SQLite.py manuál

Pre použitie je potrebné v rovnakom (najoptimálnejšie koreňovom) priečinku projektu mať:

Sqlite.py  
gefco.db  
config.py

Prerekvizity:

pip install python-dateutil

zbytok by mal byť „vstavaný“ ale môže sa rôzniť.

Odporúčam importovať cez:

from sqlite import General, User\_Role, Customer, Vehicle, User, Config, Shipment, Pattern, Work\_statement, Stillage\_type, Advanced\_user, Stillage, Pattern\_Item, synchronize\_db\_server\_client, synchronize\_db\_client\_server, all\_user\_codes

!!!!!!!!!!!!!

Čomu sa vyvarovať:

1. Systém si id / kódy užívateľov generuje sám, nikdy ich nemeňte.
2. SQLite nepozná časové dátové typy. Niektoré dokáže rozpoznať a uložiť si ich ako text, bohužiaľ, nie všetky. Preto každý objekt, ktorý „imituje“ sql timestamp treba vkladať ako text, databáza si s tým vie poradiť. Príklad: str(datetime.datetime(2022, 9, 20,13,00))
3. Každá tabuľka obsahuje stĺpec „last\_sync“. Tento stlpec obsahuje čas poslednej zmeny a je ďalej použitý pri synchronizácií. Odporúčam NECHYTAŤ.
4. Za žiadnych okolností sa NECHYTAŤ listu „tabulky“. Obsahuje zoznam všetkých tabuliek. V prípade akejkoľvek zmeny nebude databáza fungovať.

!!!!!!!!!!!!!!

funkcie:

all\_user\_codes()   
– vráti kódy všetkých užívateľov v databáze ako množinu  
!Zámer funkcie je ďalšie využitie v iných štruktúrach, je povolené ju ale využiť

synchronize\_db\_server\_client()   
– stiahne dáta z centrálnej databázy do lokálnej a všetky záznamy, ktoré chýbajú, alebo sú novšie nahradí.  
Funkcia nemá žiadnu návratovú hodnotu

synchronize\_db\_client\_server()  
 - nahrá dáta z lokálnej databázy do centrálnej a všetky dáta, ktoré sú novšie alebo chýbajú, nahradí  
Funkcia vracia zoznam výsledkov komunikácie pre každú tabuľku  
  
Triedy:

Každá tabuľka, ktorá bola „zakreslená“ v ERD má svoju triedu. Každá trieda obsahuje premenné, ktoré názvom a typom odzrkadľujú stĺpce z ERD.  
  
Každá trieda okrem svojich premenných obsahuje nasledujúce metódy:

vrat\_vsetky() – vráti všetky riadky danej tabuľky ako list v ktorom sú riadky reprezentované dátovou štruktúrou dictionary

vrat\_vsetky(True) – vráti všetky riadky danej tabuľky ako zoznam tried

stiahni(id) – Na základe id stiahne dáta z lokálnej databázy a vytvorí triednu reprezentáciu daného riadku. Vracia triedu obsahujúcu dáta. V prípade, že dané id neexistuje, vráti None

nahraj(\*\*obsah) – Nahrá riadok so vstupnými dátami do databázy a vráti triedu naplnenú o vstupné parametre.Ak trieda má priradené id, ktoré už v databáze je, vráti None

.update() vloží všetky premenné v triede do databázy. Pokiaľ trieda nemá pridelené id, vráti None inak vracia triedu

over\_zmazanie(). Overí či daný záznam je označený ako zmazaný, ak áno vráti True, ak doplnok je prázny alebo None vráti False

ak záznam alebo jeho id etc neexistuje vráti None a ak doplnok obsahuje inú hodnotu, vráti tú

zmazat() - Označí záznam ako zmazaný. Vráti False ak záznam už je označený ako zmazaný, None ak neexistuje alebo nieje setnuté id a true ak sa vykonaním zmaže

zmazat(doplnok) správanie ako zmazat() akurát namiesto mazacej konštanty do stlpcu 'doplnok' vloží zvolený doplnok

vrat\_vykonavatela() Vráti None ak záznam neexistuje, inak vráti vykonavatela

nastav\_vykonavatela(int kod) Nastaví do stlpcu 'vykonavatel' zadane id uzivatela, ktorý akciu vykonal (nejedná sa

o žiaden typ kľúča, teda hodnota reálne nikde inde figurovať nemusí). Vráti None ak záznam neexistuje alebo

kod nie je typu int. Vráti false ak už takýto vykonavatel je nahrany a ak sapodarí vykonavatela

zapísať vráti true

Niektoré ďalšie triedy budú mať svoje unikátne metódy v závislosti od individuálnych potrieb implementácie.

Config:  
def edit\_configs() – vráti zoznam riadkov tabulky Customer ,Config, Vehicle, ktoré sú vavzájom prepojené

Pattern:

def Data\_on\_editing(self) - vráti zoznam riadkov tabulky Customer ,Pattern, Pattern\_Item,Stillage\_type ktoré sú vavzájom prepojené