Universe Design Techniques Proven to Boost Front-End Performance

David G. Rathbun

Originally presented at BI2012 Conference

Abstract

- This session dives deep into universe development and examines when, why, and how to tweak your existing Business Objects universes for optimal report performance — and when you may need to build new ones. Explore proven techniques for extending a universe to ensure more efficient queries, an optimized end-user experience, and more timely and efficient BI operations. Acquire tips to perform index awareness, such as choosing a value from the list of values (LOV) directly within the query panel, rather than using prompts. Learn how to use aggregate awareness to set up complex logic and step through a demo to see how this results in significantly improved performance on the front end. Gain insight into whether and when to leverage shortcut joins to boost query speed. Explore universe design techniques that provide the best performance when pointing to a data source outside your Business Objects system. View detailed demonstrations of various advanced universe design techniques and leave with proven strategies for incorporating them into your own environment.

About Dave

- Dedicated to BusinessObjects solutions since 1995
  - Consultant and trainer for fifteen years
  - Currently BI Solutions Architect for PepsiCo
  - Note: Content is my own and does not reflect my employer
- 17 consecutive years presenting at major BI conferences
  - United States, Europe, Australia
- Charter member of BOB
  - http://busobj.forumtopics.com

BOB BusinessObjects Board

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

- SAP Mentor for 2009 – 2012

SAP Mentor

Demonstration Platform

- Demonstration universes
  - eFashion
  - Island Resorts Marketing
- Software configuration
  - SAP BusinessObjects Enterprise XI 3.1
  - Oracle 10g
- Business Objects toolset
  - Web Intelligence Rich Client
  - Universe Designer

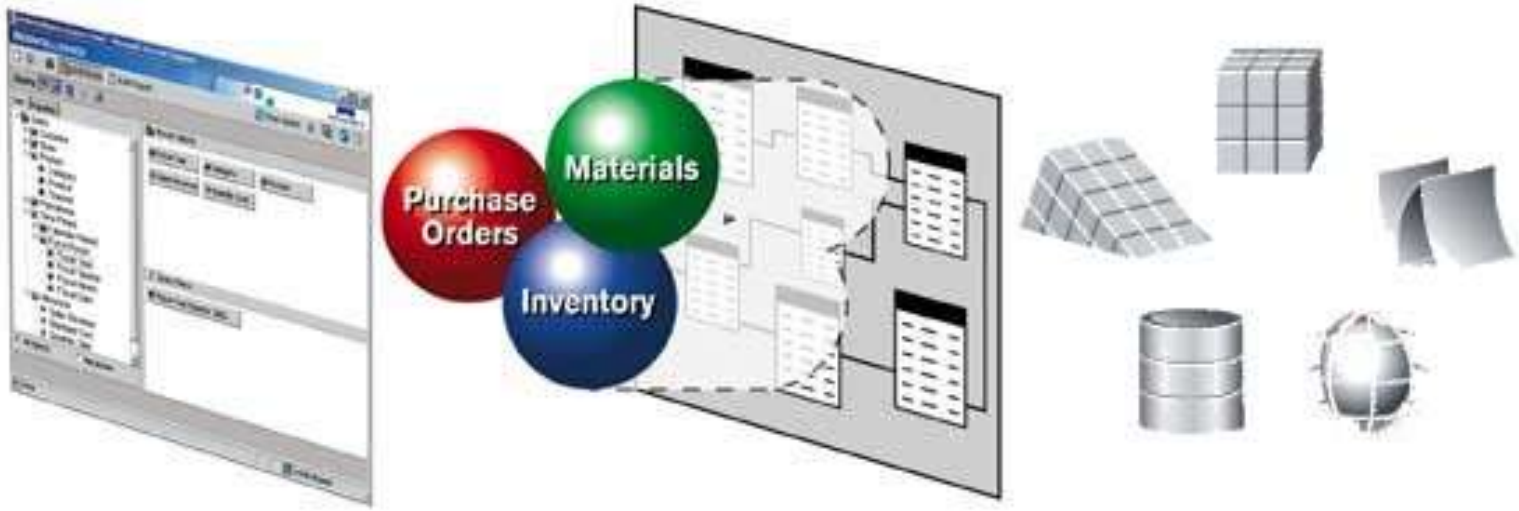
💻 Demonstration slides will be highlighted with this icon

What We'll Cover …

- Universe Concepts
- Index Awareness
- Shortcut Joins
- Aggregate Awareness
- Wrap-up

What Is a Universe?

- The universe is the core of your Business Objects installation
- Presents a semantic layer that translates database structures into business terms



Database Requirements

- None!
- A universe can be built on:
  - Relational model
  - Operational data store (ODS)
  - Star schema
  - Snowflake schema
  - Cubes
- Some designs do have more issues than others
  - Single-fact star schemas are probably the easiest …
  - … but they're still not immune from challenges

What's Important?

- Primary focus on universe design is to provide the correct results
- Secondary concerns are:
  - User-friendly
  - Ease of maintenance
  - Performance
- Today we will assume our universe is correct, and we will discuss various techniques that can be used to improve performance

What We'll Cover …

- Universe Concepts
- Index Awareness
- Shortcut Joins
- Aggregate Awareness
- Wrap-up

What Is Index Awareness?

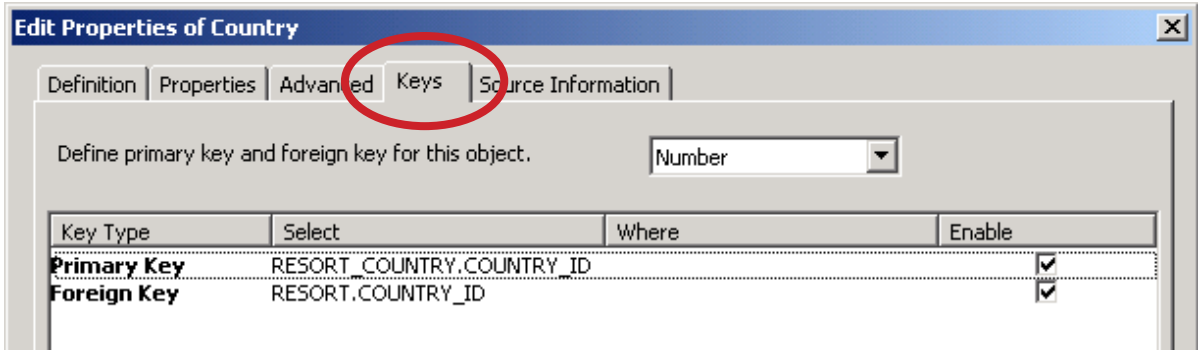
- What does the Universe Designer help say?

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

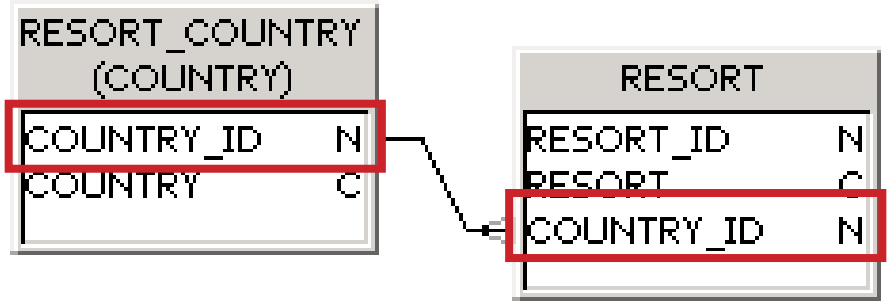
- That's great, but what does it mean?
- How do I set it up?
- When does it work?
- When does it not work?

