كيف تعمل مكتبة Sqlite3En

المحتويات:

المشكلة

المشكلة هي في تشفير قواعد البيانات من نوع SQLite على لغة البرمجة بايثون.

هناك طريقتان لتشفير قواعد البيانات من نوع SQLite اما **بتشفير كل القيم الموجودة ضمن ملف قاعدة البيانات** ولهذه الطريقة عدة مشاكل

او **تشفير قاعدة البيانات كاملة كملف** يتم قراءته ثم تشفيره باستخدام الدالة open ثم read ثم تشفير ما يتم قراءته ثم كتابة البيانات المشفرة باستخدام الدالة write ولهذه الطريقة مشاكل ايضا

لنناقش مشاكل الطريقتين:

مشاكل الطريقة الاولى

عند تشفير القيم لديك احتمالان **اما ان تتعامل مع قاعدة البيانات المشفرة قيمها كما هي**

اي عندما يريد المستخدم ادخال قيمة جديدة لقاعدة البيانات مثلا نقوم باخذ هذه القيمة وتشفيرها ووضعها في قاعدة البيانات المشفرة وهي طريقة سريعة وفعالة لاضافة البيانات ولكنها لا تستطيع سحب البيانات من القاعدة المشفرة باستخدام الدالة SELECT كمثال على ذلك

لو اراد المستخدم مثلا معرفة رقم تلفون موظف ضمن قاعدة البيانات المشفرة فسوف يستخدم الدالة

SELECT phone FROM db_table WHERE name = ‘employee name’

اي انه يطلب رقم التلفون باستخدام اسم الموظف المعطى فتذهب قاعدة البيانات للبحث عن اسم الموظف ومن ثم رقم هاتفه ولكنها لن تجده لان الاسم المعطى من قبل المستخدم غير مشفر والاسم في قاعدة البيانات مشفر فلا يوجد تطابق بينهما ولا يمكن لقاعدة البيانات ايجاده .

قد تتسائل لماذا لا نقوم بتشفير اسم الموظف المعطى من قبل المستخدم ثم البحث باستخدام الاسم المشفر, السبب هو انه عند تشفير اسم او نص ما مرتين فانه في كل مرة ينتج عنه قيم مشفرة مختلفة لان التشفير الحاسوبي يدخل به عدة عوامل منها الزمن وبعض القيم العشوائية لذلك من المستحيل تطابق قيمتين مشفرتين معا حتى لو كانتا لنفس العبار وبالتالي لن تستطيع قواعد البيانات ايجاد رقم التلفون ايضا (يوجد طريقة تشفير تعطي نفس القيمة المشفرة دائما لكنها غير آمنة للاستخدام)

الاحتمال الثاني هو ان تقوم بفك تشفير كامل قيم قواعد البيانات اولا في الذاكرة العشوائية حيث لا يستطيع احد قراءتها ثم العمل عليها مباشرة ولكن هذه الطريقة ستجعل البرنامج الذي تصنعه بطيء عند الاقلاع (خلال عملية فك التشفير) اذا كانت قواعد البيانات كبير الحجم وهي حل مناسب في حالة قواعد البيانات الصغيرة

لذلك لم نفضل استخدام الطريقة الاولى وهي تشفير قيم قواعد البيانات

مشاكل الطريقة الثانية

تشفير قاعدة البيانات كاملة كملف يتم قراءته ثم تشفيره , وذلك باستخدام الدالة open ثم read ثم تشفير ما يتم قراءته ثم كتابة البيانات المشفرة باستخدام الدالة write

مشكلة هذه الطريقة هي انه لا يمكنك الاتصال بالملف الناتج المشفر مباشرة واستخدامه كقاعدة بيانات باستخدام الدالة connect وذلك لان الملف الناتج ليس قاعدة بيانات وانما ملف مشفر فينبغي اولا فك تشفيره من جديد كي تنتج قواعد البيانات غير المشفرة ثم نتصل بها باستخدام الدالة connect وعند فك التشفير لا بد من وضع الملف غير المشفر ضمن القرص الصلب او الهارد كي نستطيع الاتصال به واستخدامه من قبل الدالة connect, ولا نستطيع فك تشفيره ووضعه بالذاكرة العشوائية لان الدالة connect لا تتيح لنا الاتصال به عندها اذ لا يمكن الاتصال بملف موجود بالذاكرة العشوائية باستخدام connect- ولكن يمكن انشاء قاعدة بيانات جديدة بالذاكرة العشوائية فقط واضافة معلومات لها- ولكي نضيف المعلومات لقاعدة البيانات الجديدة لا بد من قراءتها اولا بشكل غير مشفر وذلك من القرص الصلب بالتالي يمكن للمستخدم ان يجد الملف غير المشفر على القرص الصلب او الهارد والحصول على المعلوات وسرقة قاعدة البيانات.

