|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SQL functions for Spatial-MetaData and Spatial-Index handling** | | | | |
| **Function** | **Syntax** | **OGC defined** | **required module** | **Summary** |
| **InitSpatialMetaData** | InitSpatialMetaData( void ) : *Integer*  InitSpatialMetaData( transaction *Integer* ) : *Integer*  InitSpatialMetaData( mode *String* ) : *Integer*  InitSpatialMetaData( transaction *Integer* , mode *String* ) : *Integer* |  | base | Creates the **geometry\_columns** and **spatial\_ref\_sys** metadata tables the return type is Integer, with a return value of 1 for TRUE or 0 for FALSE   * if the *optional* argument **transaction** is set to **TRUE** the whole operation will be handled as a single *Transaction* (faster): the default setting is **transaction=FALSE** (slower, but safer). * if the *optional* argument **mode** is not specified then all possible ESPG SRID definitions will be inserted into the **spatial\_ref\_sys** table. * if the **mode** arg **'WGS84'** (alias **'WGS84\_ONLY'**) is specified, then only WGS84-related EPSG SRIDs will be inserted * if the **mode** arg **'NONE'** (alias **'EMPTY'**) is specified, no EPSG SRID will be inserted at all |
| **InitAdvancedMetaData** | InitAdvancedMetaData( void ) : *Integer*  InitAdvancedMetaData( transaction *Integer* ) : *Integer* |  | base | This one simply is an utility function intended to create several ancillary metadata tables required by **libspatialite v.5** and subsequent versions. the return type is Integer, with a return value of 1 for TRUE or 0 for FALSE  **Note**: will be automatically called by both **InitSpatialMetaData()** or **InitSpatialMetaDataFull()**, but could be usefull for recovering old databases created by earlier versions; it's absolutely harmless because any existing MetaTable will be left untouched. |
| **InitSpatialMetaDataFull** | InitSpatialMetaDataFull( void ) : *Integer*  InitSpatialMetaDataFull( transaction *Integer* ) : *Integer*  InitSpatialMetaDataFull( mode *String* ) : *Integer*  InitSpatialMetaDataFull( transaction *Integer* , mode *String* ) : *Integer* |  | base | This one simply is a *convenience function* accepting the same arguments of **InitSpatialMetaData()** (with identical meaning). The intended scope is to fully initialize all *metadata tables* required by *libspatialite 5.0.0* (and following versions). This function will internally call in a single pass:   * *InitSpatialMetaData()* * *InitAdvancedMetaData()* * *CreateRasterCoveragesTable()* * *CreateVectorCoveragesTables()* * *CreateStylingTables()*   the return type is Integer, with a return value of 1 for TRUE or 0 for FALSE |
| **CreateMissingSystemTables** | CreateMissingSystemTables( void ) : *Integer*  CreateMissingSystemTables( relaxed *Integer* ) : *Integer*  CreateMissingSystemTables( relaxed *Integer* , transaction *Integer* ) : *Integer* |  | base | This function will create any missing ancillary metadata table required by **libspatialite v.5** and subsequent versions.   * the *optional* argument **relaxed** has the same interpretation as in **CreateStylingTables()**: if it is explicitly set as **TRUE** then a *relaxed* version of the validating Triggers will be installed (not checking for strict *XSD schema validation*). * if the *optional* argument **transaction** is explicitly set as **TRUE** then the whole operation will be atomically confined within a monolithic SQL transaction   The return type is Integer, with a return value of 1 on success. An exception will be raised on invalid arguments or on failure.  **Note**: this SQL function is intended for safely upgrading old databases created by earlier versions; it's absolutely harmless because any existing MetaTable will be left untouched. |
| **InsertEpsgSrid** | InsertEpsgSrid( srid *Integer* ) : *Integer* |  | base | Attempts to insert into **spatial\_ref\_sys** the EPSG definition uniquely identified by **srid** [the corresponding EPSG SRID definition will be copied from the inlined dataset defined in **libspatialite**]  the return type is Integer, with a return value of 1 for success or 0 for failure |
