Implementing Easik 1.0

Robert Rosebrugh[*], Vera Ranieri, Kevin Green and Rob Fletcher
Department of Mathematics and Computer Science
Mount Allison University

[*]Research partially supported by grants from NSERC Canada, NSERC USRA and Mount Allison University.

Contents

1 Design concepts

For some years, Johnson, Rosebrugh and others (see, for example [3], [4], [5]) have been developing a data model called the *Sketch Data Model* (SkDM). This new model is related to the entity relational attribute (ERA) model whose suggestive and simple graphical expression via entity-relational diagrams (ERDs) is an important reason for its success as a database design paradigm. The Sketch Data Model enhances the constraint expressiveness of the ERA model by using the category-theoretic concept of *sketch*[1]. Like the ERA model, the SkDM is well adapted to presentation and manipulation with modern graphical user interfaces (GUIs).

Easik implements the SkDM enhancement of the ERA design paradigm. The SkDM includes the graphical simplicity of the ERA model and adds precise and powerful constraint expressiveness. Along with this, the SkDM allows a cleaner visual design than ERDs since it needs fewer graphical types.

We note that a variety of software is available to provide a graphical environment for designing ERDs, and for translating these into a database description in a language such as SQL. See `http://en.wikipedia.org/wiki/Entity-relationship_model` for some examples.

Easik is a Java application that provides a graphical design environment for the Entity-Attribute(EA) Sketches of SkDM. The EA sketch that a user designs can be saved as an XML document. The XML document can be exported to a database description in SQL which includes triggers and procedures to enforce the constraints expressed graphically, and further provides connectivity to database management systems via JDBC.

The application opens with a graphical canvas and tools available for the creation of entities, attributes and edges joining entities. The graphics package used in Easik is JGraph. EA sketch constraints, which include and extend the usual constraints of ERA diagrams, may also be specified using the graphical interface.

The XML document holding EA sketch information (entities, attributes, edges and constraints) follows an XML schema written in XSD so as to standardize data storage. The XML document also stores information about the sketch, such as the authors, modification dates, and data types defined for the sketch. Easik can define an SQL database directly from this file.

Generation of data description for an SQL database from a stored EA sketch begins with a table per entity and any key information implied by the graph of the EA sketch. Attributes are directly encoded as columns of entity tables. In addition, the constraints of the EA sketch are encoded as *triggers* and *stored procedures.* In Section 5 we explain the design of this implementation and its limitations.

1.1 Users of the Product

One of the goals of Easik is to have a product which can be used in both academia and industry. The product can be used by those in academia to facilitate database design understanding, while those in industry will appreciate the direct translation from a graphical database to a usable database. Organizations using common database platforms such as MySQL or Oracle can present a graphical data model and then implement it in a database while maintaining specified constraints.

1.2 EA Sketches as Data Models

Like ERA diagrams, EA sketches have *entities* represented as nodes of a graph. Entity nodes may have *attributes*. Attributes are also represented as nodes of a graph in ERA diagrams. For economy of screen space they are included within their entity in Easik. An EA sketch does not have the relationship nodes that appear in an ERA diagram. These are replaced by a *tabulation* of the relationship making the relationship an entity in its own right and specifying edges directed to the participating entities. This transformation has also been noted by Date [2]. For example, where `Registration` might be a relationship from `Student` to `Course` in an ERA diagram, the same information is represented by three entity nodes and two directed edges in an EA sketch as

$$\texttt{Student} \longleftarrow \texttt{Registration} \longrightarrow \texttt{Course}$$

Easik is an application that provides a graphical method of designing Entity-Attribute(EA) sketches, followed by the definition of a database. The graphical representation should accurately represent any EA database, i.e. accurately represent entities, attributes, and constraints. The interpretation of this graphical database into a functioning database should then be available to the user, for use with a variety of SQL and eventually other database platforms. The database created should contain all information contained in the EA sketch. This includes any information such as constraints, foreign keys, and attribute types.

2 Project requirements

This section describes characteristics of the implementation that were in the design, with reference to the software used.