الحل

الحل لهذه المعضله كان على الشكل التالي

لتشفير قواعد البيانات نقوم اولا بالاتصال بقاعدة البيانات الاساسية غير المشفرة والتي نرغب بتشفيرها ثم انشاء ما يدعى بالـ cursor object وهو مؤشر لقاعدة البيانات ثم نقوم بتحوليه لقائمة بايثون list باستخدام الدالة list او cursor.fetchall

بالتالي تتحول قواعد البيانات لـ list تحوي tuple اي حولنا قاعدة البيانات كاملة لقائمة List بايثون ثم نقوم بتحويل القائمة الاخيرة الى نص باستخدام الدالة str ثم نشفر ذلك النص ونكتبه بعد تشفيره باستخدام الدالة write في ملف ينتهي باللاحقة Kn. فاصبح لدينا ملف Kn. يحتوي كل قاعدة البيانات المشفرة .

لفك التشفير

نقوم اولا بانشاء قاعدة بيانات جديدة وفارغة ضمن الذاكرة العشوائية باستخدام الدالة connect ثم نقوم بقراءة ملف Kn الذي يحوي البيانات المشفرة باستخدام الدالة read ثم فك تشفير النص الناتج فنحصل على نص غير مشفر ثم تحويل النص لقائمة بايثون جديدة list باستخدام الدالة eval ثم اضافة القائمة الناتجة list الى قاعدة البيانات التي في الذاكرة العشوائية والتي انشأناها منذ قليل فنحصل على قاعدة بيانات ضمن الذاكرة العشوائية تحوي المعلومات غير المشفرة المطلوبة بسرعة كبيرة نسبيا

ويستطيع المستخدم او البرنامج التعامل مع قاعدة البيانات الموجودة بالذاكرة العشوائية بكل سهولة كونها غير مشفرة

حفظ البيانات المعدلة من قبل المستخدم وتشفيرها

عندما يقوم المستخدم او البرنامج بتعديل قواعد البيانات التي بالذاكرة ينبغي تشفيرها مجددا ووضع الملف المشفر على القرص الصلب كي لا تضيع المعلومات في حال انقطاع التيار الكهربائي مثلا كون الذاكرة العشوائية لا تحتفظ بالمعلومات بشكل دائم.

لذلك بعد قيام اي تعديل على قواعد البيانات ضمن الذاكرة نقوم بقراءة كامل قواعد البيانات مجددا وتحويها لـ cursor ثم قائمة list ثم نص str ثم تشفير لملف Kn مجددا وكتابته على القرص الصلب بالتالي نحصل على ملف Kn يحوي قاعدة البيانات بعد تعديلها بشكل مشفر.

ميزات هذه الطريقة انها سريعة بفك التشفير وبالتشفير نسبيا

ولكن لها مشكلة واحدة وهي حفظ المعلومات بعد تعديلها

ففي حال كانت قواعد البيانات كبيرة فان القيام بعملية الحفظ السابقة تحتاج لوقت طويل نسبيا 3-10 ثواني وهو امر غير مقبول في بعض البرامج كبرامج المبيعات والمحاسبة مثلا حيث ستقوم بابطاء طابور البيع في حال كان المحل او الماركت الذي يستخدم البرنامج مزدحم.

لحل المشكلة السابقة قمنا ببعض التعديلات على عملية **التشفير وفك التشفير والحفظ بعد التعديل** وهي الالية المستخدمة في مكتبة Sqlite3En :

اولا تشفير قاعدة البيانات تقوم المكتبة اولا بتقسيم قاعدة البيانات المراد تشفيرها في حال كانت كبيرة الحجم الى عدة قواعد بيانات صغيرة الحجم حيث تحوي كل قاعدة صغيرة على 500 عنصر او سطر row فقط

ثم تشفير كل قواعد البيانات الصغيرة الناتجة الى ملفات Kn

فك التشفير

نقوم بإنشاء قاعدة بيانات جديدة في الذاكرة العشوائية فارغة ثم نقوم بقراءة كل ملفات Kn السابقة وفك تشفيرها مجددا ووضعها في قاعدة بيانات الذاكرة العشوائية اي اعادة جمع كل ملفات Kn في الذاكرة العشوائية بشكل غير مشفر فتنتج لدينا قاعدة البيانات الاساسية بشكل غير مشفر بالذاكرة العشوائية مجددا ويسهل التعامل معها حينها.

حفظ التعديلات

عند قيام المستخدم او البرنامج بتعديل قيمة ما تقوم مكتبة Sqlit3En بمعرفة موقع التعديل اي في اي سطر row من قاعدة البيانات التي في الذاكرة العشوائية حصل التعديل ثم تعديل ملف Kn الذي يحوي ذلك السطر row المعدل فقط بالتالي نحصل على سرعة في عملية الحفظ والتعديل .

تمت بفضل الله

يوسف بدر يوسف