| **AddGeometryColumn** | AddGeometryColumn( table *String* , column *String* , srid *Integer* , geom\_type *String* [ , dimension *String* [ , not\_null *Integer* ] ] ) : *Integer* | X | base | Creates a new geometry column updating the Spatial Metadata tables and creating any required trigger in order to enforce constraints  **geom\_type** has to be one of the followings:   * 'POINT', 'POINTZ', 'POINTM', 'POINTZM' * 'LINESTRING', 'LINESTRINGZ', 'LINESTRINGM', 'LINESTRINGZM' * 'POLYGON', 'POLYGONZ', 'POLYGONM', 'POLYGONZM' * 'MULTIPOINT', 'MULTIPOINTZ', 'MULTIPOINTM', 'MULTIPOINTZM' * 'MULTILINESTRING', 'MULTILINESTRINGZ', 'MULTILINESTRINGM', 'MULTILINESTRINGZM' * 'MULTIPOLYGON', 'MULTIPOLYGONZ', 'MULTIPOLYGONM', 'MULTIPOLYGONZM' * 'GEOMETRYCOLLECTION', 'GEOMETRYCOLLECTIONZ', 'GEOMETRYCOLLECTIONZM', 'GEOMETRYCOLLECTIONZM' * 'GEOMETRY', 'GEOMETRYZ', 'GEOMETRYM', 'GEOMETRYZM'   **dimension** has to be one of the followings:   * **'XY'** or **2**: 2D points, identified by **X** and **Y** coordinates * **'XYM'**: 2D points, identified by **X** and **Y** coordinates. Each point stores an **M**-value (*measure*) as well * **'XYZ'** or **3**: 3D points, identified by **X**, **Y** and **Z** coordinates * **'XYZM'** or **4**: 3D points, identified by **X**, **Y** and **Z** coordinates. Each point stores an **M**-value (*measure*) as well   Please note: the **dimension** argument is optional; when specified, if specified, iy's expected to be consistent with the declared **geom\_type**  the return type is Integer, with a return value of 1 for TRUE or 0 for FALSE  the optional 6th arg [**not\_null**] is a non-standard extension required by the peculiar SQLite arch:   * if set to **0** [*false*], then the Geometry column will accept NULL values as well. This is the default behaviour * if set to any **<> 0 value** [*true*], then the Geometry will be defined using a **NOT NULL** clause |
| **AddTemporaryGeometryColumn** | AddTemporaryGeometryColumn( db-prefix *String* , table *String* , column *String* , srid *Integer* , geom\_type *String* [ , dimension *String* [ , not\_null *Integer* ] ] ) : *Integer* |  | base | Almost the same as **AddGeometryColumn()**, with a critical difference:   * **db-prefix** is the *schema-name* of some **attached database** * Such an *Attached Database* must necessarily be of the **:memory:** type being previously created by issuing an SQL statement like: **ATTACH DATABASE ':memory:' AS prefix;** * If the above prerequisite is not fullfilled this function will always fail. |
| **RecoverGeometryColumn** | RecoverGeometryColumn( table *String* , column *String* , srid *Integer* , geom\_type *String* [ , dimension *Integer* ] ) : *Integer* |  | base | Validates an existing ordinary column in order to possibly transform it in a real geometry column, thus updating the Spatial Metadata tables and creating any required trigger in order to enforce constraints  the return type is Integer, with a return value of 1 for TRUE or 0 for FALSE |
| **DiscardGeometryColumn** | DiscardGeometryColumn( table *String* , column *String* ) : *Integer* |  | base | Removes a geometry column from Spatial MetaData tables and drops any related trigger the column itself still continues to exist untouched as an ordinary, unconstrained column  the return type is Integer, with a return value of 1 for TRUE or 0 for FALSE |
| **RegisterVirtualGeometry** | RegisterVirtualGeometry( table *String* ) : *Integer* |  | base | Registers a VirtualShape or VirtualGeoJSON table into the Spatial MetaData tables; the VirtualShape table should be previously created by invoking **CREATE VIRTUAL TABLE ... USING VirtualShape(...)**  the return type is Integer, with a return value of 1 for TRUE or 0 for FALSE |
| **DropVirtualGeometry** | DropVirtualGeometry( table *String* ) : *Integer* |  | base | Removes a VirtualShape or VirtualGeoJSONtable from the Spatial MetaData tables, dropping the VirtualTable table as well.  the return type is Integer, with a return value of 1 for TRUE or 0 for FALSE |
| **CreateSpatialIndex** | CreateSpatialIndex( table *String* , column *String* ) : *Integer* |  | base | Builds an RTree **Spatial Index** on a geometry column, creating any required trigger required in order to enforce full data coherency between the main table and Spatial Index  the return type is Integer, with a return value of 1 for TRUE or 0 for FALSE  -1 will be returned if any physical column named "rowid" (caseless) shadowing the real ROWID is detected. |
