Android Application Design

Software System Design

Zhu Hongjun

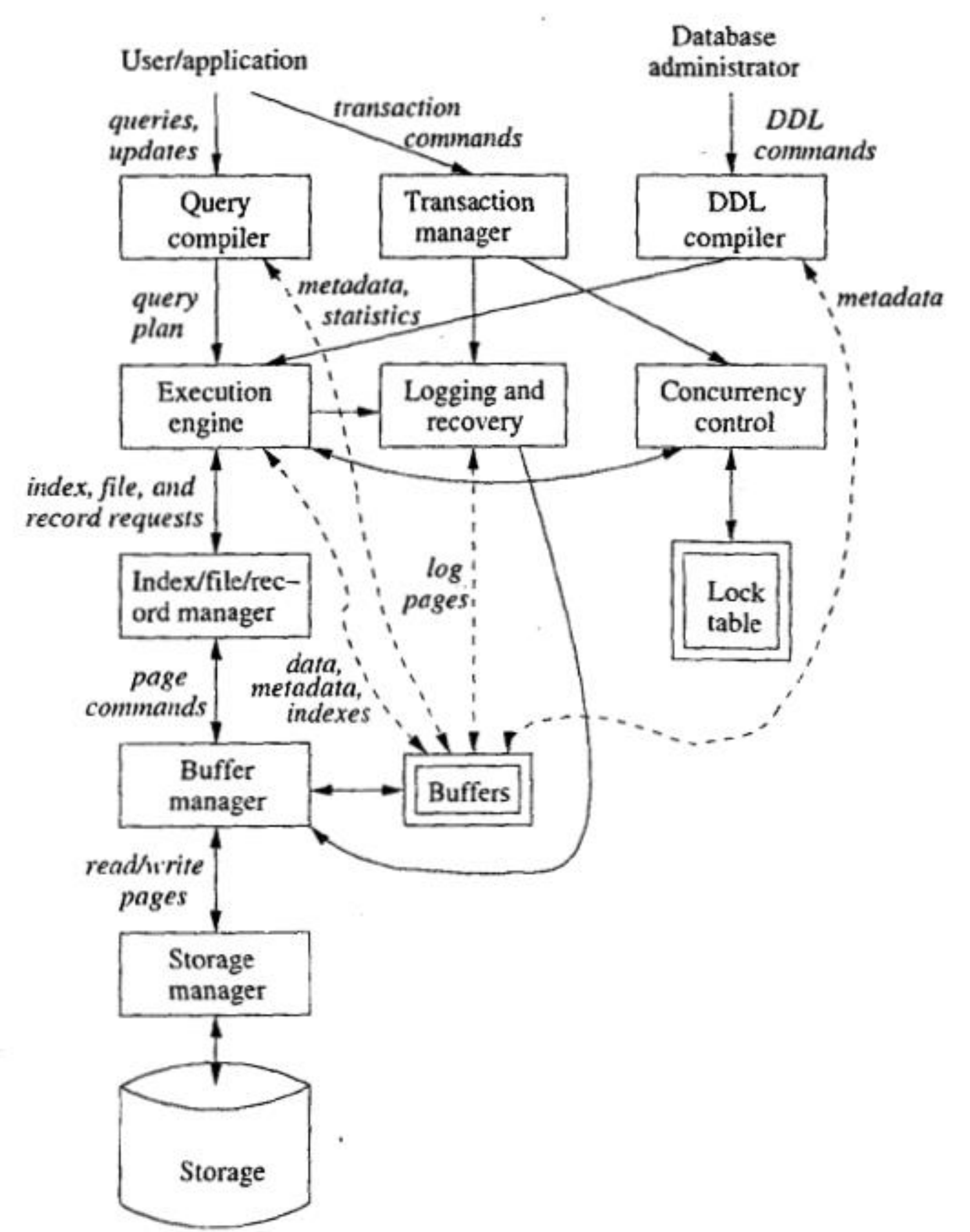
Session 4: Design of Database

- Basics
- E-R Modeling
- Relational Data Modeling
- SQLite and ContentProvider Programming
- Conclusions

Basics

- Database
  - used to maintain internal records, to present data to customers and clients on the Mbrld-Wide- Web, and to support many other commercial processes
- DBMS
  - a powerful tool for creating and managing large amounts of data efficiently and allowing it to persist over long periods of time, safely

DBMS components



Basics

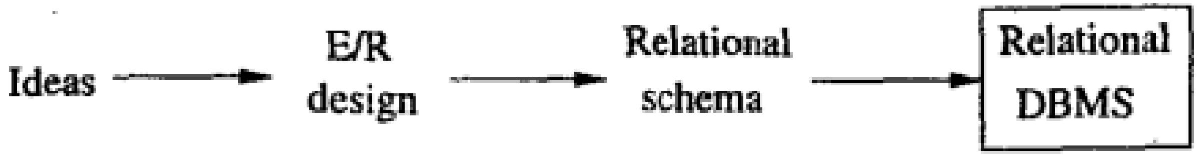
- Design of database
    - the process of producing a detailed data model of a database
    - also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the database management system (DBMS)

Basics

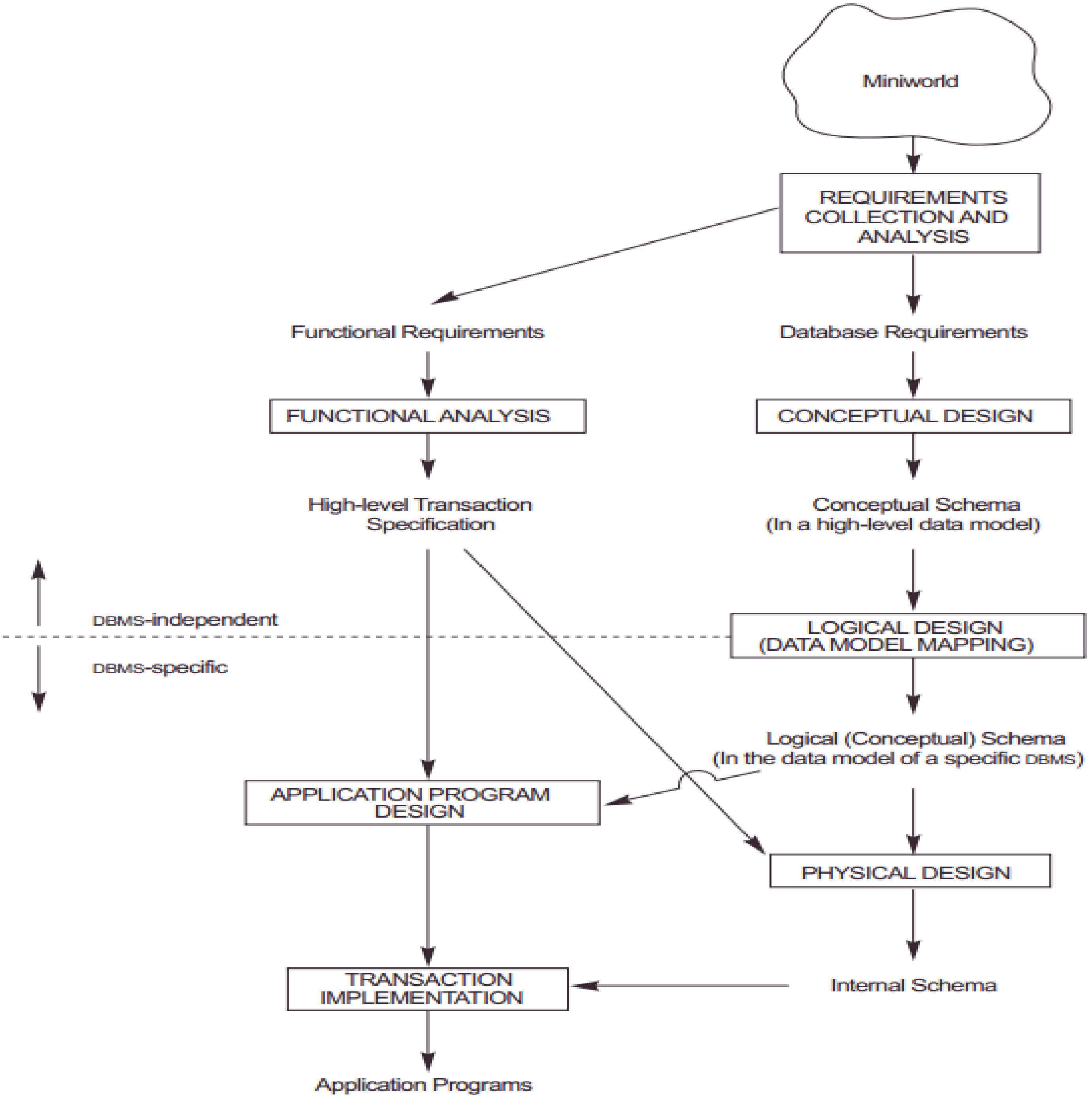
- Design process
  - Determine the purpose of the database
  - Find and organize the information required
  - Divide the information into entities/tables
  - Turn information items into attributes/columns
  - Specify indetified/primary keys
  - Set up the relationships/constraints
  - Refine the design
  - Apply the normalization rules

Basics

- Design process (cont.)
  - The process of designing a database begins with an analysis of what information the database must hold and what are the relationships among components of that information



