Intermediate Report

CPTW Software Group

Faculty Sponsor:

Suneuy Kim - San Jose State University Professor

San Jose State University - College of Engineering

Principal Investigators:

Diego Cruz

Jared Park

Andrew Tedijanto

Shuti Wang

# 

# 1 - Table of Contents

[**1 - Table of Contents**](#_fm4cutrenj4d) **1**

[**2 - Abstract**](#_7mdjjr6f4sw9) **2**

[**3- Background**](#_1t3h5sf) **2**

[**4 - Complete Database Schema**](#_4d34og8) **3**

[4.1 - Database Relation Schema Form(s)](#_a1ykpntsqoxg) 5

[**5 - Results**](#_1ksv4uv) **6**

[5.1 - Initial Relations](#_x6p6v4rwikjs) 6

[5.1a - Generator Relation](#_me6ljldxa3c5) 6

[5.1b - Fx Relation](#_pv9ithtb3zi7) 7

[5.1c - Downloaded Relation](#_5uri1nawstc4) 7

[5.1d - Plugins Relation](#_mxz1erm4dfy0) 8

[5.1e - User Relation](#_5npfzkjkd75n) 8

[5.2 - Distinct Functions](#_5pa6gsjliasc) 9

[5.3 - SQL](#_sqzpoxmkl75l) 14

[5.3a - Select Statements](#_dp4677cdx2sj) 14

[5.3b - Update Statements](#_egwg5hcessv0) 16

[5.3c - Delete Statements](#_ul7cbm9m3ydc) 16

[5.3d - Insert Statements](#_h43hpjjsg1b5) 16

[5.3e - Triggers](#_hpd61o3twfkn) 17

[5.3f - Stored Procedures and Archiving](#_xwr4kjtef17o) 18

[5.3g - Key Constraint and Foreign Key Constraint Violations](#_l2291lezayix) 18

# 

# 2 - Abstract

Our project is a representation of plugin management for audio processing software commonly used in audio engineering applications and music production. The data set is inspired by the current software inside of the King Library 4th Floor Sound Studio managed by Student Computing Services as well as the software inside of the personal workstation of Diego Cruz. The data set structure is partially inspired by the plugin databases that exist for Digital Audio Workstation software such as Image-Line FL Studio and Cockos REAPER. The group sought to improve upon these already-established plugin databases by introducing new attributes as well as introducing a modified structure expressed through tables.

# 3- Background

The design of this project is in the context of a desktop machine geared for audio production that exists on the 4th floor of the King Library, inside of the sound studio. There exists a clear and present need to manage the plugins (and their relevant licensing) inside of the sound studio for internal use among the Student Computing Services and IT employees.

Plugins are software tools that are used in the audio production process. The most common formats of plugins are VST/VST3 (Virtual Studio Technology- Windows and Mac), AU (Audio Unit - Mac Only), and AAX (Pro Tools). They are commonly divided into effects and generators, which then have several different subtypes. There exist both free and paid plugins, as well as plugins that do not require a license to use and those that do. Interestingly some free plugins do require a license to be able to use them, and just about every paid plugin requires a license to fully use the plugin.

FX plugins take an already-generated sound and manipulate it using various methods. For instance, an equalizer can influence the overall sound by applying boosts/cuts/ dynamic modulations with certain frequencies, a reverb can give a sound the impression that it is playing in a certain acoustic environment, and a delay can give a sound the impression that it is playing in a space where reverberations can bounce back and give echo feedbacks. There are many different types of effects, but the aforementioned are some of the most commonly used effects.

Generator plugins are able to generate a sound, often resampling them or synthesizing them. There are two common types of generators that cover 97% of all generator plugins that exist. There exists ROMplers, which use a multisample library of a sound and add parameters where you can add some effects onto them. These do not generate the sound from scratch, so there is little that can be done to change the fundamentals of the sound. On the other hand, softsynths are synthesizer plugins that can generate a sound from scratch, often using basic wave shapes, wavetables or a few other methods to create a sound from the ground up.

