Table of Contents

Evolution Requirements

Primary Specification:

The administrator shall be able to import new ingredients, SKUs, and product lines into the system by means of an import compatible with modern spreadsheet software (CSV, XLSX, or similar). The customer is accepting proposals on the format.

Additional Specifications:

1. The import interface shall include documentation as to the import format.
2. The import action shall only occur if the entire input is free of name conflicts or otherwise problematic issues; if such issues arise, the precise nature of the error should be presented to the administrator in enough detail that it can be corrected.
3. If an import contains identical record(s) to those already in the system, such records should be ignored.
4. If an import contains record(s) that match on name or a unique numeric identifier, the user should be warned about all such records in detail, and if the user approves, the records should be modified to match the imported data.
5. After a successful import, a count and list of records that were added, updated, and ignored should be provided.
6. The system shall be able to export any of the above data in a format compatible with import. The specific records exported should be filterable by the same means defined in the “view options” described for each record in the requirements above (reqs 2.1.2.1, 2.3.2.1). *Note: This allows for an export/modify/import workflow when large-scale changes are needed*

Proposal

Overview

There are three types of data that need to be imported/exported at the same time: SKUs, Ingredients, and Product Lines. Each of these can be naturally and nicely fit into a single table, so it remains to represent the relations (SKU-Ingredient) and (SKU-Product Line). Two additional tables are used to express these relations.

CSV files do a good job representing table data, and are extremely easy to parse. In addition, CSVs are among the most space-efficient formats, so long as the tables they are representing are themselves the minimal. As discussed above, **four** CSV (comma-separated values) files are specified, and their headers are defined below:

Filename Prefix	**Header**
skus	**`SKU#`**`,Name,Case UPC,Unit UPC,Unit size,Count per case,Product Line Name,Comment`
ingredients	**`Ingr#`**`,Name,Vendor Info,Size,Cost,Comment`
product_lines	**`Name`**
formulas	**`SKU#`**`,Ingr#,Quantity`

Definitions

- Evolution requirements: the requirements set forth in the original requirements for the various evolutions. **In case of conflict, the evolution requirements supersede those in this document.**
- CSV file: a file compliant with RFC4180.
- Type of file: each row in the table above is a type of file.
- Record: a row in any of the CSV files.
- Attribute: a column in any of the CSV files.
- Unique identifier: a column designated as unique, either in this document or in Evolution requirements.
- Primary key: an attribute that is **bolded** in the table above.
- Autogeneratable: an attribute is autogeneratable, and may be autogenerated, if it is so permitted in the Evolution requirements.
- Collision: a record that is identical in one or more unique identifier(s) to one existing in the system.
- Duplicate record: a record that is identical in any of its unique identifier(s) to another record in the same CSV file.
- Ambiguous record: a record whose values of unique identifiers refer to more than one record in the system.
- Session: a group of one or more request-response cycles in which a user submits/obtains a logical set of file(s) to the system.
- Batch: all files in a session.
- Batched Mode: an import/export workflow, in which multiple files may be submitted/obtained per session.
- Individual Mode: an import/export workflow in which only one file may be submitted/obtained per session.
- Referential integrity: the property that all unique identifier(s) must refer to exactly one record, that already exists in the system, or, when in Batched Mode, exists in another file of the same batch.

File format

1. The names of the files must be prefixed by an identifying string, as defined above. The filenames must end in ".csv". More precisely, the filenames must constitute matches for the following regular expressions conforming to PCRE standard:
    - SKUs: `skus(\S)*\.csv`
    - Ingredients: `ingredients(\S)*\.csv`
    - Product Lines: `product_lines(\S)*\.csv`
    - Formulas: `formulas(\S)*\.csv`
2. The CSV files must be properly delimited and escaped. RFC4180 specifies the format of a CSV file. Note specifically:
    - Spaces (and other space-like characters) must not be ignored if they are part of a field.
    - Null values, if they are used, must be denoted by an empty string (i.e., two consecutive commas, or a comma followed by a `CRLF` character).
3. A valid header, as defined above, must be included as the first line in each file. The header is case-sensitive.
4. The columns must conform to the data types specified on the Evolution requirements.
5. Unless otherwise specified in the Evolution requirements, all VARCHAR-typed columns must be no more than 1000 characters.
6. Unless otherwise specified in the Evolution requirements, all Integer-typed columns must be within range of $[-2^{31}, 2^{31})$ (i.e., a 32-bit signed integer).
7. Unless otherwise specified in the Evolution requirements, all floating-point numbers must be in decimal, and have a leading 0 (before the decimal place) if the value is less than 1.0.
8. Each file must contain no more than 1,000,000 lines. The **records** may be in no particular order.

