Android sqlite db.query where clause

Provides S'Lite database management methods. S'LiteData has methods for creating, deleting, executing S'L commands, and other common database management tasks. Here's an example of the Notepad application in SDK to create and manage the database. Database names should be unique in the app, not all applications. In addition to the default BINARY collaboration, Android delivers two more, LOCALIZED, which changes with the current location of the system, and UNICODE, which is a Unicode mapping algorithm and is not adapted to the current locale. S'LiteDatabase interface. CursorFactory is used to return Cursor sub-contracts when a request is called. Class S'LiteDatabase. OpenParams Wrapper for configuration settings that are used to open int S'LiteDatabase CONFLICT_ABORT When a limit is breached, ROLLBACK is not performed, so changes from previous commands within the same transaction are saved. int CONFLICT_FAIL When a restriction is breached, the team is interrupted with the reverse code SQLITE_CONSTRAINT. int CONFLICT_IGNORE When a restriction is breached, one line that contains a violation of the restriction is not inserted or changed. CONFLICT_NONE use the following when the conflict is not specified. int CONFLICT_REPLACE When a UNIQUE restriction is breached, pre-existing lines that cause a restriction violation are removed before the current line is inserted or updated. int CONFLICT_ROLLBACK When a restriction is breached, there is an immediate ROLLBACK, thus ending the current transaction, and the team is interrupted with the SQLITE_CONSTRAINT return code. int CREATE_IF_NECESSARY Open Flag: Flag for openDatabase (File, S'LiteDatabase. OpenParams) to create a database file if it doesn't exist yet. int ENABLE_WRITE_AHEAD_LOGGING Open Flag: Flag for openDatabase (File, S'LiteDatabase. OpenParams) to open the database file with the record forward registration enabled by default. MAX_SQL_CACHE_SIZE the maximum value that setMaxSqlCacheSize (int) can set. int NO_LOCALIZED_COLLATORS Open flag: Flag for openDatabase (file, S'LiteDatabase. OpenParams) to open a database without the support of localized collitors. int OPEN_READONLY Open Flag: Flag for openDatabase (File, S'LiteDatabase. OpenParams) to open a reading-only database. int OPEN_READWRITE Open Flag: Flag for openDatabase (File, S'LiteDatabase. OpenParams) to open a database for reading and writing. If the drive is full, it may fail even before you actually write anything. int SQLITE_MAX_LIKE_PATTERN_LENGTH The maximum length of SHABOLON LIKE or GLOB Pattern Mapping Algorithm used in the default IMPLEMENTATION OF LIKE and GLOB S'Lite can show performance O (N-2) (where N is symbols in the pattern) for certain pathological cases. Invalid startTransaction () Starts a transaction in EXCLUSIVE mode. Void Void Starts the transaction in IMMEDIATE mode. Invalid startTransactionWithListener (S'LiteTransactionListener transaction) starts the transaction in EXCLUSIVE mode. Invalid startTransactionWithListenerNonExclusive (S'LiteTransactionListener transaction) starts the transaction in IMMEDIATE mode. The S'LiteStatement (String sql) compiles the SL statement into a reusable pre-written statement. S'LiteDatabase (S'LiteDatabase Factory). CursorFactory) Create a database supported by S'Lite memory. S'LiteDatabase createsInMemory (S'LiteDatabase). OpenParams openParams) Create a database supported by S'Lite memory. Int delete (Table Line, Row, WhereClause, String, Where The Args) Convenience method for deleting rows in the database. static boolean deleteDatabase (file file) removes the database, including its log file and other supporting files that may have been created by the database engine. invalid disableWriteAheadLogging () This method disables features enabled by enableWriteAheadLogging. This method allows you to run multiple threads in a single database in parallel. the invalid end of the transaction. emptiness execPerConnectionS'L (String sql, Object) bindArgs) Follow this statement on all connections to this database. Invalid execS'L (String sql) Perform one S'L statement that is not SELECT or any other S'L statement that returns the data. void execS'L (String sql, Object) bindArgs) Perform one S'L statement that is not SELECT/INSERT/UPDATE/DELETE. static line findEditTable (String Tables) Finds the name of the first table that is edited. GetAttachedDbs returns a list of complete path names for all attached databases, including the main database, by performing 'pragma database_list' in the database. GetPath gets way to the database file. This method was highlighted at API 15. This method no longer serves any useful purpose and has been decreced. int getVersion () gets a version of the database. boolean inTransaction () Returns correctly if the current thread has a pending transaction. Long insertion (table line, nullColumnHack line, ContentValues value) Convenience method for inserting a string into the database. Long InsertorThrow (Table strings, nullColumnHack line, ContentValues Values) Convenience method for inserting strings into the database. Long insertwithOnConflict (String Table, NullColumnHack, ContentValues initialValues, int conflictAlgorithm) Common method for inserting a string into the database. boolean isDatabaseIntegrityOk () Launches 'pragma integrity_check' on this base (and all the attached databases) and returns correctly if given a given, String, String, String. (and all its attached databases) integrity_check false otherwise. boolean isDbLockedByCurrentThread () Returns correctly if the current thread contains an active connection to the database. boolean isDbLockedByOtherThreads () This method was mutilated in API 16. Always comes back false. Don't use this method. boolean isOpen () Returns correctly if the database is currently open. boolean isReadOnly () Returns correctly if the database is only open as read. boolean isWriteAheadLoggingEnabled () Returns correctly if the record forward log was included for this database. invalid markTableSyncable (Line Table, Line deletedTable) This method has been deprecated in API level 15. This method no longer serves any useful purpose and has been decreced. invalid markTableSyncable (Line Table, Line ForeignKey, Line updateTable) This method has been deprecated in API level 15. This method no longer serves any useful purpose and has been decreced. boolean needUpgrade (int newVersion) returns correctly if the new version code is larger than the current version of the database. S'LiteDatabase openDatabase (String Way, S'LiteDatabase. CursorFactory, int flags) Opens a database according to the flags of OPEN_READWRITE OPEN_READONLY CREATE_IF_NECESSARY and/or NO_LOCALIZED_COLLATORS. S'LiteDatabase openDatabase (File Path, S'LiteDatabase. OpenParams openParams) Open the database according to the stated OpenParams static S'LiteDatabase openDatabase (String Path, S'LiteDatabase. CursorFactory Plant, int Flags, DatabaseError Handler ErrorHandler) Open the database according to the flags of the OPEN_READWRITE OPEN_READONLY CREATE_IF_NECESSARY and NO_LOCALIZED_COLLATORS static system S'LiteDatabase openOrCreateDatabase (file file, factory S'LiteDatabase. CursorFactory) Equivalent of openDatabase (file.getPath), plant, CREATE_IF_NECESSARY). S'LiteDatabase openOrCreateDatabase (String Way, S'LiteDatabase. CursorFactory, Error DatabaseErrorHandlerHandler) Equivalent to openingDatabase (way, plant, CREATE_IF_NECESSARY, errorHandler). S'LiteDatabase openOrCreateDatabase (String Path, S'LiteDatabase. CursorFactory) Equivalent for openDatabase (way, plant, CREATE_IF_NECESSARY). Cursor query (boolean distinct, Table Line, String Columns, Line Choice, Line ChoiceArgs, GroupBy Row, Line, Line orderBy, Limit Line) Request this URL by returning the cursor over the set of results. Request Cursor (Table Line, Strings, Line Choice, Line ChoiceArgs, GroupBy, Row, Row, String OrderBy, Limit Line) Request