Recognizing and resolving Chasm and Fan traps when designing universes

---

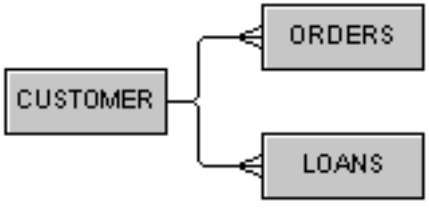
Relational databases can return incorrect results due to limitations in the way that joins are performed in relational databases. Unlike loops, which return fewer rows, the Chasm and the Fan traps are two common circumstances which return too many rows. You can use Designer to resolve both types of problems in your universe schema.

What is a Chasm trap?

The Chasm trap occurs when two “many to one” joins converge on a single table. You will get incorrect results when the following circumstances exist:

- A “many to one to many relationship” exists among three tables in the universe structure
- The query includes objects based on two tables both at the “many” end of their respective joins
- There are multiple rows returned for a single dimension

In the example below a customer can place many orders/and or place many loans:



If you want to run a query that returns the total order and loan values for a customer Paul, you would get the following results:

CUSTOMER.NAME	ORDERS.DATE	ORDERS.TOTAL_VALUE	LOANS.DATE	LOANS.TOTAL_VALUE
Paul	12/01/99	100.00	05/08/97	50.00
Paul	14/04/99	150.00	05/08/97	50.00
Paul	20/09/99	150.00	05/08/97	50.00
Paul	12/01/99	100.00	03/06/97	100.00
Paul	14/04/99	150.00	03/06/97	100.00
Paul	20/09/99	150.00	03/06/97	100.00
		Sum = 800		Sum = 450

This is obviously an incorrect result. A Cartesian product of the CUSTOMER, ORDERS, and LOAN tables has been returned. The correct results should be:

Total orders value for Paul is 400

Total loans value for Paul is 150

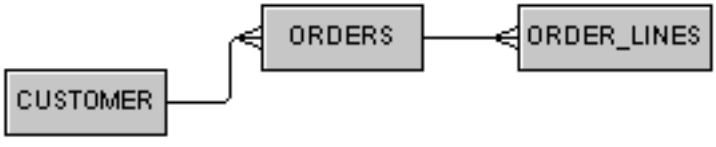
How do you resolve a Chasm trap?

Depending on whether the target users for your universe are BusinessObjects or WebIntelligence users, you can use Designer in the following ways to resolve the Chasm trap problem:

Solution	Description
Define a context for each table at the “many” end of the joins.	In our example you could define a context from CUSTOMER to ORDERS and from CUSTOMER to LOANS. This creates two SQL statements and two separate tables in Business Objects, avoiding the creation of a Cartesian product. Using contexts is the most effective way to solve Chasm traps.
Select the option Multiple SQL Statements for Each Measure from the Universe Parameters dialog box.	Only applies to measures. You force the SQL generation engine in Reporter to generate SQL queries for each measure that appears in the Query panel. You cannot use this solution to generate multiple SQL statements for dimensions.

What is a Fan trap?

The Fan trap occurs when a “one to many” join links a table which is in turn linked by another “one to many” join. The fanning out effect of “one to many” joins can cause incorrect results to be returned when a query includes objects based on both tables. A simple example of a Fan trap is shown below:



When you run a query that asks for the total orders by each order line, for a particular customer, an incorrect result is returned as you are performing an aggregate function on the

table at the “one” end of the join, while still joining to the “many” end. For example if you run the following query against the three tables:

```
SELECT
   CUSTOMER.NAME,
   SUM(ORDERS.TOTAL_VALUE),
   SUM(ORDER_LINES.QTY_SOLD)
FROM
   CUSTOMER,
   ORDERS,
   ORDER_LINES
WHERE
   (CUSTOMER.CUST_ID=ORDERS.CUST_ID) AND
   (ORDERS.ORDER_ID=LINES.ORDER_ID) AND
   (CUSTOMER.NAME='PAUL')
GROUP BY
   CUSTOMER.NAME
```

The result is a Cartesian product which produces an inflated figure for the total number of orders placed by the customer Paul, as shown in the following table:

CUSTOMER.NAME	ORDERS.TOTAL_VALUE	ORDER_LINES.PROD_ID	ORDER_LINES.QTY_SOLD
Paul	550000	JAGUAR	3
Paul	550000	PEUGEOT	5
Paul	550000	VW	1
	Sum = 1650000		Sum = 9

How do you resolve a Fan trap?

There are two ways to solve a Fan trap problem.

- Using an alias and the aggregate awareness function. This is the most effective way to solve the Fan trap problem.
- Altering the SQL parameters for the universe. This only works for measure objects.

Both of these methods are described below.

Using an alias and aggregate awareness

You create an alias table and use the aggregate awareness function. You cannot use this option if you have incompatible objects. You can do this as follows:

1. Create an alias for the table that is producing the multiplied aggregation
2. Create a one to one join between the original table and the alias table
3. Modify the select statement for the columns that are summed so that the columns in the alias table are summed and not the original table
4. Apply the @AggregateAware function to the select statement. for example:
   `@AggregateAware(SUM(ORDERS.TOTAL_VALUE) , SUM(ORDERS_2.TOTAL_VALUE))`

Altering the SQl parameters for the universe

You select the option Multiple SQL Statements for Each Measure. You force the SQL generation engine in Reporter to generate separate SQL queries for each measure that appears in the Query panel. You find this option on the SQL page of the Universe Parameters dialog box. You cannot use this method to generate multiple queries for dimensions.