# 4 - Complete Database Schema

DROP DATABASE IF EXISTS PLUGINS;

CREATE DATABASE PLUGINS;

USE PLUGINS;

DROP TABLE IF EXISTS Plugins;

CREATE TABLE Plugins(

plugin VARCHAR(50), # plugin uses as join attributes(name) in generator and fx table

developer VARCHAR(30),

type VARCHAR(20), # fx or generator

PRIMARY KEY(plugin)

);

DROP TABLE IF EXISTS Generator;

CREATE TABLE Generator(

name VARCHAR(50) PRIMARY KEY, # name of the plugin

format VARCHAR(10),

subtype VARCHAR(50),

isFree BOOLEAN DEFAULT FALSE,

hasLicense BOOLEAN DEFAULT FALSE,

lastUpdate timestamp not null on update current\_timestamp # 0000-00-00 00:00:00

);

DROP TABLE IF EXISTS Fx;

CREATE TABLE Fx(

name VARCHAR(50) PRIMARY KEY, # name of the plugin

format VARCHAR(10),

subtype VARCHAR(50),

isFree BOOLEAN DEFAULT FALSE,

hasLicense BOOLEAN DEFAULT FALSE,

lastUpdate timestamp not null on update current\_timestamp

);

DROP TABLE IF EXISTS User;

CREATE TABLE User(

uID INT AUTO\_INCREMENT,

name VARCHAR(30),

age INT,

downLoaded INT,

PRIMARY KEY(uID)

);

DROP TABLE IF EXISTS Downloaded;

CREATE TABLE Downloaded(

uID INT,

plugin VARCHAR(50),

downloadDate timestamp not null on update current\_timestamp,

PRIMARY KEY(uID, plugin, downloadDate),

FOREIGN KEY(uID) references User (uID) on delete cascade,

FOREIGN KEY(plugin) references Plugins (plugin) on delete cascade

);

DROP TABLE IF EXISTS Archive;

CREATE TABLE Archive(

name VARCHAR(50) primary key, # name of the plugin

format VARCHAR(10),

developer VARCHAR(30),

type VARCHAR(20), # fx or generator

subtype VARCHAR(50),

isFree BOOLEAN DEFAULT FALSE,

hasLicense BOOLEAN DEFAULT FALSE,

lastUpdate timestamp not null on update current\_timestamp,

uID INT,

downloadDate timestamp on update current\_timestamp

);

## 

## 4.1 - Database Relation Schema Form(s)

Step 1: start with all attributes

R(plugin, format, developer, type, subtype, isFree, hasLicense, lastUpdate, uID, name, age, downloaded, downloadDate,)

plugin -> developer, type

plugin(name)-> format, subtype, isFree, hasLicense, lastUpdate

uID -> name, age, downloaded

uID, plugin, downloadDate -> uID, plugin, downloadDate

{plugin}+ : {plugin, developer, type, format, subtype, isFree, hasLicense, lastUpdate} is not a super key, BCNF violation.

{uID}+: {uID, name, age, downloaded} is not a super key, BCNF violation

{uID, plugins, downloadDate}+ : {uID, name, age, downloaded, plugins, downloadDate} is not a super key, BCNF violation.

Step 2: decompose

Decompose 1:

R(plugin, developer, type, format, subtype, isFree, hasLicense, lastUpdate)

plugin -> developer, type

plugin(name)-> format, subtype, isFree, hasLicense, lastUpdate

{plugin}+ : {plugin, developer, type, format, subtype, isFree, hasLicense, lastUpdate} is a super key, no BCNF violation.

Since we have 2 types, we decided to further decompose into 2 tables based on the type (Fx, Generator)

Each type will have it attributes: (plugin(name)-> format, subtype, isFree, hasLicense, lastUpdate)

Decompose 2:

R(uID, name, age, downloaded, plugin, downloadDate, downloadDate)

uID -> name, age, downloaded

uID, plugin, downloadDate -> uID, plugin, downloadDate

{uID}+: {uID, name, age, downloaded} is not a super key, BCNF violation

{uID, plugins, downloadDate}+ : {uID, plugins, downloadDate, name, age, downloaded} is super key, no BCNF violation.