| **CreateTemporarySpatialIndex** | CreateTemporarySpatialIndex( db-prefix *String* , table *String* , column *String* ) : *Integer* |  | base | Almost the same as **CreateSpatialIndex()**, but specifically intended to support Geometry columns created by **AddTemporaryGeometryColumn()**   * **db-prefix** is the *schema-name* of some **attached database** * Such an *Attached Database* must necessarily be of the **:memory:** type being previously created by issuing an SQL statement like: **ATTACH DATABASE ':memory:' AS prefix;** * **table** and **column** must identify an already existing Temporary Geometry located on that Database. * If the above prerequisites are not fullfilled this function will always fail. |
| **CreateMbrCache** | CreateMbrCache( table *String* , column *String* ) : *Integer* |  | base | Builds an **MbrCache** on a geometry column, creating any required trigger required in order to enforce full data coherency between the main table and the MbrCache  the return type is Integer, with a return value of 1 for TRUE or 0 for FALSE |
| **DisableSpatialIndex** | DisableSpatialIndex( table *String* , column *String* ) : *Integer* |  | base | Disables an RTree **Spatial Index** or **MbrCache**, removing any related trigger  the return type is Integer, with a return value of 1 for TRUE or 0 for FALSE |
| **CheckShadowedRowid** | CheckShadowedRowid( table *String* ) : *Integer* |  | base | Checks if some table has a physical column named "rowid" (caseless) shadowing the real ROWID.  the return type is Integer, with a return value of 1 for TRUE or 0 for FALSE NULL will be returned if the requested table doesn't exist. |
| **CheckWithoutRowid** | CheckWithoutRowid( table *String* ) : *Integer* |  | base | Checks if some table was created by specifying a **WITHOUT ROWID** clause.  the return type is Integer, with a return value of 1 for TRUE or 0 for FALSE NULL will be returned if the requested table doesn't exist. |
| **CheckSpatialIndex** | CheckSpatialIndex( void ) : *Integer*  CheckSpatialIndex( table *String* , column *String* ) : *Integer* |  | base | Checks an RTree **Spatial Index** for validity and consistency   * if no arguments are passed, then any RTree defined into **geometry\_columns** will be checked * otherwise only the RTree corresponding to **table** and **column** will be checked   the return type is Integer, with a return value of 1 for TRUE or 0 for FALSE NULL will be returned if the requested RTree doesn't exist  -1 will be returned if any physical column named "rowid" (caseless) shadowing the real ROWID is detected. |
| **RecoverSpatialIndex** | RecoverSpatialIndex( [ no\_check : *Integer* ] ) : *Integer*  RecoverSpatialIndex( table *String* , column *String* [ , no\_check : *Integer* ] ) : *Integer* |  | base | Recovers a (*possibly broken*) RTree **Spatial Index**   * if no arguments are passed, then any RTree defined into **geometry\_columns** will be recovered * otherwise only the RTree corresponding to **table** and **column** will be recovered * the *optional* argument **no\_check** will be interpreted as follows:   + if **no\_check = FALSE** (*default*) the RTree will be checked first: and only an invalid RTree will be then actually rebuilt   + if **no\_check = TRUE** the RTree will be unconditionally rebuilt from scratch   the return type is Integer, with a return value of 1 for TRUE or 0 for FALSE NULL will be returned if the requested RTree doesn't exist  -1 will be returned if any physical column named "rowid" (caseless) shadowing the real ROWID is detected. |
| **GetSpatialIndexExtent** | GetSpatialIndexExtent( db\_prefix *String* , table *String* , column *String* ) : *Geometry* |  | base | Retrieves the Full Extent from an RTree **Spatial Index** supporting a **SpatialTable**/**SpatialView** or a **GeoPackage-Geometry**.   * The **db\_prefix** argument specifies which one of the ATTACHED databases is expected to contain the RTree; if **NULL** then the **MAIN** database will be assumed by default. * The **table** and **column** arguments specifying which Table and Geometry is intended.   returns a Rectangle (Polygon Geometry). The Polygon is defined by the corner points of the RTree's Full Extent [(MINX, MINY),(MAXX, MINY), (MAXX, MAXY), (MINX, MAXY), (MINX, MINY)]. **NULL** will be returned on invalid arguments, or on a not existing Table/Geometry, or if no coresponding Spatial Index exists.  Plase note: retrieving the Full Extenxt from an RTree is a very quick operation even when performed on a Table/Geometry containing many million rows. Also: The returned Geometry will be a Spatialite-Binary, even when a GeoPackage-Table/Geometry has been requested. |
