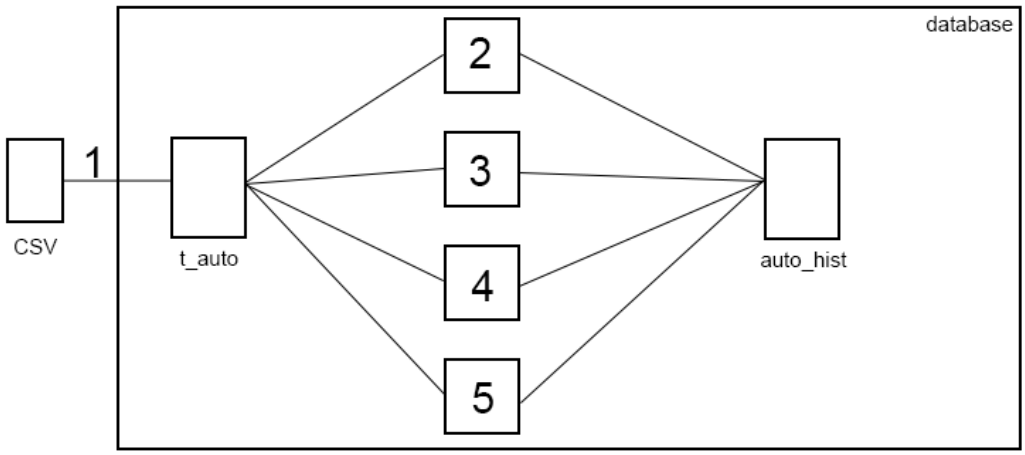
Инкрементальная загрузка

Зачастую процессы, разрабатываемые дата инженерами, направленны на формирование среза состояния системы и сохранение ее в историческую таблицу. Давайте разберем небольшой пример, направленный на более глубокое понимание процесса инкрементальной загрузки.

У нас есть CSV файл с данными, которые мы собрали с сайта auto.ru. Это данные об автомобильных объявлениях. Мы построим процесс инкрементальной загрузки этих данных в историческую таблицу.

В первую очередь давайте обсудим план процесса



Давайте разберем эту схему подробнее по шагам:

1) **загрузка данных**

Первым шагом необходимо загрузить данные в базу. Загрузка может вестись из множества разнообразных источников, однако в нашем примере мы рассмотрим загрузку данных о состоянии системы в виде csv файла.

Далее необходимо сравнить нашу промежуточную таблицу **t_auto** с выгрузкой из CSV с актуальным срезом из таблицы **auto_hist.** Это позволит сформировать несколько промежуточных таблиц

Новые объявления (пункт 2)

Новые объявления можно найти сравнив по ключу **t_auto и auto_hist** и определить записи, которые есть в **t_auto,** но отсутствуют в **auto_hist**.

Удаленные объявления (пункт 3)

Процесс нахождения удаленных записей очень поход на нахождение новых записей. Сделать это можно сравнив по ключу **t_auto и auto_hist** и определить записи, которые есть в **auto_hist,** но отсутствуют в **t_auto**.

Измененные объявления (пункт 4)

Измененные объявления, определить их можно сравнив значения из **t_auto и auto_hist** по ключу и найти записи, у которых ключ совпадает, и одно из бизнес полей отличается.

Н**еизмененные объявления (пункт 5)**

Это объявления, которые присутствуют по ключу в **t_auto и auto_hist** и полностью совпадают по всем полям. Эти записи не изменились с прошлой загрузки данных и не интересуют нас.

После того как мы сформируем промежуточные таблицы (пункты 2-4) нам необходимо произвести преобразования над таблицей **auto_hist.**

1) У всех удаленных записей (данные из таблицы пункт 3) необходимо преобразовать в **auto_hist** и указать им end_dttm текущий момент времени.

2) У всех объявлений, которые есть в таблице с измененными данными (пункт 4), и которые в **auto_hist** являются актуальными на данный момент необходимо изменить end_dttm на текущий момент времени.

3) Необходимо добавить в **auto_hist** записи о новых объявлениях (данные из пункта 2)

4) Необходимо добавить в **auto_hist** записи об измененных данных, но уже в измененном виде (пункт 4)

Данный подход позволяет обеспечить хранение всех срезов системы в исторической таблице.