Decompose 3

R(uID, name, age, downloaded}

uID -> name, age, downloaded

{uID}+: {uID, name, age, downloaded} is a super key, no BCNF violation

Step 3 all the tables we have:

Plugins(plugin, developer, type)

Generator(name, format, subtype, isFree, hasLicense, lastUpdate)

Fx(name, format, subtype, isFree, hasLicense, lastUpdate)

User(uID, name, age, downloaded)

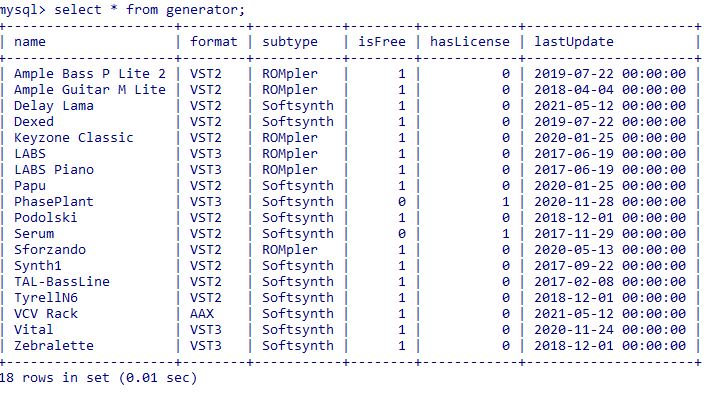
Downloaded(uID, plugin, downloadDate)

Archive table will have attributes :Archive(plugin(name), format, developer, type, subtype, isFree, hasLicense, lastUpdate, uID, downloadDate) with plugin(name) as the primary key.

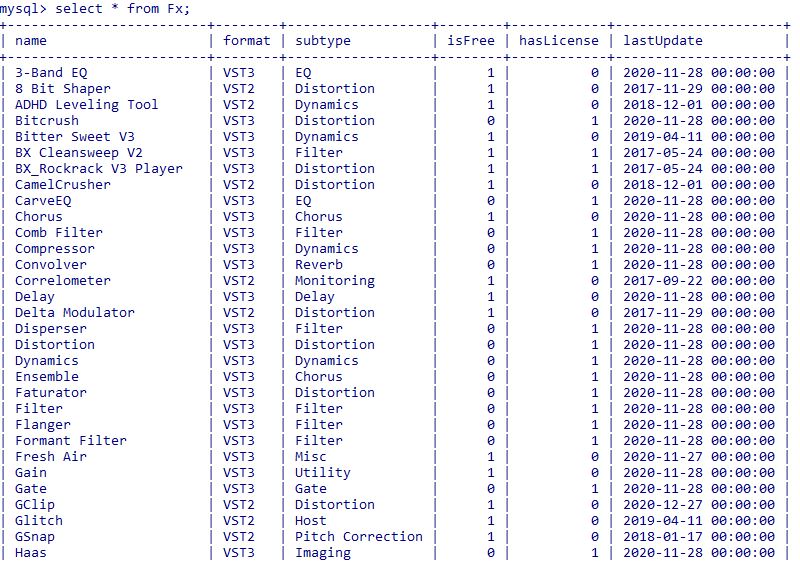
# 5 - Results

## 5.1 - Initial Relations

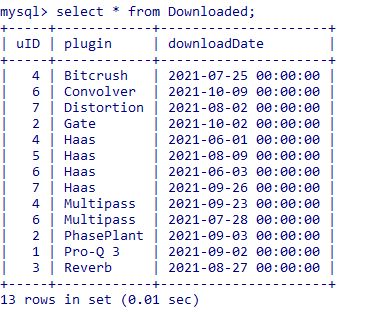
### 5.1a - Generator Relation



### 5.1b - Fx Relation



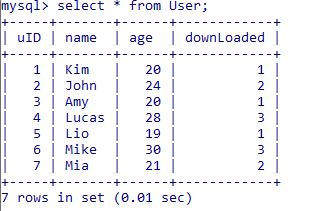
### 5.1c - Downloaded Relation



### 5.1d - Plugins Relation

[insert image here]

### 5.1e - User Relation



## 

## 5.2 - Distinct Functions

// select 1

// this select query will find developers who make both fx plugins and generator plugins and find // total plugins each of these selected developers create.

select developer, count(developer) as numOfPlugins from plugins old

where type !=any (select type from plugins where developer = old.developer)

group by (developer);