| **InvalidateLayerStatistics** | InvalidateLayerStatistics( [ void ) : *Integer*  InvalidateLayerStatistics( table *String* [ , column *String* ] ) : *Integer* |  | base | Immediately and unconditionally invalidates the internal Layer Statistics   * if no arguments are passed, then internal statistics will be invalidated for any possible Geometry Column defined in the current DB * otherwise statistics will be updated only for Geometry Columns corresponding to the given table   the return type is Integer, with a return value of 1 for TRUE or 0 for FALSE  Please note: will effectively work only on behalf DB-files supporting the more recent metatables layout introduced starting since version **4.x**; in any other case will always return an error and no action will happen. |
| **UpdateLayerStatistics** | UpdateLayerStatistics( void ) : *Integer*  UpdateLayerStatistics( table *String* [ , column *String* ] ) : *Integer* |  | base | Updates the internal Layer Statistics [Feature Count and Total Extent]   * if no arguments are passed, then internal statistics will be updated for any possible Geometry Column defined in the current DB * otherwise statistics will be updated only for Geometry Columns corresponding to the given table * When called from inside a GeoPackage Database, the **gpkg\_contents** entry will be updated with the extent information.   the return type is Integer, with a return value of 1 for TRUE or 0 for FALSE |
| **GetLayerExtent** | GetLayerExtent( table *String* [ , column *String* [ , mode *Boolean*] ] ) : *Geometry* |  | base | Return the Envelope corresponding to the Total Extent (*bounding box*] of some Layer; if the Table/Layer only contains a single Geometry column passing the column name isn't strictly required. The returned extent will be retrieved from the Statistics tables:   * if the third argument **mode** is set to TRUE a **PESSIMISTIC** strategy will be applied, i.e. an attempt will be made in order to update the Statistics tables before returning the Envelope. * otherwise the returned Envelope will simply reflect the current values stored into the Statics tables as they are (**OPTIMISTIC** strategy, adopted by default).   NULL will be returned if any error occurs or if the required table isn't a Layer. |
| **CreateRasterCoveragesTable** | CreateRasterCoveragesTable( *void* ) : *Integer* |  | base | Creates the **raster\_coverages** table required by **RasterLite-2**  the return type is Integer, with a return value of 1 for TRUE (success) or 0 for FALSE (failure) |
| **ReCreateRasterCoveragesTriggers** | ReCreateRasterCoveragesTriggers( *void* ) : *Integer* |  | base | (Re)Creates all Triggers supporting the **raster\_coverages** table required by **RasterLite-2**  the return type is Integer, with a return value of 1 for TRUE (success) or 0 for FALSE (failure) |
| **CreateVectorCoveragesTables** | CreateVectorCoveragesTables( *void* ) : *Integer* |  | base | Creates the **vector\_coverages** and **vector\_coverages\_srid** tables required by **RasterLite-2**  the return type is Integer, with a return value of 1 for TRUE (success) or 0 for FALSE (failure) |
| **ReCreateVectorCoveragesTriggers** | ReCreateVectorCoveragesTriggers( *void* ) : *Integer* |  | base | (Re)Creates all Triggers supporting the **vector\_coverages** table required by **RasterLite-2**  the return type is Integer, with a return value of 1 for TRUE (success) or 0 for FALSE (failure) |
| **RebuildGeometryTriggers** | RebuildGeometryTriggers( table\_name *String* , geometry\_column\_name *String* ) : *integer* |  | base | This function will reinstall all geometry-related Triggers for the named table.  the return type is Integer, with a return value of 1 for TRUE (success) or 0 for FALSE (failure) |
| **UpgradeGeometryTriggers** | UpgradeGeometryTriggers( transaction *Integer* ) : *integer* |  | base | This function will upgrade all geometry-related Triggers to the latest version (all DB tables declaring at least one Geometry will be affected by the upgrade). If the **transaction** argument is set to TRUE then the whole upgrade will be safely executed within an internally defined SQL transaction.  Please note: DB-files created using obsolete versions of SpatiaLite (< 4.0.0) will not be upgraded.  the return type is Integer, with a return value of 1 for TRUE (success) or 0 for FALSE (failure) |