2.1 Solution environment

The graphical front end of Easik is written in Java (version 1.5). The program is compatible with a variety of operating systems. The storage of all saved files is accomplished through XML, and the translation between XML and Java is accomplished through SAX.

Both Java GUI implementation and database manipulation can be costly of computing resources. Because of this, database interaction with the Easik front end product should be minimized.

Easik should only be responsible for the data definition phase of the database design, with the user then being able to access the newly designed database through their platform of choice.

2.2 External software

Easik currently uses JGraph to provide the graphical rendering. Any database implementation requires an ambient database management system (DBMS), such as MySQL, Oracle or DB2, to create a database.

Translation of the graphic to a typesettable version uses GasTeX, a $\TeX$ macro package for graph design. Both the GasTeX package and a $\TeX$ installation are required to implement this feature.

2.3 Functional and data

- The product and sum constraints, commutative diagrams, and pullbacks should be represented graphically and should be able to be translated into a database.
- Entities must have exactly one designated *primary key.*
- Attributes should be based on pre-existing datatypes for a particular database platform specified by the user.
- Unique Keys must be definable within the sketch and be translatable to a DBMS.
- Database translation should be accomplished automatically.
- Database translation should include all relevant information about constraints, primary keys, and foreign keys.

2.4 Look and feel

- All user input should be gathered through the GUI environment.
- User should be able to select a path of edges of any length to create constraints
- User should be able to add, delete, edit and rename all sketch elements.
- Attributes should be displayed in an easy to understand manner without adding clutter to the overall design.

2.5 Usability

- User should have a choice of many popular database platforms with which to interact, including Oracle, DB2, and MySQL.
- User should be able to create a text file containing SQL for database generation.
- Drivers for any interactions with database platforms should be included in the software package.

- User should also be able to create a database based on XML.
- User should be able to translate an Easik sketch into a TEX file for image creation.

2.6 Security requirements

Databases often contain data with access restrictions, so users should not be able to access a database without supplying a user name and password.

Easik itself is a design tool and restricted access is not intended.

Accessing a database remotely often does not allow the user privileges to create a new database. As such, should the database storage location be remote, the user must be made aware that the database must be created first using a database interaction platform, such as phpMyAdmin or Cpane, at which point the full database can be accessed.

3 Progress to date

The GUI of the application accurately displays entities and their attributes, along with a variety of SkDM constraints.

Graphically displaying the sketch (database schema) is accomplished using an open source package known as JGraph. This package interprets information sent by the program and displays it for the user as a directed graph which the user can manipulate.

Sketches to be retrieved at a later date are saved in a user-named XML file. The file format is predefined using an XML schema, and is parsed using SAX during loading. Stored in the XML file is all information about the entities, attributes, edges, constraints, and their positions in the screen representation.

Users can also export sketches to SQL text or to a functioning SQL database system. Testing has been completed on databases created within the MySQL DBMS. Functionality of databases is constrained by conditions outlined in section 5.

4 Export to an SQL based database or schema

Easik allows the user to export an EA sketch to an SQL database schema generically, or to create a database in one of several database management systems (DBMS). This exportation occurs using Java Database Connectivity (JDBC) and drivers for each platform. These drivers were downloaded from the respective DBMS's web sites, and included in the Easik package.

4.1 Database Platforms

Easik provides functionality for the following database platforms: MySQL 5.0, Oracle 10*g* Release 2, and DB2.

4.1.1 MySQL

MySQL 5.0 is the required version for Easik database exportation as it is the earliest version that integrates foreign keys, triggers, and stored procedures. Earlier versions of MySQL may compile when the database is created, however the system will not enforce foreign key restrictions and any defined constraints.

To allow full functioning of foreign keys in MySQL, the `INNODB` engine is required. This engine is used for all tables, regardless of whether they contain foreign keys, for simplicity. For more information on this engine, and the MySQL 5.0 database, see [6].

4.1.2 Oracle

Oracle provides many versions of its database software, to allow for different user requirements. Currently, the only Oracle DBMS version supported by Easik is Oracle10*g* Release 2. For this version, the database driver used is the Oracle Call Interface (OCI) which allows for access only to client-side installations. This driver cannot interact with applets.

For more information on this engine, and the Oracle 10*g* DBMS, see [7].

