**ЛАБОРАТОРНАЯ РАБОТА №8,9**

**Тема: Сохранение аудио и видео данных в базе данных. Типы аудио и видео данных. Поиск аудио и видео данных в базе данных.**

**Цель работы.** С помощью функции BLOB вставить в базу данных аудио и видео данные.

**Задания по теме:**

1. Сохранить аудио и видео данные в базе данных.

2. Поиск аудио и видео данных из базы данных.

**Форма отчёта**

1. Оформить отчет по лабораторной работе, используя литературу и различные источники.

2. Вставить в отчет скриншоты.

**Теоретическая часть**

Large Objects (LOB) это тип данных используемый для хранения больших объектов - различные форматы текстов, изображения, видео, звуковые файлы. Использование LOB для хранения данных позволяет эффективно манипулировать данными в приложении.

Компоненты LOB. LOB состоит из локатора и значения.

Локатор – это внутренний указатель на фактическое значение большого объекта.

Значение – это реальное содержимое объекта.

LOB хранит локатор в таблице и данные в различных сегментах, за исключением случая, когда размер данных меньше 4000 байт.

Максимальный размер LOB составляет (4 GB (4 294 967 295 байт) - 1).

**Типы данных LOB. Табл.2**

|  |  |
| --- | --- |
| SQL Datatype | Описание |
| BLOB | Двоичный большой объект (Binary Large OBject)  Хранит данные в двоичном формата, используется для хранения изображений, аудио и видео, а также компилированного программного кода |
| CLOB | Символьный большой объект (Character Large Object).Хранит текстовые данные в формате БД |
| NCLOB | Национальный символьный большой объект (National Character Set Large Object).Хранит текстовые данные в национальной кодировке. |
| BFILE | Внешний двоичный файл (External Binary File)  Файл хранящийся вне базы данных, как файл операционной системы, но доступный из таблиц БД. BFILEs имеет доступ только для чтения.  Когда LOB удаляется, Oracle сервер не удаляет сам файл.  BFILE поддерживает только случайное (не последовательное) чтение, не участвует в транзакции. |

**Виды LOB.**

Как внутренние, так и внешние большие объекты могут быть использованы как столбцы таблицы, переменные в pl/sql, атрибуты объектов.

*Internal LOB*

Внутренние большие объекты - хранятся в табличных пространствах БД, поддерживаются следующие типы данных: BLOB, CLOB, and NCLOB.

*Persistent and Temporary LOB*

Внутренние большие объекты могут быть временными или постоянными.

Постоянные LOB существуют в строках таблицы БД.

Временные LOB создается для использования только в пределах локального приложения.

Временный экземпляр становится постоянным если его вставить в строку таблицы.

Свойства Temporary LOB

• Данные хранятся не в таблице, а во временном табличном пространстве

• Временные LOB быстрее чем постояннные, так как не генерируется redo и rollback данные

• Используется только в сесии. По окончанию сесии удаляется

• Для создания временного LOB используется DBMS\_LOB.CREATETEMPORARY

*External LOB*.

Внешние большие объекты - вид данных, который хранится в файлах операционной системы, вне табличных пространств БД. Используется тип данных – BFILE. BFILE может быть только внешним.

Использование пакета DBMS\_LOB

DBMS\_LOB предоставляет методы манипулирования внутренними и внешними LOBами.

Методы DBMS\_LOB можно условно разделить на два типа mutators и observers:

• mutators - могут изменять LOB : APPEND, COPY, ERASE, TRIM, WRITE, FILECLOSE, FILECLOSEALL, and FILEOPEN.

• observers - могу читать LOB: COMPARE, FILEGETNAME, INSTR, GETLENGTH, READ, SUBSTR, FILEEXISTS, and FILEISOPEN.

Для инициализации LOB локаторов используются следующие функции:

• EMPTY\_CLOB() - функция возвращает LOB локатор для CLOB колонки

• EMPTY\_BLOB() - функция возвращает LOB локатор для BLOB колонки

CLOB и BLOB колонки, так же могут быть инициализированы с помощью символьной или raw строки, если длина не превышает 4000 байт.

**Практическая часть**

This section describes how to use audio types with object views. Just as a view is a virtual table, an object view is a virtual object table.

Oracle provides object views as an extension of the basic relational view mechanism. By using object views, you can create virtual object tables -- of either built-in or user-defined types -- from data stored in the columns of relational or object tables in the database.

Object views can offer specialized or restricted access to the data and objects in a database. For example, you might use an object view to provide a version of an employee object table that does not have attributes containing sensitive data or a deletion method. Object views also let you try object-oriented programming without permanently converting your tables. Using object views, you can convert data gradually and transparently from relational tables to object-relational tables.