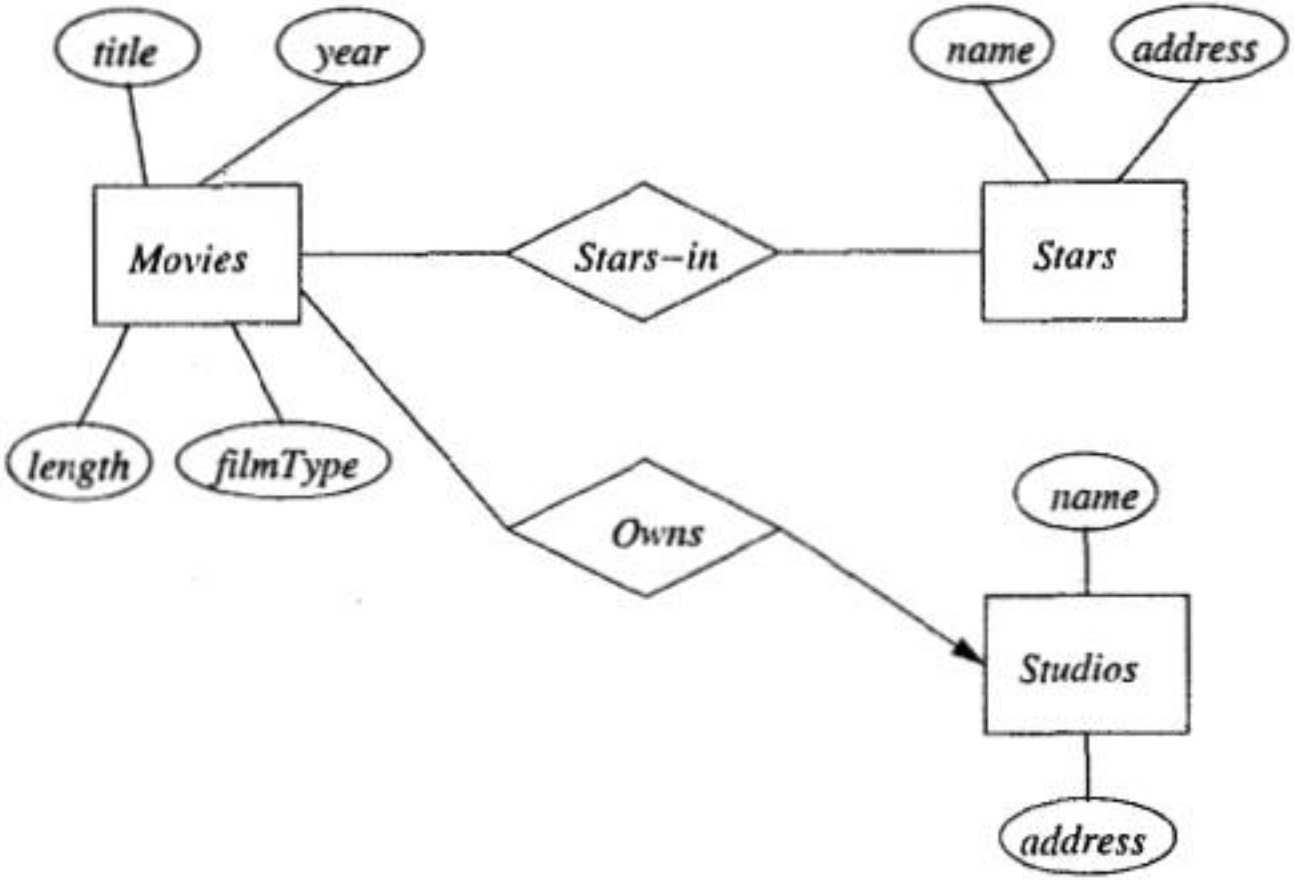
Design process


Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Database Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

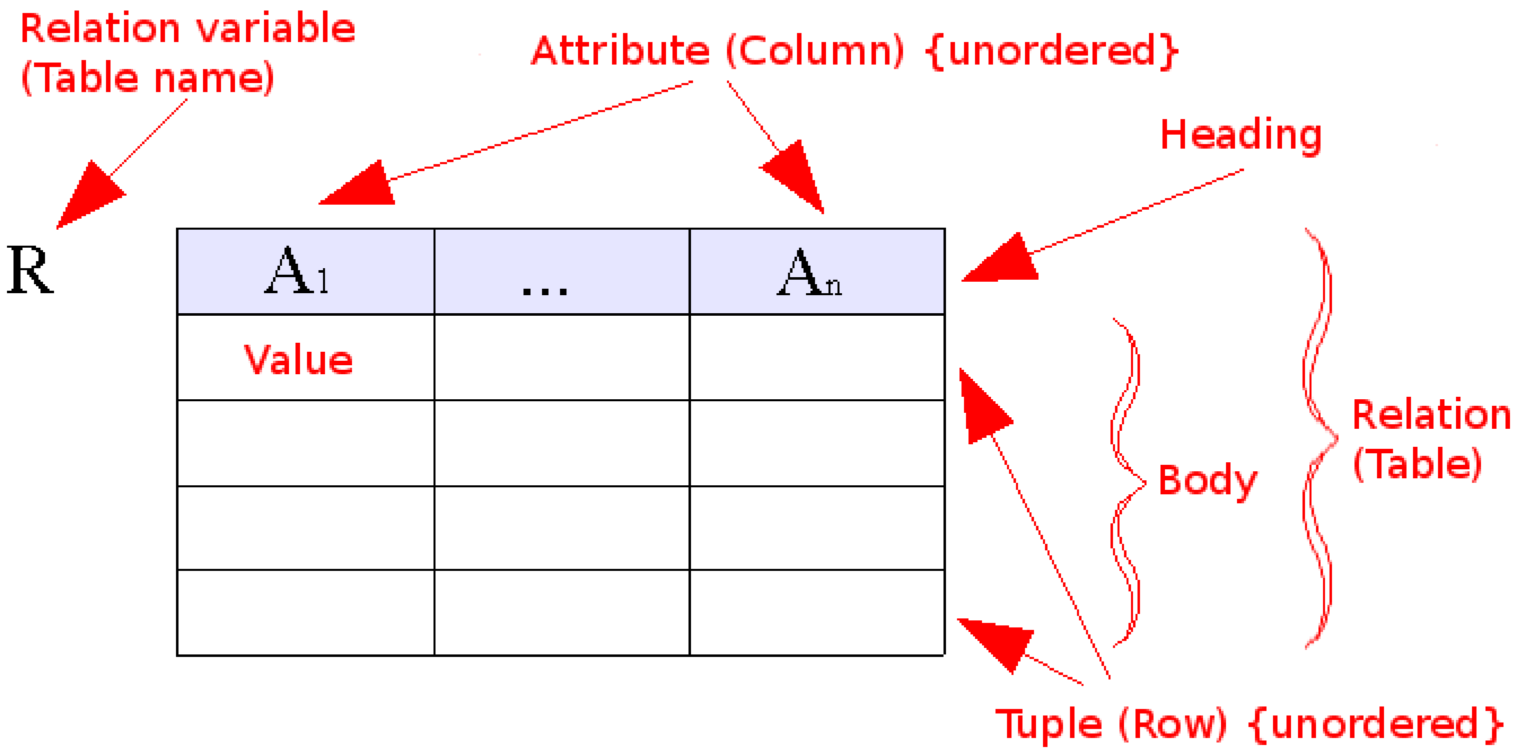
Basics

- Database modeling
  - E-R model
    - the structure of data is represented graphically using three principal element types
  - Relational model
    - the structure of data is represented using tables and views

E-R model


title
year
name
address
Movies
Stars-in
Stars
length
filmType
Owns
name
Studios
address

Relation/Table



Relation variable (Table name)

Attribute (Column) {unordered}

Heading

R

A_1	…	A_n
Value		

Body

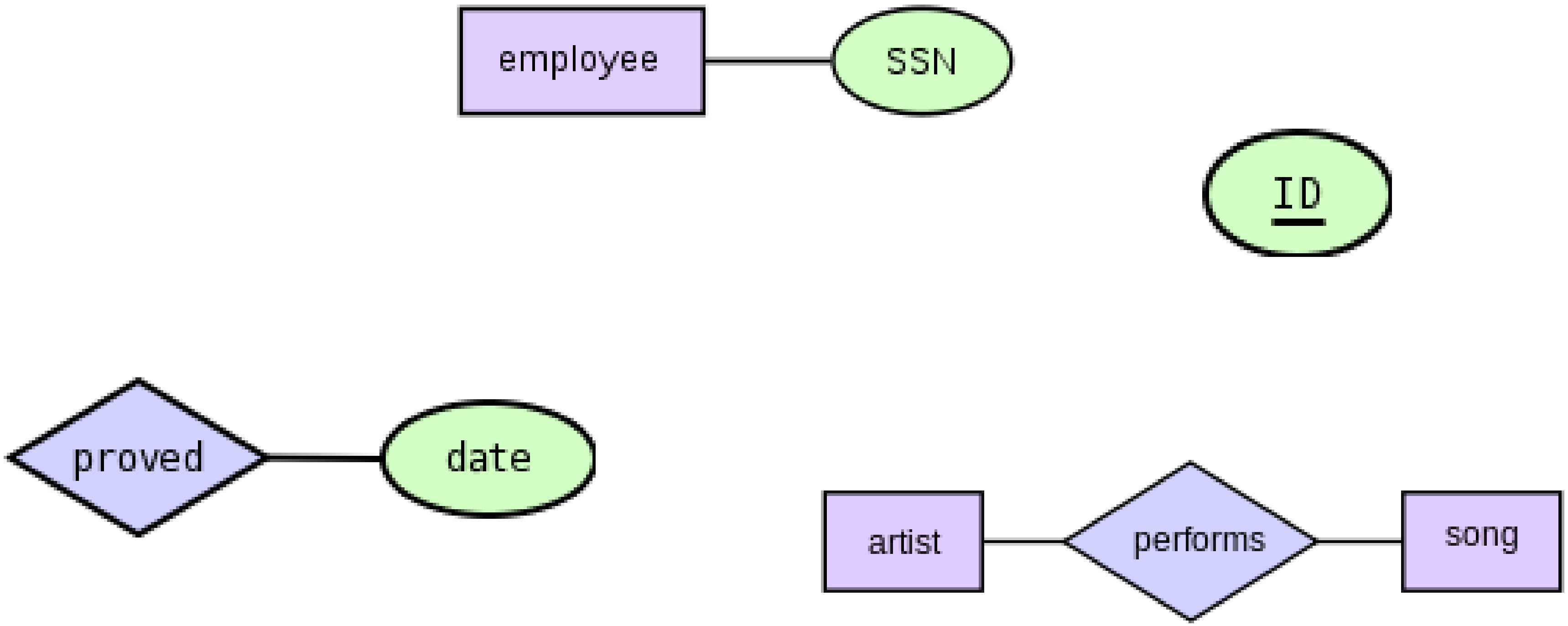
Relation (Table)