4.1.3 DB2

While Easik has code to generate a DB2 database, this functionality has not been tested.

4.2 Export to SQL Text

From within Easik, it is also possible to export SQL Text, formatted for the desired database platform. This code contains the correct declarations for the different DBMS, and differs only slightly from the code passed to the database through the automatic database generation capability of Easik.

When exporting to SQL Text, the user is prompted to provide the name of the database. This is used to create the database and also to ensure that the desired database is used for further commands.

The SQL Text is written sequentially, so that if the SQL Text is parsed in a different order than the one presented, it may not be possible for the database to be created.

5 SQL limitations of Easik

Different database platforms implement various levels of SQL functionality. The freeware MySQL version 5.0 was taken as the SQL standard. This platform suffers the limitation that triggers cannot access any table other than the table for which the trigger was called. For this reason, triggers are used primarily to call stored procedures, which do not have this restriction.

Subject to the limitations described below, constraints are translated into relational database constraints using triggers and stored procedures.

5.1 Primary Keys

Each entity translates to a table of the database schema. Tables are created with the same name as the entity, and are defined with all attributes and unique keys defined within Easik.

When defining an SQL table, a column of the table may be defined as the primary key. This implies that all entries under this column must be unique. Because Easik relies on triggers and stored procedures to enforce constraints, the user is not permitted to define a primary key. Instead, Easik automatically defines one for the user, and uses this column to ensure accurate enforcement of foreign keys and user defined constraints.

The primary key for a table is defined as the column with the header with the name of the table followed by a predefined Easik identification marker. The column is set to be of type `INTEGER`, with the properties `AUTO INCREMENT`, and `NOT NULL` as defined by SQL.

For example, if a table has the name `STUDENTS`, then the primary key column definition has the following syntax:

```
students_id INTEGER NOT NULL AUTO INCREMENT PRIMARY KEY
```

This primary key is then used as reference to create any foreign keys required by edges between entities in the sketch.

5.2 Foreign keys and Easik edges

Foreign keys are defined in a relational database as one or more columns of a table T whose values are required to match the primary key of a second table S. Within Easik, an edge from entity T to entity S is represented by a foreign key. These are accurately translated into SQL relational tables by the following syntax in the definition of table T:

```
FOREIGN KEY (s_id) REFERENCES S (s_id) ON UPDATE CASCADE ON DELETE CASCADE
```

Where *s_id* is a column in both S and T.

Foreign keys are cascaded on delete to ensure that all elements which reference the key are also deleted if the key is deleted. This property maintains the integrity of certain constraints such as product constraints, and commutative diagrams. Because foreign keys are restricted to values held by primary key entries in the referenced table, foreign keys are not permitted to be `NULL`.

5.3 Unique keys

Unique keys can be defined using Easik and must have at least one attribute. Unique keys determine a set of columns in the database whose values on the set are unique.

Within SQL databases, unique keys have the default condition that their entries are allowed to be `NULL`, and two elements of the unique key set whose entries are all `NULL` are still defined to be unique. Easik relies on this property to accurately insert elements into a table where mandated by a constraint. For this reason, it is forbidden to make a unique key `NOT NULL`.

An exception to this rule is created when the user defines an injective edge in Easik. Injective edges require the condition that the foreign key creates a unique key group with the foreign key as the sole member. As foreign keys are not permitted to be `NULL`, a unique key which is defined by a foreign key inherently contains the `NOT NULL` definition.

5.4 Constraints

Easik allows the user to define commutative diagrams and three types of constraints. The constraints for which Easik currently provides functionality are:

- Sum Constraints
- Product Constraints
- Pullback Constraints

Each of these constraints, along with commutative diagrams, can be accurately translated into SQL database schemas that enforce the constraints. Enforcement is achieved through various methods, with the caveat that databases created with Easik must, after creation, be accessed only by users with limited privileges to maintain constraint integrity.

The Fail Table To help maintain a record of actions by the user which lead to prohibited conditions as defined by constraints, a `Fail` table is created to log these failures.

