# **COMP-1701 - Transferring Data to Databases**

## **Data Science & Machine Learning (DSML) - RRC Polytech**

Module E.3 - ERD – Duplicates and UNIQUEness  
  
CORRECTION – SQL Keyword Conflict

2 Fields from the ERD for people need to be adjusted from **user\_add** and **date\_add**, to **user\_act** and **date\_act**. Reason for this is that using ***date\_add*** as a field, conflicts with a function called **DATE\_ADD**(). Noting, we do not use SQL Keywords as table or column names (this extends to other SQL concepts as well).

After the rename, these should be above the active column. Use the = sign for each column to move them down.

Noting adjustment should match this diagram:

A screen shot of a graph

Description automatically generated

**Duplication of Data**

The biggest issue with regards to data, is adding duplicate string values within a column. For example, our town\_city\_name, prov\_state\_pref, and country\_name columms, are not a very good way to setup the table.

Our total width for all 3 columns are 60 + 50 + 75 = 185 characters in width. The values within would be smaller of course, as for our 2 top records, ‘Winnipeg’ ‘Manitoba’ ‘Canada’ would repeat over several data rows.

Also, noting the addr\_type will duplicate, which is adding 15 more characters, to an even 200 characters adding girth to our table.

A screenshot of a computer

Description automatically generated

Other issues with this free form entry, is that one can imagine all of the typos that would exist in each of those fields. Also Winnipeg could technically get assigned to the province of Saskatchewan, and again at the country level, this can cause several issues within our datasets.

Additionally, you’ve seen signup forms online before, typically these province/state fields and countries are in dropdown listings…and you would not type these in, you would select them from that list.

The way we prevent this is to create proper JOINs to other tables.

**Naming Conventions**

For the 4 new tables, we are creating, we will use a simple prefix to keep all the tables together within the table listings: **geo**\_

The tables will be:

* geo\_address\_type (table alias: gat)
  + will hold the House, Apartment, Warehouse, values
  + JOINs directly to people table
* geo\_country (table alias: gco)
  + will hold the countries required for our system
* geo\_region (table alias: grg)
  + will hold the province, state, prefecture, …shires, etc
  + will contain a country\_id column, as all region\_name’s will be within a country
* geo\_towncity (table alias: gtc)
  + will hold the town cities required for our system
  + will contain a region\_id columns, as all towns/cities are in specific regions.

**Table: Address Type (geo\_address\_type : gat)**

Changing the addr\_type VARCHAR in people to an addr\_type\_id TINYINT NOT NULL value.

A screenshot of a computer code

Description automatically generated

Using a DROP/CREATE – TRUNCATE/INSERT/SELECT blocks we can add, this directly below the DROP/CREATE/USE database section. This table should be created prior to the people table.

-- 4567890123456789012345678901234567890123456789012345678901234567890  
-- TABLE: geo\_address\_type -------------------------------------------  
  
DROP TABLE IF EXISTS geo\_address\_type;  
CREATE TABLE IF NOT EXISTS geo\_address\_type (  
 addr\_type\_id TINYINT AUTO\_INCREMENT  
 , addr\_type VARCHAR(15) NOT NULL  
 , active BIT NOT NULL DEFAULT 1  
 , CONSTRAINT gat\_PK PRIMARY KEY(addr\_type\_id)  
 , CONSTRAINT gat\_UK UNIQUE (addr\_type ASC)  
);

TRUNCATE TABLE geo\_address\_type;

INSERT INTO geo\_address\_type (addr\_type)  
VALUES ('Apartment')  
 ,('Building')  
 ,('Condominium')  
 ,('Head Office')  
 ,('Townhouse')  
 ,('Warehouse')  
 ,('Other')  
;

SELECT gat.addr\_type\_id, gat.addr\_type, gat.active  
FROM geo\_address\_type gat  
WHERE gat.active=1;  
  
-- JOIN

A screenshot of a computer

Description automatically generated

**Table: Country-Region-TownCity Logic**

To remove freeform entry values within People, we need to create the following tables, where country would group region(s) (ie: provinces, states, prefectures, etc), then regions would be grouped into the town/city tables.

**Table: Country (geo\_country : gco)**

This table needs to be created before the region table, as we cannot have a region without a country and several regions are within a country.

A white grid with black text

Description automatically generated

Using a DROP/CREATE – TRUNCATE/INSERT/SELECT blocks we can add, this directly below the DROP/CREATE/USE database section. This table should be created after the geo\_address\_type table.