Setting Up Index Awareness

- Select an object and navigate to the Keys tab



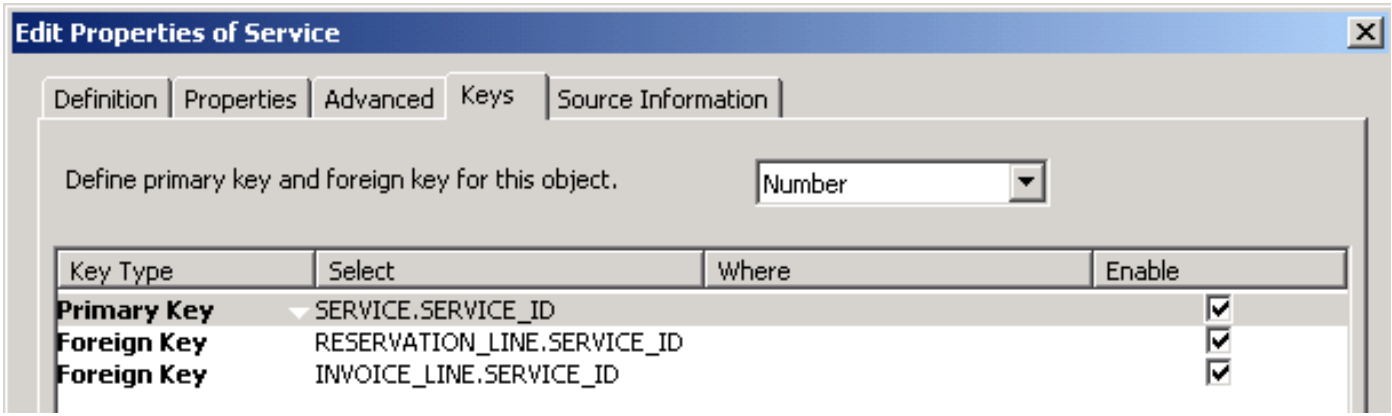
- Key info is driven by the table relationships

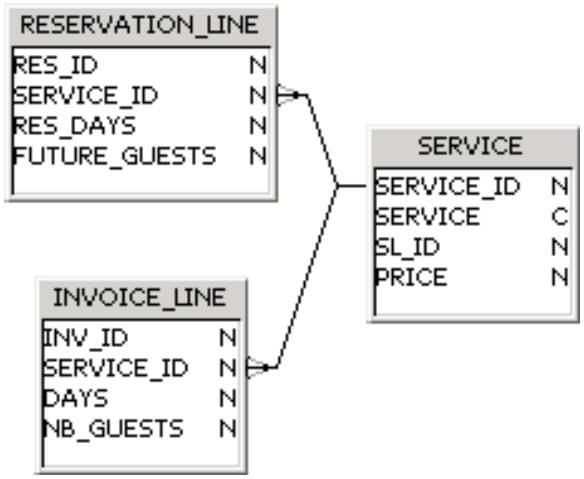


- Repeat for each object from that table

Using Multiple Keys

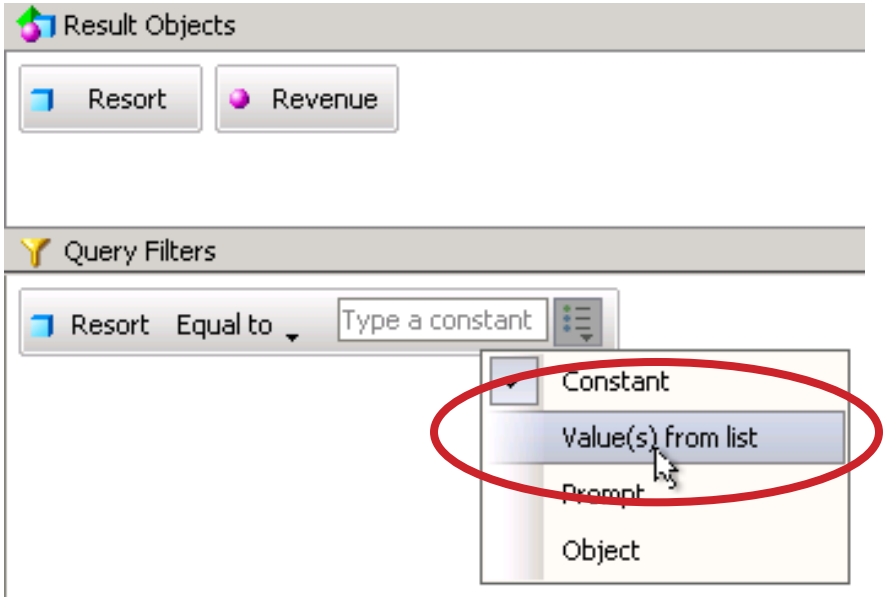
- Index aware can go in more than one direction





What Does It Do For Me?

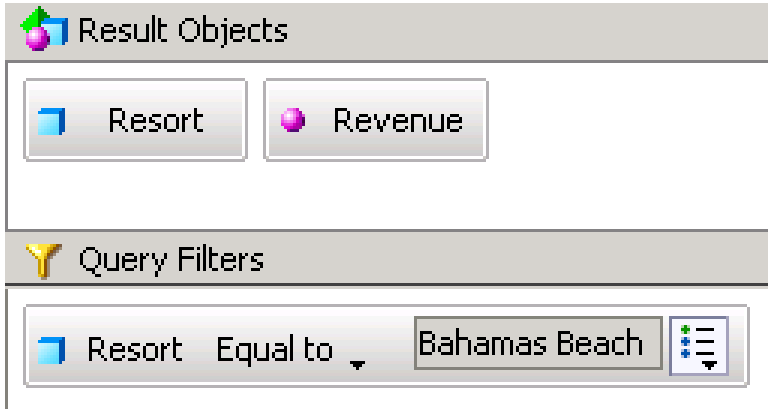
- Index awareness is not an “always on” feature
  - In some cases it works
  - In others it does not
- When it works …
  - Keys are defined
  - User selects from the list of values
- When it does not work
  - User types the value manually
  - User responds to a prompt



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
```



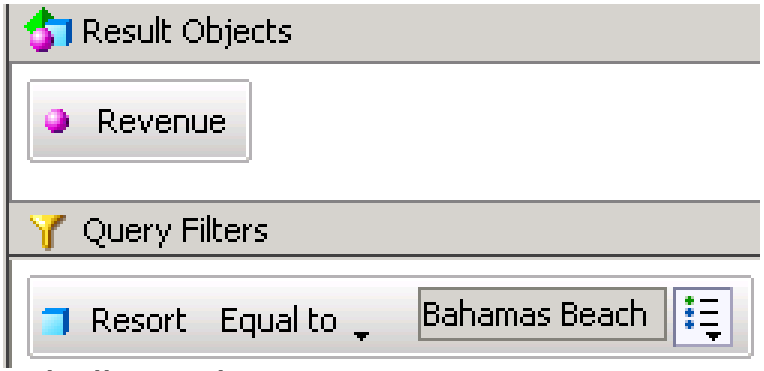
Demonstration 1 – Index aware queries

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

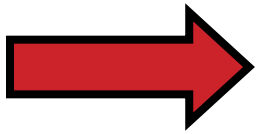


Demonstration 2 – Index aware query with no result dimension

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)
```

