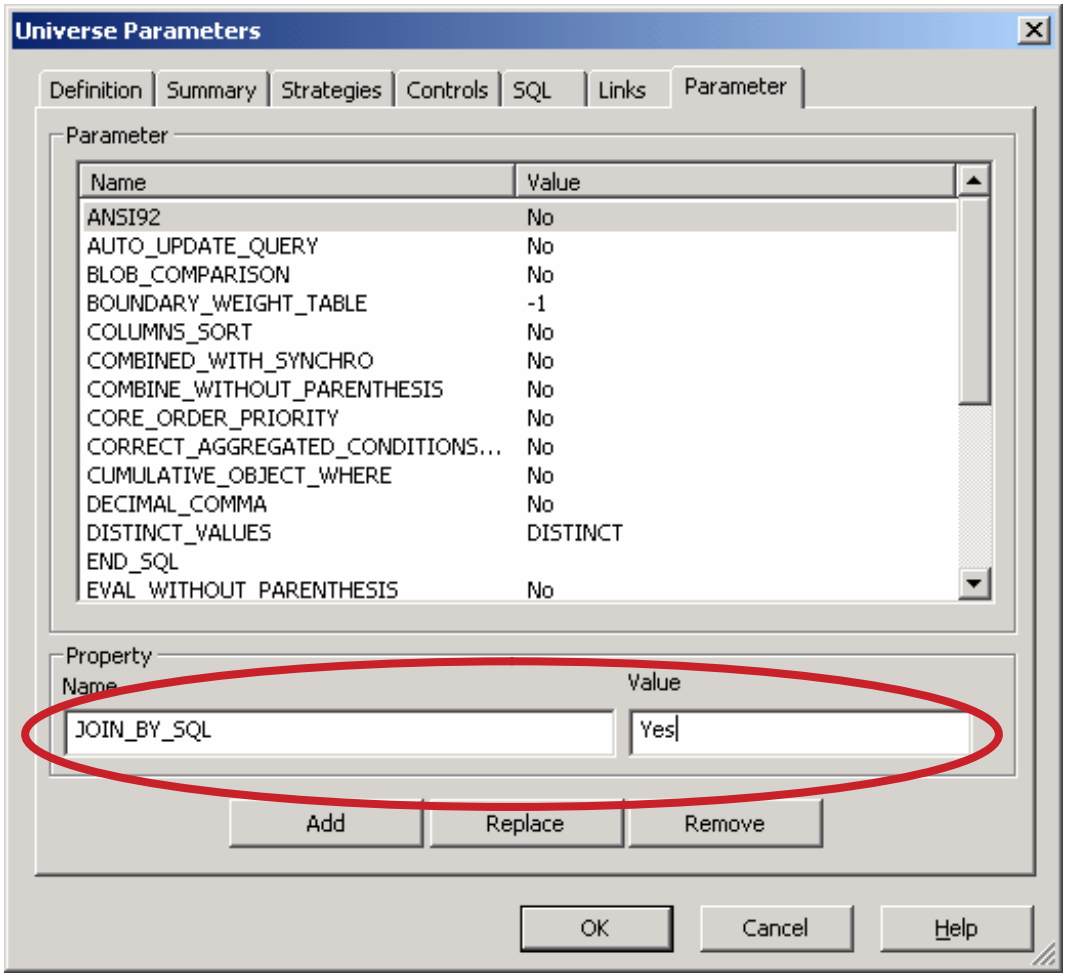
IMPACT OF MULTIPLE QUERIES

- One connection for each query
- One download (fetch) phase for each query
- Resulting data sets are joined (merged) by the client
- Higher load on the Web Intelligence server
- Consider the following
  - Four data providers: CY YTD, PY YTD, CY MTD, and PY MTD
  - Each returns 5,000 rows
  - Four fetch phases
  - 20,000 rows of data on the client
  - Client merges the data
- Where is the power?

 Demonstration 10 – Synchronized queries across contexts

ADDING THE PARAMETER

- File + Parameters
- Parameter tab
- See if it's there…
- … if not, add it in
  - Name = JOIN_BY_SQL
  - Value = Yes



AFTER JOIN_BY_SQL

- The same query generates one combined query

```
SELECT
  NVL( F__1.Axis__1,F__2.Axis__1),
  F__1.M__7,
  F__2.M__20
FROM
( SELECT … ) F__1
FULL OUTER JOIN
( SELECT … ) F__2
ON (F__1.KEY__79 = F__2.KEY__79)
```