Import Workflow

Exactly one of the following import workflows must be used. They are defined as **Batched Mode**, and **Individual Mode**, and are described in the following sections. The implementer must indicate clearly on relevant workflow UIs, which mode is implemented. The implementer may choose a different mode for this workflow than the one for the export workflow.

Batched Mode

Batched mode allows multiple files to be submitted in the same import session, and is the customer's preferred mode of operation.

1. At most one file of each **type** may be submitted in one import **session**. These files are also defined as a **batch**.
2. If the **batch** contains fewer than 4 files, **referential integrity** must still be preserved.
3. In one import **session**, the files may be uploaded separately, or in a ZIP-formatted archive. The system must support both formats. The archive file must have the following structure
   ```
   (root)
     |-- <file1>.csv
     |-- <file2>.csv
     |-- ...
   ```
4. If a **batch** contains more than one file, the files may be processed in any order.
5. If not automatically recognized, all submitted **CSV file(s)** may be assumed to have MIME type `text/csv`, as stipulated in RFC4180 §3. All submitted ZIP-formatted archives may be assumed to have MIME type `application/zip`.
6. The following checks must be performed on each file individually. If any of the check fails, the **batch** must be rejected as a whole, and an appropriate error message must be displayed for the administrator.
   - File validity: the file must be standards-compliant, and contain the required headers in the order specified.
   - **Duplicate records**: the file(s) must not contain **duplicate records** in itself/themselves. The file(s) may contain **collision(s)** to **records** existing in the system. Refer to the Collision Resolution algorithm in the next section when a **collision** is detected.
   - Data validity: for example, the UPC numbers must conform to the UPC-A standard, as specified in the **Evolution requirements**.
   - Empty attributes: all required attribute(s) must be supplied, except when both of the following are true:
     - It is marked as **autogeneratable** in the **Evolution requirements**;
     - The record whose required attribute is empty contains no **collision**.

7. **Referential integrity** must be checked upon successful receipt of all files in a **batch**. Note that the following referential relations exist among the files / corresponding entries in the system:
    - In `skus.csv`,
        - `Product Line Name` must refer to exactly one entry in `product_lines.csv` or be existing in the system.
    - In `formulas.csv`:
        - `SKU#` must refer to exactly one entry in `skus.csv` or be existing in the system.
        - `Ingr#` must refer to exactly one entry in `ingredients.csv` or be existing in the system.

Individual Mode

Individual mode limits the number of files that may be uploaded in an import session to one, and enforces stricter requirements on **referential integrity**.

1. Only one file may be submitted in one import **session**. This file may be of any type.
2. All submitted **CSV file(s)** may be assumed to have MIME type `text/csv`, as stipulated in RFC4180 §3.
3. The following checks must be performed on each file individually. If any of the check fails, the file must be rejected as a whole, and an appropriate error message must be displayed for the administrator.
    - File validity: the file must be standards-compliant, and contain the required headers in the order specified.
    - **Duplicate records**: the file must not contain **duplicate records** in itself. The file may contain **collision(s)** to **records** existing in the system. Refer to the Collision Resolution algorithm in the next section when a **collision** is detected.
    - Data validity: for example, the UPC numbers must conform to the UPC-A standard, as specified in the **Evolution requirements**.
    - Empty attributes: all required attribute(s) must be supplied, except when both of the following are true:
        - It is marked as **autogeneratable** in the **Evolution requirements**;
        - The record whose required attribute is empty contains no **collision**.
4. **Referential integrity** must be preserved in the file.
   Note that the following referential relations exist among the files / corresponding entries in the system:
    - In `skus.csv`,
        - `Product Line Name` must be existing in the system.
    - In `formulas.csv`:
        - `SKU#` must be existing in the system.
        - `Ingr#` must be existing in the system.

Additional Requirements on Importing Formulas

If an SKU# is contained in a formulas file, all (ingredient, quantity) tuples must be interpreted as replacements, rather than additions. That is, all existing (ingredient, quantity) tuples existing in the database for this SKU must be deleted, before importing the formulas file.

Collision Resolution

As described in the previous section, **collisions** may be contained in the **CSV file(s)** submitted by the user. When a collision is detected in a file, the following algorithm, described in pseudocode in procedure `add_record`, must be used to resolve collision.