| **SQL functions supporting the MetaCatalog and related Statistics** | | | | |
| **Function** | **Syntax** | **OGC defined** | **required module** | **Summary** |
| **CreateMetaCatalogTables** | CreateMetaCatalogTables( transaction *Integer* ) : *Integer* |  | base | Creates both **splite\_metacatalog** and **splite\_metacatalog\_statistics** tables; *splite\_metacatalog* will be populated so to describe every table/column currently defined within the DB. if the first argument **transaction** is set to **TRUE** the whole operation will be handled as a single *Transaction* (faster): the default setting is **transaction=FALSE** (slower, but safer). the return type is Integer, with a return value of 1 for TRUE or 0 for FALSE |
| **UpdateMetaCatalogStatistics** | UpdateMetaCatalogStatistics( transaction *Integer* , table\_name *String* , column\_name *String* ) : *Integer*  UpdateMetaCatalogStatistics( transaction *Integer* , master\_table *String* , table\_name *String* , column\_name *String* ) : *Integer* |  | base | Updates the **splite\_metacatalog\_statistics** table by computing the statistic summary for the required table/column. if the first argument **transaction** is set to **TRUE** the whole operation will be handled as a single *Transaction* (faster): the default setting is **transaction=FALSE** (slower, but safer).  the first form (using *three* arguments) will simply attempt to update the statistic summary for a single table/column as identified by their names: a matching row is expected to be found in *splite\_metacatalog*.  the second form (using *four* arguments) allows to update the statistic summary for many table/columns in a single pass. in this case *master\_table* should identify an existing table: *table\_name* and *column\_name* should identify two columns belonging to this tables. the statistic summary for every table/columns fetched from the master table will then be updated: any mismatching table/column will be simply ignored.  the return type is Integer, with a return value of 1 for TRUE or 0 for FALSE |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SQL functions supporting ISO Metadata** | | | | |
| **Function** | **Syntax** | **OGC defined** | **required module** | **Summary** |
| **CreateIsoMetadataTables** | CreateIsoMetadataTables() : *Integer*  CreateIsoMetadataTables( relaxed *Integer* ) : *Integer* |  | libxml2 | Creates a set of tables supporting **ISO Metadata**.   * if the *optional* argument **relaxed** is specified (any value), then a *relaxed* version of the validating Triggers will be installed.   the return type is Integer, with a return value of 1 for TRUE (success) or 0 for FALSE (failure): -1 will be returned on invalid arguments. |
| **RegisterIsoMetadata** | RegisterIsoMetadata( scope *String* , metadata *BLOB* ) : *Integer*  RegisterIsoMetadata( scope *String* , metadata *BLOB* , id Integer ) : *Integer*  RegisterIsoMetadata( scope *String* , metadata *BLOB* , fileIdentifier *String* ) : *Integer* |  | libxml2 | Inserts (or updates) an **ISO Metadata** definition.   * **scope** can be one of *undefined*, *fieldSession*, *collectionSession*, *series*, *dataset*, *featureType*, *feature*, *attributeType*, *attribute*, *tile*, *model*, *catalogue*, *schema*, *taxonomy*, *software*, *service*, *collectionHardware*, *nonGeographicDataset*, *dimensionGroup*. * **metadata** is expected to be a valid XmlBLOB storing some IsoMetadata payload. * the first form (two arguments only) always performs an *INSERT*; if one the optional arguments **id** or **fileIdentifier** an *UPDATE* could be eventually performed if a corresponding metadata row is already defined.   the return type is Integer, with a return value of 1 for TRUE (success) or 0 for FALSE (failure): -1 will be returned on invalid arguments. |
| **GetIsoMetadataId** | GetIsoMetadataId( fileIdentifier *String* ) : *Integer* |  | libxml2 | Return the unique **id** corresponding to the **ISO Metadata** definition identified by **fileIdentifier**.  If no corresponding ISO Metadata definition exists, this function will always return *ZERO*; *-1* will be returned for invalid arguments. |