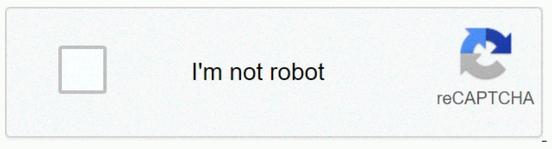

Continue

Android sqlite database foreign key example

In this tutorial, you will learn- SQLite limitsSulite limits and rules for values in a column to confirm the inserted data. Column constraints are defined when you create a table in the column definition. SQLite Primary KeyAll primary key column values must be unique, not null The primary key can be applied to only one column or combination of columns, in the latter case, the combination of column values must be unique in all rows in the tables. There are many different ways to define a primary key in a table, such as: Column definition itself: Syntax: ColumnName INTEGER NOT NULL PRIMARY KEY; As a separate definition: PRIMARY KEY(ColumnName); To create a column combination as primary key: PRIMARY KEY(ColumnName1, ColumnName2); Null constraintSQLite is not null limit preventing a column from being null: ColumnName INTEGER NOT NULL;DEFAULT ConstraintSQLite, if you do not insert a column value, a default value will be inserted instead. For example: ColumnName INTEGER DEFAULT 0; If you type an insert statement and do not specify a value for this column, the column value will be a unique constraint of 0. For example: EmployeeId INTEGER NOT NULL UNIQUE; This will make the EmployeeId value unique when duplicate values are not allowed. Note that this applies only to the values in the EmployeeId column. In the case of the SQLite CHECK constraint, the condition is limited to check the inserted value if the value does not meet the condition, it will not be inserted. INTEGER QUANTITY IS NOT A ZERO CHECK(> 10); You cannot insert a value less than 10 in the Quantity column. What is SQLite External KEY? A SQLite foreign key is a constraint that checks the existence of values present in one table to another table that is associated with the first table in which the foreign key is defined. When you work with multiple tables, you have two tables that are bound to each other by one common column. And if you want to ensure that the value inserted in one of them must exist in the other column of the table, then you should use the Foreign Key limit on the column together. In this scenario, when you try to insert a value in this column, the foreign key will ensure that the inserted value exists in the table column. Note that foreign key restrictions are not enabled by default in SQLite, you have to bring them first by running the following command: PRAGMA foreign_keys = ON; SQLite introduced foreign key restrictions starting with version 3.6.19. For example, let's say if we have two tables; Students and departments. The Students table contains the student list, and the chapter table contains a list of departments. Each student belongs to a department; that is, each student has a departmentId column. Now we 'll see whether the foreign key constraint can be useful to ensure that the department ID value in the student table must be in the Departments table. So, if we created a foreign key limit in the DepartmentId table in the Students table, each inserted departmentId must be submitted to the Departments table. CREATE TABLE [Departments] ([DepartmentId] INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, [DepartmentName] NVARCHAR(50) NULL); CREATE TABLE [Students] ([StudentId] INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL, [StudentName] NVARCHAR(50) NULL, [DepartmentId] INTEGER NOT NULL, [DateOfBirth] DATE NULL, FOREIGN KEY(DepartmentId) REFERENCE Departments(DepartmentId)); To check how foreign key constraints can prevent an undefined element or value that should be inserted into a table that has a relationship to another table, we will consider this example. In this example, the Departments table contains the foreign key constraint for the Student table, so the Students table must contain any DepartmentId inserted in the Students table. If you try to insert an departmentId value that is not in the department table, the foreign key constraint will not allow you to do so. Place chapters in the IT and Arts chapters in the table as follows: INSERT VALUES INTO CHAPTERS(1, 'IT'); INSERT INTO Departments VALUES(2, 'Art'); For both notifications, you must insert two chapters into the department table, you can