Liquibase Recipes at Appian

What do you want to do to the database?	Here's how	Status
Add a non-nullable column to an existing table	[link]	Reviewed
Define a nullable column whose non-null values have a uniqueness constraint applied to them	[link]	Reviewed

Explanation

This is a list of Liquibase "best practice" recipes to follow at Appian. The recipes are divided into two sections, **Proposed** and **Reviewed**. Anyone can add to this document, but please follow these steps:

1. Add your recipe to the Proposed section, including your name and the date
2. Ask for a code review in the Liquibase Reviewers chat room
3. Ask for a code review in the Databases and Search (DaS) chat room
4. Once you're approved by both a Liquibase approver and a DaS member, move your recipe to the Reviewed section

Reviewed Recipes

↓↓↓↓↓ Please don't add to the section below unless you've gone through the review process in the Introduction section ↓↓↓↓↓

Add a **non**-nullable column to an existing table

[Added 12/20/21 by Dave Lum, reviewed by Erol Guven]

This recipe is for when you have an existing table and you want to add a new column to it with the following properties:

- The column should never be null
- All pre-existing rows should set the new column to a fixed default value
- If our software forgets to set a value for this column in a newly-inserted row in the future, we want the DB to flag the bug by exploding

If you search, you can find two different ways to do this in our Appian code base– a three-changeSet approach and a two-changeSet approach. The two-changeSet approach shown below has less clutter and is better when the table has a lot of existing columns because it doesn't involve a separate UPDATE statement.

```
- changeSet:
    id: '000340.1.0'
    author: appian
    comment: Add icf_status column to dpkg table
    addColumn:
      tableName: dpkg
      column:
        name: icf_status
        type: ${byteType}
        defaultValue: 0
        constraints:
          nullable: false

- changeSet:
    author: appian
    id: '000340.1.1'
    comment: Remove the default value
    dropDefaultValue:
      tableName: dpkg
      columnName: icf_status
      columnDataType: ${byteType}
```

Define a nullable column whose non-null values have a uniqueness constraint applied to them

[Added 12/20/21 by Dave Lum, reviewed by Erol Guven]

This recipe shows how to define a new column that has the following properties:

- It may contain null values
- It's indexed, so querying for a **non**-null value is fast
- The DB enforces a uniqueness constraint, ensuring that the non-null values never overlap

This turns out to be the default behavior for unique indexes on most of our supported DB servers– but for DB2 and SQL Server it's not the default behavior, and some tweaking is required.

```
- changeSet:
    id: '000348.1.0'
    author: appian
    comment: Create the version-related uuid_if_current column
    changes:
      -  addColumn:
          tableName: record_type
          column:
            name: uuid_if_current
            type: ${uuidType}
            constraints:
              nullable: true
```

```
- changeSet:
    id: '000348.2.0'
    author: appian
    comment: Add a unique index to the uuid_if_current column
    changes:
      - createIndex:
          indexName: rec_type_uuid_if_current_idx
          tableName: record_type
          unique: true
          columns:
            - column:
                name: uuid_if_current
      - modifySql:
          # SQL Server unique indexes disallow multiple NULL rows unless you do this:
          dbms: mssql
          append:
            value: ' WHERE ([uuid_if_current] IS NOT NULL)'
      - modifySql:
          # DB2 unique indexes disallow multiple NULL rows unless you do this:
          dbms: db2
          append:
            value: ' EXCLUDE NULL KEYS'
```

^^^^^ Please don’t add to the section above unless you’ve gone through the review process in the Introduction section ^^^^^

Proposed Recipes

Add a non-nullable boolean column to an existing table

Proposed By	Patrick Lynch (from material by George Ivan)
Proposed On	Dec 16, 2024
Liquibase Reviewer	<<pending>>
DaS Squad Reviewer	<<pending>>

When adding a non-nullable boolean column to an existing table, Liquibase offers the following options for how to configure the value of the new column:

- `defaultValueBoolean` sets a default value at the schema level for the new column. It's beneficial for future inserts where the value for the new column might not be provided.
- `valueBoolean` updates existing rows with a specific value when the column is added. This ensures that all existing data complies with the new column's constraints right away.

For best practices on adding a non-nullable boolean column to a table with existing records, you should consider the following approach:

1. Use `valueBoolean` to ensure all existing rows are updated with a specific boolean value as part of the `addColumn` operation. This step ensures immediate compliance with the non-nullable constraint for existing data.
2. Consider also specifying `defaultValueBoolean` if you want new rows inserted after the column addition to have a default value when the column value is not explicitly provided. This is more about future-proofing your table schema.

In summary, `valueBoolean` is all that is required to add a non-nullable boolean column to an existing table. Use of `defaultValueBoolean` is not required but should be considered as a defensive programming pattern.

```
- changeSet:
    id: '000566.1.1'
    author: appian
    comment: Add is_deleted field to mining_kpi
    changes:
      -  addColumn:
          tableName: mining_kpi
          column:
            name: is_deleted
            type: ${booleanType}
            defaultValueBoolean: false
            valueBoolean: false
            constraints:
              nullable: false
```

[Your Recipe Name]

Proposed By	<<Your name>>
Proposed On	<<Today's date>>
Liquibase Reviewer	<<pending>>
DaS Squad Reviewer	<<pending>>

[Your Liquibase code and explanation here]

[Your Recipe Name]

Proposed By	<<Your name>>
Proposed On	<<Today’s date>>
Liquibase Reviewer Reviewer	<<pending>>
DaS Squad Reviewer	<<pending>>

[Your Liquibase code and explanation here]