this table by returning the cursor over the set of results. Cursory Query (boolean, String Table, Strings, Choice Row, String ChoiceArgs, String GroupBy, Row, Row The limit line, CancellationSignal cancellationSignal) Request this URL by returning the cursor over the set of results. Results. Request (String Table, Columns of Strings, Line Choice, String selectionArgs, String GroupBy, Row, Line orderBy) Request this table, returning courser over a set of results. Cursory requestWithFactory (S'LiteDatabase. CursorFactory cursorFactory, boolean, row table, column line, line choice, String' selectionArgs, GroupBy row, row orderBy, Limit Line, CancellationSignal cancellationSignal) Request this URL, return Course over a set of results. Cursory requestWithFactory (S'LiteDatabase. CursorFactory cursorFactory, boolean, table, column line, line choice, String' selectionArgs, GroupBy row, row orderBy, Line Limit) Request this URL, returning the cursor over the set of results. Cursor raw'query (String sql, String' selectionArgs, CancellationSignal cancellationSignal) launches S'L and returns the results cursor. Cursor raw'query (String sql, String' selectionArgs) launches the provided S'L and returns the cursor on the set of results. Courser raw'yWithFactory (S'LiteDatabase. CursorFactory cursorFactory, String sql, String' selectionArgs, String editTable, CancellationSignal cancellationSignal) launches S'L and returns the test rate. Courser raw'yWithFactory (S'LiteDatabase. CursorFactory cursorFactory, String sql, String' selectionArgs, String editTable) launches S'L and returns the cursor over the set of results. static int releaseMemory () Attempts to release the memory that S'Lite holds, but does not require normal operation. Long Replacement (String Table, String nullColumnHack, ContentValues initialValues) Convenience method to replace the string in the database. Long to replaceOrThrow (String Table, NullColumnHack, ContentValues initialValues) A convenient method for replacing the line in the database. Invalid setCustomAggregateFunction (String functionName, BinaryOperator aggregateFunction) Register a custom aggregated feature that can be called from the expressions of S'L. SetCustomScalarFunction (String functionName, UnaryOperator scalarFunction) Register a custom scalal feature that can be called from S'L expressions. The invalid setForeignKeyConstraintsEnabled (boolean enable) determines whether foreign key restrictions are included in the database. void setLocale (Locale locale) Sets a location for this database. This method now does nothing. Don't use it. The invalid Set of MaxSqlCacheSize (int cacheSize) sets the maximum cache size of the prepared statement for this database. The long set of MaximumSize (long numBytes) sets the maximum size of the database will grow up to. Invalid setPageSize (long numBytes) Sets page size Data. The invalid set ofTransactionSuccessful () marks the current transaction as a success. Invalid Set Version (int version) Installs Version. ToString returns the view of the object line. Update int (String Table, ContentValues Values, Row, WhereClause, Row, WhereArgs) Convenience method to update rows in the database. Int updateWithOnConflict (String Table, ContentValues Values, Row, WhereClause, String, WhereArgs, Int conflictAlgorithm) Convenience method to update lines in the database. invalidateSql (String sql, CancellationSignal cancellationSignal) Checks that the S'L SELECT statement is valid by compiling it. boolean yieldIfContended () This method has been unified at API 15 level. if the DB is blocked more than once (due to nested transactions), the lock will not give way. Use yieldIfContended Safely instead. boolean yieldIfContendedSafely () Temporarily stop the transaction so that other threads are started. boolean yieldIfContendedSafely (Long SleepAfterYieldDelay) Temporarily finish the transaction to allow other threads to work. From the java.lang.Object Object class, the clone creates and returns a copy of this object. boolean (Object obj) indicates whether any other object is equal to this. invalid completion () Is called by the garbage collector at the facility when the garbage collection determines that there are no more references to the object. The final class of the getClass returns the time class of the subject. int hashCode () Returns the hash code value to the object. the final invalid to notify () will wake up one thread that is waiting on the monitor of this object. ToString returns the view of the object line. The final expectation of emptiness (long time out, int nanos) triggers anticipation of the current thread until another thread triggers the notification method () or the notifyAll method for that object, or some other thread interrupts the current thread, or a certain amount of real time has passed. The final expectation of emptiness (long time) triggers the wait for the current thread until another notification method () or notifyAll method has been triggered for that object, or a certain amount of time has passed. the final expectation of emptiness () causes the current thread to wait until another thread triggers the notification method () or the notifyAll method for that object. From the java.io.Closeable interface, the abstract void closes this thread and releases any system resources associated with it. Public static final int CONFLICT_ABORT When a restriction is breached, ROLLBACK is not performed, so changes from previous commands within the same transaction are saved. This is the default behavior. Permanent value: 2 (0x0000002) public static final int CONFLICT_FAIL When a limit violation occurs, the command is interrupted with the reverse code SQLITE_CONSTRAINT. But any in the database, the team's input before the meeting with a violation of the restriction, are stored and not snable. Permanent value: 3 (0x00000003) (0x0000003) static final CONFLICT_IGNORE When a restriction is breached, one line that contains a violation of the restriction is not inserted or changed. But the team continues to perform normally. Other lines before and after the line that violate the restriction continue to be inserted or updated normally. The error doesn't come back. Permanent value: 4 (0x0000004) public static final int CONFLICT_NONE Use the following when the action of the conflict is not specified. Permanent value: 0 (0x00000000) public static final int CONFLICT_REPLACE When THE UNIQUE limit is breached, pre-existing lines that cause a limitation violation are removed before the current line is inserted or updated. This way, the insertion or update always happens. The team continues to run as normal. The error doesn't come back. If you break the NOT NULL limit, NULL is replaced by the default for that column. If the column doesn't have a default, the ABORT algorithm is used. If you break the CHECK limit, ignore is used. When this conflict resolution strategy removes strings to meet the limit, it does not cause deletion triggers on those lines. This behavior may change in a future release. Permanent value: 5 (0x00000005) public static final int CONFLICT_ROLLBACK When a limit violation occurs, there is an immediate ROLLBACK, thus ending the current transaction, and the team is interrupted by the SQLITE_CONSTRAINT return code. If the transaction is not active (except for the implied transaction created in each team), then this algorithm works in the same way as ABORT. Permanent value: 1 (0x00000001) public static final int MAX_SQL_CACHE_SIZE Absolute maximum value that can be setMaxSqlCacheSize (int). Each extract prepared ranges from 1K to 6K, depending on the complexity of the SCL statement and the scheme. A large S'L cache can use a significant amount of memory. Permanent value: 100 (0x00000064) public static final int NO_LOCALIZED_COLLATORS Open Flag: Flag for openDatabase (File, S'LiteDatabase. OpenParams) to open a database without the support of localized callouters. This results