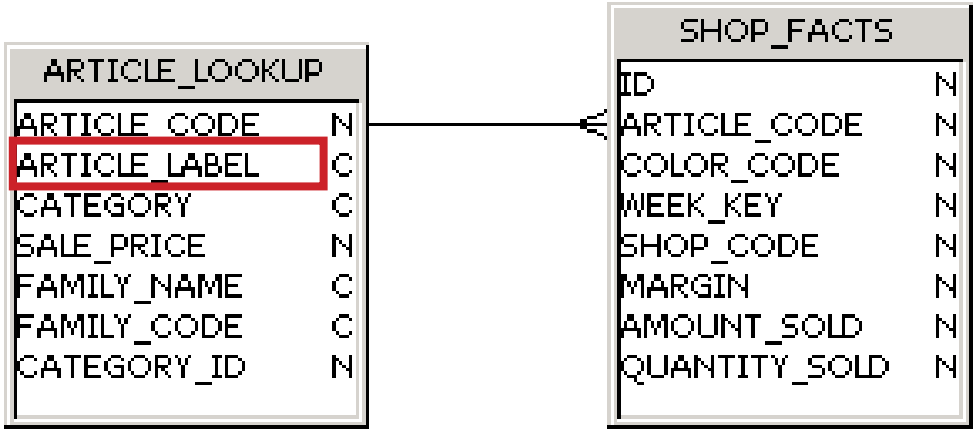


Sales revenue
$4,309

Demonstration 3 – Index aware fact table query

Index Aware Limitations

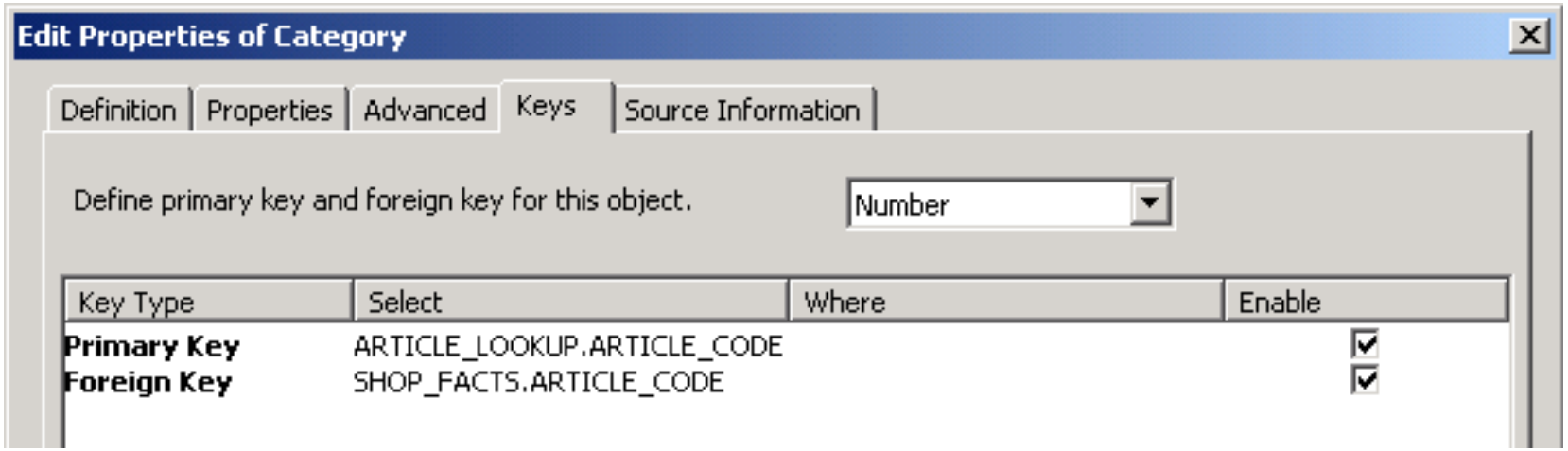
- Keys are defined for individual objects rather than tables
  - Object LOV is altered automatically
  - That is why the LOV must be invoked
- Only unique values are candidates
  - Snow-flaking may be required to fully leverage this feature
- Let's review a simple case in Universe Designer



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

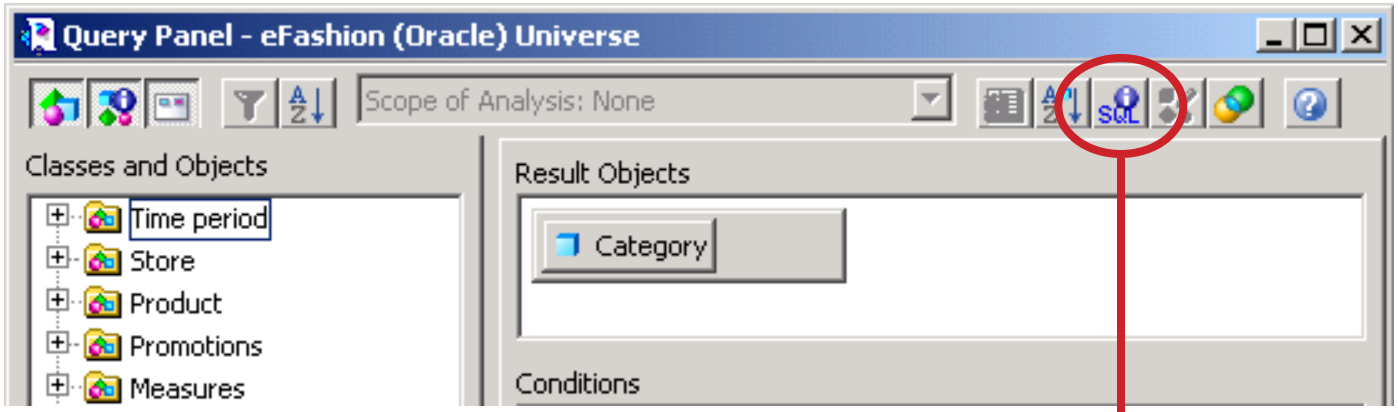
Category Keys Defined

- 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

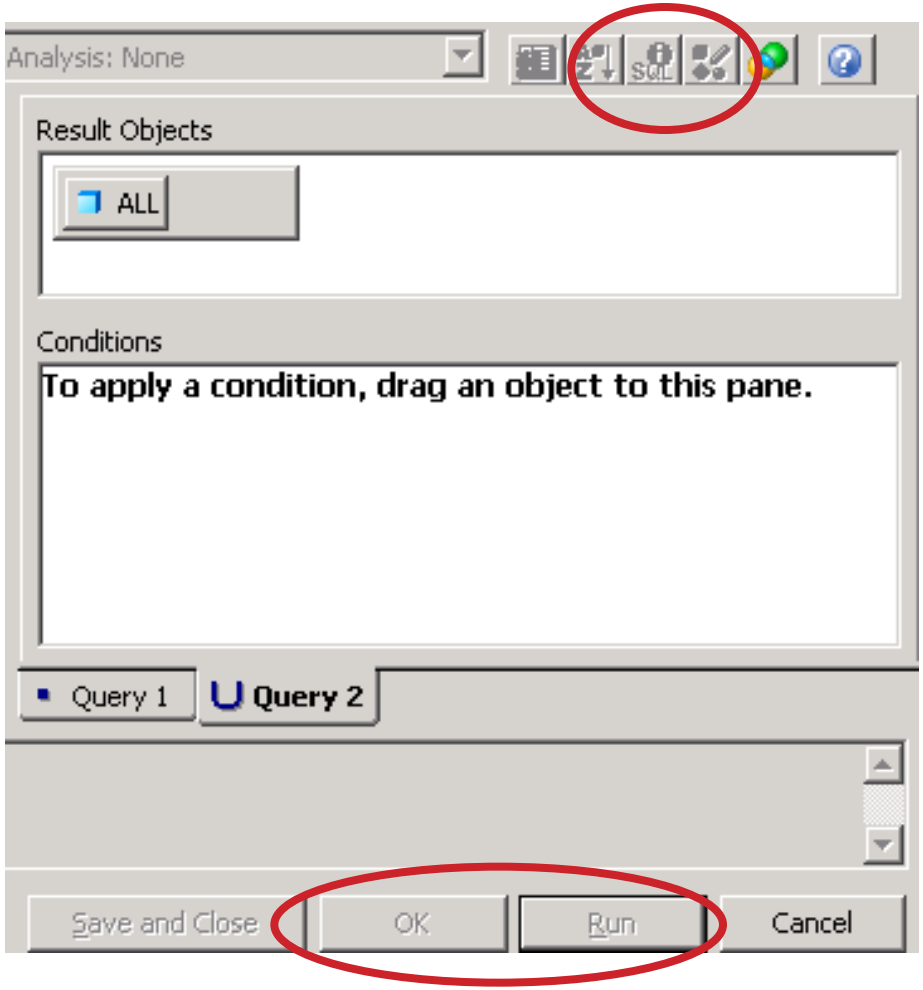



```
SELECT DISTINCT
  ARTICLE_LOOKUP.CATEGORY,
  ARTICLE_LOOKUP.ARTICLE_CODE
FROM
  ARTICLE_LOOKUP
```

