Implementing Common Database Constraints

15-721 Final Presentation

Wuwen Wang, Yingjing Lu, Yi Zhou

Constraints - Overview

Constraints are an important feature in DBMSs to ensure database integrity

- PRIMARY KEY
- UNIQUE
- FOREIGN KEY
- Other constraints

Goals & Progress

75% goal: implement basic constraints for create_table

- UNIQUE
- PK √
- FK

100% goal: implement foreign key for create_table, and enforce it in insertion

(Almost there)

125% goal: implement ALTER operation to support dynamic change of constraints

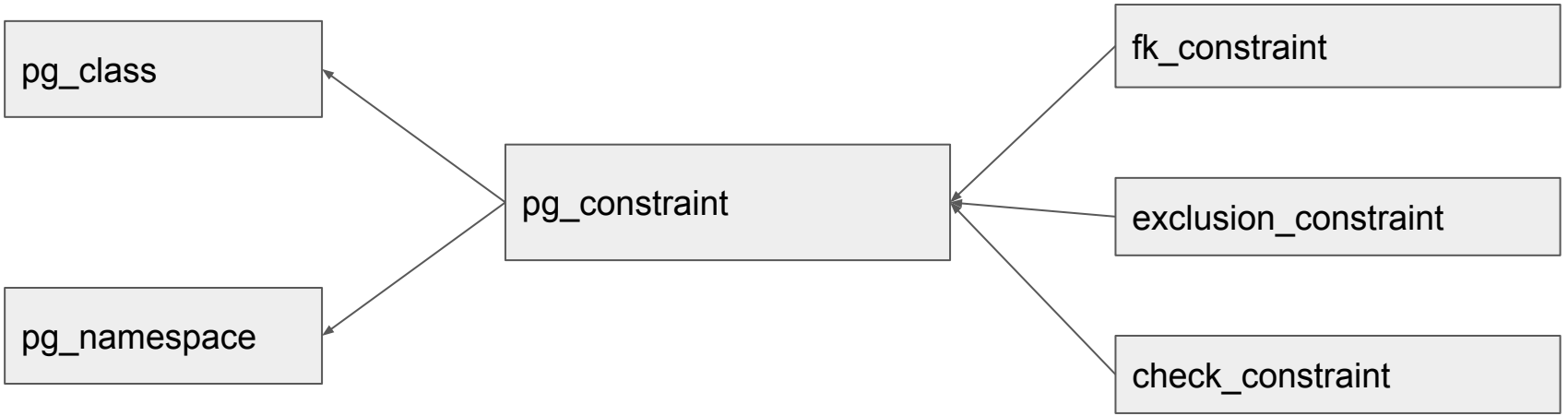
Progress

Completed & Tested:

- Create Constraints (PK, NOTNULL, UNIQUE, FK)
- PK, UNIQUE, FK Constraint Verification
- Offloaded UNIQUE and PK from Index schema to constraint checking
- Multi Column PK, UNIQUE, FK Constraint Support
- Delete Constraint When Deleting Table

Constraint Storage Schema

Refactored the original pg_constraint schema for better flexibility and expandability



Procedure

CREATE TABLE TableA (id INT PRIMARY KEY, data INT, data2 INT UNIQUE);

DDLExecutors::CreateTableExecutor

-> CreateConstraintsAndIndices

-> CatalogAccessor

-> DatabaseCatalog::CreateConstraint

- Write constraint information into the pg_constraint

*For FK, also write information into fk_constraint

- Build bw_tree for constraints to improve scan performance

Procedure

INSERT INTO TableA VALUES (1, 2, 2);

INSERT INTO TableA VALUES (2, 2, 2);

InsertTranslator::GenConstraintVerify

-> ast::Builtin::VerifyTableInsertConstraint

-> StorageInterface::VerifyTableInsertConstraint

-> DatabaseCatalog::VerifyTableInsertConstraint

- PK/UNIQUE: Scan through bw_tree and check if input matches
- FK: Scan through bw_tree of referenced table and check if input matches

Testing

	Unittest	Runtime Query
Create Constraint & Index	√	√
Create Multi Column Constraint	√	√
Delete Constraint	√	√
Get Constraint	√	√
Enforce UNIQUE	√	√
Enforce PK	√	√
Enforce FK	In progress	In progress
Enforce Multi Column Constraint	In progress	In progress

Surprises & Challenges

- When the ProjectedRow for insert, update, etc contains VARLEN column, entry data cannot be directly retrieved by using column oid and offset according to the target table schema. We are still trying to figure this out.
- We are using String as a fake array, since VARBINARY can only be used internally. It needs to be improved after Terrier correctly support VARBINARY.
- Some functionalities of constraints such as ALTER can be implemented after Terrier supports index update, since it only allows index modification during table creation.

Code Quality

Strengths:

- Modularity on FK, EXCLUSION, CHECK constraint creation and storage. Allowing them to be further expanded without affecting implementation of existing ones

Weaknesses:

- Array storation
- Index update
- Abort handling

Demo

Future WORK

Cascade for Update,
Delete and Drop

*Implement online ALTER statement on constraints after the support of index-update

In progress for testing before the final code drop:

- UPDATE/DELETE FK CASCADE
- DROP TABLE CASCADE
- NOTNULL verification

Constraint Storage Schema