A table with the name `Fail` has been included in all Easik generated databases as a place to record errors when attempting inserts into certain constraints. From this table, the user can determine the exact reason why an insert or update failed. References to this table may be removed at a later date if the user feels that it interferes with efficient database manipulation.

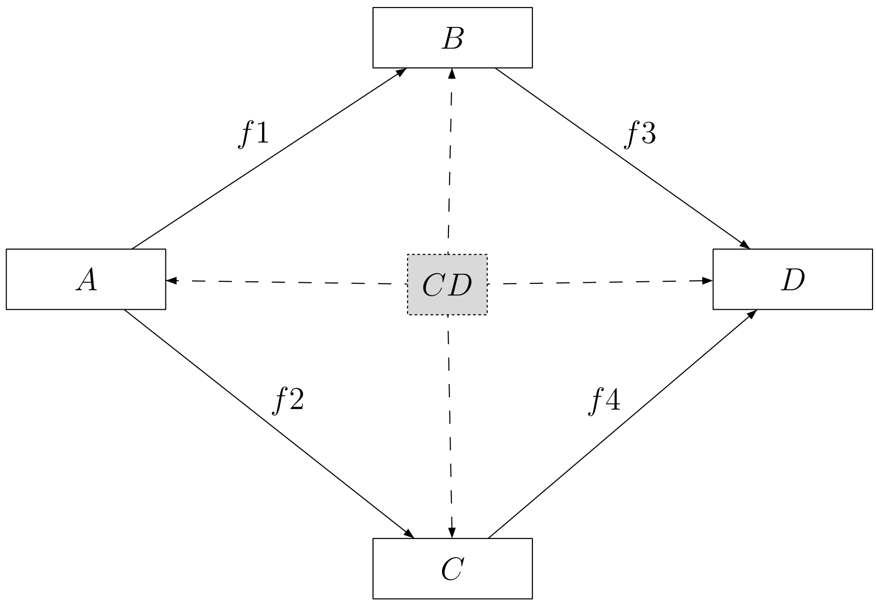
5.4.1 Commutative Diagrams

A commutative diagram for an EA sketch is:

> an ordered pair of paths in the directed graph of the sketch that have the same source and target.

When an EA sketch defined by Easik is realized as an SQL database, a commutative diagram constraint is satisfied if, for every row in the table for the source entity, following either path (of edges) in the commutative diagram reaches the same row in the table for the target entity. Recall that the edges are realized in the database schema by foreign keys.

The following illustrates a commutative diagram between the source entity A and the target entity D, travelling through tables B and C.



When an Easik commutative diagram is translated into a constraint on an SQL database, the following restrictions are required to ensure the commutative diagram conditions are satisfied:

1. Any insert in table A must lead to the same insert in table D when traced on either path.
2. Updating or deleting entries from a table in the commutative diagram will update or delete any referenced entries in all other tables participating in the commutative diagram if needed.

The first condition is enforced by a trigger activated after insert of any entry into the domain table. The trigger calls a procedure which ensures that foreign key entries determined by the two paths terminate at the same entry in the target table. Should this not be the case, an entry is created into the *Fail* table, and the insertion itself fails. The second condition is enforced through the use of foreign keys and the restrictions placed on them in the foreign key definition of the table.

5.4.2 Sum Constraints

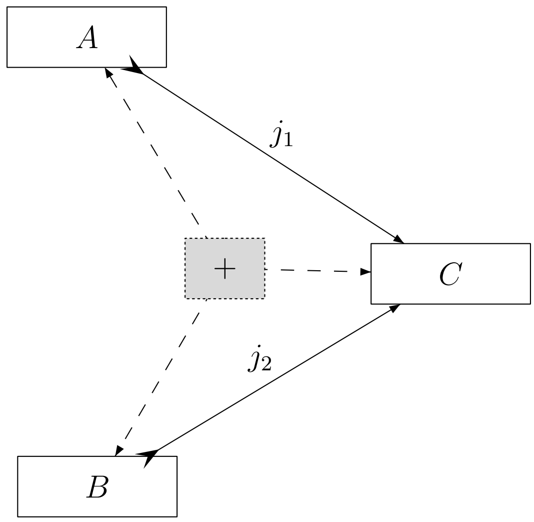
A sum constraint for an EA sketch is:

> entities $A_1, \ldots, A_n$ (the summands), an entity T (the sum), and paths $(j_i : A_i \longrightarrow T)_{1 \leq i \leq n}$ (the injections).