Tuple (Row) {unordered}

E-R modeling

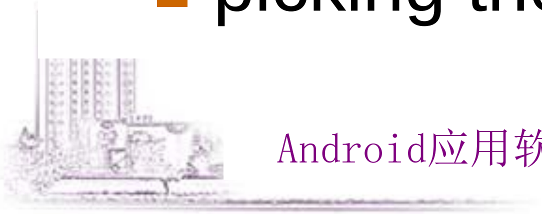
- Entity
  - an entity may be defined as a thing capable of an independent existence that can be uniquely identified
  - represented by rectangles
- Attribute
  - an attribute is used to describe features of entity or relationship
  - represented by ovals
- Relationship
  - a relationship captures how entities are related to one another
  - represented by diamonds

Notations of E-R diagram



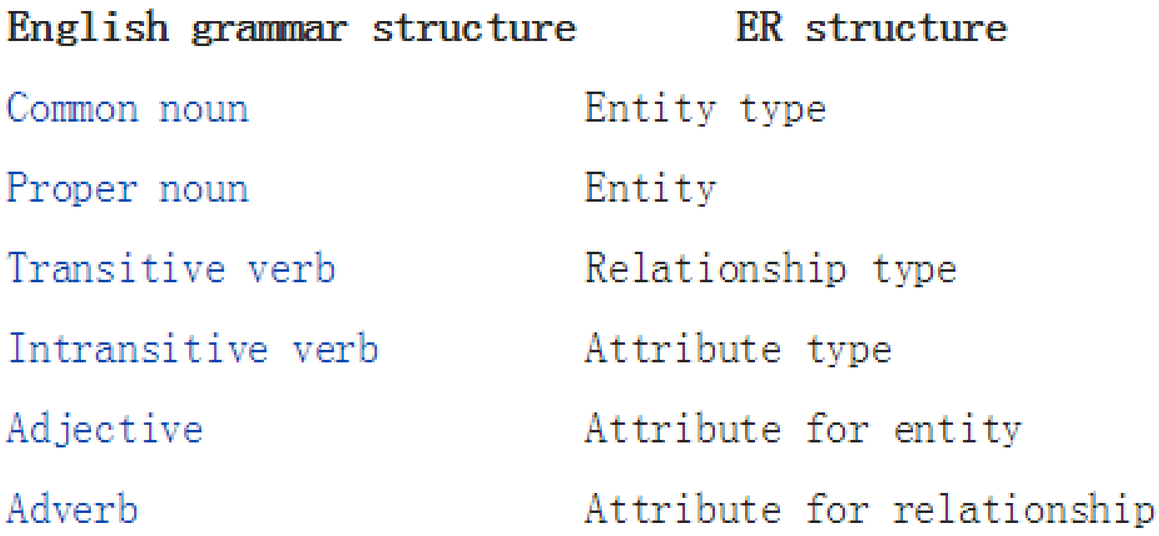
E-R modeling

- Design principles
  - faithfullness
    - entity sets and their attributes should reflect reality
  - avoid redundancy
  - simplicity counts
    - Avoid introducing more elements into your design than is absolutely necessary
  - choosing the right relationship
  - picking the right kind of element



E-R modeling

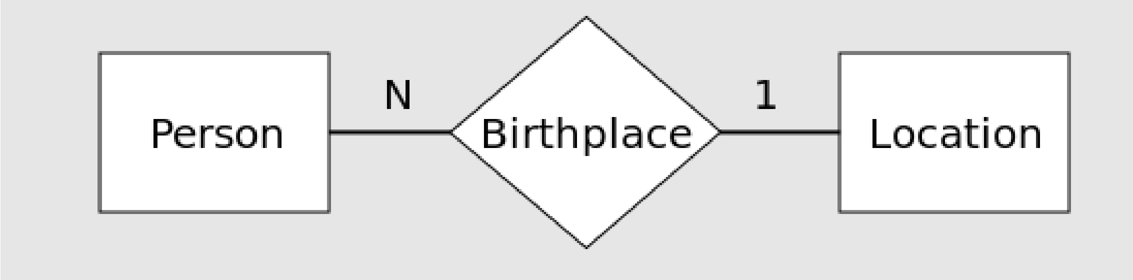
- Mapping natural language



English grammar structure	ER structure
Common noun	Entity type
Proper noun	Entity
Transitive verb	Relationship type
Intransitive verb	Attribute type
Adjective	Attribute for entity
Adverb	Attribute for relationship

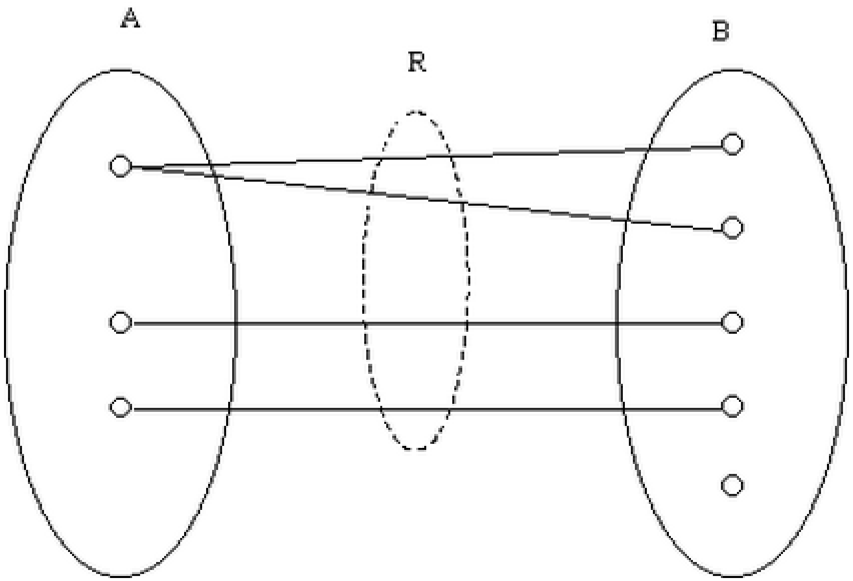
E-R modeling

- Role naming
  - signifies role that a participating entity plays in each relationship instance
  - name roles with nouns such as owner or possession

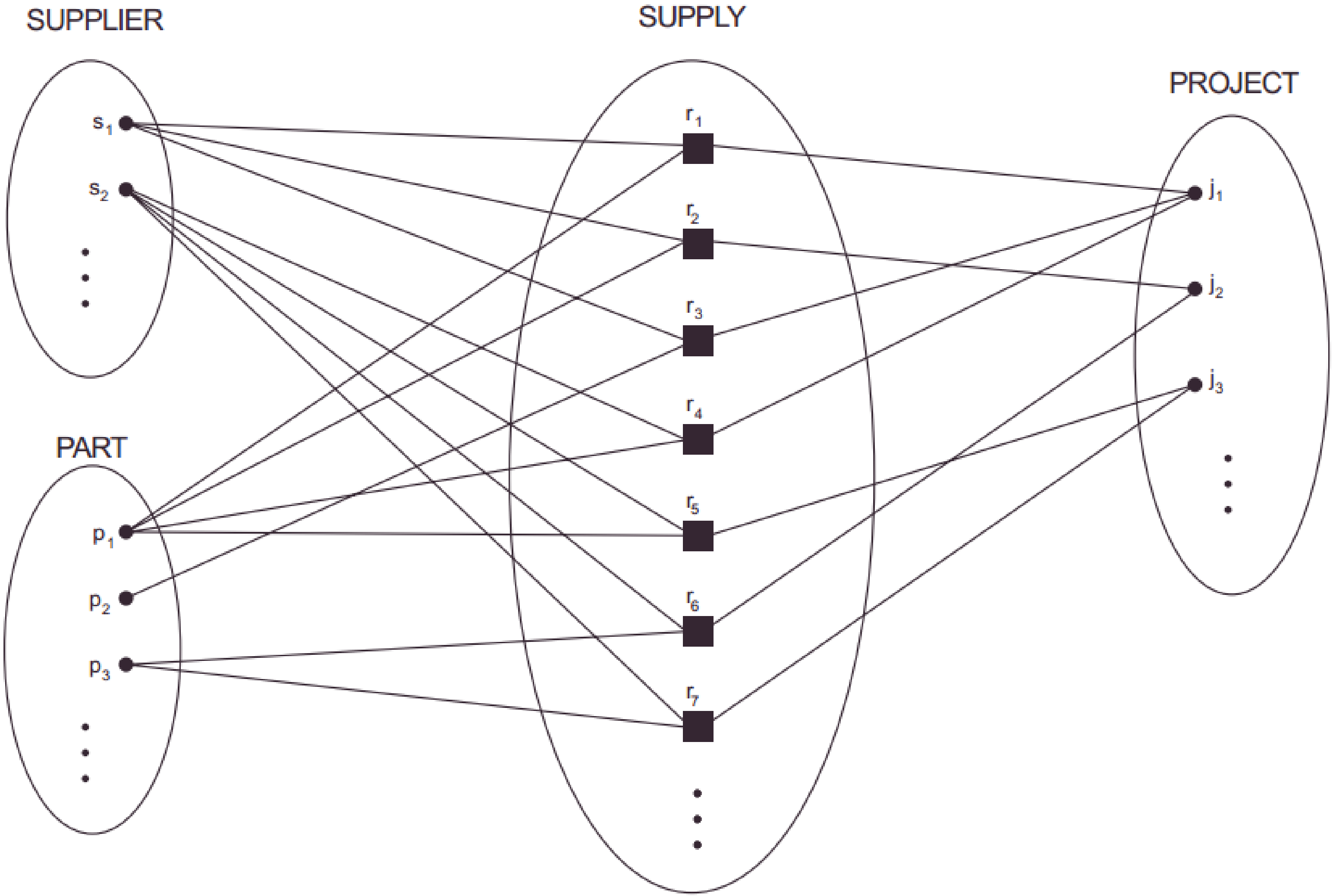


E-R modeling

- Cardinality
  - concerns the number of entities within each entity type that can be linked by a given relationship type
  - 1:1, 1:N, M:N