- KEY__79 is the Resort ID

```
SELECT
 NVL( F__1.Axis__1,F__2.Axis__1 ),
 F__1.M__7,
 F__2.M__20
FROM
   (
   SELECT
    max( RESORT.resort  ) AS Axis__1,
    sum(INVOICE_LINE.DAYS * INVOICE_LINE.NB_GUESTS * SERVICE.PRICE) AS M__7,
    RESORT.RESORT_ID AS Key__79
   FROM
    RESORT,
    INVOICE_LINE,
    SERVICE,
    SERVICE_LINE
   WHERE
    ( RESORT.RESORT_ID=SERVICE_LINE.RESORT_ID  )
    AND  ( SERVICE_LINE.SL_ID=SERVICE.SL_ID  )
    AND  ( SERVICE.SERVICE_ID=INVOICE_LINE.SERVICE_ID  )
   GROUP BY
    RESORT.RESORT_ID
    )
   F__1
   FULL OUTER JOIN
   (
   SELECT
    max( RESORT.resort  ) AS Axis__1,
    sum(INVOICE_LINE.NB_GUESTS) AS M__20,
    RESORT.RESORT_ID AS Key__79
   FROM
    RESORT,
    INVOICE_LINE,
    SERVICE_LINE,
    SERVICE
   WHERE
    ( RESORT.RESORT_ID=SERVICE_LINE.RESORT_ID  )
    AND  ( SERVICE_LINE.SL_ID=SERVICE.SL_ID  )
    AND  ( SERVICE.SERVICE_ID=INVOICE_LINE.SERVICE_ID  )
    AND  ( SERVICE_LINE.SERVICE_LINE = 'Accommodation'  )
   GROUP BY
    RESORT.RESORT_ID
    )
   F__2
   ON ( F__1.Key__79=F__2.Key__79  )
```

IMPACT OF SINGLE QUERY

- One connection
- One download (fetch) phase
- Resulting data sets are joined (merged) by the server
- Lower load on the Web Intelligence server
- Reconsider the following
  - Four data providers: CY YTD, PY YTD, CY MTD, and PY MTD
  - Each returns 5,000 rows
  - Database merges the result rows via a Full Outer Join pass
  - One fetch phase
  - 5,000 rows of data on the client
  - Client data merge step is gone

💻 Demonstration 10 (Repeat) – JOIN_BY_SQL combined SQL across contexts

PUTTING IT ALL TOGETHER

- Will index awareness, shortcut joins, and JOIN_BY_SQL all play nicely together?
  - Select a region from a list of values
  - Select two measures that will require separate query passes
  - Include values from the snowflake tables
  - What will the query look like?

TRIPLE PLAY COMBINATION

- Full outer join? Yes
- Shortcut join? Yes

```
FROM   YEAR_LOOKUP,
         SHOP_FACTS
WHERE  (
SHOP_FACTS.YEAR_ID =
YEAR_LOOKUP.YEAR_ID )
```

- Index aware? Yes

```
SHOP_FACTS.REGION_ID = 2
```

Year	Sales revenue	Discount
1999	$2,937,592	$572,165
2000	$4,883,547	$237,177
2001	$5,319,184	$718,027

```
SELECT
NVL( F__1.Axis__1,F__2.Axis__1 ),
F__1.M__147,
F__2.M__191
FROM
(
SELECT
max( YEAR_LOOKUP.YEAR ) AS Axis__1,
sum(SHOP_FACTS.Amount_sold) AS M__147,
YEAR_LOOKUP.YEAR_ID AS Key__562
FROM
YEAR_LOOKUP,
SHOP_FACTS
WHERE
( SHOP_FACTS.YEAR_ID=YEAR_LOOKUP.YEAR_ID )
AND
SHOP_FACTS.REGION_ID = 2
```

💻 Demonstration 11 – Triple play (index aware, shortcuts, and JOIN_BY_SQL)

CASE STUDY

- Current project details
  - Teradata 6.2
  - Billions of rows in the fact table
  - Not using @Aggregate_Aware()
    - All aggregation is done by the database
- Index aware has not been implemented due to the number of canned reports with prompts
- Shortcut joins behavior is set to global to leverage snowflake design
- Implementing JOIN_BY_SQL resulted in a 50% reduction in query time and a 75-85% reduction in report size

CONCLUSION

- Entering index keys on appropriate objects can generate very efficient SQL
  - But many things can get in the way of its use
- Shortcut joins can be very helpful for tuning
  - Be sure to know how they are going to interact with the rest of your universe
- JOIN_BY_SQL pushes the data synchronization process back to the database engine
  - It works best with measures
- All of these options can make your universe better but will not fix structural issues
- Make sure your universe works before adding these tweaks

WHAT IS COMING IN PART TWO?

- Some interesting tricks with derived tables and some fancy report variables
  - Making up fake data
  - Getting creative with the DrillFilters() function
- Lots of good stuff about List of Values (LOV) customizations
  - Using “ALL” to simulate optional prompts (standard approach)
  - Using “ALL” with a derived table approach (new technique)
  - Working with cascading LOV queries
  - Adding an “ALL” option to cascading LOV queries

Q&A

- Thank you for your time and attention today
- Questions
  - David G. Rathbun, Integra Solutions
- Contact information
  - 214 637 6622
  - www.IntegraSolutions.net

INTEGRA SOLUTIONS
a business unit of Quorum Business Solutions