When an EA sketch defined by Easik is realized as an SQL database, a sum constraint is satisfied if the rows in the table T are isomorphic to the disjoint sum of the rows of the tables A_i via the injections, i.e.

$$T = \amalg_{1 \leq i \leq n} A_i = \Sigma_{1 \leq i \leq n} \{(x, i) | x \in A_i\}$$

The following diagram illustrates a simple sum constraint existing between tables A, B, and C, with A and B comprising the summands, C the sum and injections isA_1 and isA_2.



When an Easik sum constraint is translated into an SQL constraint, the following restrictions ensure that the sum constraint conditions are enforced:

1. A path leading from a summand entity to the sum entity must begin with an edge that is injective.
2. An insert into a summand table automatically adds a row in each table along the path, including adding a row to the sum table.
3. The removal of a row in a summand table automatically deletes the entry along each table of the path, including deleting the row from the sum table.
4. The user is not permitted to add entries directly into the sum table.
5. Should the user delete an entry from the sum table, the corresponding entry is deleted from the summand.

The first condition is enforced by creating a foreign key in the summand table so that the foreign key defines a unique key group alone. The second and third conditions are enforced using triggers which call stored procedures to ensure the correct rows are added and deleted from all tables along a path. To ensure the last condition is met, users of the database created using Easik must not be given permission to `INSERT` into the sum table. Because stored procedures can have different permissions than users, this does not cause any conflict with automatic addition in the sum table by a trigger called for each entry into a base table.

5.4.3 Product Constraints

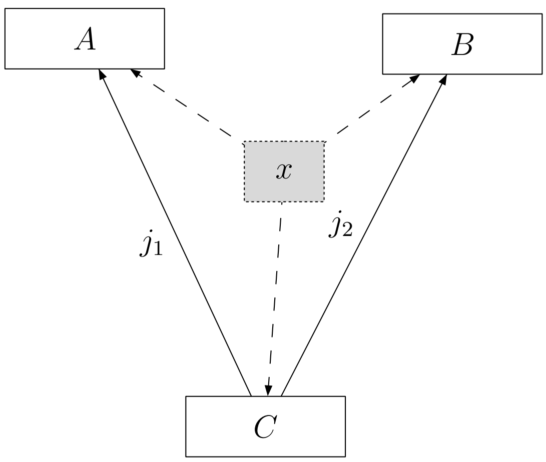
A product constraint for an EA sketch is:

> entities $A_1, \ldots, A_n$ (the factors), an entity T (the product), and paths $(j_i : T \longrightarrow A_i)_{1 \leq i \leq n}$ (the functions).

When an EA sketch defined by Easik is realized as an SQL database, a product constraint is satisfied if the rows in the table T are isomorphic to the cross product of the rows of the tables A_i via the functions, i.e.

$$T = \prod_{1 \leq i \leq n} A_i = \{(x_1, \ldots, x_n) | x_i \in A_i, 1 \leq i \leq n\}.$$

The following diagram illustrates a simple product constraint existing between tables A, B, and C, with A and B the factors, and C the product table.



When an Easik product constraint is translated into an SQL constraint, the following restrictions are put in place to ensure the product constraint conditions are maintained:

1. The product table must contain an entry for every element defined in the cross product of all base tables.

2. The removal of an entry from a base table should remove any related entries from the product table.

3. The removal of an entry from the product table should remove any related entries from the base tables.

The first condition is maintained by a trigger called every time an entry is added to a base table. This trigger then calls a procedure which automatically adds an entry along the path from the product table for every new entry that must be added to the product table. These entries form the group of entries which are the cross product of all entries from any other factor, crossed with the new entry.

The second condition is maintained by the foreign key, and the *cascade* feature of that key.

The third condition is maintained by a trigger placed on the product table. The trigger is activated when the user deletes any element from the product table. The trigger then calls a procedure which will delete any associated elements from the factor tables. Because the product table is defined as a cross product of all entries from the factor tables, deleting one element from the product table will delete all elements from the factor tables and from the product table. Users should avoid deleting entries from the product table, and instead perform deletions by deleting entries from the factors.