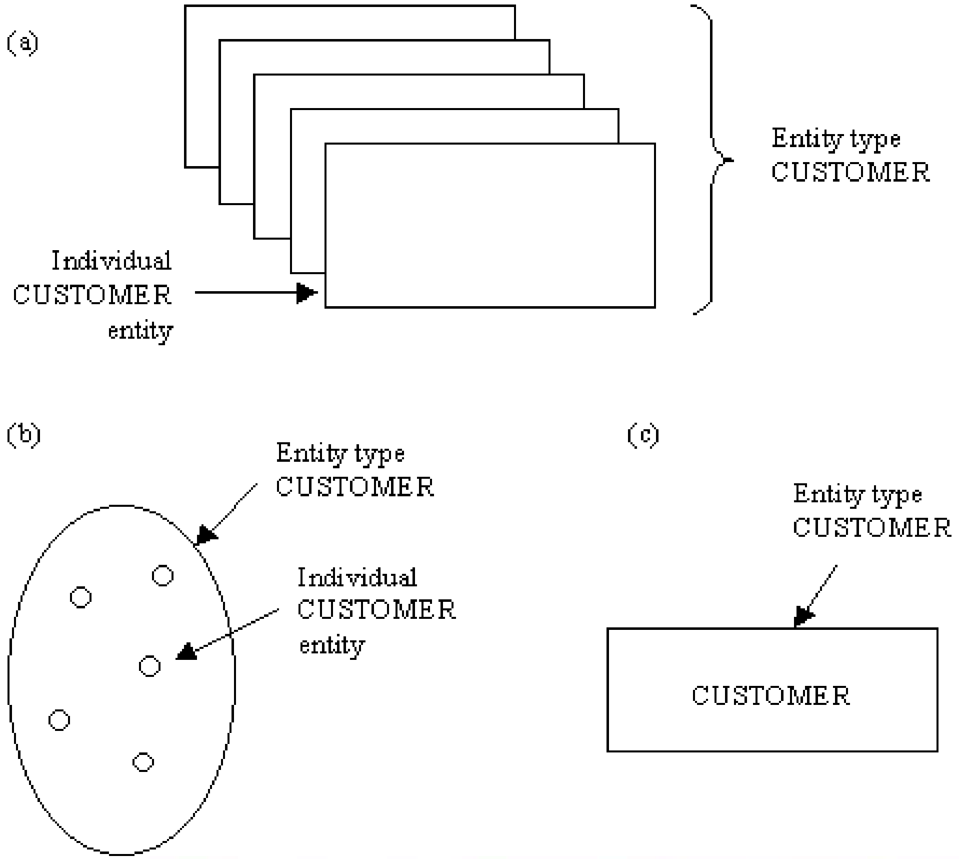


M:N relationship



E-R modeling

- Key
  - used to identify each entity from all the others in the entity type
  - primary key
  - unique key
  - foreign key



Relational Data Modeling

- The relational model gives us a single way to represent data: as a two-dimmsional table called a relation
- Each row corresponds to one entity instance
- and each column corresponds to one of the attributes of the entity instances

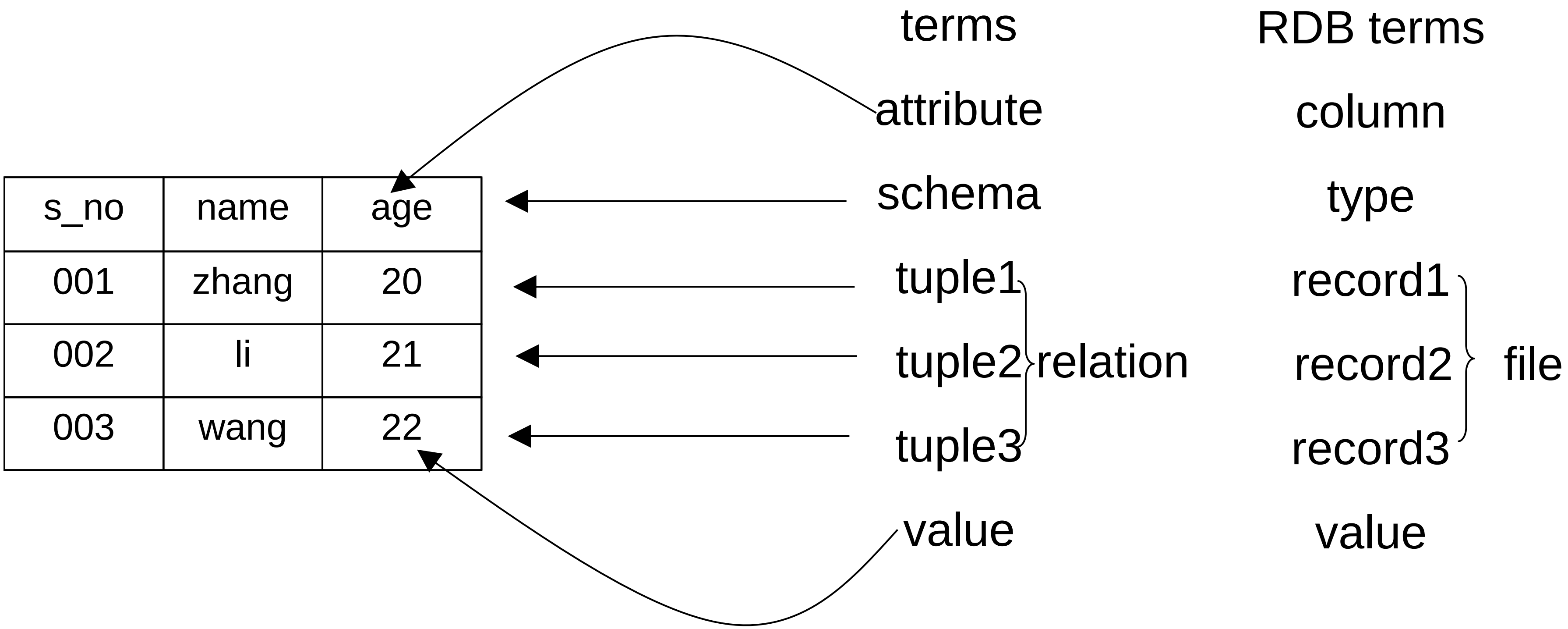
Relational Data Modeling

- Attributes
  - an attribute describes the meaning of entries in the column below
- Schemas
  - The name of a relation and the set of attributes for a relation is called the schema for that relation

Relational Data Modeling

- Tuples
  - The rows of a relation, other than the header row containing the attribute names, are called tuples
- Domains
  - The components of any tuple of the relation must have, in each component, a value that belongs to the domain of the corresponding column

Relaiton instance



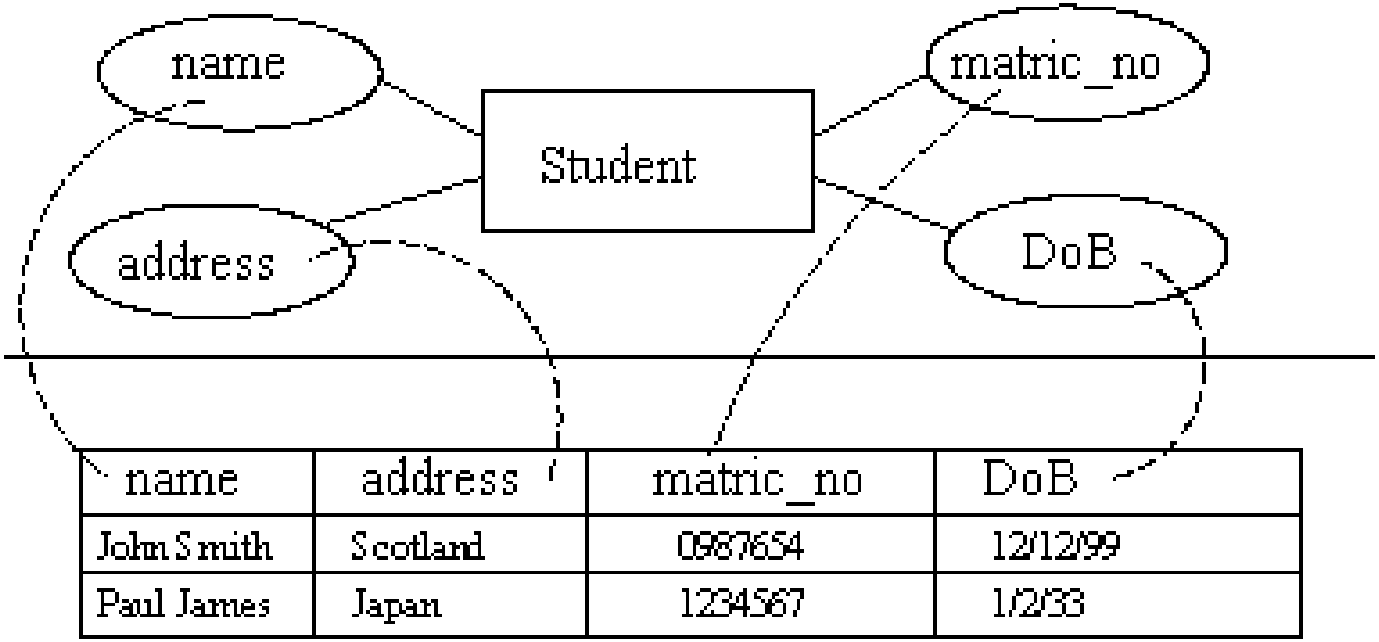
s_no	name	age
001	zhang	20
002	li	21
003	wang	22

terms		RDB terms	
attribute		column	
schema		type	
tuple1		record1	
tuple2	relation	record2	file
tuple3		record3	
value		value	