in the localized collaboration not being created. You have to be consistent when using this flag to use the database settings created with. Permanent value: 16 (0x000000010) public static final int OPEN_READWRITE Open Flag: Flag for openDatabase (File, S'LiteDatabase. OpenParams) to open a database for reading and writing. If the drive is full, it may fail even before you actually write anything. attention that the value of this flag is 0, so it's the default. Permanent value: 0 (0x00000000) public static final int SQLITE_MAX_LIKE_PATTERN_LENGTH Maximum Length pattern LIKE or GLOB Pattern Mapping Algorithm used in default implementations LIKE and GLOB S'Lite, can demonstrate performance O (N-2) (N-2) N is the number of pattern symbols) for some pathological cases. To avoid denial-of-service attacks, the length of the LIKE or GLOB pattern is limited to SQLITE_MAX_LIKE_PATTERN_LENGTH bytes. This default limit is 50,000. A modern workstation can estimate even a pathological LIKE or GLOB pattern of 50,000 bytes relatively quickly. The denial of service issue comes into play only when the length of the template hits millions of bytes. However, because the most useful LIKE or GLOB templates are no more than a few dozen bytes, paranoid app developers may want to reduce this setting to something in the range of several hundred if they know that external users are able to generate arbitrary patterns. Permanent value: 50,000 (0x0000c350) public void startTransaction () Starts a transaction in EXCLUSIVE mode. Deals can be made. When the external transaction is completed, all the work done in that transaction and all the transactions invested will be made or rolled back. Changes will be rolled back if any transaction is completed without being marked as clean (by calling setTransactionSuccessful). Otherwise they will be committed. Here's the standard idioms for transactions: db.beginTransaction; Give it a shot... db.setTransactionSuccessful(); After all, db.endTransaction - Public void of the beginningtransctionNonExclusive () Begins the transaction in IMMEDIATE mode. Deals can be made. When the external transaction is completed, all the work done in that transaction and all the transactions invested will be made or rolled back. Changes will be rolled back if any transaction is completed without being marked as clean (by calling setTransactionSuccessful). Otherwise they will be committed. Here is a standard idioms for transactions: db.beginTransactionNonExclusive; Give it a shot... db.setTransactionSuccessful(); After all, db.endTransaction - Public void of the beginningtransactionListener (S'LiteTransactionListener transaction) Starts the transaction in EXCLUSIVE mode. Deals can be made. When the external transaction is completed, all the work done in that transaction and all the transactions invested will be made or rolled back. Changes will be rolled back if any transaction is completed without being marked as clean (by calling setTransactionSuccessful). Otherwise they will be committed. Here is a standard idioms for transactions: db.beginTransactionWithListener (listener); Give it a shot... db.setTransactionSuccessful(); After all, db.endTransaction Listener S'LiteTransactionListener transaction options: The listener who needs to be notified when the transaction starts is made, or rolled back, either explicitly or by calling to yieldIfContendedSafely. public emptiness (S'LiteTransactionListener) Starts a transaction in IMMEDIATE mode. Transactions can be Nested. When the external transaction is completed, all the work done in that transaction and all the transactions invested will be made or rolled back. Changes will be rolled back if any transaction is completed without being marked as clean (by calling setTransactionSuccessful). Otherwise they will be committed. Here is a standard idioms for transactions: db.beginTransactionWithListenerNonExclusive (listener); Give it a shot... db.setTransactionSuccessful(); After all, db.endTransaction Listener S'LiteTransactionListener transaction options: The listener who needs to be notified when the transaction starts is made, or rolled back, either explicitly or by calling to yieldIfContendedSafely. S'LiteDatabase (S'LiteDatabase Factory). CursorFactory) Create a database supported by S'Lite memory. Its contents will be destroyed when the database is closed. Sets the database localized to the current local database of the system. Call setLocale (Locale) if you want something different. S'LiteDatabase factory options. CursorFactory: An additional factory class that is designed to instantly use the cursor when the request is called This value can be zero. Returns S'LiteDatabase copy s'LiteDatabase This value can not be zero. Throws S'LiteException if the database can not be created public static database S'LiteDatabase to createInMemory (S'LiteDatabase. OpenParams openParams) Create a commemorative database supported by S'Lite. Its contents will be destroyed when the database is closed. Sets the database localized to the current local database of the system. Call setLocale (Locale) if you want something different. Options open Params S'LiteDatabase. OpenParams: The configuration options that are used to open S'LiteDatabase This value cannot be zero. Returns S'LiteDatabase copy s'LiteDatabase This value can not be zero. Throws S'LException if the database can not be created a public int remove (Line table, string, whereClause, String, where args) Convenience method to delete strings in the database. Table Line Options: Table to Remove From WhichClause String: Optional Where to Apply When Removing. Passage of zero will remove all lines. whereArgs String: You can include ?s in a position where the values from where the Args will be replaced. Values will be tied as strings. Returns int the number of lines affected if whereClause is transmitted, 0 otherwise. To remove all the lines and get the score to pass 1 as whereClause. Public static boolean deleteDatabase (file file) removes the database, including its log file and other supporting files that may have been created by the database engine. Settings file: The way the database file is published. That's the value could be zero. Returns boo Truelean if the database has been successfully removed. Public boolean enableWriteAheadLogging () This method allows you to run multiple threads in a single database in parallel. It's This. this is by opening multiple connections to the database and using different database connections for each query. The database log mode has also been modified to allow you to write to act simultaneously with readings. When the forward log is not enabled (by default), it is not possible to read and write occur in the database at the same time. Before changing the database, the author implicitly acquires an exclusive database lock that prevents readers from accessing the database until the recording is complete. In contrast, when you turn the forward recording (by calling this method), recording operations take place in a separate log file that allows readings to act simultaneously. While the recordings are underway, readers on other threads will perceive the state of the database as it was before the recording began. When the entries are completed, readers on other threads will perceive the new state of the database. It is a good idea to enable the record to advance registration whenever the database is simultaneously available and modified by multiple threads at the same time. However, the forward log entry uses significantly more memory than conventional logging because there are multiple connections to the same database. So if the database is only used by one thread, or if concurrency optimization is not very important, then the forward log should be disabled. After this method is called, the parallel execution of requests is turned on as long as the database remains open. To disable the query, either call offWriteAheadLogging, or

