CSE 530A

BLOBs and CLOBs

Washington University
Fall 2013
BLOBs and CLOBs

• BLOBs are binary large objects
• CLOBs are character large objects
• Unfortunately, there is no SQL standard for handling BLOBs and CLOBs
  – Most DBMSs have their own, incompatible methods
JDBC

• Incompatibilities among BLOB implementations can be hidden by APIs
  – JDBC has a standard way of accessing BLOBs
byte[] data = ...;

// Create an InputStream from the data
ByteArrayInputStream stream = new ByteArrayInputStream(data);

PreparedStatement stmt = conn.prepareStatement("INSERT INTO my_blobs VALUES (?, ?)");
stmt.setLong(1, id);

// Set the stream for the BLOB field
stmt.setBinaryStream(2, stream);

stmt.executeUpdate();
JDBC

• Reading from a BLOB

PreparedStatement stmt = conn.prepareStatement("SELECT id, value FROM my_blobs WHERE id = 1");
ResultSet rs = stmt.executeQuery();
while (rs.next()) {
    InputStream istream = rs.getBinaryStream(2);
    ...
}
PostgreSQL

• Creating a BLOB column
  – Varies among DBMSs
  – PostgreSQL stores BLOBs in a special table (pg_largeobject)
    • The OID (object ID in the pg_largeobject table) is stored in the referencing table

CREATE TABLE my_blobs (id integer, value oid);
JPA/Hibernate

• Use a field of type java.sql.Blob
• Use the @Lob annotation

```java
@Entity
@Table(name="documents")
public class Document {

    private Blob content;

    @Column(name="content")
    @Lob
    public Blob getContent() {
        return content;
    }

    ...
```
JPA/Hibernate

• Saving a BLOB

...  

Blob blob = Hibernate.createBlob(istream);  
document.setContent(blob);  

...
JPA/Hibernate

• Reading a BLOB

...  
InputStream istream = document.getContent().getBinaryStream();  
...

...
Alternatives

• Should BLOBs actually be stored in the database?
  – Pros
    • Single location for all data (everything is in the database)
  – Cons
    • Databases are not optimized for handling BLOBs
    • Inconsistencies in implementation and support among DBMSs
    • Reading and writing happen in transactions
      – Can cause contention because of long transactions
    • Can vastly increase size of database
    • BLOBs are often limited in size to 1 or 2 GB
Alternatives

• Might be better to store the BLOBs outside the database and store some sort of key to retrieve the BLOB in the database
  – Could store the BLOBs as files in a network mounted file system and save the file path to the BLOBs in the database
  – Pros
    • Could use a specially-optimized content server or distributed content network to serve BLOBs
    • Reading and writing happen outside of transactions
      – Faster transactions
    • Could support larger BLOBs
  – Cons
    • Reading and writing happen outside of transactions
      – Lose the benefit of transaction
      – But many DBMSs don't apply transactions to BLOB functions anyway
    • Data now comes from multiple sources
    • Need to maintain content server for BLOBs
CLOBs

• Like BLOBs, CLOBs are not standardized among DBMSs

• Many DBMSs (like PostgreSQL) have a TEXT type that can be used as a CLOB
  – Often limited in size to 1 or 2 GB
  – Large CLOBs can be inefficient