Relational Data Modeling

- From E/R Diagrams to Relational Designs
  - Turn each entity set into a relation with the same set of attributes, and
  - Replace a relationship by a relation whose attributes are the keys for the connected entity sets
  - Combining relations
  - Nomalization
  - Reviewing

Mapping ER Models into Relations

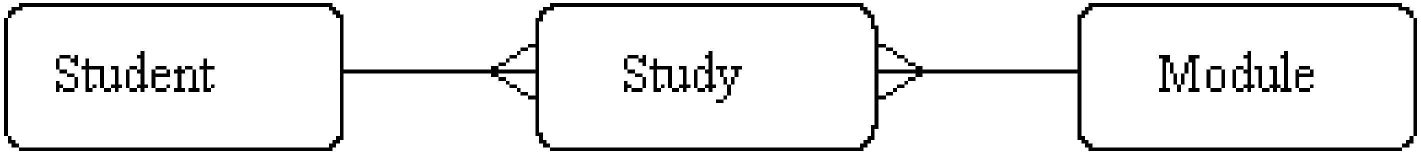


name	address	matric_no	DoB
John Smith	Scotland	0987654	12/12/99
Paul James	Japan	1234567	1/2/33

Relational Data Modeling

- Discovering Keys for Relations
  - If the relation comes from an entity set then the key for the relation is the key attributes of this entity set
  - If the relationship is many-many, then the keys of both connected entity sets are the key attributes for R
  - If the relationship is many-one from entity set E1 to entity set E2, then the key attributes of E1 are key attributes of R, but those of E2 are not
  - If the relationship is one-one, then the key attributes for either of the connected entity sets are key attributes of R

Mapping ER Models into Relations



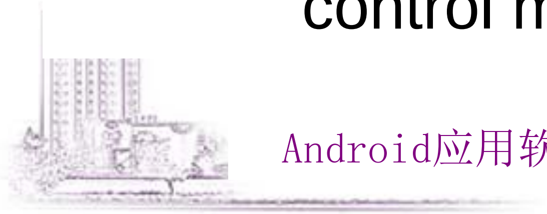
Student(matric_no, st_name, dob)
Module(module_no, m_name, level, credits)
Studies(*matric_no,module_no*)

Relational Data Modeling

- Constraints
  - primary key constraints
  - unique key constraints
  - referential integrity constraints
  - domain integrity constraints
  - gerneral integrity constraints

Relational Data Modeling

- Trigger & stored procedure
  - The trigger is mostly used for maintaining the integrity of the information on the database
  - A stored procedure is a subroutine available to applications that access a relational database system
    - stored in the database data dictionary
    - typical use for stored procedures include data validation (integrated into the database) or access control mechanisms



SQLite and ContentProvider Programming

- SQLite
  - is a software library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine
  - is the most widely deployed SQL database engine in the world
  - the source code for SQLite is in the public domain
  - http://www.sqlite.org/

SQLite and ContentProvider Programming

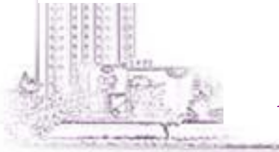
- Google uses SQLite in their Desktop for Mac, in Google Gears, in the Android cell-phone operating system, and in the Chrome Web Browser
- Apple uses SQLite for many functions within Mac OS X, including Apple Mail, Safari, and in Aperture. Apple uses SQLite in the iPhone and in the iPod touch and in iTunes software

SQLite and ContentProvider Programming

- Datatypes In SQLite Version 3
  - NULL: the value is a NULL value
  - INTEGER: the value is a signed integer, stored in 1, 2, 3, 4, 6, or 8 bytes depending on the magnitude of the value
  - REAL: the value is a floating point value, stored as an 8-byte IEEE floating point number
  - TEXT: the value is a text string, stored using the database encoding (UTF-8, UTF-16BE or UTF-16LE)
  - BLOB: the value is a blob of data, stored exactly as it was input

SQLite and ContentProvider Programming

- Android provides full support for SQLite databases
- Any databases you create will be accessible by name to any class in the application, but not outside the application
- Using SQLiteOpenHelper manage database creation and version management



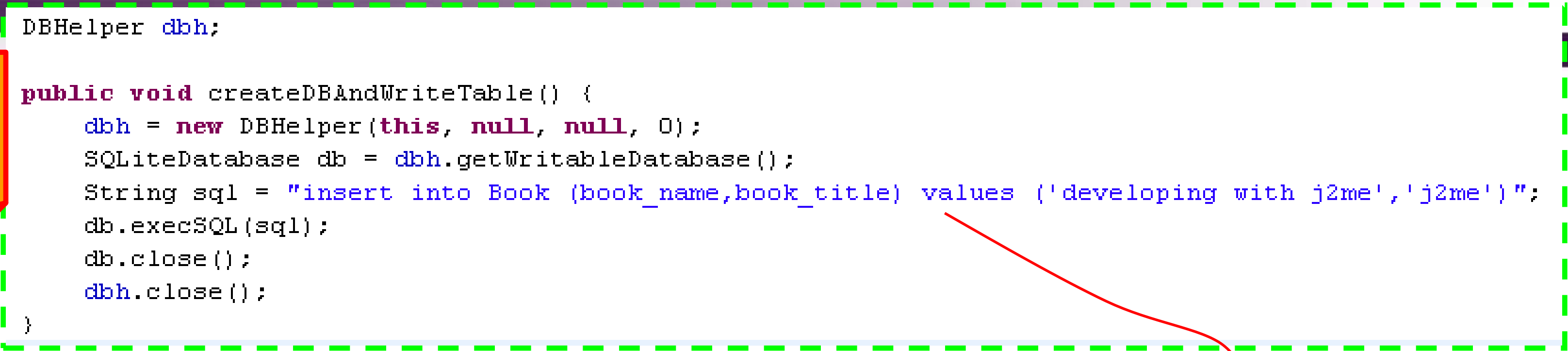
SQLite and ContentProvider Programming

- SQLiteOpenHelper
  - An abstract class, being used to create a helper object to create, open, and/or manage a database
- SQLiteDatabase
  - A class exposes methods to manage a SQLite database
  - Database names must be unique within an application, not across all applications

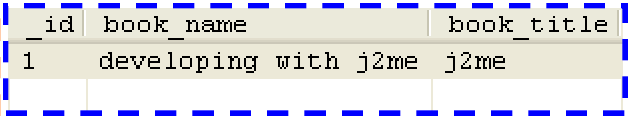
SQLite and ContentProvider Programming

- SQLiteDatabase (cont.)
  - You can execute SQLite queries using the SQLiteDatabase query() methods, which accept various query parameters
    - Every SQLite query will return a Cursor that points to all the rows found by the query
  - You can also use the SQLiteDatabase delete()/update() methods to manage your data in database