-- TABLE: geo\_country ------------------------------------------------  
  
DROP TABLE IF EXISTS geo\_country;  
CREATE TABLE IF NOT EXISTS geo\_country (  
 co\_id TINYINT AUTO\_INCREMENT  
 , co\_name VARCHAR(60) NOT NULL  
 , co\_abbr CHAR(2) NOT NULL  
 , active BIT NOT NULL DEFAULT 1  
 , CONSTRAINT gco\_PK PRIMARY KEY(co\_id)  
 , CONSTRAINT gco\_UK\_name UNIQUE (co\_name ASC)  
 , CONSTRAINT gco\_UK\_abbr UNIQUE (co\_abbr ASC)  
);

TRUNCATE TABLE geo\_country;

INSERT INTO geo\_country (co\_name, co\_abbr)  
VALUES ('Canada','CA')  
 , ('Japan','JP')  
 , ('South Korea','KR')  
 , ('United States of America','US');

SELECT gco.co\_id, gco.co\_name, gco.co\_abbr  
 , gco.active  
FROM geo\_country gco  
WHERE gco.active=1;

-- JOIN

A screenshot of a computer

Description automatically generated

**Table: Region (geo\_region : grg)**

This table needs to be created before the town/city table, as we cannot have a town/city without a region (province, state, etc) and several town/cities are within a region.

A screenshot of a computer

Description automatically generated

Using a DROP/CREATE – TRUNCATE/INSERT/SELECT blocks we can add, this directly below the DROP/CREATE/USE database section. This table should be created directly after the geo\_country table is.

-- TABLE: geo\_region -------------------------------------------------  
  
DROP TABLE IF EXISTS geo\_region;  
CREATE TABLE IF NOT EXISTS geo\_region (  
 rg\_id SMALLINT AUTO\_INCREMENT  
 , rg\_name VARCHAR(50) NOT NULL  
 , rg\_abbr CHAR(2)   
 , co\_id TINYINT NOT NULL  
 , active BIT NOT NULL DEFAULT 1  
 , CONSTRAINT grg\_PK PRIMARY KEY(rg\_id)  
 , CONSTRAINT grg\_UK   
 UNIQUE (co\_id ASC, rg\_name DESC)  
);

TRUNCATE TABLE geo\_region;

INSERT INTO geo\_region (rg\_name, rg\_abbr, co\_id)  
VALUES ('Manitoba', 'MB', 1)  
 , ('Tokyo', '', 2)  
 , ('Osaka', '', 2)  
 , ('Gyeonggi', '', 3)  
 , ('California', '', 4)  
 , ('Texas', '', 4)  
 , ('Washington', '', 4);

SELECT grg.rg\_id, grg.rg\_name, grg.rg\_abbr  
 , grg.active  
FROM geo\_region grg  
WHERE grg.active=1;  
  
-- JOIN

**Table: Town/City (geo\_towncity : gtc)**

This table needs to be created before the town/city table, as we cannot have a town/city without a region (province, state, etc) and several town/cities are within a region.

A white grid with black text

Description automatically generated

Using a DROP/CREATE – TRUNCATE/INSERT/SELECT blocks we can add, this directly below the DROP/CREATE/USE database section. This table should be created after the geo\_region table and just before the people table.

-- TABLE: geo\_towncity -----------------------------------------------  
  
DROP TABLE IF EXISTS geo\_towncity;  
CREATE TABLE IF NOT EXISTS geo\_towncity (  
 tc\_id MEDIUMINT AUTO\_INCREMENT  
 , tc\_name VARCHAR(15) NOT NULL  
 , rg\_id SMALLINT NOT NULL  
 , active BIT NOT NULL DEFAULT 1  
 , CONSTRAINT gtc\_PK PRIMARY KEY(tc\_id)  
 , CONSTRAINT gtc\_UK   
 UNIQUE (rg\_id ASC, tc\_name ASC)  
);

TRUNCATE TABLE geo\_towncity;

INSERT INTO geo\_towncity (tc\_name, rg\_id)  
VALUES ('Winnipeg', 1)  
 , ('Chiyoda', 2)  
 , ('Minato', 2)  
 , ('Kadoma', 3)  
 , ('Suwon', 4)  
 , ('Seoul', 4)  
 , ('Los Altos', 5)  
 , ('Santa Clara', 5)  
 , ('Round Rock', 6)  
 , ('Redmond', 7);

SELECT gtc.tc\_id, gtc.tc\_name, gtc.rg\_id  
 , gtc.active  
FROM geo\_towncity gtc  
WHERE gtc.active=1;  
  
-- JOIN

Noting a few things here, the PRIMARY KEY will AUTO\_INCREMENT, starting at 1, and up by 1. The PK CONSTRAINT is defined after the last column was defined for each table.

The UNIQUE CONSTRAINTs will ensure we cannot add 2 of any of the previous entries, thus prevents duplicates. It should be just below the PK CONSTRAINT in each table.

And the active BIT, which defaults to 1, in this case meaning show the record.

Within the people table, you can update the addr\_type VC(15) field, to addr\_type\_id TINYINT NOT NULL and town\_city\_name VC to tc\_id MEDIUMINT NOT NULL

So, for every block you create, you will have a DROP/CREATE & TRUNCATE/INSERT/SELECT. After the final SELECT, we will have you develop the JOINs, after we update the new people table changes.

Next Up:

Altering the people table, to create JOINs from the tables created in this lecture.

## 