5.4.4 Pullback Constraints

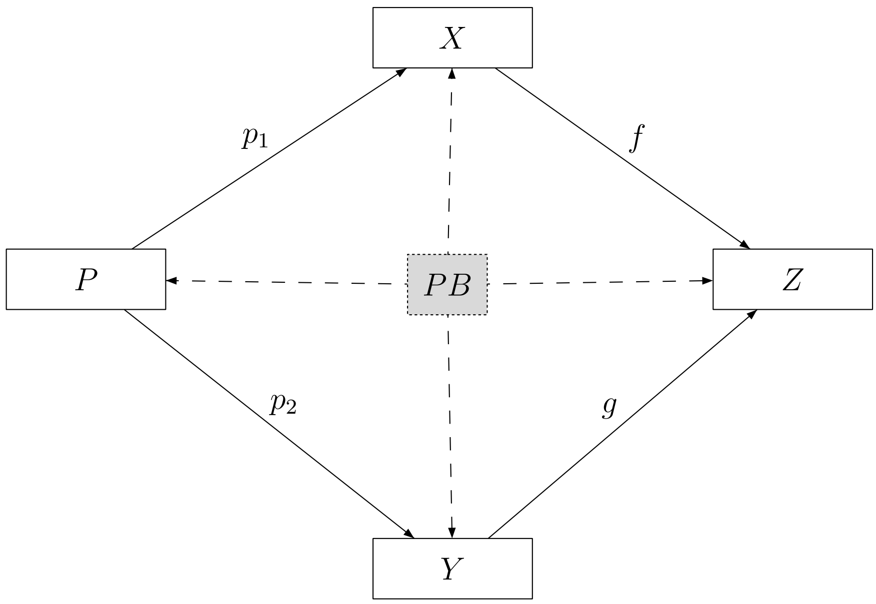
A pullback constraint for an EA sketch is:

> entities X, Y and Z , and entity P (the pullback), paths f, g respectively from X to Z and Y to Z, and paths p_1 from P to X, and p_1 from P to Y(the projections).

When an EA sketch defined by Easik is realized as an SQL database, a pullback constraint is satisfied if the rows in the table P are equal to the join of the tables X, Y over Z as given by the projection maps p_1, p_2 and the diagram commutes, i.e.

$$P = X \times_Z Y = \{(x, y) \in X \times Y | f(x) = g(y)\}$$

The following diagram illustrates a pullback constraint with the definitions as above.



When an Easik pullback constraint is translated into an SQL constraint, the following restrictions are put in place to ensure the pullback conditions are maintained:

1. If when an entry is inserted into X, and for some entries in Y, $f(x) = g(y)$, then entries should be added into P to reflect this. A similar case occurs for an entry into Y.

2. An item should be inserted into the pullback table P only if it satisfies the pullback conditions.

3. Deleting entries from X or Y, deletes corresponding entries in P as necessary.

4. Deleting an entry from P deletes the corresponding entry in Z.

The first condition is maintained by a set of triggers on the factors. When an insert occurs on either of the factor tables, a procedure is called by the trigger to test whether the other factor has a corresponding entry. If a corresponding entry exists, an insert is made into the pullback table.

The second condition is maintained by a check which ensures any entry added to the pullback table satisfies the pullback condition. If the entry does not satisfy the pullback condition, insertion fails. The check allows any automatic insert caused by condition one, and effectively ensures that no other insert can occur.

The third condition is maintained through the cascade property of foreign keys, while the last condition is maintained through a trigger which calls a procedure which removes any necessary entries from Z.

5.5 Multiple constraints

Because the Easik SQL database design requires automatic inserts, the implementation of constraints may conflict with other constraints, making one or more non-functional. Many designs will avoid such interactions, but it is not possible to guarantee no conflicts.