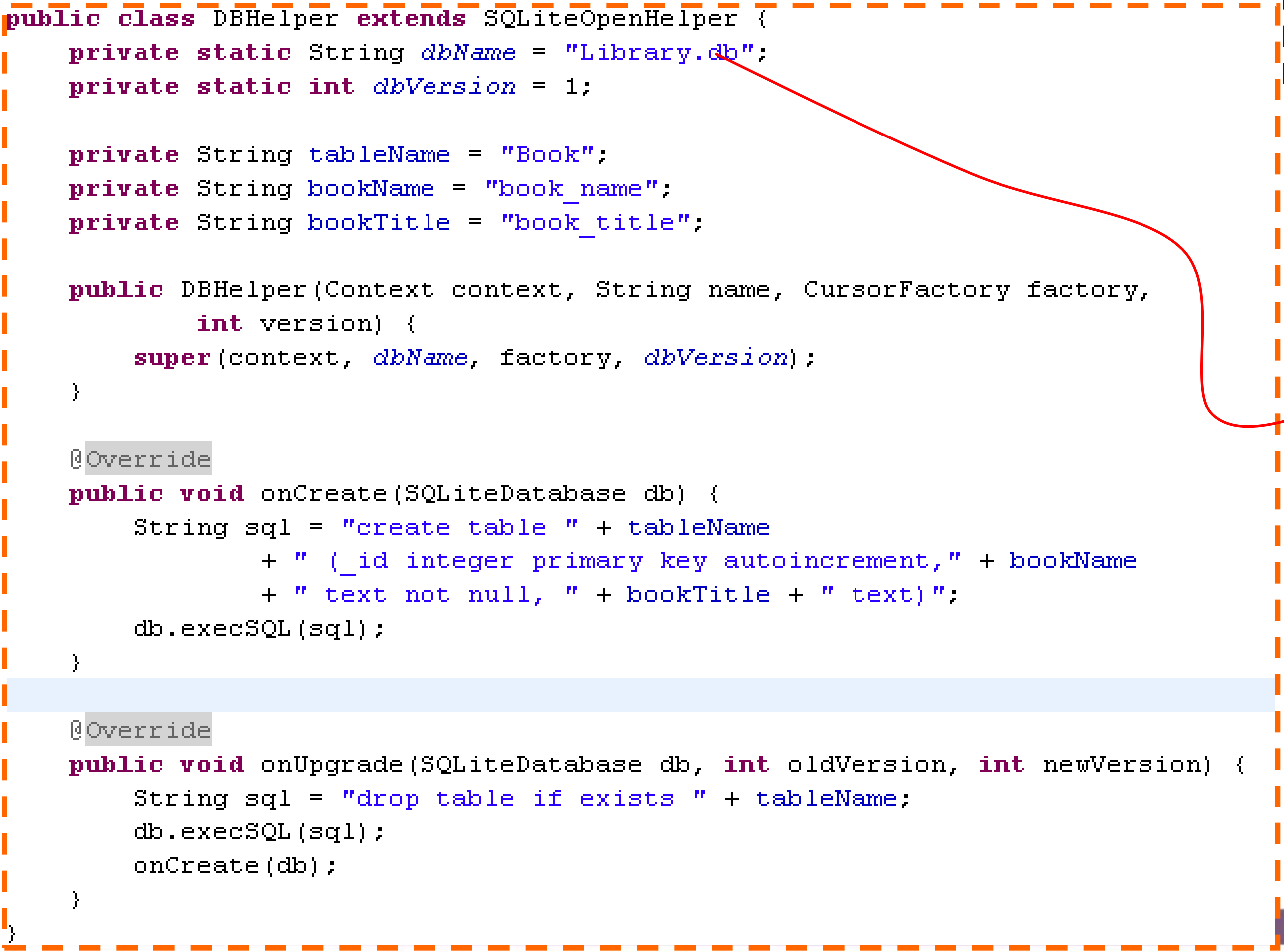
Using SQLiteDB



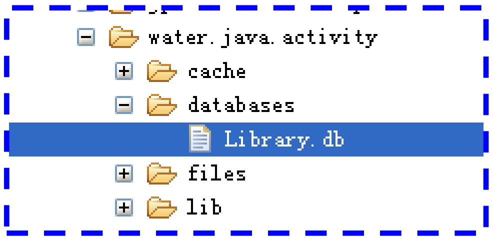
```java
DBHelper dbh;

public void createDBAndWriteTable() {
    dbh = new DBHelper(this, null, null, 0);
    SQLiteDatabase db = dbh.getWritableDatabase();
    String sql = "insert into Book (book_name,book_title) values ('developing with j2me','j2me')";
    db.execSQL(sql);
    db.close();
    dbh.close();
}
```



_id	book_name	book_title
1	developing with j2me	j2me



```java
public class DBHelper extends SQLiteOpenHelper {
    private static String dbName = "Library.db";
    private static int dbVersion = 1;

    private String tableName = "Book";
    private String bookName = "book_name";
    private String bookTitle = "book_title";

    public DBHelper(Context context, String name, CursorFactory factory,
            int version) {
        super(context, dbName, factory, dbVersion);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        String sql = "create table " + tableName
                + " (_id integer primary key autoincrement," + bookName
                + " text not null, " + bookTitle + " text)";
        db.execSQL(sql);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        String sql = "drop table if exists " + tableName;
        db.execSQL(sql);
        onCreate(db);
    }
}
```



```
⊟ water.java.activity
    ⊞ cache
    ⊟ databases
          Library.db
    ⊞ files
    ⊞ lib
```

SQLite and ContentProvider Programming

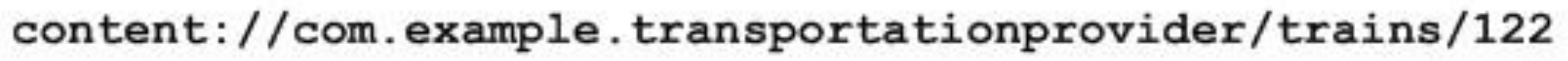
- ContentProvider
  - ContentResolver
    - You can use the ContentResolver's methods to interact with whatever content providers you're interested in
    - There's just a single instance of each type of ContentProvider
    - But it can communicate with multiple ContentResolver objects in different applications and processes

SQLite and ContentProvider Programming

- ContentProvider (cont.)
  - Data model
    - Content providers expose their data as a simple table on a database model
    - Every record includes a numeric _ID field that uniquely identifies the record within the table
    - A query returns a Cursor object that can move from record to record and column to column to read the contents of each field

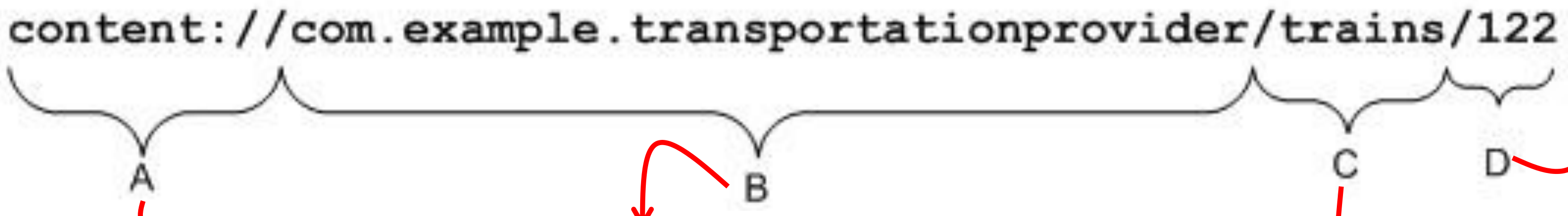
SQLite and ContentProvider Programming

- ContentProvider (cont.)
  - URIs
    - Each content provider exposes a public URI (wrapped as a Uri object) that uniquely identifies its data set
    - All URIs for providers begin with the string "content://"



```
content://com.example.transportationprovider/trains/122
```

Content URI



content://com.example.transportationprovider/trains/122

Standard prefix indicating that the data is controlled by a content provider

The authority part of the URI; it identifies the content provider

The path that the content provider uses to determine what kind of data is being requested

The ID of the specific record being requested, if any

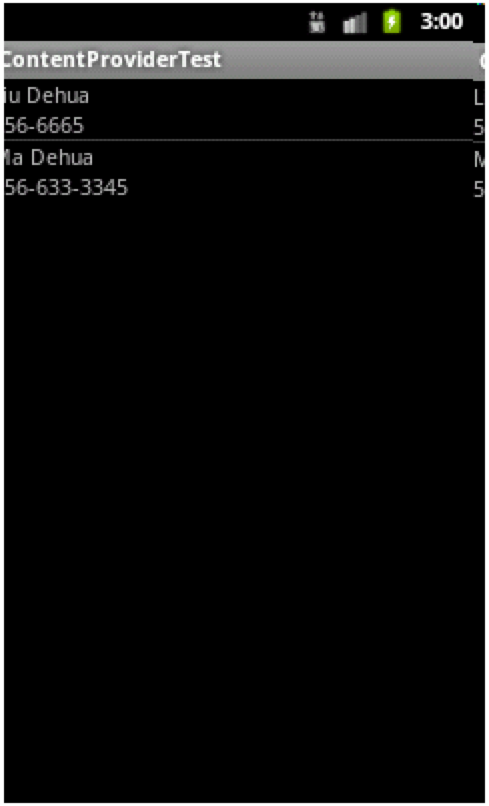
SQLite and ContentProvider Programming

- Querying a Content Provider
  - The URI that identifies the provider
  - The names of the data fields you want to receive
  - The data types for those fields
  - If you're querying a particular record, you also need the ID for that record

Using Content Provider to Query Contact

```
<uses-permission android:name="android.permission.READ_CONTACTS"></uses-permission>
```

```
public ArrayList<HashMap<String, Object>> initListData() {
    ArrayList<HashMap<String, Object>> al = new ArrayList<HashMap<String, Object>>();
    HashMap<String, Object> hm = null;
    Uri contactURI = ContactsContract.CommonDataKinds.Phone.CONTENT_URI;
    ContentResolver cr = getContentResolver();
    String[] columns = new String[] {
            ContactsContract.CommonDataKinds.Phone._ID,
            ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME,
            ContactsContract.CommonDataKinds.Phone.NUMBER };
    Cursor c = cr.query(contactURI, columns, null, null,
            ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME + " ASC");
    if (c.moveToFirst()) {
        int nameIndex = c
                .getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME);
        int numberIndex = c
                .getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER);
        String name;
        String number;
        do {
            hm = new HashMap<String, Object>();
            name = c.getString(nameIndex);
            number = c.getString(numberIndex);
            hm.put("name", name);
            hm.put("number", number);
            al.add(hm);
        } while (c.moveToNext());
    }
    c.close();
    return al;
}
```



3:00

ContentProviderTest

iu Dehua
56-6665

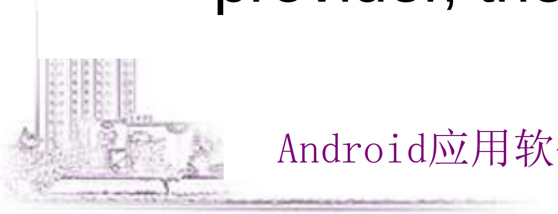
la Dehua
56-633-3345

After finishing cursor operation, if you don't need it any more, you should close it for releasing memory resources