ensure that both values were inserted by running the SELECT * FROM Departments query then: Then try inserting a new student with a departmentid that is not in the department table: INSERT INTO Students(StudentName,DepartmentId) VALUES('John', 5); The queue will not be inserted, and you receive an error message that states that: The foreign key constraint failed. Summary: In this tutorial, you will learn how to use the SQLite foreign key limit to enforce the relationship between related tables. SQLite foreign key limit supportSQLite has supported the foreign key limit since version 3.6.19 SQLITE_OMIT_TRIGGER SQLITE_OMIT_FOREIGN_KEY. Use the following command to verify that the current version of SQLite supports foreign key restrictions: The command returns an integer: 1: enable, 0: disabled. If the command does not return anything, it means that the SQLite version does not support foreign key restrictions. If the SQLite library is compiled with foreign key restriction support, the application can use the PRAGMA foreign_keys to enable or disable foreign key restrictions at runtime. To disable the foreign key limit:PRAGMA foreign_keys = OFF; To enable the foreign key limit:PRAGMA foreign_keys = ON;Introduction to the SQLite foreign key limitLet starts with two tables: suppliers and supplier_groups :CREATE TABLE vendors(supplier_id integer PRIMARY KEY, supplier_name text IS NOT NULL, group_id integer NOT NULL); CREATE TABLE supplier_groups (group_id PRIMARY KEY, group_name IS NOT NULL); Supposing each supplier belongs to one and only one group of suppliers. And each supplier group can have zero or many vendors. The relationship between supplier_groups and suppliers tables is one-to-many. In other words, each row in the vendor table has the appropriate row supplier_groups table. At this time, you cannot prevent adding a row to a vendor table without a corresponding row in supplier_groups table. Additionally, you can remove the line supplier_groups by deleting or updating the corresponding lines in the vendor table. This can leave the orphaned lines in the vendor table. Use foreign key constraints to enforce supplier_groups in the table. To add a foreign key constraint to a vendor table, change the create table definition as follows:DROP TABLE vendors; CREATE TABLE suppliers (supplier_id INTEGER PRIMARY KEY, supplier_name TEXT NOT NULL, group_id INTEGER NOT NULL, FOREIGN KEY (group_id) REFERENCE supplier_groups (group_id)); The supplier_groups table is called the parent table, which is the table that the foreign key refers to. The vendor table is known as a child table, which is the table that is subject to the foreign key limit. The group_id column in the supplier_groups is called a primary key, which is the parent table column or set of columns that the foreign key constraint references apply to. Typically, the parent key is the parent table's primary key. The group_id column code for a table is called a child key. Typically, the child key references to the parent table's primary key. SQLite foreign key constraint exampleSealsealsealt, insert three rows into supplier_groups table. INSERT SUPPLIER_GROUPS (group_name) VALUES (Domestic) (Global) (One-off time); Secondly, place a new vendor in the vendor table together with the vendor group supplier_groups the table. INSERT VENDORS (supplier_name, group_id) VALUES (HP, 2); This statement works perfectly fine. Thirdly, an attempt to insert a new vendor group that is not in the table is not included in supplier_groups. INSERT VENDORS (supplier_name, group_id) VALUES (ABC Inc., 4); SQLite checked the foreign key limit, rejected the change, and issued the following error message:[SQLITE_CONSTRAINT] Stop due to constraint violation (foreign key limit failed)SQLite foreign key limit actionsWhing would happen if a row is deleted from table supplier_groups? Should all vendor lines in the relevant table also be deleted? The same questions for the update operation. To specify how the foreign key constraint works each time the basic image is deleted or updated, use the ON DELETE or ON UPDATE foreign_key_columns action in the PARENT_TABLE(parent_key_columns) FOR UPDATE action DELETE action; SQLite supports the following actions:NULLSET DEFAULTRESTRICTNO