// select 2

// this select query will find the type of plugin that is downloaded the most by users

select type from plugins join (

select plugin, count from (

select plugin, count(plugin) as count from downloaded group by plugin) as d1

where d1.count >= all (

select count(plugin) from downloaded group by plugin))d2

where d2.plugin = plugins.plugin;

// select 3

// this query will find users who downloaded both fx and generator plugins

select name from user where uid = (select uID from

(select uID, plugin from downloaded natural join plugins

where plugin != all (select name from fx)) t1 join

(select uID, plugin from downloaded natural join plugins

where plugin = any (select name from fx)) t2 using(uID));

//select 4

//select the names of the plugins that have been downloaded from Fx and Generator after the

//year 2020

select distinct generator.name from generator

left outer join downloaded on generator.name = downloaded.plugin

where generator.lastUpdate > '2019-12-31' and downloaded.downloadDate > '2019-12-31'

union all

select distinct fx.name from fx

right outer join downloaded on fx.name = downloaded.plugin

where fx.lastUpdate > '2019-12-31' and downloaded.downloadDate > '2019-12-31';

//select 5

//select number of generator plugins and fx plugins in VST3 format

select count(distinct generator.name) as genCount, count(distinct fx.name) as fxCount

from generator, FX

where generator.format = 'VST3' and fx.format = 'VST3';

//select 6

//select average number of plugins in fx table that are free

select avg(stcounts.ct)

from(

select count(subtype) as ct

from fx

where fx.isFree = 1

group by subtype

having count(subtype) > 1

) as stcounts;

//select 7

//select all users that have the same number of downloads

select distinct u1.name as User1, u2.name as User2

from User u1, User u2

where u1.name < u2.name and u1.downLoaded = u2.downLoaded

group by u1.uID;

//select 8 find developers made only one type of plugins and not the other and the total number //of plugins that developer made

select developer, type, count(developer) as numOfPlugins from plugins where developer != any (select developer from plugins old where type != any (

select type from plugins where old.developer = developer) group by developer) group by developer;

// select 9 : select user whose download is not free but has license

select user.name from (select uid from (select \* from downloaded left outer join generator on plugin = name union select \* from downloaded left outer join fx on plugin = name) t where t.name is not null and isFree = 0 and hasLicense = 1 group by(uid)) s natural join user;

// update 1: update the plugin ‘Delay Lama’ with the current time.

update generator set lastUpdate = now() where name = 'Delay Lama';

// update 2 : update generator by inverting the isFree variable for plugins with the subtype

// ‘ROMpler’

update generator set isFree = not isFree where subtype = ‘ROMpler’;

// delete 1 : work with trigger 2

// delete the uid =7, and plugin = ‘Haas’ and downloadDate=’2021-09-26’ will delete the record

// from the downloaded table as well as update the user table for the downloaded times for uid 7

delete from downloaded where uid = 7 and plugin ='Haas' and downloadDate='2021-09-26';

// delete 2 // expiration after 4 months

//delete plugins that have not been updated in 4 months

delete from fx

where NOW() > DATE\_ADD(fx.lastUpdate, INTERVAL 4 MONTH);

// insert 1: work with trigger 1

// for insert into downloaded table where user download new plugins at current time

insert into downloaded values(1, 'Haas', now());

// insert 2: insert a new user

insert into user(name, age, downloaded) values('Anna', 20, 0);

// trigger 1:

// create trigger validateUserForDownload before insert into downloaded table, need to check //if user is an existing user, and will add 1 to total download times for that user in the user table.

delimiter //

drop trigger if exists validateUserForDownload;

create trigger validateUserForDownload

before insert on downloaded

for each row

if new.uID in (select uID from user)

then update User set downLoaded = downLoaded +1 where uID = new.uID;

end if;//

delimiter ;

// trigger 2:

// delete record from downloaded table, the user table downloaded plugin’s time will be reduced // as well, this trigger will not check delete record input for validation. So it is allow delete //multiple record, as long as the value in the where clause exist in the downloaded table

delimiter //

drop trigger if exists deleteDownloadRecord;

create trigger deleteDownloadRecord

before delete on downloaded

for each row

begin

update User set downloaded = downloaded -1 where uid = old.uid;

end;//

delimiter;

// stored procedure table archive will store old plugins that its lastUpdate was 2020 and earlier.