close the database and reopen it. The maximum number of connections used to run requests side-by-side depends on the device's memory and possibly other properties. If the request is part of the transaction, it is performed in the same database processing as the transaction. Writers must use startTransactionNonExclusive () or start TransactionWithListenerNonExclusive (android.database.sqlite.S'LiteTransactionListener) to start the transaction. The in-exclusive mode allows the database file to be readable by other threads that run requests. If the database has any attached databases, requests cannot be made in parallel. Similarly, a forward log is not supported only for reading databases or memory databases. In such cases, include WriteAheadLogging () returns false. The best way to turn on the record-forward registration is to pass the flag ENABLE_WRITE_AHEAD_LOGGING openDatabase (file picture, S'LiteDatabase. OpenParams). This is more effective than calling enableWriteAheadLogging. S.LiteDabab dB and S'LiteDatabase.openDatabase (db_filename, cursorFactory, SQLiteDatabase.CREATE_IF_NECESSARY SQLiteDatabase.ENABLE_WRITE_AHEAD_LOGGING, Another way to allow you to write a forward log is to call enableWriteAheadLogging after the database is opened. S.LiteDabaab dB dB cursorFactory, SQLiteDatabase.CREATE_IF_NECESSARY, myDatabaseErrorHandler); db.enableWriteAheadLogging(); For more information on how forward writing works, visit S'Lite Write-Ahead Logging. Returns boolean True if the entry forward log is included. Throws IllegalStateException if there are transactions in the process at the time when this method is called. WAL mode can only be changed if there are no transactions. See also: ENABLE_WRITE_AHEAD_LOGGINGdisableWriteAheadLogging () the public invalid end of the Transaction () End of the transaction. See startTransaction for notes on how to use this and when transactions are made and rolled back. public void execPerConnectionS'L (String sql, Object) bindArgs) Follow this statement on all connections to this database. This approval will be immediately performed on all existing connections and will be automatically performed on all future connections. Some examples of using changes such as PRAGMA trusted_schema-OFF or features such as SELECT icu_load_collation. If you run these statements with execS'L (String), they will only apply to one database connection; using this method ensures that they apply evenly to all current and future connections. Sql String Options: S'L Statement to Be Executed. Several statements separated by a comma are not supported. This value cannot be zero. bindArgs Object: Arguments that should be tied to the statement of S'L. This value can be zero. Public void execS'L (String sql) Perform one S'L statement that is not SELECT or any other S'L statement that returns the data. It does not have the means to return any data (such as the number of lines affected). Instead, it is recommended to use the insert (java.lang.String, java.lang.String, android.content.ContentValues), update (java.lang.String, android.content.ContentValues, java.lang.String, java.lang.String) and others, whenever possible. When you use enableWriteAheadLogging, journal_mode is automatically controlled by this class. Thus, don't set journal_mode with a PRAGMA statement journal_mode if your app uses enableWriteAheadLogging () Please note that PRAGMA values that are applied based on connectivity should not be configured using this method; instead, execPerConnectionS'L (String, Object) should be used to ensure that they are evenly applied to all current and future connections. Sql String options: S'L statement to be executed. Several statements separated by a comma are not supported. Throws S'LException if the S'L line is invalid public invalid execS'L (String sql, Object) Follow one S'L statement that is not SELECT/INSERT/UPDATE/DELETE. For INSERT operators, use any of the following statements. For UPDATE operators, use any of the following statements. For DELETE, use any of the following statements. (java.lang.String, java.lang.String, java.lang.String) java.lang.String) java.lang.String) For example, the following good candidates for using this method are: ALTER TABLE CREATE or DROP table/trigger/view/index/virtual table REINDEX RELEASE SAVEPOINT PRAGMA, which does not return data when using enableWriteAheadLogging, journal_mode is automatically managed by this class. Thus, don't set journal_mode with a PRAGMA statement journal_mode if your app uses enableWriteAheadLogging () Please note that PRAGMA values that are applied based on connectivity should not be configured using this method; instead, execPerConnectionS'L (String, Object) should be used to ensure that they are evenly applied to all current and future connections. Sql String options: S'L statement to be executed. Several statements separated by a comma are not supported. bindArgs Object: only byte, String, Long and Double are supported in bindArgs. Throws S'LException if the S'L line is an invalid public static line findEditTable (String Tables) finds the name of the first table that is edited. Table Options Line: Table List Returns a line of the first table of listed public lists of the 'database_list' list of the database. Returns the pair (database name, database file path) or zero if the database is not open. Returns long the new maximum size of the database to the public long getPageSize () Returns the current size of the page database, in bytes. Returns the long size of the database page, in bytes public String getPath () Gets way to the database file. Returns the Path line to the database file. public int getVersion receives a version of the database. Returns int version of the database public boolean inTransaction () Returns correctly if the current thread has a transaction pending. Returns boo-true if the current thread is in a transaction. Public long insertion (table line, nullColumnHack line, ContentValues values) Convenience method for inserting a string into the database. Table Line Options: Table to insert a line into nullColumnHack String: optional; could be zero. S'L does not allow you to insert a completely blank row without naming at least one column name. If the values provided are empty, the column names are not known and an empty line cannot be inserted. If you're not set to zero, nullColumnHack provides a zero-name column name to explicitly insert NULL in case your values are empty. ContentValues: This map contains the initial column values for the line. The keys should be column names and values, The column values returns a long line ID to a newly inserted line, or -1, if there is an error by a public long insertOrThrow (Table Line, nullColumnHack, ContentValues'lt;/String, String, String) A convenient method of inserting a line into a database. Table Line Options: Table to insert a line into nullColumnHack String: optional; could be zero. S'L does not allow you to insert a completely blank row without naming at least one column name. If the values provided are empty, the column names are not known and an empty line cannot be inserted. If you're not set to zero, nullColumnHack provides a zero-name column name to explicitly insert NULL in case your values are empty. ContentValues: This map contains the initial column values for the line. The keys should be column names and values that column values returns to a long line ID of a newly inserted line, or -1, if there is an error, throws android.database.S'LException S'LException public long inserteWithOnConflict (table, line nullColumnHack, ContentValues initialValues, int conflictAlgorithm) Table Line Options: Table to insert a line into nullColumnHack String: optional; could be zero. S'L does not allow you to insert a completely blank row without naming at least one column name. If the initial estimates provided are empty, the names of the columns are not known and the blank row cannot be inserted. If