- Key column is what drives index aware

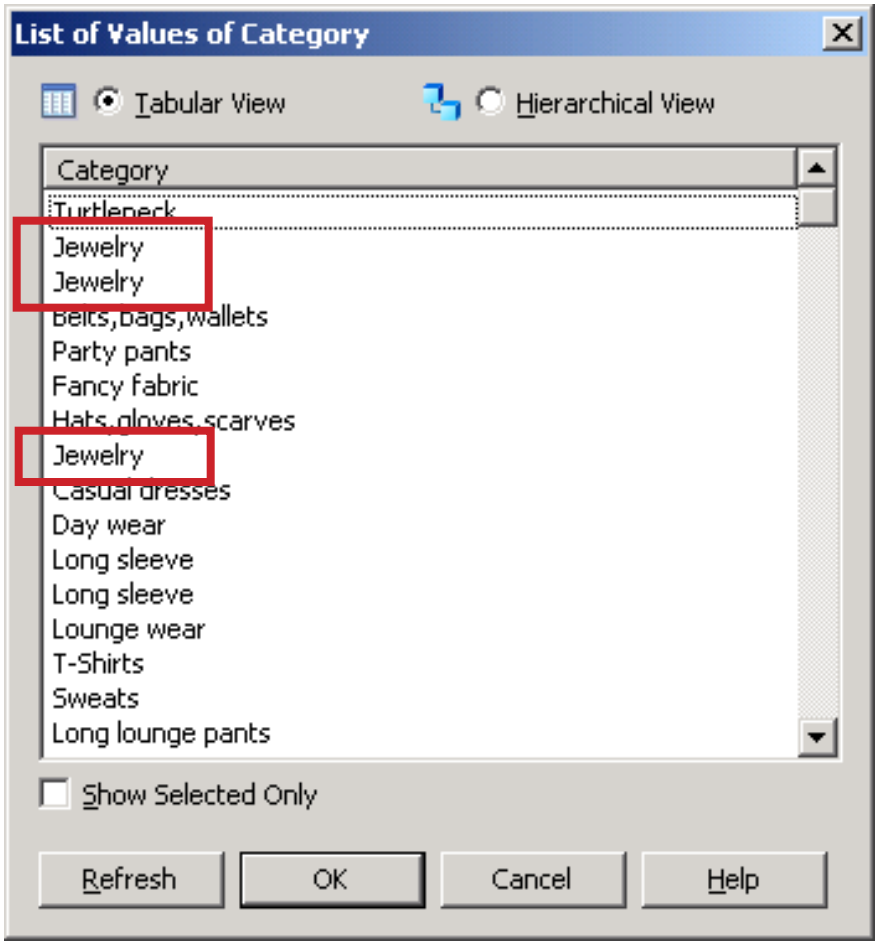
Cannot Add “ALL” to the LOV

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



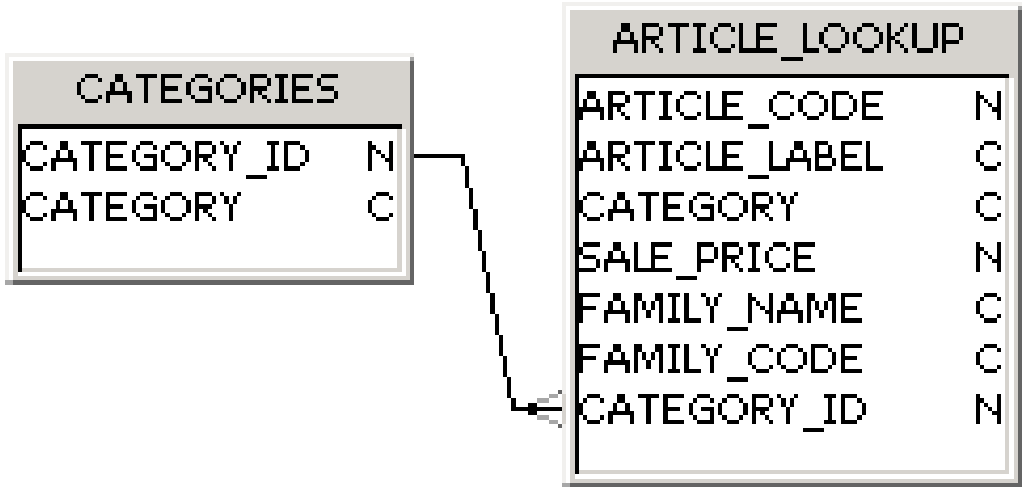
LOV Values Look Strange

- 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 (cont.)