//this procedure will first insert into archive using data from plugins join fx and plugins join

//generator using common attribute plugin name and the given parameter ‘last\_update’ . Then

//delete old plugs info from plugins, fx and generator table using common attributes and given

//parameters. Archive table also include user who has downloaded that plugins

drop procedure if exists archive;

delimiter //

create procedure archive(IN last\_update timestamp)

begin

insert into archive (name, format, developer, type, subtype, isFree, hasLicense, lastUpdate, uID, downloadDate) select name, format, developer, type, subtype, isFree, hasLicense, lastUpdate, uID, downloadDate from plugins join fx on fx.name = plugins.plugin and fx.lastUpdate < last\_update left outer join downloaded on downloaded.plugin = plugins.plugin;

insert into archive (name, format, developer, type, subtype, isFree, hasLicense, lastUpdate, uID, downloadDate) select name, format, developer, type, subtype, isFree, hasLicense, lastUpdate, uID, downloadDate from plugins join generator on generator.name = plugins.plugin and generator.lastUpdate < last\_update left outer join downloaded on downloaded.plugin = plugins.plugin;

delete from plugins where plugins.plugin in (select name from fx where lastUpdate < last\_update);

delete from plugins where plugins.plugin in (select name from generator where lastUpdate < last\_update);

delete from fx where lastUpdate < last\_update;

delete from generator where lastUpdate < last\_update;

end //

delimiter;

call archive('2020-01-01');

## 

## 5.3 - SQL

### 5.3a - Select Statements

// select 1

// this select query will find developers who make both fx plugins and generator plugins and find // total plugins each of these selected developers create.

select developer, **count**(developer) as numOfPlugins from plugins old

where type !=any (select type from plugins where developer = **old.developer**)

**group by** (developer);

// select 2

// this select query will find the type of plugin that is downloaded the most by users

select type from plugins **join** (

select plugin, count from (

select plugin, **count**(plugin) as count from downloaded group by plugin) as d1

where d1.count >= all (

select **count**(plugin) from downloaded **group by** plugin))d2

where d2.plugin = plugins.plugin;

// select 3

// this query will find users who downloaded both fx and generator plugins

select name from user where uid = (select uID from

(select uID, plugin from downloaded natural **join** plugins

where plugin != all (select name from fx)) t1 **join**

(select uID, plugin from downloaded natural join plugins

where plugin = any (select name from fx)) t2 **using**(uID));

//select 4

//select the names of the plugins that have been downloaded from Fx and Generator after the

//year 2020

select distinct generator.name from generator

**left outer join** downloaded on generator.name = downloaded.plugin

where generator.lastUpdate > '2019-12-31' and downloaded.downloadDate > '2019-12-31'

union all

select distinct fx.name from fx

**right outer join** downloaded on fx.name = downloaded.plugin

where fx.lastUpdate > '2019-12-31' and downloaded.downloadDate > '2019-12-31';

//select 5

//select number of generator plugins and fx plugins in VST3 format

select count(distinct generator.name) as genCount, count(distinct fx.name) as fxCount

from generator, FX

where generator.format = 'VST3' and fx.format = 'VST3';

//select 6

//select average number of plugins in fx table that are free

select avg(stcounts.ct)

from(

select count(subtype) as ct

from fx

where fx.isFree = 1

**group by** subtype

**having** count(subtype) > 1

) as stcounts;

//select 7

//select all users that have the same number of downloads

select distinct u1.name as User1, u2.name as User2

from User u1, User u2

where u1.name < u2.name and u1.downLoaded = u2.downLoaded

group by u1.uID;

//select 8 find developers made only one type of plugins and not the other and the total the //plugins that developer made

select developer, type, **count**(developer) as numOfPlugins from plugins where developer **!= any** (select developer from plugins old where type **!= any** (

select type from plugins where **old**.developer = developer) **group by** developer) **group by** developer;

// select 9 : select user whose download is not free but has license

select user.name from (select uid from (select \* from downloaded **left outer join** generator on plugin = name **union** select \* from downloaded **left outer join** fx on plugin = name) t where t.name is not null and isFree = 0 and hasLicense = 1 **group by**(uid)) s natural join user;