not set at zero, the nullColumnHack option provides a zero name column name explicitly inserted by NULL in case your originalValues is empty. At the beginningvalues ContentValues: this map contains the initial column values for the line. The keys should be the names of the column and the values that the column values conflictedAlgorithm int: to insert conflict resolver Returns long, the line ID of the newly inserted line OR -1, if there was either the conflictAlgorithm entry option CONFLICT_IGNORE or error. public boolean isDatabaseIntegrityOk () Runs a 'pragma integrity_check' on this database (and all attached databases) and returns correctly if the database data (and all attached databases) pass integrity_check, otherwise false. If the result is false, then this method registers errors about the integrity_check command. Please note that integrity_check in the database can take a long time. Returns boolean is true if the database data (and all its attached databases) integrity_check, false otherwise. Public boolean isDbLockedByCurrentThread () Returns correctly if the current stream keeps an active connection to the database. The name of this method comes from a time when having an active connection to the database meant that the thread kept the actual lock in the database. There is no longer a true database lock, although threads can be blocked if they can't get a database connection to perform a specific operation. Returns boo Truelean if The stream is actively connected to the database. Added to API 1, API level Public boolean isDbLockedByOtherThreads () This method has been deprecated in API level 16. Always comes back false. Don't use this method. Always comes back false. There is no longer a concept of blocking the database, so this method always returns false. Public boolean isOpen () Returns correctly if the database is currently open. Returns boo Truelean if the database is currently open (not closed). Public boolean isReadOnly () Returns correctly if the database is only open as read. Returns boo Truelean if the database is only open to reading. Added to api level 1 Deprecated in API level 15 public void markTableSyncable (Line Table, Line deletedTable) This method has been deprecated in API level 15. This method no longer serves any useful purpose and has been decreced. To resize this table as synchronous. When the update occurs in this table, _sync_dirty field will be installed to ensure that the synchronization works properly. Table Line Options: Table for Mark as Synchronized deletedTable String: Remote Table That Corresponds to a Synchronized Table Added to API Level 1 Deprecated in API Level 15 Public Void MarkTableSyncable (Line Table, Line ForeignKey, StringTable Update) This method has been deprecated in API level 15. This method no longer serves any useful purpose and has been decreced. Mark this table as synchronized, while _sync_dirty in another table. When you update this table, you'll _sync_dirty the line box in updateTable with _id in foreignKey to ensure that the synchronization works properly. Table Line Options: An update on this table will synchronize the removal time of foreignKey String: it is a column in the table whose value is _id in the updateTable String: it is a table that will have its _sync_dirty public boolean needUpgrade (int newVersion) Returns true if the new version code is larger than the current version of the database. NewVersion int options: New version code. Returns boo True if the new version code is larger than the current version of the database. Cursor Public Request (boolean distinct, Table Line, Column Line, Line Choice, Row selectionArgs, String GroupBy, Row, Line OrderBy, Line Limit) Request this URL by returning the cursor over the set of results. Options different boolean: true if you want each row to be unique, false otherwise. Table Line: The name of the table to compile the request is against. Column Row: List of columns to return. Passing zero will return all columns, which is not recommended to prevent the reading of data from the store that will not be used. Choice line: A filter announcing which lines to return, formatted as a S'L WHERE clause (except where itself). Walkthrough return all the lines for this table. selectionArgs String: You can include ?s in a selection that will be replaced by values from selectionArgs, in order for them to appear in Values will be tied as strings. groupBy String: A filter announcing how to group lines formatted as the S'L GROUP BY clause (except for GROUP BY). Passage of zero will result in the strings not being grouped. with the line: The filter announces which groups of strings to include in the cursor if you use a grouping of strings formatted as the O'8 HAVING clause (except for HAVING itself). Passage zero will result in the inclusion of all groups of strings and is required when the grouping of strings is not used. orderBy String: How to order lines formatted as the O'8 ORDER BY clause (except order BY itself). Passage of zero will use the default sorting order, which may be disorderly. Line limit: Limits the number of lines returned by the request, formatted as a LIMIT position. Passage zero means no LIMIT position. Returns the Cursor A Cursor, which is located before the first entry. Note that the Cursors are out of sync, see the cursor public query (Table Line, Line, Line Choice, GroupBy Choice, Row, Line, OrderBy Line, Limit Line) Request this table by returning the Cursor over the set of results. Table Line Options: Table Name to compile a request against. Column Row: List of columns to return. Passing zero will return all columns, which is not recommended to prevent the reading of data from the store that will not be used. Choice line: A filter announcing which lines to return, formatted as a S'L WHERE clause (except where itself). Passage of zero will return all lines for this table. selectionArgs String: You can include ?s in a selection that will be replaced by selectionArgs values in order for them to appear in the selection. Values will be tied as strings. groupBy String: A filter announcing how to group lines formatted as the S'L GROUP BY clause (except for GROUP BY). Passage of zero will result in the strings not being grouped. with the line: The filter announces which groups of strings to include in the cursor if you use a grouping of strings formatted as the O'8 HAVING clause (except for HAVING itself). Passage zero will result in the inclusion of all groups of strings and is required when the grouping of strings is not used. orderBy String: How to order lines formatted as the O'8 ORDER BY clause (except order BY itself). Passage of zero will use the default sorting order, which may be disorderly. Line limit: Limits the number of lines returned by the request, formatted as a LIMIT position. Passage zero means no LIMIT position. Returns the Cursor A Cursor, which is located before the first entry. Note that The Cursors are not See Public Request Cursor (boolean separately, Cord Table, Columns, String Choice, String selectionArgs, GroupBy Row, Row having, Line orderBy, String Limit, CancellationSignal CancellationSignal Request this URL by returning the cursor over the set of results. Options different boolean: true if you want each row to be unique, false otherwise. Table Line: The name of the table to compile the request is against. Column Row: List of columns to return. Passing zero will return all columns, which is not recommended to prevent the reading of data from the store that will not be used. Choice line: A filter announcing which lines to return, formatted as a S'L WHERE clause (except where itself). Passage of zero will return all lines for this table. selectionArgs String: You can include ?s in a selection that will be replaced by selectionArgs values in order for them to appear in the selection. Values will be tied as strings. groupBy String: A filter announcing how to group lines formatted as the