If it is determined by the algorithm that an existing **record** is to be updated, a warning must be displayed to the administrator in detail about this update. The administrator's consent must be obtained prior to committing any update to the system. Implementation may combine multiple such warnings in one **session** and, if the user consents, update all **records** in collision at once.

If it is determined by the algorithm that an import should fail, or if the user did not provide consent in case of **collision(s)**, the import must be aborted. Any changes to the system in the same session must be rolled back if the import is aborted.

```
function search(table, key=value):
    returns a record with key = value, or does not exist

function fail(msg):
    Abort and rollback changes
    Display detailed error message to administrator

function confirm(old_record, new_record):
    Display detailed information to administrator
        about the update from old_record to new_record
    Returns true iff the user consents

function table(record):
    returns the table that would contain the record

procedure add_record(new_record):
    t <- table(new_record)
    primary_match <- search(t, key=new_record.primary_key)
    unique_keys <- {t.keys} \ t.primary_key
    matches <- {}
    for key in unique_keys:
        record <- search(t, key=new_record.key)
        add record to matches
    if matches contains more than 1 record:
        fail("Ambiguous record")
    if matches contains 1 record:
        if matches != {primary_match}:
            fail("Ambiguous record")
    if primary_match exists:
        if confirm(primary_match, new_record):
            update primary_match to new_record
    else:
        insert new_record
```

Here are some examples of collisions and their resolutions. The examples are listed in (primary key, unique key, non-key) triplets.

Existing records	**Imported records**	**Resolution**
(1, Chocolate, A) (2, Cheese, B)	(1, Apple, A)	(1, Apple, A) (2, Cheese, B)
(1, Chocolate, A) (2, Cheese, B)	(1, Cheese, A)	*Fail: Ambiguous record*
(1, Chocolate, A) (2, Cheese, B)	(2, Cheese, C)	(1, Chocolate, A) (2, Cheese, C)
(1, Chocolate, A) (2, Cheese, B)	(1, Chocolate, B)	(1, Chocolate, B) (2, Cheese, B)
(1, Chocolate, A) (2, Cheese, B)	(3, Cheese, B)	*Fail: Ambiguous record* (Note: primary key cannot be updated)
(1, Chocolate, A) (2, Cheese, B)	(1, Cheese, A) (2, Chocolate, B)	*Fail: Ambiguous record*
(1, Chocolate, A) (2, Cheese, B)	(1, Chocolate, C) (2, Apple, D)	(1, Chocolate, C) (2, Apple, D)
(1, Chocolate, A) (2, Cheese, B)	(3, Apple, C)	(1, Chocolate, A) (2, Cheese, B) (3, Apple, C)

Export Workflow

Exactly one of the following exported workflows must be used. They are defined as **Batched Mode**, and **Individual Mode**, and are described in the following sections. The implementer must indicate clearly on relevant workflow UIs, which mode is implemented. The implementer may choose a different mode for this workflow than the one for the import workflow.

Batched Mode

Batched mode allows multiple files to be obtained in one export session, and is the customer's preferred mode of operation.

1. At most one file of each **type** may be exported in one export **session**.
2. All exported file(s) must have compatible formats and names such that it/they can be used as-is in an import **session**.
3. In one export **session**, the file(s) may be transferred to the user individually (e.g., as clickable, downloadable link(s)), or in a ZIP-formatted archive file. The system must support both export formats.
4. The **CSV file(s)**, whether by themselves, or included in the archive file, should follow the naming convention above.
5. All exported **CSV file(s)** must have MIME type `text/csv`. The archive file must have MIME type `application/zip`.
6. If the export were in CSV format, the **record(s)** must be filterable by the facilities defined in the **Evolution requirements**. If the export were in archive format, the **record(s)** need not be filterable.
7. The archive file, if used, must be free of intermediary directories. In other words, the structure of the archive should be
   ```
   (root)
     |-- <file1>.csv
     |-- <file2>.csv
     |-- ...
   ```
8. The file(s) may be included in the archive in no particular order.

Individual Mode

Individual mode limits the number of files that may be obtained in one export session to one.

1. One file may be exported in one export **session**.
2. The user must be able to specify the records to be exported.
3. An exported file must have a compatible format and name such that it can be used as-is in an import **session**.

4. The **CSV file** should follow the naming convention above.
5. The exported **CSV file** must have MIME type `text/csv`.