- 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

Specifying Primary Key on Designer Prompts

- The primary_key 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)
```

💻 Demonstration 6 – Prompts with index aware and primary_key

Things That Do Not Work

- Manual LOV selections are required
- 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

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

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
- No aggregate tables
- No “ALL” option
- Limited to key (unique) dimensions

What We'll Cover …

- Universe Concepts
- Index Awareness
- Shortcut Joins
- Aggregate Awareness
- Wrap-up

What Are Shortcut Joins?

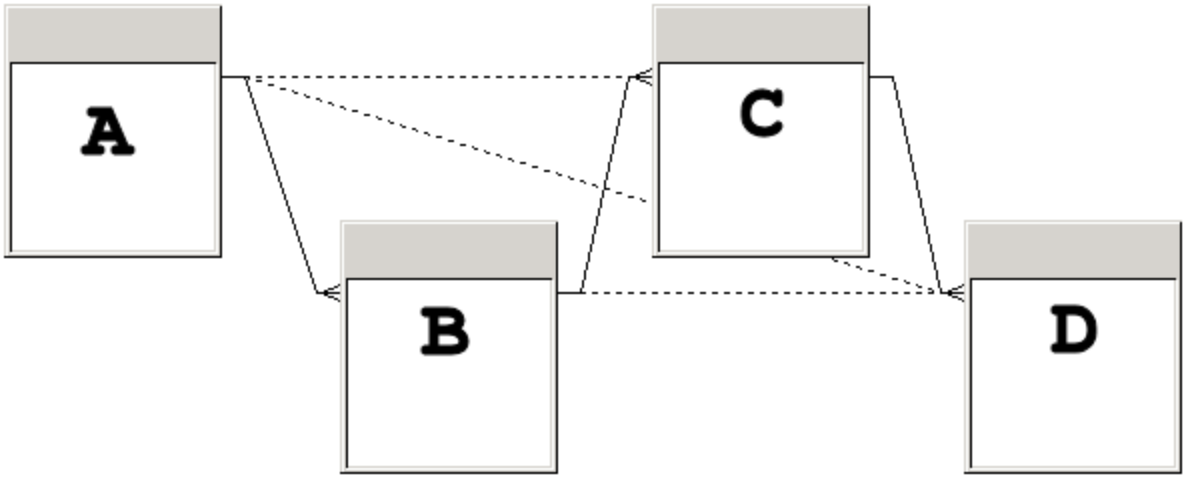
- What does the Universe Designer help say?

  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?

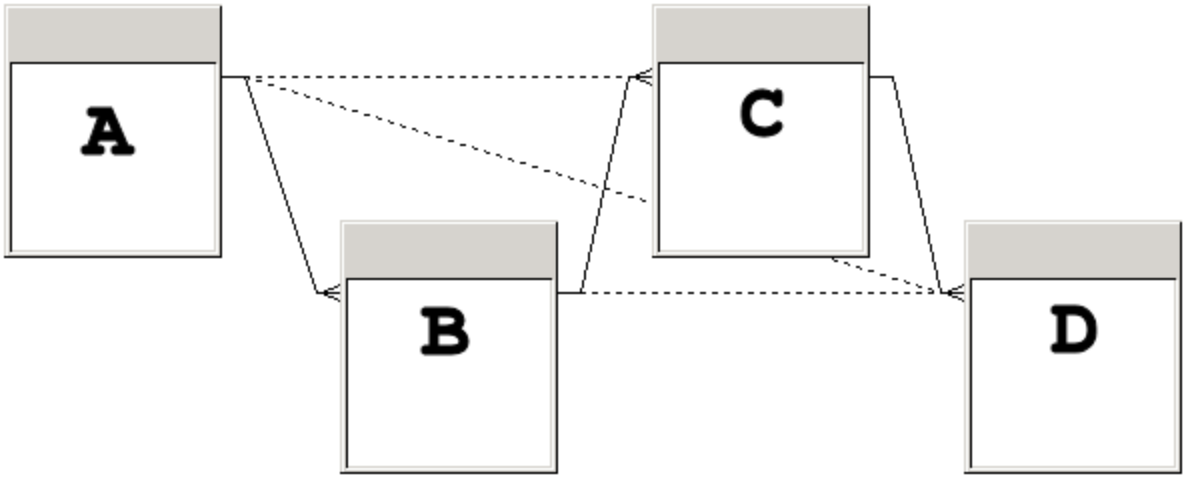
- Multiple shortcut joins can be used at the same time
- … but only in the right situation
- For example, consider these four tables:



- 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

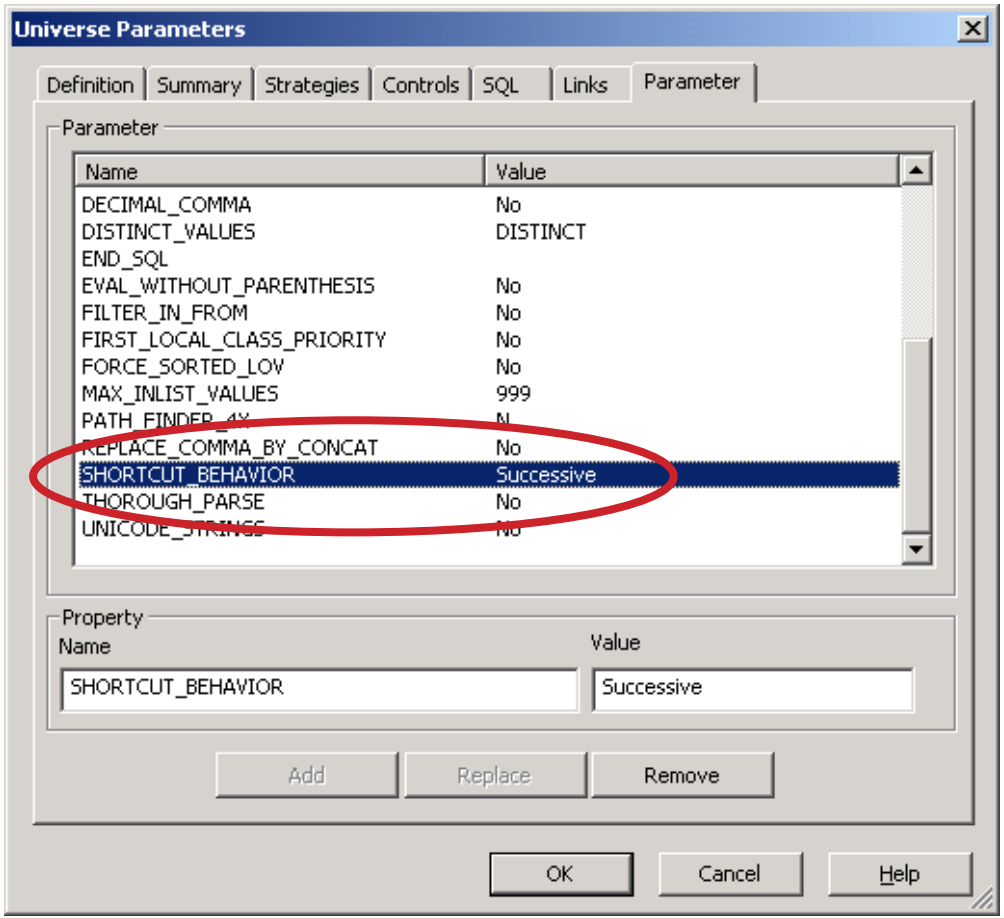
- Query includes objects from table A, C, and D
- Shortcut A – C is considered as it drops table B, which is not needed
- Shortcut A – D is considered but cannot be used as it drops table C, which is needed
- Therefore the only valid query path is A – C – D
- This avoids a “backwards” join, which could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

SHORTCUT_BEHAVIOR

- There is a parameter in universe parameters that configures the shortcut process
  - 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?

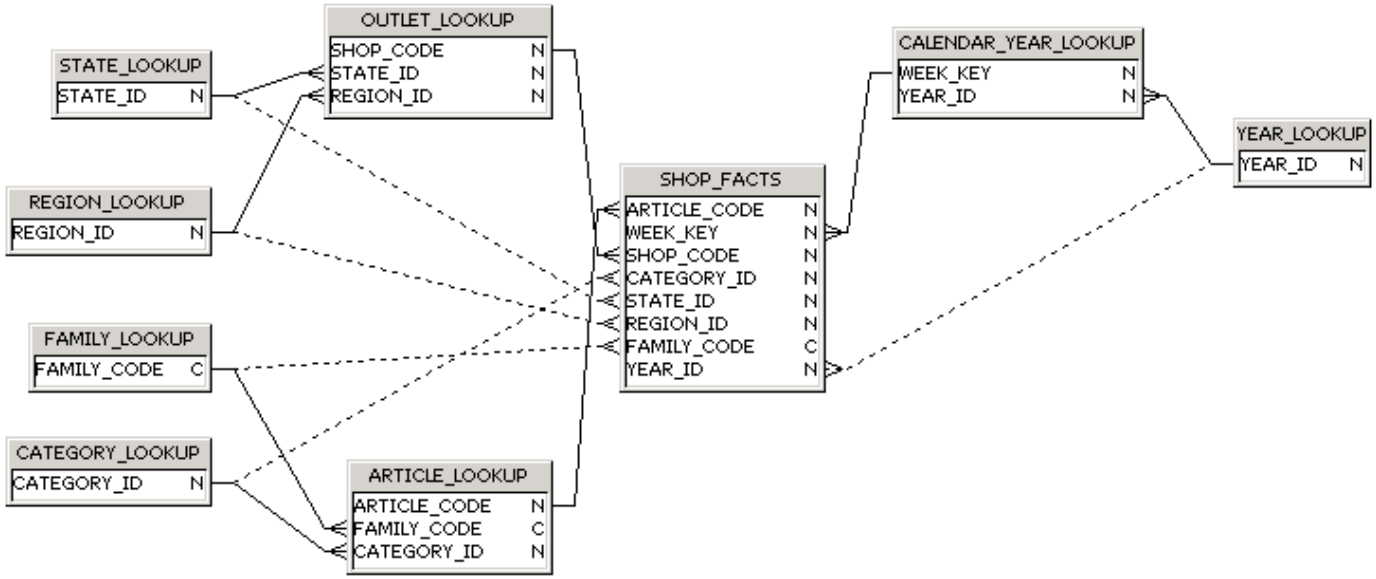
Universe Parameter Setting

- 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 snow-flaked
- 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”

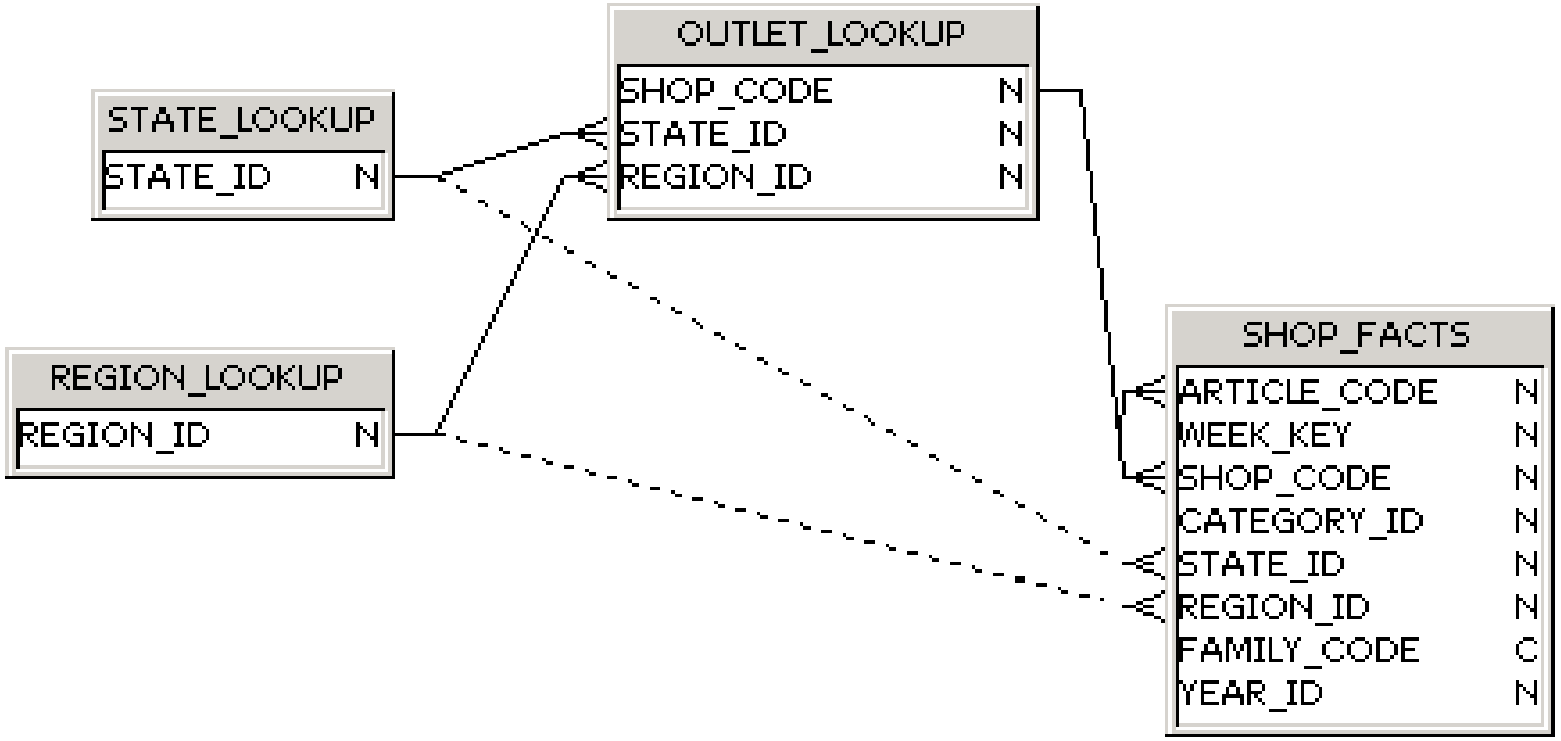
- Removing the OUTLET_LOOKUP table makes a more efficient query in the snowflake universe