S'L GROUP BY clause (except for GROUP BY). Passage of zero will result in the strings not being grouped. with the line: The filter announces which groups of strings to include in the cursor if you use a grouping of strings formatted as the O'8 HAVING clause (except for HAVING itself). Passage zero will result in the inclusion of all groups of strings and is required when the grouping of strings is not used. orderBy String: How to order lines formatted as the O'8 ORDER BY clause (except order BY itself). Passage of zero will use the default sorting order, which may be disorderly. Line limit: Limits the number of lines returned by the request, formatted as a LIMIT position. Passage zero means no LIMIT position. cancellations. signal to cancel the operation, or zero if not. If the operation is cancelled, OperationCanceledException will be abandoned when the request is performed. Returns the Cursor A Cursor, which is located before the first entry. Note that the Cursors are out of sync, see the cursor public request (Table Line, Column Line, Line choice, String selectionArgs, GroupBy Line, Row, Line orderBy) Request this table by returning the cursor over the set of results. Table Line Options: Table Name to compile a request against. Column Row: List of columns to return. Passing zero will return all columns, which is not recommended to prevent the reading of data from the store that will not be used. Choice line: A filter announcing which lines to return, formatted as a S'L WHERE clause (except where itself). Passage of zero will return all lines for this table. selectionArgs String: You can include ?s in a selection that will be replaced by selectionArgs values in order for them to appear in the selection. Values will be tied as strings. groupBy String: A filter announcing how to group formatted as the S'L GROUP BY clause (except for group BY itself). Passage of zero will result in the strings not being grouped. Line availability: filter filter which group lines to include in the cursor, if the string grouping is used, are formatted as a S'L HAVING clause (except for the HAVING itself). Passage zero will result in the inclusion of all groups of strings and is required when the grouping of strings is not used. orderBy String: How to order lines formatted as the O'8 ORDER BY clause (except order BY itself). Passage of zero will use the default sorting order, which may be disorderly. Returns the Cursor A Cursor, which is located before the first entry. Please note that the Cursors are out of sync, see the public request for CursorWithFactory (S'LiteDatabase). CursorFactory cursorFactoryFactory, boolean, Row Table, Column Line, Line Choice, String' selectionArgs, GroupBy Row, Row having, Line OrderBy, Line Limit, CancellationSignal cancellationSignal) Request this URL, returning cursor over the result of the set. CursorFactory S'LiteDatabase options. CursorFactory: Plant cursor for use, or zero for plant default different boolean: true if you want each row to be unique, false otherwise. Table Line: The name of the table to compile the request is against. Column Row: List of columns to return. Passing zero will return all columns, which is not recommended to prevent the reading of data from the store that will not be used. Choice line: A filter announcing which lines to return, formatted as a S'L WHERE clause (except where itself). Passage of zero will return all lines for this table. selectionArgs String: You can include ?s in a selection that will be replaced by selectionArgs values in order for them to appear in the selection. Values will be tied as strings. groupBy String: A filter announcing how to group lines formatted as the S'L GROUP BY clause (except for GROUP BY). Passage of zero will result in the strings not being grouped. with the line: The filter announces which groups of strings to include in the cursor if you use a grouping of strings formatted as the O'8 HAVING clause (except for HAVING itself). Passage zero will result in the inclusion of all groups of strings and is required when the grouping of strings is not used. orderBy String: How to order lines formatted as the O'8 ORDER BY clause (except order BY itself). Passage of zero will use the default sorting order, which may be disorderly. Line limit: Limits the number of lines returned by the request, formatted as a LIMIT position. Passage zero means no LIMIT position. cancellations. signal to cancel the operation, or zero if not. If the operation is cancelled, OperationCanceledException will be abandoned when the request is performed. Returns the object A Cursor, which is located before the first entry. Please note that the Cursors are out of sync, see the public request for CursorWithFactory (S'LiteDatabase). CursorFactory) (S'LiteDatabase. CursorFactory) The boolean separately, String Table, String Columns, String Choice, String selectionArgs, GroupBy Row, Row having, Line orderBy, String Limit given URL, returning Cursor over result set. CursorFactory S'LiteDatabase options. CursorFactory: Plant cursor for use, or zero for plant default different boolean: true if you want each row to be unique, false otherwise. Table Line: The name of the table to compile the request is against. Column Row: List of columns to return. Passing zero will return all columns, which is not recommended to prevent the reading of data from the store that will not be used. Choice line: A filter announcing which lines to return, formatted as a S'L WHERE clause (except where itself). Passage of zero will return all lines for this table. selectionArgs String: You can include ?s in a selection that will be replaced by selectionArgs values in order for them to appear in the selection. Values will be tied as strings. groupBy String: A filter announcing how to group lines formatted as the S'L GROUP BY clause (except for GROUP BY). Passage of zero will result in the strings not being grouped. with the line: The filter announces which groups of strings to include in the cursor if you use a grouping of strings formatted as the O'8 HAVING clause (except for HAVING itself). Passage zero will result in the inclusion of all groups of strings and is required when the grouping of strings is not used. orderBy String: How to order lines formatted as the O'8 ORDER BY clause (except order BY itself). Passage of zero will use the default sorting order, which may be disorderly. Line limit: Limits the number of lines returned by the request, formatted as a LIMIT position. Passage zero means no LIMIT position. Returns the Cursor A Cursor, which is located before the first entry. Please note that the Cursors are out of sync, see the public Cursor raw'query (String sql, String' selectionArgs, CancellationSignal cancellationSignal) launches the S'L provided and returns the test cursor to the set of results. Sql Line Options: S'L request. The S'L line should not be; discontinued selectionArgs String: You can turn ?s in where the item in the request will be replaced by values from selectionArgs. Values will be tied as strings. cancellations. signal to cancel the operation, or zero if not. If the operation is cancelled, OperationCanceledException will be abandoned when the request is performed. Returns the Cursor A Cursor, which is located before the first entry. Note that the Cursors are out of sync, see public Cursor raw'query (String sql, String' selectionArgs) launches provided by S'L and returns set of results. Sql Line Options: S'L request. The S'L line should not be; Discontinued Line: You can include ?s in where the item in the request will be replaced by values from selectionArgs. Values will be tied as strings. Returns the Cursor A Cursor, which is located before the first entry. Please note that the Cursors are out of sync, see Public Cursor raw'yWithFactory (S'LiteDatabase. CursorFactory cursorFactory, String sql, String' selectionArgs, String editTable, CancellationSignal cancellationSignal) launches S'L