***Define a Relational Table Containing No ORDAudio Object***

*create table flat (*

*id NUMBER,*

*description VARCHAR2(4000),*

*localData BLOB,*

*srcType VARCHAR2(4000),*

*srcLocation VARCHAR2(4000),*

*srcName VARCHAR2(4000),*

*upDateTime DATE,*

*local NUMBER,*

*format VARCHAR2(31),*

*mimeType VARCHAR2(4000),*

*comments CLOB,*

*encoding VARCHAR2(256),*

*numberOfChannels NUMBER,*

*samplingRate NUMBER,*

*sampleSize NUMBER,*

*compressionType VARCHAR2(4000),*

*audioDuration NUMBER,*

*)*

*--*

*-- Store audio data as SecureFiles LOBs.*

*--*

*LOB(localData) STORE AS SECUREFILE;*

***Define an Object View Containing an ORDAudio Object and Relational Columns***

*create or replace view object\_audio\_v as*

*select*

*id,*

*ORDSYS.ORDAudio(T.description,*

*ORDSYS.ORDSource(*

*T.localData, T.srctype, T.srcLocation, T.srcName, T.updateTime,*

*T.local),*

*T.format,*

*T.mimeType,*

*T.comments,*

*T.encoding,*

*T.numberOfChannels,*

*T.samplingRate,*

*T.sampleSize,*

*T.compressionType,*

*T.audioDuration)*

*from flat T;*

Object views provide the flexibility of looking at the same relational or object data in more than one way. Therefore, you can use different in-memory object representations for different applications without changing the way you store the data in the database.

Create and Populate the soundtable Table

The create\_soundtable.sql script creates and populates the soundtable table. This table contains a BLOB column; it is created to demonstrate how to populate a table with an Oracle Multimedia ORDAudio column from a table with a BLOB column.

This script creates the soundtable table, inserts a row with an empty BLOB, loads the BLOB with audio data, and then checks the length of the BLOB data. You can replace the name of the data file in the create\_soundtable.sql script with the name of the data file you plan to use.

Before running this script, ensure that you have the CREATE TABLE privilege.

-- create\_soundtable.sql

--

-- Create the soundtable table. This table is used ONLY to show

-- how to copy data from a BLOB column to an ORDAudio column.

--

-- Insert a row into the table with an empty BLOB.

-- Load the row with BLOB data by pointing to the audio file to

-- be loaded from the directory specified using the BFILE data

-- type.

-- Close the files and commit the transaction.

-- Check the length of the BLOB loaded. Is the length

-- what you are expecting?

--

SET SERVEROUTPUT ON;

CREATE TABLE soundtable ( id number,

sound BLOB default EMPTY\_BLOB() )

--

-- Store audio data as SecureFiles LOBs.

--

LOB(sound) STORE AS SECUREFILE;

--

INSERT INTO soundtable(id, sound) VALUES (1, EMPTY\_BLOB());

COMMIT;

DECLARE

f\_lob BFILE := BFILENAME('MEDIA\_DIR','aud1.wav');

b\_lob BLOB;

length INTEGER;

BEGIN

SELECT sound INTO b\_lob FROM soundtable WHERE id=1 FOR UPDATE;

-- Open the LOBs.

dbms\_lob.open(f\_lob, dbms\_lob.file\_readonly);

dbms\_lob.open(b\_lob, dbms\_lob.lob\_readwrite);

-- Populate the BLOB from the 'aud1.wav' file in the BFILE.

dbms\_lob.loadfromfile (b\_lob, f\_lob, dbms\_lob.getlength(f\_lob));

-- Close the LOBs.

dbms\_lob.close(b\_lob);

dbms\_lob.close(f\_lob);

COMMIT;

-- Check the length of the LOB.

SELECT dbms\_lob.getlength(t.sound) INTO length FROM soundtable t WHERE id = 1;

DBMS\_OUTPUT.PUT\_LINE('The length is '|| length);

END;

Create the audio\_table Table

The create\_audtable.sql script creates the audio\_table table with the two columns id and audio. The audio column is defined as type ORDAudio.

Before running this script, ensure that you have the CREATE TABLE privilege.

-- create\_audtable.sql

--

CREATE TABLE audio\_table ( id NUMBER,

audio ORDAudio )

LOB(audio.source.localData) STORE AS SECUREFILE;

**Задание**: Выполнить заполнение СУБД Oracle звуковыми и видеофайлами. Оформить отчет.

**Контрольные вопросы.**

Опишите процедуру заполнения базы данных мультимедийными данными.