### 

### 5.3b - Update Statements

// update 1: update the plugin ‘Delay Lama’ with the current time.

update generator set lastUpdate = now() where name = 'Delay Lama';

// update 2 : update generator by inverting the isFree variable for plugins with the subtype

// ‘ROMpler’

update generator set isFree = not isFree where subtype = ‘ROMpler’;

### 5.3c - Delete Statements

// delete 1 : work with trigger 2

// delete the uid =7, and plugin = ‘Haas’ and downloadDate=’2021-09-26’ will delete the record

// from the downloaded table as well as update the user table for the downloaded times for uid 7

delete from downloaded where uid = 7 and plugin ='Haas' and downloadDate='2021-09-26';

// delete 2 // expiration after 4 months

//delete plugins that have not been updated in 4 months

delete from fx

where NOW() > DATE\_ADD(fx.lastUpdate, INTERVAL 4 MONTH);

### 5.3d - Insert Statements

// insert 1: work with trigger 1

// for insert into downloaded table where user download new plugins at current time

insert into downloaded values(1, 'Haas', now());

// insert 2: insert a new user

insert into user(name, age, downloaded) values('Anna', 20, 0);

### 

### 5.3e - Triggers

// trigger 1:

// create trigger validateUserForDownload before insert into downloaded table, need to check //if user is an existing user, and will add 1 to total download times for that user in the user table.

delimiter //

drop trigger if exists validateUserForDownload;

create trigger validateUserForDownload

before insert on downloaded

for each row

if new.uID in (select uID from user)

then update User set downLoaded = downLoaded +1 where uID = new.uID;

end if;//

delimiter ;

// trigger 2:

// delete record from downloaded table, the user table downloaded plugin’s time will be reduced // as well, this trigger will not check delete record input for validation. So it is allow delete //multiple record, as long as the value in the where clause exist in the downloaded table

delimiter //

drop trigger if exists deleteDownloadRecord;

create trigger deleteDownloadRecord

before delete on downloaded

for each row

begin

update User set downloaded = downloaded -1 where uid = old.uid;

end;//

delimiter;

### 

### 5.3f - Stored Procedures and Archiving

// stored procedure table archive will store old plugins that its lastUpdate was 2020 and earlier.

//this procedure will first insert into archive using data from plugins join fx and plugins join

//generator using common attribute plugin name and the given parameter ‘last\_update’ . Then

//delete old plugs info from plugins, fx and generator table using common attributes and given

//parameters. Archive table also include user who has downloaded that plugins

drop procedure if exists archive;

delimiter //

create procedure archive(IN last\_update timestamp)

begin

insert into archive (name, format, developer, type, subtype, isFree, hasLicense, lastUpdate, uID, downloadDate) select name, format, developer, type, subtype, isFree, hasLicense, lastUpdate, uID, downloadDate from plugins join fx on fx.name = plugins.plugin and fx.lastUpdate < last\_update left outer join downloaded on downloaded.plugin = plugins.plugin;

insert into archive (name, format, developer, type, subtype, isFree, hasLicense, lastUpdate, uID, downloadDate) select name, format, developer, type, subtype, isFree, hasLicense, lastUpdate, uID, downloadDate from plugins join generator on generator.name = plugins.plugin and generator.lastUpdate < last\_update left outer join downloaded on downloaded.plugin = plugins.plugin;

delete from plugins where plugins.plugin in (select name from fx where lastUpdate < last\_update);

delete from plugins where plugins.plugin in (select name from generator where lastUpdate < last\_update);

delete from fx where lastUpdate < last\_update;

delete from generator where lastUpdate < last\_update;

end //

delimiter;

call archive('2020-01-01');

### 5.3g - Key Constraint and Foreign Key Constraint Violations

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla libero velit, interdum ac elit non, ornare euismod orci. Quisque varius vehicula felis, ac bibendum enim tincidunt ut. Pellentesque mattis mattis diam, fermentum porttitor libero. Ut molestie justo ut leo pulvinar lobortis. Quisque faucibus egestas scelerisque. Aenean interdum efficitur finibus. Fusce mattis vitae nibh nec rutrum. Nulla tristique leo nisl, in suscipit dolor eleifend sit amet. Fusce vitae ultrices ligula. Aenean posuere semper dui vel varius.