```
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” (cont.)

- A global setting changes the shortcut algorithm



SHORTCUT_BEHAVIOR “Global” (cont.)

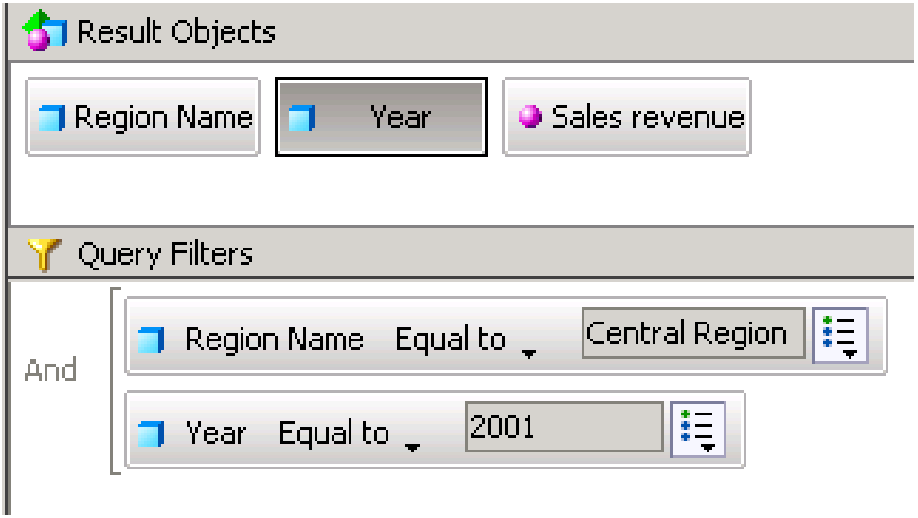
- Two potential shortcuts
  - STATE_LOOKUP to SHOP_FACTS
  - REGION_LOOKUP to SHOP_FACTS
- Either would drop the OUTLET_LOOKUP table
- Successive processes joins one at a time
  - OUTLET_LOOKUP won’t be dropped because the “other” join requires that table
- Global processes all at once and recognizes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

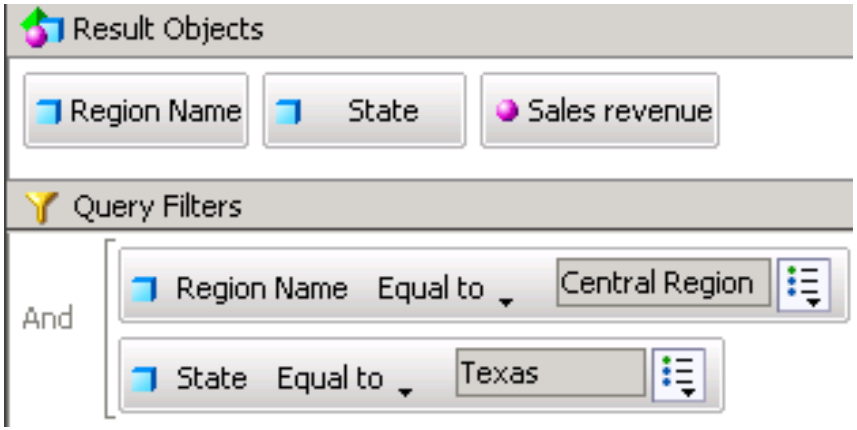
SQL Shortcuts All the Way Around

- 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

What We'll Cover …

- Universe Concepts
- Index Awareness
- Shortcut Joins
- Aggregate Awareness
- Wrap-up

What Is Aggregate Awareness?