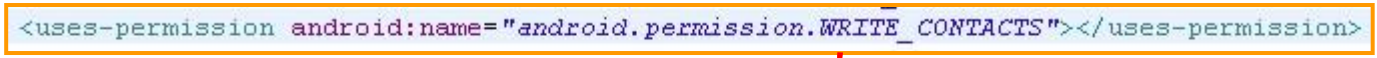
SQLite and ContentProvider Programming

Modifying Data

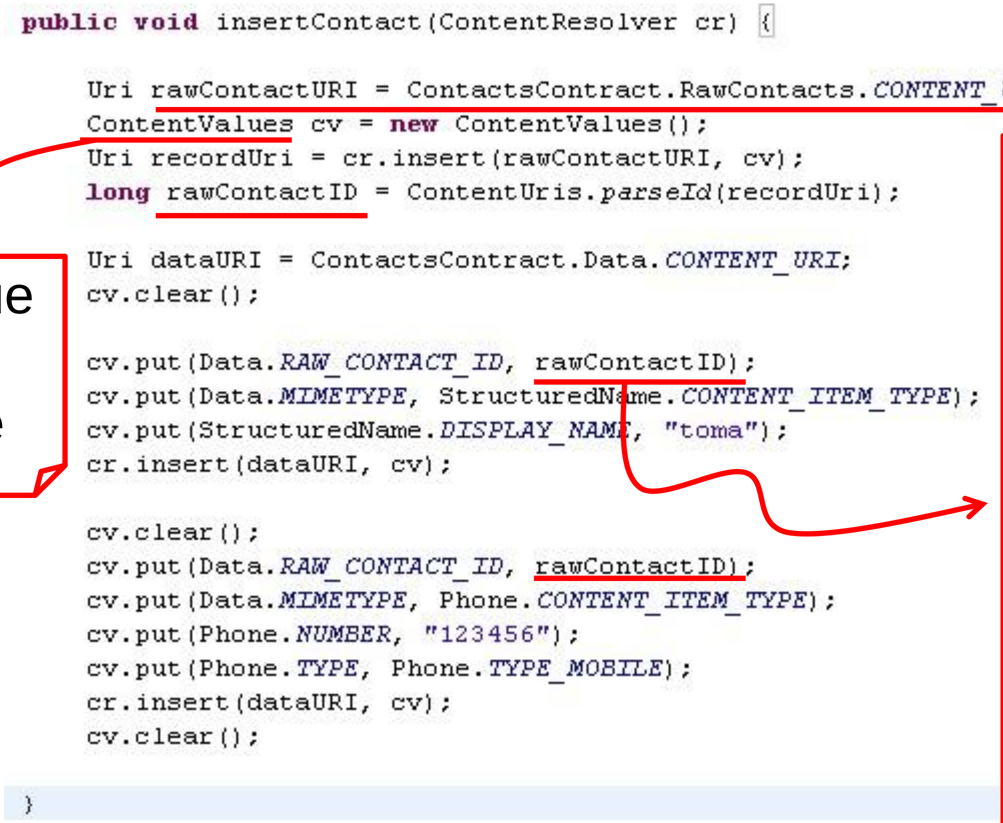
- Data kept by a content provider can be modified by
  - Adding new records
  - Adding new values to existing records
  - Batch updating existing records
  - Deleting records
- Some content providers require a more restrictive permission for writing data than they do for reading it. If you don't have permission to write to a content provider, the ContentResolver methods will fail



Using Content Provider to Insert Contact Records



```
<uses-permission android:name="android.permission.WRITE_CONTACTS"></uses-permission>
```

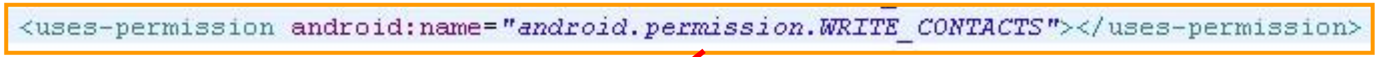


```
public void insertContact(ContentResolver cr) {

    Uri rawContactURI = ContactsContract.RawContacts.CONTENT_URI;
    ContentValues cv = new ContentValues();
    Uri recordUri = cr.insert(rawContactURI, cv);
    long rawContactID = ContentUris.parseId(recordUri);

    Uri dataURI = ContactsContract.Data.CONTENT_URI;
    cv.clear();

    cv.put(Data.RAW_CONTACT_ID, rawContactID);
    cv.put(Data.MIMETYPE, StructuredName.CONTENT_ITEM_TYPE);
    cv.put(StructuredName.DISPLAY_NAME, "toma");
    cr.insert(dataURI, cv);

    cv.clear();
    cv.put(Data.RAW_CONTACT_ID, rawContactID);
    cv.put(Data.MIMETYPE, Phone.CONTENT_ITEM_TYPE);
    cv.put(Phone.NUMBER, "123456");
    cv.put(Phone.TYPE, Phone.TYPE_MOBILE);
    cr.insert(dataURI, cv);
    cv.clear();

}
```

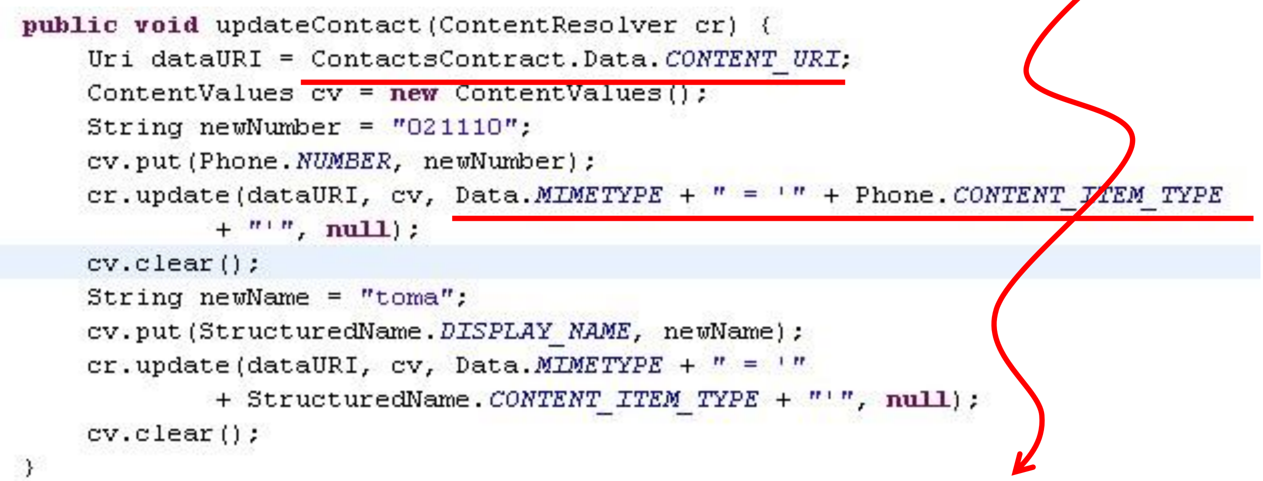
Key-value data structure

To insert a contact record, you should have the permission of write_contacts for writing data and add a new record into table of raw_contacts firstly. Beacause raw_contacts is the basic table for others of contacts. And their relation is set up by column raw_contact_id

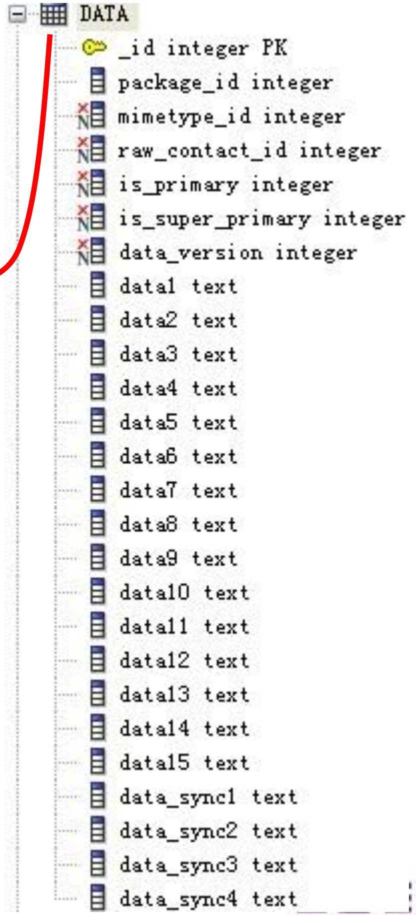
Using Content Provider to Update Contact Records



```
<uses-permission android:name="android.permission.WRITE_CONTACTS"></uses-permission>
```



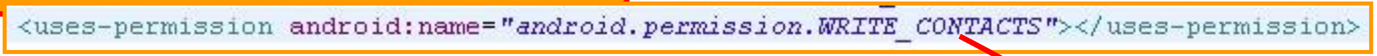
```
public void updateContact(ContentResolver cr) {
    Uri dataURI = ContactsContract.Data.CONTENT_URI;
    ContentValues cv = new ContentValues();
    String newNumber = "021110";
    cv.put(Phone.NUMBER, newNumber);
    cr.update(dataURI, cv, Data.MIMETYPE + " = '" + Phone.CONTENT_ITEM_TYPE
            + "'", null);
    cv.clear();
    String newName = "toma";
    cv.put(StructuredName.DISPLAY_NAME, newName);
    cr.update(dataURI, cv, Data.MIMETYPE + " = '"
            + StructuredName.CONTENT_ITEM_TYPE + "'", null);
    cv.clear();
}
```