and returns the test rate. Settings of the coursetorActor S'LiteDatabase. CursorFactory: Cursor factory for use, or null for sql String factory by default: S'L request The S'L line should not be; discontinued selectionArgs String: You can turn ?s in where the item in the request will be replaced by values from selectionArgs. Values will be tied as strings. editTable String: the name of the first table that edits the cancellation Of the CancelSigned: Signal to cancel the operation in the process, or zero if not. If the operation is cancelled, OperationCanceledException will be abandoned when the request is performed. Returns the Cursor A Cursor, which is located before the first entry. Please note that the Cursors are out of sync, see the public Cursor raw'yWithFactory (S'LiteDatabase. CursorFactory cursorFactory, String sql, String' selectionArgs, String editTable) launches S'L and returns the cursor over the set of results. Settings of the coursetorActor S'LiteDatabase. CursorFactory: Cursor factory for use, or null for sql String factory by default: S'L request The S'L line should not be; discontinued selectionArgs String: You can turn ?s in where the item in the request will be replaced by values from selectionArgs. Values will be tied as strings. editTable String: The name of the first table that is edited returns to the Cursor Cursor object that is located before the first entry. Note that the Cursors are out of sync, see the public static releaseMemory () Attempts to release the memory that S'Lite holds but does not require normal operation. Usually this memory comes from the cache of the page. Returns int the number of bytes actually released by public long replace (Line Table, Row nullColumnHack, ContentValues initialValues) Convenience method to replace the line in the database. Inserts a new line if the string doesn't exist yet. Table Line Options: A table in which to replace the nullColumnHack String line: optional; could be zero. S'L does not allow you to insert a completely blank row without naming at least one column name. If the initial estimates provided are empty, the names of the columns are not known and the blank row cannot be inserted. If not set at zero, nullColumnHack provides the name of the zero column name Insert NULL in case your initial estimates are empty. At the beginningvalues ContentValues: this map contains the initial column values for the line. The keys should be column names and column values. Returns a long line ID to a newly inserted line, or -1, if there has been a public long replaceOrThrow error (Table Line, NullColumnHack, ContentValues initialValues) Convenience method to replace the line in the database. Inserts a new line if the string doesn't exist yet. Table Line Options: A table in which to replace the nullColumnHack String line: optional; could be zero. S'L does not allow you to insert a completely blank row without naming at least one column name. If the initial estimates provided are empty, the names of the columns are not known and the blank row cannot be inserted. If not set at zero, the nullColumnHack option provides a zero name column name explicitly inserted by NULL in case your originalValues is empty. At the beginningvalues ContentValues: this map contains the initial column values for the line. The keys should be column names and column values. Returns a long line ID to the newly inserted line, or -1, if there is an error, throws android.database.S'LException public blank setCustomAggregateFunction (String functionName, BinaryOperator'lt'lt; aggregateFunction) Register a custom aggregate function that can be called from the expressions of S'L. For example, registering a custom aggregation function called LONGEST can be used in a request, such as SELECT LONGEST (name) from employees. This method follows the reduction thread outlined in Stream.reduce (BinaryOperator), and the custom aggregation function is expected to be the associative accumulation function defined by this class. When you try to register multiple features with the same function name, S'Lite will replace any previously defined features with the latest definition, no matter what type of function they are. S'Lite does not support unregistered features. FunctionName String options: The insensitive name to register this feature under, is limited to 255 UTF-8 bytes in length. This value cannot be zero. aggregateFunction BinaryOperator: A functional interface that will be called when using the function name in the S'L statement. Arguments from the S'L statement are transferred to the functional interface, and the return values from the functional interface are returned back to the SL statement. This value cannot be zero. Throws S'LiteException if the user function cannot be registered. See also: CustomScalarFunction (String, UnaryOperator) public void setCustomScalarFunction (String functionName, UnaryOperator)lt;String'gt; scalarFunction) a user-scale feature that can be removed from S'L expressions. For example, registering a custom scalable feature called REVERSE can be a string-gt; is used in the query as SELECT REVERSE (name) from employees. When you try to register multiple features with the same function name, S'Lite will replace any previously defined features with the latest definition, no matter what type of function they are. S'Lite does not support unregistered features. FunctionName String options: The insensitive name to register this feature under, is limited to 255 UTF-8 bytes in length. This value cannot be zero. scalarFunction UnaryOperator: A functional interface that will be called when using the function name in the S'L statement. Arguments from the S'L statement are transferred to the functional interface, and the return values from the functional interface are returned back to the SL statement. This value cannot be zero. Throws S'LiteException if the user function cannot be registered. See also: The CustomAggregateFunction (String, BinaryOperator) public void setForeignKeyConstraintsEnabled (boolean enable) Determines whether the database includes restrictions on foreign keys. By default, foreign key restrictions do not apply to the database. This method allows the application to include foreign key restrictions. It should be called every time the database is open to ensure that external key limitations for the session are included. A good time to name this method immediately after calling openOrCreateDatabase (file, S'LiteDatabase. CursorFactory) or S'LiteOpenHelper'onConfigure callback. When foreign key restrictions are disabled, the database does not check whether changes to the database will violate the limitations of foreign keys. Similarly, if external key constraints are disabled, the database will not perform cascading removal or update triggers. As a result, the state of the database can become inconsistent. To check the integrity of the database, call DatabaseIntegrityOk. This method should not be called at the time of the transaction. For more information on supporting foreign key restrictions, visit S'Lite Foreign Key Constraints. Options allow boolean: The truth is to include foreign key restrictions, false to disable them. Locale locale sets the locale for this database. It does nothing if this database NO_LOCALIZED_COLLATORS flag set or has been open only to the public. Locale Locale Options: New Locale. Throws S'LException if the locale cannot be installed. The most common reason for this is that there is no collaborator available for the place you request. In this case, the database remains unchanged. Added to API Level 1 Deprecated in API Level 16 Public Void setLockingEnabled (boolean lockingEnabled) This was deprecated in API level 16. This method now does nothing. Don't use it. Check whether the S'LiteData base is safe by locking around critical sections. It's quite expensive, so if you know that your DB will only be used only stream, then you have to set this on false. This