- What does the Universe Designer help say?

  Aggregate awareness is the ability of a universe to make use of aggregate tables in a database. These are tables that contain pre-calculated data. You can use the @Aggregate_Aware function in the Select statement for an object that directs a query to be run against aggregate tables rather than a table containing non aggregated data.

  Using aggregate tables speeds up the execution of queries, improving the performance of SQL transactions.

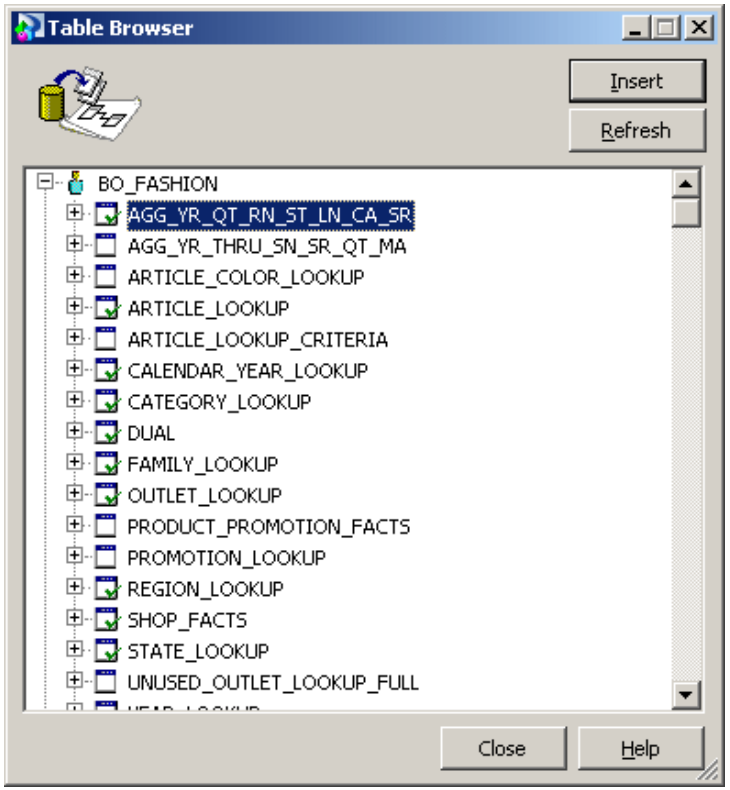
- Aggregate awareness lets a universe designer take advantage of summary tables
  - The process of building and populating these tables is beyond the scope of this session

Aggregate Awareness Implementation

- Aggregate awareness requires the following steps:
  - Creating/populating aggregate structures in the database
  - Referencing those aggregate tables in the universe
  - Setting up rules so the query tools know which aggregate sources are available
- Side note: Many databases offer alternative solutions
  - Teradata offers Aggregate Join Indexes or AJIs
  - Oracle offers Materialized Views
  - DB2 offers MQTs or Materialized Query Tables
- These solutions do not require any effort by the universe designer

Adding Aggregate Structures

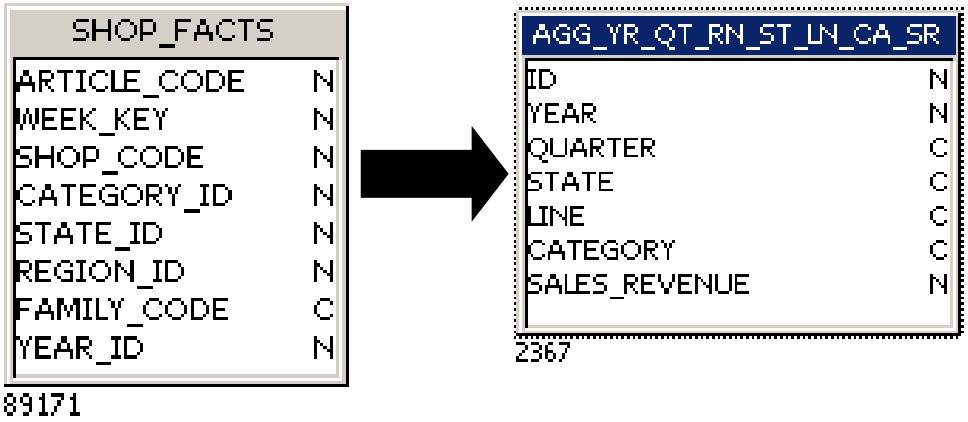
- Adding aggregate tables is done like any other table



- In this case the aggregate table stands alone with no joins

Designing Aggregate Structures

- Look for a reduction of 10:1 or more
  - Goal is to provide noticeable performance impact with minimal ETL effort



- What to remove?
  - Look at standard queries that users run
  - Time makes a great aggregate option
  - Geography is another typical candidate

Creating Objects

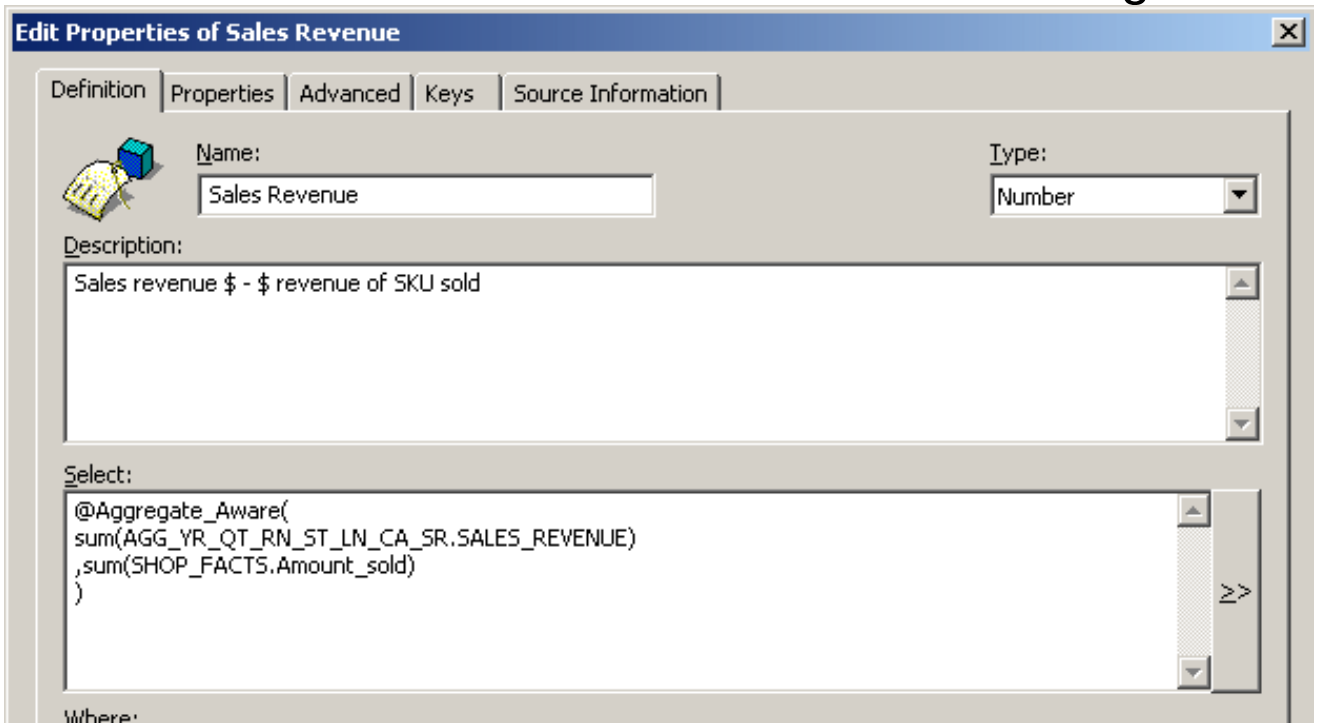
- Aggregate tables can contain dimensions or measures
  - Most typically objects are measures
  - Dimension objects can also be aggregate aware
- Syntax

```
@Aggregate_Aware(
 Sum(top_level_table.measure)
,Sum(second_level_table.measure)
,…
,Sum(lowest_level_table.measure)
)
```