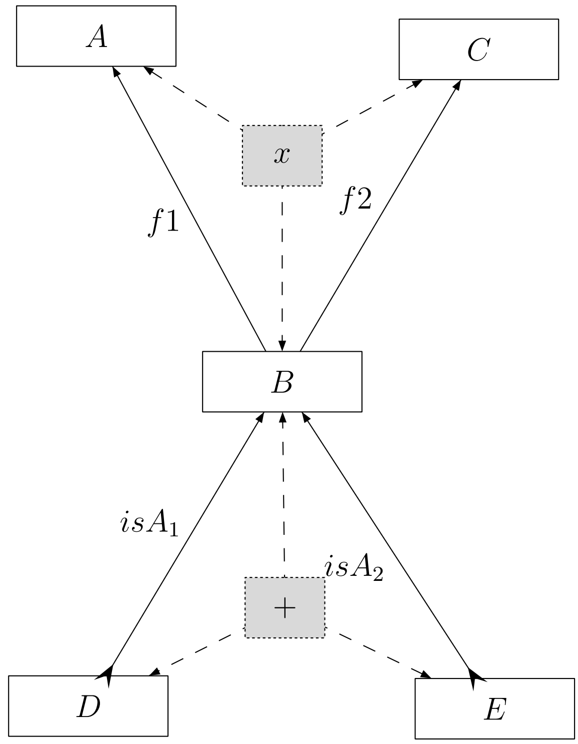
The following is a description of some constraint configurations which may conflicts. These are minimal cases. Any configuration which includes any of the following will subsequently be non-functional. Images are provided to allow for better clarity.

Constraints and foreign keys Sum constraints, product constraints, and pullback constraints all effect automatic inserts to maintain the constraint conditions. The procedures calling these inserts currently cannot register whether a table has foreign keys unrelated to the constraint. Because foreign keys are defined as `NOT NULL`, any automatic insert into a table with an unknown foreign key will fail.

Currently, there is no way from within Easik to allow for automatic inserts of this kind. Users may wish to modify procedures to include references to these foreign keys and insert a temporary default value for the foreign key. This can only be accomplished from within the database platform by an administrator, or by modifying an SQL text file.

Sum constraints When a sum constraint is defined, the procedure called when an entry is inserted into a summand table automatically inserts an entry into the sum table. Users should not be permitted to add entries directly into the sum table.

The following sketch is an example of a sketch that will not function properly when exported to an SQL database:



In this sketch, inserting an entry into a product factor table automatically triggers an insert into the product table. Though an insert into the sum table is permitted through the procedure (though not directly by the user), this insert would explicitly violate the sum condition. Though the database may still function, the constraint conditions would now be violated, compromising database integrity.

6 Future work

Easik demonstrates the feasibility of implementing SkDM. There are, however, two serious limitations in the current version. First, the SkDM allows arbitrary finite limit constraints while Easik implements only products and pullbacks. Second, constraint enforcement is achieved by triggers and procedures. This has the desirable property of being part of the SQL database description. Unfortunately, it limits designs to those where constraints do not interact. Allowing interactive updating of tables (via php, for example) would loosen this restriction.

Some additional desirable enhancements:

- User should be able to have multiple sketches open concurrently without opening multiple instances of Easik.
- Support for views would use the functionality of the previous item.
- Export functions should be able to support multiple edges between two entities.
- Tex export should allow layout specifying diagram size.

All of these items will be addressed in future releases.

References

[1] M. Barr and C. Wells. *Category theory for computing science.* Prentice-Hall, second edition, 1995.

[2] C. J. Date. *Introduction to Database Systems.* Addison-Wesley, eighth edition, 2004.

[3] Michael Johnson and Robert Rosebrugh. Sketch data models, relational schema and data specifications. Proceedings of CATS '02, *Electronic Notes in Theoretical Computer Science* 61(6), 1–13, 2002.

[4] Michael Johnson and Robert Rosebrugh. Three approaches to partiality in the sketch data model. Proceedings of CATS '03, *Electronic Notes in Theoretical Computer Science*, 78, 1–18, 2003.

[5] Michael Johnson, Robert Rosebrugh, and R. J. Wood. Entity-relationship-attribute designs and sketches. *Theory and Applications of Categories* 10, 94–112, 2002.

[6] `www.mysql.org`

[7] `www.oracle.com`