is true by default. The Enabled boolean lock options: Set to make sure that locks that are false otherwise public void setMaxSqlCacheSize (int cacheSize) sets the maximum size of the prepared cache statement for this database. (cache size - number of compiled-sql statements stored in the cache). The maximum cache size can only be increased from its current size (default 10). If this method is called with a smaller size than the current maximum value, then IllegalStateException is thrown. This method is safe for threads. Throws IllegalStateException if the introductory cache is MAX_SQL_CACHE_SIZE. The public long set of MaximumSize (long numBytes) sets the maximum size of the database will grow up to. The maximum size cannot be set below the current size. numBytes Long Options: The maximum size of the database in the bytes returns for a long time the new maximum size of the database of public void setPageSize (long numBytes) Sets the size of the database page. The page size should be two in force. This method does not work if any data has been recorded in the database file and should be called immediately after the database has been created. NumBytes options are long: the size of the database page, in bytes the public blank setTransactionSuccessful () marks the current transaction as successful. Don't do any more database work between calling this and calling endTransaction. Do as little work without databases as possible in this situation too. If any errors collide between this and the end of the transaction, the transaction will still be made. IllegalStateException is cast if the current thread is not in the transaction or the transaction is already marked as successful. public void setVersion (int version) sets the version of the database. Int Options: The new version of the public toString database returns the view of the object line. Typically, the toString method returns the line that textually represents that object. The result should be a short but informative presentation that is easy for a person to read. It is recommended that all subclasses override this method. The toString method for the class object returns a line consisting of the class name in which the object is a copy, a symbol on the 'q' sign, and an unsigned six-social representation of the object's hash code. In other words, this method returns a line equal to the value: getClass (.). Settings table line: ContentValues update table: map from column names to new column values. null is valid which will be transferred to NULL. WhereClause String: an optional WHERE item to apply when updated. Passage of zero zero update all the lines. whereArgs String: You can include ?s in a position where the values from where the Args will be replaced. Values will be tied as strings. Returns int the number of lines affected by the public int updateWithOnConflict (Line Table, ContentValues Values, Row, WhereClause, String, WhereArgs, Int conflictAlgorithm) Convenience method to update lines in the database. Settings table line: ContentValues update table: map from column names to new column values. null is a valid value to be translated into NULL. WhereClause String: an optional WHERE item to apply when updated. Passage zero will update all lines. whereArgs String: You can include ?s in a position where the values from where the Args will be replaced. Values will be tied as strings. conflictAlgorithm int: To update conflict resolver Returns int, the number of lines affected by public invalidation checkSql (String sql, CancellationSignal cancellationSignal) checks that the S'L SELECT statement is valid when compiling it. If the S'L statement is not valid, this method will throw S'LiteException. Sql String: S'L to test This value can't be zero. cancellations. signal to cancel the operation, or zero if not. If the operation is cancelled, OperationCanceledException will be abandoned when the request is performed. This value can be zero. Throws S'LiteException if the sql is invalid Added to API level 1 Deprecated in API level 15 public boolean yieldIfContended () This method has been led away in API level 15. if the DB is blocked more than once (due to nested transactions), the lock will not give way. Use yieldIfContended Safely instead. Temporary end of transaction to allow other threads to work. The deal is expected to be a success for the time being. Don't call setTransactionSuccessful before calling it. When this returns, a new transaction will be created but not marked as successful. Returns boolean correctly if the transaction was received by public boolean yieldIfContended Safe () Temporarily finish the transaction to allow other threads to work. The deal is expected to be a success for the time being. Don't call setTransactionSuccessful before calling it. When this returns, a new transaction will be created but not marked as successful. This assumes that there are no nested transactions (startTransaction has been called only once) and will make an exception if this is not the case. Returns boolean correctly if the transaction was received by a public boolean yieldIfContended Safe (Long SleepAfterYieldDelay) to temporarily finish the transaction to allow other threads to work. The deal is expected to be a success for the time being. Don't call setTransactionSuccessful before calling When this returns, a new transaction will be created but not marked as successful. This suggests that there are no nested (beginTransaction has been called only once) and will make an exception if it is not. The sleepAfterYieldDelay options are long: if you're 0, sleep so long before you start a new transaction, if the lock was actually given. This will allow other background threads to make more progress than if we started the transaction immediately. Returns boolean is true if the transaction has been obtained protected void completion () Called by the garbage collector at the facility when the garbage collection determines that there are no more references to the object. Subclass redefines completion method to remove system resources or perform other cleaning. The general contract completion is that it is called if and when Java™ the virtual machine has determined that there are no more means by which this object can be accessed on any thread that is not yet dead, except as a result of actions taken by completing some other objects or classes that are ready to be completed. The completion method can take any action, including ensuring that other threads are available again; the usual goal of completion, however, is to perform clean-up actions before the object is irretrievably discarded. For example, a completion method for an object representing an I/O connection may perform explicit I/O transactions to break the connection before the object is permanently removed. The Object class completion method does not perform any special actions; it just comes back normally. Object subclasses can override this definition. The Java programming language does not guarantee which thread will trigger the completion method for a given object. However, it is guaranteed that the thread that causes the completion will not have any user-visible synchronization locks when the completion call is called. If an untrained exception is abandoned by completion method, the exception is ignored and the end of that object is completed. Once the completion method has been called for the object, no further action will be taken until the Java virtual machine is again determined that there are no more means by which the object can be accessed by any thread that has not yet died, including the possible actions of other objects or classes that are ready to be completed, after which the object can be discarded. The completion method is never called more than once by a Java virtual machine for any given object. Any exception thrown by the completion method stops the completion of the facility, but is otherwise ignored. Ignore.