ACTIONCASCADEIn practice, the values of the parent primary key are unchanged, so update rules are less important. The more important a rule is a DELETE rule that specifies an action when deleting the basics. We will check each step by using the following examples to SEMP NULLWhen the B key in the parent b changes, delete, or update the corresponding child keys in the entire child table set to NULL. First, drop and create vendors in the table by using the SET NULL group_id the:DROP TABLE vendors key; CREATE TABLE SUPPLIERS (supplier_id INTEGER PRIMARY KEY, supplier_name TEXT NOT NULL, group_id INTEGER, FOREIGN KEY (group_id) SUPPLIER_GROUPS (group_id) ON UPDATE SET NULL ON DELETE NULL); Secondly, insert a few lines into the vendor table:INSERT INSERT supplier_name (group_id) IN VALUES(XYZ Corp, 3); INSERT SUPPLIERS (supplier_name, group_id) VALUES (ABC Corp, 3); Thirdly, delete supplier group ID 3 from supplier_groups:DELETE FROM supplier_groups WHERE group_id = 3; Fourthly, query data from the vendor table. The group_id the corresponding row column in the vendor table set to NULL. SET DEFAULTThe SET DEFAULT action sets the foreign key value to the default value specified in the column definition when you create the table. Because the values in group_id are zero, if you delete a row from the supplier_groups table, group_id values will be set to NULL. By default, the foreign key constraint starts and performs a check. Ierobežota_darbība prevents you from changing or deleting values in the parent parent key of the parent parent of the parent. First, drop and create a vendor table with the RESTRICT foreign key group_id:DROP TABLE for vendors; CREATE TABLE suppliers (supplier_id INTEGER PRIMARY KEY, supplier_name TEXT NOT NULL, group_id INTEGER, FOREIGN KEY (group_id) REFERENCE SUPPLIER_GROUPS (group_id) ON UPDATE RESTRICT ON DELETE RESTRICT ON DELETE ON DELETE DELETE RESTRICT); Secondly, insert a line with 1.INSERT SUPPLIERS (group_id supplier_name, group_id) VALUES(XYZ Corp, 1) supplier_name suppliers in the table; Thirdly, remove the group of suppliers with ID 1 from supplier_groups table:DELETE FROM supplier_groups WHERE group_id = 1; SQLite issued the following error:[SQLITE_CONSTRAINT] Stop due to constraint violation (foreign key limit failed)To fix this, you must first delete all rows from the vendor table that has the table group_id 1:DELETE FROM suppliers WHERE group_id =1; You can then delete vendor group 1 from supplier_groups table:DELETE FROM SUPPLIER_GROUPS WHERE group_id = 1;NO ACTIONNEIM NO ACTION does not mean that bypassing the foreign key constraint. It has the same effect as THE LIMIT. The CASCADEThe CASCADE operation distributes changes from the parent table to the child table by updating or deleting the primary key. First , insert the table:supplier_groups INTO supplier_groups (group_name) VALUES (Domestic), (Global) (Single); Secondly, drop and create table vendors with the CASCADE foreign key group_id :D ROP TABLE suppliers; CREATE TABLE SUPPLIERS (supplier_id INTEGER PRIMARY KEY, supplier_name TEXT NOT NULL, group_id INTEGER, FOREIGN KEY (group_id) REFERENCE SUPPLIER_GROUPS (group_id) ON UPDATE CASCADE ON DELETE CASCADE (UPDATE CASCADE ON DELETE CASCADE); Thirdly, to include some suppliers in the table suppliers:INSERT INTO suppliers (supplier_name, group_id) VALUES(XYZ Corp, 1); INSERT SUPPLIERS (supplier_name, group_id) VALUES(ABC Corp, 2); Fourthly, group_id group of suppliers is named 100:UPDATE supplier_groups SET group_id = 100 WHERE group_name = Domestic; Fifth, query data from the Vendors table:As you can see the value in the XYZ Corp column, group_id table suppliers changed from 1 to 100 when we updated the group_id table suplier_groups. This is the result of ON UPDATE CASCADE's operation. Sixthly, delete vendor group ID 2 supplier_groups table:DELETE SUPPLIER_GROUPS WHERE group_id = 2; Seventhly, query data from the Vendors table: Vendor ID 2, whose group_id is 2 was deleted when vendor group ID 2 was removed from supplier_groups table. This is the effect of the ON DELETE action CASCADE. In this tutorial, you learned about the sqlite foreign key limit and how to use them to enforce relationships between related tables. Was this tutorial useful ? Yes , No ? YesNo JāNo