INSTITUTO FEDERAL DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA

Introdução à Programação Android

Prof. Samir Bonho

Aula 6

Florianópolis, 02 de Abril de 2014

Sumário

- ✓ Persistência de Dados: Banco de dados
- ✓ SQLite

- ✓ Fontes:
  - ✓ Android Developers. Disponível em http://developer.android.com/

Banco de dados

Banco de Dados

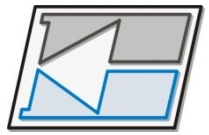
- Consiste em tabelas onde dados podem inseridos, modificados ou excluídos
- De forma simplória uma tabela pode ser vista como uma planilha do Excel
  - As colunas da planilha equivalem aos campos da tabela
  - As linhas da planilha equivalem aos registros da tabela
- Esta analogia acaba quando consideram-se índices e relações entre registros.
  - Não é o escopo do curso
- No Android, é utilizado SQLite para gerenciar as tabelas de bancos de dados
  - www.sqlite.org

SQL - Structured Query Language

- Linguagem padrão para consulta a bancos de dados
- Construções básicas
  - `SELECT Campo1, Campo2, ... FROM Tabela1, Tabela2 WHERE <condição> ORDERBY CampoX;`
  - `INSERT INTO Tabela(Campo1, Campo2, ...) VALUES (Valor1, Valor2, ...);`
  - `DELETE FROM Tabela WHERE <condição>;`
  - `UPDATE Tabela SET Campo1 = <expressao1>, Campo2 = <expressao2> WHERE <condição>;`
- Criação e exclusão de tabelas []=opcional
  - ```
    CREATE TABLE Tabela (
          id INTEGER PRIMARY KEY AUTOINCREMENT,
          Campo1 Tipo1 [[UNIQUE] NOT NULL],
          Campo2 Tipo2 [NOT] NULL,
          ...);
    ```
  - `DROP TABLE IF EXISTS Tabela`
    ```

SQL

- Exemplos de <expressão>
  - Preço – Taxas
  - LENGTH(Tabela.Nome)
  - Tabela.Idade IS [NOT] NULL
- Exemplos de <condição>
  - (Nome = “Maria” OR Nome = “João”) AND Idade <= 40 AND Idade >= 18
  - LENGTH(Nome) > 4
  - Preço – Taxas > 10.0
- Tipos
  - INTEGER, REAL, TEXT



Criação do banco de dados

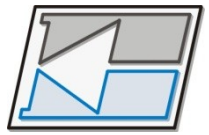
- Usando SQL via linha de comando
  - Programa na pasta do SDK do Android
    - <pasta_android>\sdk\tools\sqlite3.exe

- Nos 2 últimos, após criar, é preciso mover o arquivo para a pasta /data/data/<pacote>/databases/{nome do banco}
  - No eclipse (com emulador aberto):
    - Window > Show view > Other... > FileExplorer

Classe SQLiteOpenHelper

- Auxilia abertura e criação de um banco de dados

SQLiteOpenHelper(Context, String name, SQLiteDatabase.CursorFactory, int version)	Cria um objeto para auxiliar no gerenciamento da base de dados.
SQLiteDatabase getReadableDatabase()	Cria ou abre um banco de dados apenas para leitura.
SQLiteDatabase getWritableDatabase()	Cria ou abre um banco de dados para leitura e escrita.
void onCreate(SQLiteDatabase db)	Chamado quando o banco de dados precisa ser criado, ou seja, não existe.
void onOpen(SQLiteDatabase db)	Chamado quando o banco de dados é aberto.
void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)	Chamado quando a versão do banco de dados sendo aberto é diferente da versão existente.



Classe SQLiteDatabase

- Representa o banco de dados e executa operações de consulta, inclusão, alteração e exclusão de registros

static SQLiteDatabase openDatabase(String path, CursorFactory factory, flags)	Abre banco de dados de acordo com os flags: OPEN_READWRITE, OPEN_READONLY, CREATE_IF_NECESSARY, NO_LOCALIZED_COLLATORS.
static SQLiteDatabase openOrCreateDatabase(String path, CursorFactory factory) static SQLiteDatabase openOrCreateDatabase(File file, CursorFactory factory)	Abre ou cria banco de dados. Equivalente a usar openDatabase(…) com flags = CREATE_IF_NECESSARY

boolean	isOpen()	Verifica se está aberto
void	close()	Fecha banco de dados
void	execSQL(String sql)	Executa script SQL que não seja SELECT. Exemplo: CREATE TABLE, INSERT, UPDATE, etc.

Classe SQLiteDatabase

Cursor query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy) Cursor query(table, columns, selection, selectionArgs, groupBy, having, orderBy, limit) Cursor query(boolean distinct, table, columns, selection, selectionArgs, groupBy, having, orderBy, String limit)	Mostra e executa um SQL de consulta na forma: SELECT <distinct> <columns> FROM <table> WHERE <selection+selectionArgs> GROUP BY <groupBy> HAVING <having> ORDER BY <orderBy> LIMIT <limit>
long insert(table, null, ContentValues values)	Insere um registro e retorna o id. INSERT INTO <table> (values) VALUES (values)
int update(table, ContentValues values, whereClause, whereArgs)	Altera registro(s) e retorna quantidade de linhas modificadas. UPDATE <table> SET <values> WHERE <whereClause+whereArgs>
int delete(table, whereClause, whereArgs)	Deleta registro(s) e retorna quantidade de linhas modificadas. DELETE FROM <table> WHERE <whereClause+whereArgs>

Exercícios