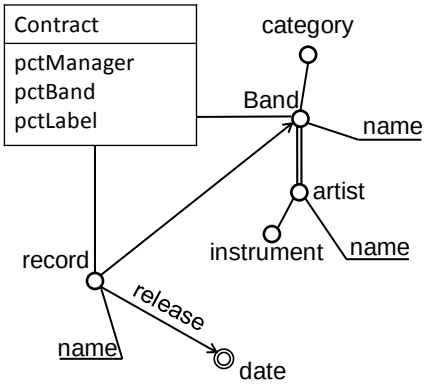
Exercises Data Warehousing
Bridge Tables and Changing Dimensions
Solutions

1. SASS (and most similar tools of other vendors as well) natively supports bridge tables. In SSAS bridge tables are considered to be measureless fact tables. Hence, when creating the cube you need to mark the bridge table as a fact table ("measurement group"). Then, in the "Dimension Usage" tab of the cube explorer you can indicate how the dimension that is connected to the fact table through the bridge, has to be used. The relationship type is "Many-to-Many" and the intermediate measure group (=fact table) will be your bridge table.

   Connect to the database "M2MDB" in BIDS. You will need all tables except Cust_Acc_Bridge. Connect the Customer dimension ("dimCustomer") to your data cube (measurement group "factTransactions").

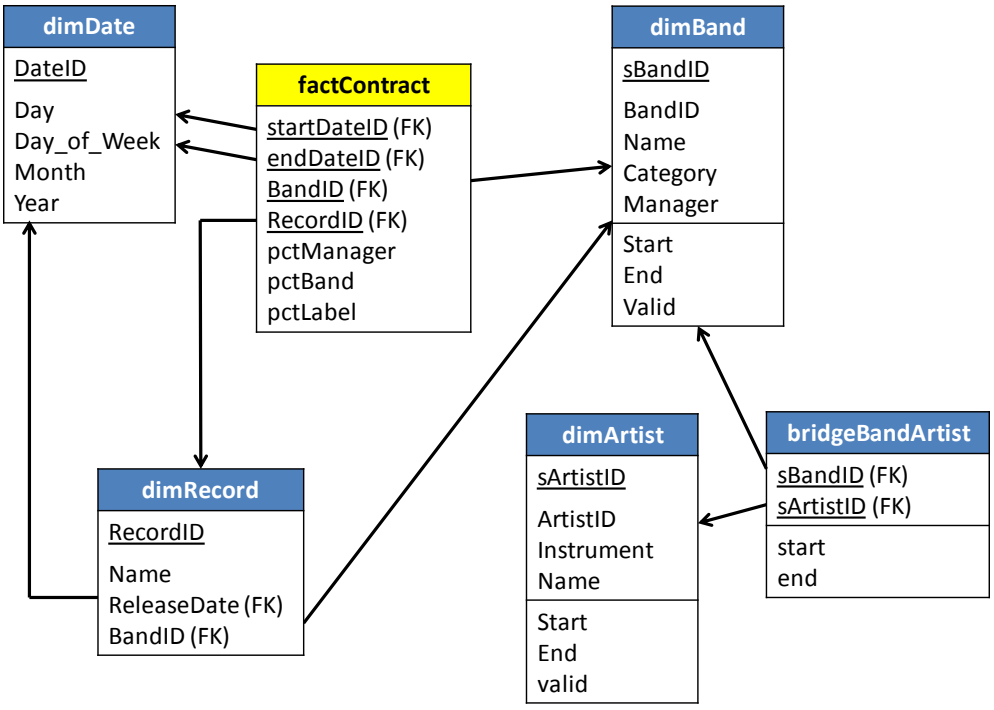
2. Consider the following DFM schema.



   Design a relational schema that implements this model. Ensure that your schema can accommodate the following changes:

   (a) Band name
   (b) Band members
   (c) Instrument

   Solution:



3. Consider the database we worked with last week, Simple_NW_DW. Suppose now that the following dimensional changes need to be accommodated for:

   (a) What changes do we need to make to the schema in order to be able to accommodate the following changes?

   - The company name and city of a customer may change.
   - The name of a product and of a category may be corrected.
   - The title of an employee may change.

   Solution: You will find the solution in Simple_NW_DW_SCD on the MSSS.

   (b) What updates to the database are needed (DELETE, INSERT, UPDATE statements) in order to encode the following changes:

   - Product name "Flotemysost" is corrected to "Flotemisost." (product ID 71)
     Solution: This is a type-1 change. Since Product is not versioned, you only have to update the existing tuple for product 71.
   - The company with ID "GREAL" changes its name back to "Great Lakes Food Market".
     Solution: This is a type-1 change. Create a new version for "GREAL" by copying all attributes from the most recent version of the tuple and changing the name. The most recent version of GREAL can be recognized by the fact that the end date equals NULL. The new version has start date the current date. Furthermore, in the newly created tuple, set the start to the current moment, and end to NULL. This is now the valid version of GREAL. For the previously most recent version of the tuple, set end to the current moment.
   - The first name of employee with ID 1 (Nancy Davolio) is misspelled. Her first name should be "Nancera" instead of "Nancy."
     Solution: This is a type-1 change. Since Employee is versioned, however, it may be the case that multiple versions have to be corrected as they all contain the same spelling mistake. Hence, for all tuples with ID 1 (OLTP key equal to 1, not the surrogate key!), make the change.
   - After Andrew Fuller resigns, Steven Buchanan becomes the new CEO.
     Solution: This is a type-2 change, since Employee is versioned. In both the most recent version of the employee record of Andrew Fuller as that of Steve Buchanan the end is set to the current moment. For Steven Buchanan a new version is made in which the position is changed and the start is the current moment and the end NULL.

4. What changes do we need to make to our data warehouse if we retrospectively learn that for one of the dimensions there was an update on one or more values of one of the tuples? Consider for instance the database of question 2. We learn now that customer "HUNGO" changed location on September 20th, 2009 (day with ID 264) and moved from Cork (GID 19) to Charleroi in Belgium (GID 18) on that day. What changes should we make to the data warehouse in order to retrospectively bring the content of the data warehouse up to date?

   Solution: TBD

5. Consider and model the following situation: we are storing all transactions of a bank. Every transaction is for an account, done by a customer, at a certain date. Furthermore we keep the amount. An account is associated to multiple customers. A bank account has attributes accountNr (=OLTP key) and a policy ("can go below zero but bounded", "can go below zero but unbounded", "cannot go below zero"). A customer is identified by his or her passport number, has a name, and is or is not a VIP customer.

   - Model this in the DFM, and translate to the relational model. Do not yet consider SCDs at this point.
   - Now consider the following changes that may take place:
     - The policy of an account may change;
     - A customer's status may change from non-VIP to VIP or vice-versa;
     - The owners of an account may change.

     How would you change your model such that these changes can be accommodated?