- I have worked with universes with over 20 levels of aggregate tables

Aggregate Aware Syntax

- Aggregate aware objects are about choices
  - Where is the best place to get the information needed?
  - Start with the smallest table and move to the largest



Demonstration 10 – Adding aggregate table, building aggregate objects

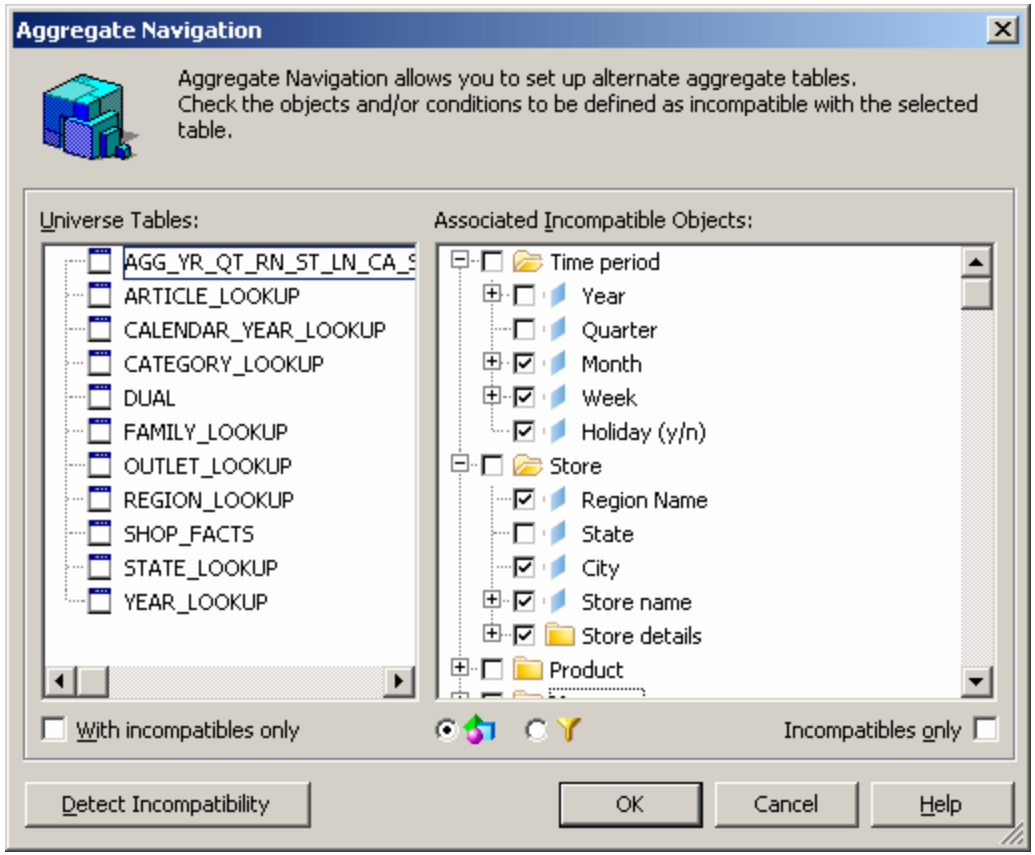
Navigating Aggregate Sources

- Aggregate tables do not have the same level of detail
  - Which objects work with the aggregate table? Which do not?
- Answer this question using Tools + Aggregate Navigation
  - Select the aggregate table
  - Mark which objects are incompatible with that table
  - Objects are incompatible with a table, not with other objects!

Demonstration 11 – Setting up aggregate navigation

Aggregate Table Definition

- Aggregate table includes Year, Quarter, State, Line, and Category
  - Therefore those objects are compatible
  - All other objects cannot be used with the aggregate source



Aggregate Navigation In Action

- Year

```
@Aggregate_Aware(AGG_YR_QT_RN_ST_LN_CA_SR.YEAR
,YEAR_LOOKUP.YEAR)
```

- State

```
@Aggregate_Aware(AGG_YR_QT_RN_ST_LN_CA_SR.STATE
,STATE_LOOKUP.STATE)
```

- Revenue

```
@Aggregate_Aware(
sum(AGG_YR_QT_RN_ST_LN_CA_SR.SALES_REVENUE)
,sum(SHOP_FACTS.Amount_sold) )
```

Demonstration 12 – Running Aggregate Aware queries

What Is Aggregate Awareness?

- What does the Designer help say?

  Aggregate awareness is the ability of a universe to make use of aggregate tables in a database. These are tables that contain pre-calculated data. You can use the @Aggregate_Aware function in the Select statement for an object that directs a query to be run against aggregate tables rather than a table containing non aggregated data.

  Using aggregate tables speeds up the execution of queries, improving the performance of SQL transactions.

- Process
  - Set up objects, build incompatibility rules, run queries
  - Aggregate tables are used when available, skipped when not
  - Queries that use summary tables run faster

What We'll Cover …

- Universe Concepts
- Index Awareness
- Shortcut Joins
- Aggregate Awareness
- Wrap-up

Additional Resources

- Many of the techniques addressed today have been covered in blog posts on my business intelligence blog
  - www.dagira.com
- What is Index Awareness?
  - www.dagira.com/2007/10/26/index-awareness-part-i-the-basics
- Everything about Shortcut Joins
  - www.dagira.com/2010/05/27/everything-about-shortcut-joins
- There are several online communities that also offer help
  - BOB at http://busobj.forumtopics.com
  - SAP at http://forums.sdn.sap.com/index.jspa

7 Key Points to Take Home

- Universe performance is a secondary concern
- Make sure structure delivers correct results first!
- Index awareness can optimize queries by using keys
- Shortcut joins can bypass unnecessary tables
- Shortcut joins are NOT used to resolve loops
- Aggregate awareness allows a universe to use summary tables
- Aggregate awareness is not compatible with index awareness

Disclaimer

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Wellesley Information Services is neither owned nor controlled by SAP.

Any opinions, comments, statements, or technical details provided in this presentation are my own personal opinions and may or may not reflect the opinions or policies of my employer or clients.