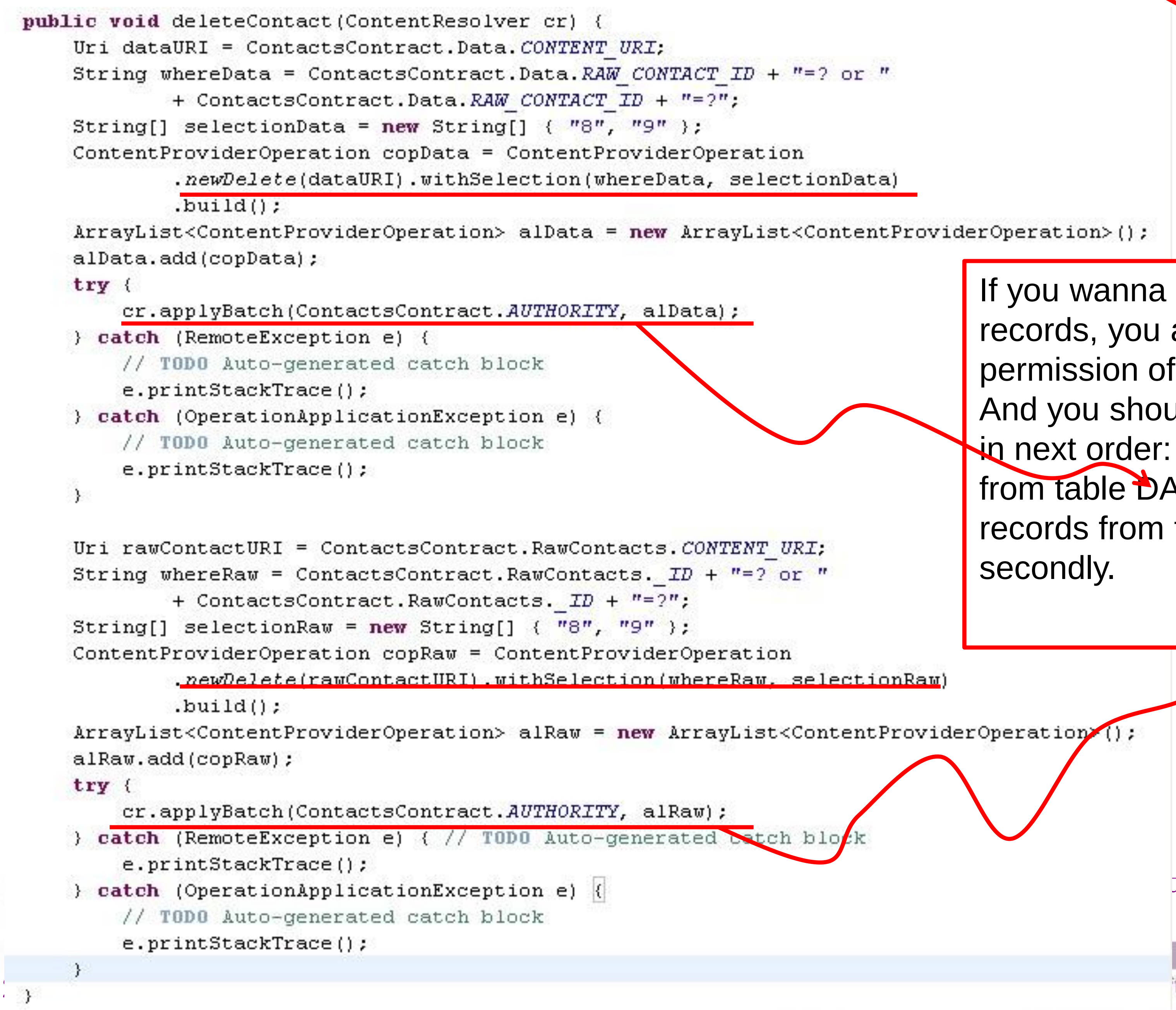
```
DATA
  _id integer PK
  package_id integer
  mimetype_id integer
  raw_contact_id integer
  is_primary integer
  is_super_primary integer
  data_version integer
  data1 text
  data2 text
  data3 text
  data4 text
  data5 text
  data6 text
  data7 text
  data8 text
  data9 text
  data10 text
  data11 text
  data12 text
  data13 text
  data14 text
  data15 text
  data_sync1 text
  data_sync2 text
  data_sync3 text
  data_sync4 text
```

If you wanna update contact records, you also need permission of write_contacts. And all of contacts data are contained in table DATA. So, you should manipulate DATA table to finish the task.

Using Content Provider to Delete Contact Records



```
<uses-permission android:name="android.permission.WRITE_CONTACTS"></uses-permission>
```



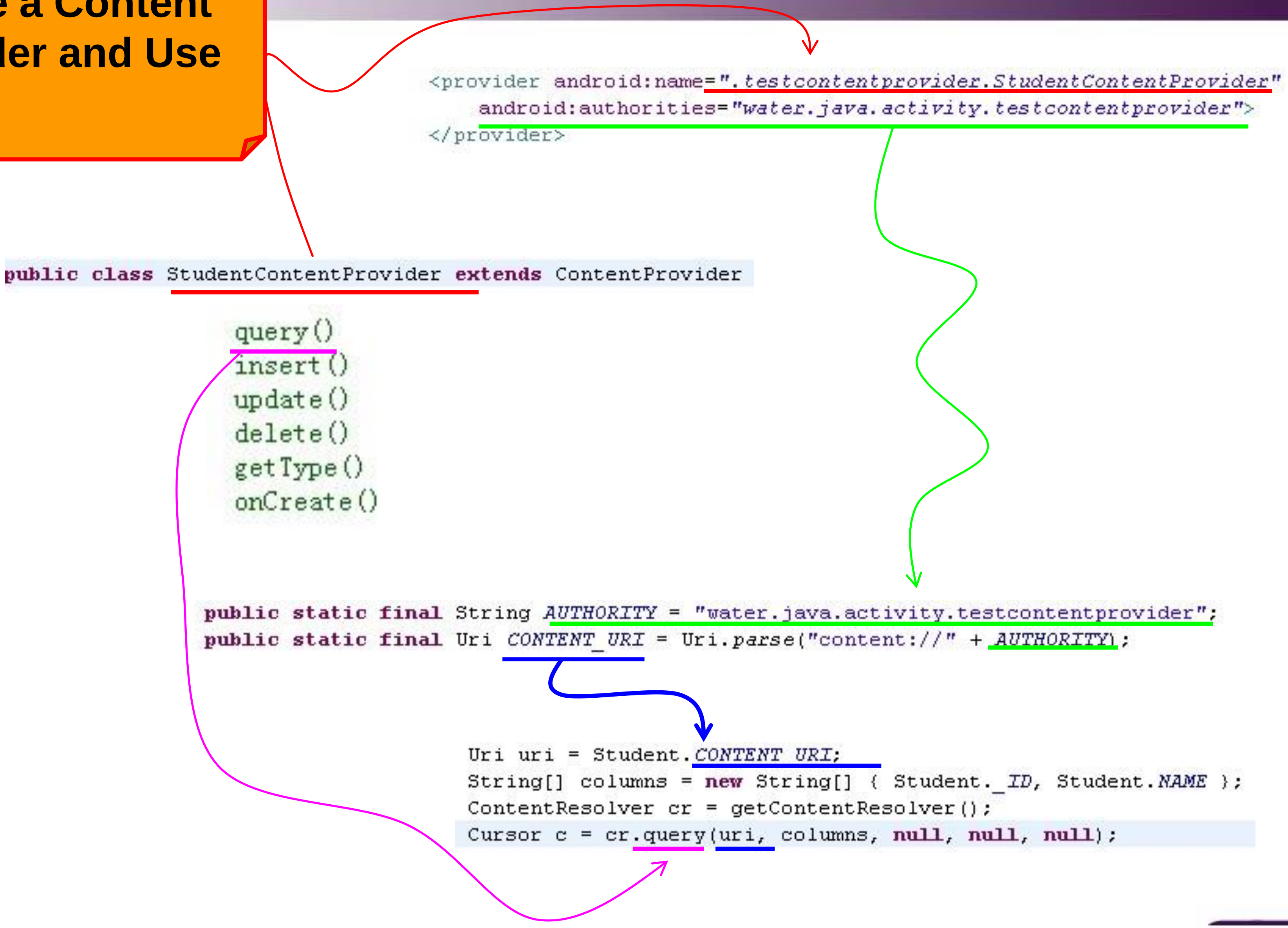
```
public void deleteContact(ContentResolver cr) {
    Uri dataURI = ContactsContract.Data.CONTENT_URI;
    String whereData = ContactsContract.Data.RAW_CONTACT_ID + "=? or "
            + ContactsContract.Data.RAW_CONTACT_ID + "=?";
    String[] selectionData = new String[] { "8", "9" };
    ContentProviderOperation copData = ContentProviderOperation
            .newDelete(dataURI).withSelection(whereData, selectionData)
            .build();
    ArrayList<ContentProviderOperation> alData = new ArrayList<ContentProviderOperation>();
    alData.add(copData);
    try {
        cr.applyBatch(ContactsContract.AUTHORITY, alData);
    } catch (RemoteException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (OperationApplicationException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    Uri rawContactURI = ContactsContract.RawContacts.CONTENT_URI;
    String whereRaw = ContactsContract.RawContacts._ID + "=? or "
            + ContactsContract.RawContacts._ID + "=?";
    String[] selectionRaw = new String[] { "8", "9" };
    ContentProviderOperation copRaw = ContentProviderOperation
            .newDelete(rawContactURI).withSelection(whereRaw, selectionRaw)
            .build();
    ArrayList<ContentProviderOperation> alRaw = new ArrayList<ContentProviderOperation>();
    alRaw.add(copRaw);
    try {
        cr.applyBatch(ContactsContract.AUTHORITY, alRaw);
    } catch (RemoteException e) { // TODO Auto-generated catch block
        e.printStackTrace();
    } catch (OperationApplicationException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

If you wanna delete contact records, you also need permission of write_contacts. And you should delete records in next order: delete records from table DATA firstly, delete records from table raw_contact secondly.

SQLite and ContentProvider Programming

- Creating a Content Provider
  - Set up a system for storing the data
  - Extend the ContentProvider class to provide access to the data
  - Declare the content provider in the manifest file for your application (AndroidManifest.xml)

Create a Content Provider and Use It



```
<provider android:name=".testcontentprovider.StudentContentProvider"
    android:authorities="water.java.activity.testcontentprovider">
</provider>
```

```
public class StudentContentProvider extends ContentProvider
```

```
query()
insert()
update()
delete()
getType()
onCreate()
```

```
public static final String AUTHORITY = "water.java.activity.testcontentprovider";
public static final Uri CONTENT_URI = Uri.parse("content://" + AUTHORITY);
```

```
Uri uri = Student.CONTENT_URI;
String[] columns = new String[] { Student._ID, Student.NAME };
ContentResolver cr = getContentResolver();
Cursor c = cr.query(uri, columns, null, null, null);
```

Conclusions

- Basics
- E-R Modeling
- Relational Data Modeling
- SQLite and ContentProvider Programming
- Conclusions