Ниже представлен код который мы реализуем в классе.

```
import sqlite3
import pandas as pd
import sys
```

```
con = sqlite3.connect('sber.db')
cursor = con.cursor()

def csv2sql(filePath, tableName):
  df = pd.read_csv(filePath)
  df.to_sql(tableName, con=con, if_exists='replace')

def showTable(tableName):
  print('_'*10+tableName +'_'*10)
  cursor.execute(f'select * from {tableName}')
  for row in cursor.fetchall():
    print(row)
  print('\n'*2)

def init():
  cursor.execute('''
    CREATE TABLE if not exists auto_hist(
      id integer primary key autoincrement,
      model varchar(128),
      transmission varchar(128),
      body_type varchar(128),
      drive_type varchar(128),
      color varchar(128),
      production_year integer,
      auto_key integer,
      engine_capacity real,
      horsepower integer,
      engine_type varchar(128),
      price integer,
      milage integer,
      start_dttm datetime default current_timestamp,
      end_dttm datetime default (datetime('2999-12-31 23:59:59')))
    ''')

  cursor.execute('''
    CREATE VIEW if not exists v_auto as
    select
      id,
      model,
      transmission,
      body_type,
      drive_type,
      color,
      production_year,
      auto_key,
      engine_capacity,
      horsepower,
      engine_type,
      price,
      milage
    from auto_hist
    where current_timestamp between start_dttm and end_dttm;
  ''')

'''
записи, которые есть в t_auto,
но отсутствуют в auto_hist
```

```
'''

def createTableNewRows():
  cursor.execute('''
    create table auto_01 as
       select
        t1.*
       from t_auto t1
       left join v_auto t2
       on t1.auto_key = t2.auto_key
       where t2.auto_key is null;
  ''')

'''
записи, которые есть в auto_hist,
но отсутствуют в t_auto
'''

def createTableDeleteRows():
  cursor.execute('''
    create table auto_02 as
      select
        t1.auto_key
       from v_auto t1
       left join t_auto t2
       on t1.auto_key = t2.auto_key
       where t2.auto_key is null;
  ''')

'''
записи, которые есть и в auto_hist и в t_auto по ключу (auto_key)
но одно из бизнесс полей отличается

'''

def createTableChangedRows():
  cursor.execute('''
    create table auto_03 as
      select
        t1.*
      from t_auto t1
      inner join v_auto t2
      on t1.auto_key = t2.auto_key
      and (t1.model <> t2.model
        or t1.transmission <> t2.transmission
        or t1.body_type <> t2.body_type
        or t1.drive_type <> t2.drive_type
        or t1.color <> t2.color
        or t1.production_year <> t2.production_year
        or t1.engine_capacity <> t2.engine_capacity
        or t1.horsepower <> t2.horsepower
        or t1.engine_type <> t2.engine_type
        or t1.price <> t2.price
        or t1.milage <> t2.milage
      )
  ''')
```

```
def updateAutoHist():
  cursor.execute('''
    UPDATE auto_hist
    set end_dttm = datetime('now', '-1 second')
    where auto_key in (select auto_key from auto_02)
    and end_dttm = datetime('2999-12-31 23:59:59')
  ''')

  cursor.execute('''
    UPDATE auto_hist
    set end_dttm = datetime('now', '-1 second')
    where auto_key in (select auto_key from auto_03)
    and end_dttm = datetime('2999-12-31 23:59:59')
  ''')

  cursor.execute('''
    INSERT INTO auto_hist(
      model,
      transmission,
      body_type,
      drive_type,
      color,
      production_year,
      auto_key,
      engine_capacity,
      horsepower,
      engine_type,
      price,
      milage
    )select
      model,
      transmission,
      body_type,
      drive_type,
      color,
      production_year,
      auto_key,
      engine_capacity,
      horsepower,
      engine_type,
      price,
      milage
    from auto_01
    ''')

  cursor.execute('''
    INSERT INTO auto_hist(
      model,
      transmission,
      body_type,
      drive_type,
      color,
      production_year,
      auto_key,
      engine_capacity,
      horsepower,
      engine_type,
      price,
```

```
      milage
    )select
      model,
      transmission,
      body_type,
      drive_type,
      color,
      production_year,
      auto_key,
      engine_capacity,
      horsepower,
      engine_type,
      price,
      milage
    from auto_03
    ''')
  con.commit()


def deleteTMPtables():
  cursor.execute('drop table if exists t_auto;')
  cursor.execute('drop table if exists auto_01;')
  cursor.execute('drop table if exists auto_02;')
  cursor.execute('drop table if exists auto_03;')


deleteTMPtables()
csv2sql(sys.argv[1], 't_auto')
init()
createTableNewRows()
createTableDeleteRows()
createTableChangedRows()
updateAutoHist()
showTable('auto_01')
showTable('auto_02')
showTable('auto_03')
showTable('auto_hist')
```