

Program 4

CS 241
Winter 1993
Craig A. Rich

Implement the generic indexed database manager whose package specification follows:

```
generic

  type ELEMENT_TYPE is private;
  type KEY_SPACE     is private;

  with function KEY (ITEM : in ELEMENT_TYPE) return KEY_SPACE;

  KEY_POSITION : NATURAL;
  KEY_LENGTH   : POSITIVE;

package INDEXED_DATABASE is

  procedure CREATE (NAME: in STRING);
  procedure OPEN   (NAME: in STRING);
  procedure CLOSE;

  procedure ADD_SEQUENTIAL_FILE (NAME      : in STRING);
  procedure ADD                  (NEW_ITEM : in ELEMENT_TYPE);
  procedure DELETE               (KEY      : in KEY_SPACE);
  function  RETRIEVE             (KEY      : in KEY_SPACE) return ELEMENT_TYPE;
  procedure UPDATE               (NEW_ITEM : in ELEMENT_TYPE);

  DATABASE_ALREADY_OPEN : EXCEPTION;
  DATABASE_NAME_ERROR   : EXCEPTION;
  DATABASE_CREATE_ERROR : EXCEPTION;
  DATABASE_NOT_FOUND    : EXCEPTION;
  DATABASE_OPEN_ERROR   : EXCEPTION;
  DATABASE_NOT_OPEN     : EXCEPTION;
  SEQ_FILE_NOT_FOUND    : EXCEPTION;
  SEQ_FILE_OPEN_ERROR   : EXCEPTION;
  RECORD_ALREADY_EXISTS : EXCEPTION;
  RECORD_NOT_FOUND      : EXCEPTION;

end INDEXED_DATABASE;
```

INDEXED_DATABASE *Implementation notes*

1. KEY_POSITION is the byte offset within ELEMENT_TYPE at which the key field physically begins. KEY_LENGTH is the physical length in bytes of the key field within ELEMENT_TYPE. This information must be supplied in the FORM parameter when creating an indexed file (see note 3).
2. The package body should declare a single logical file *i* having indexed organization and fixed-length records of type ELEMENT_TYPE.

3. CREATE connects *i* in mode INOUT_FILE to a new physical file named NAME. Use the following actual value for the FORM parameter:

```
"KEY 0;" &
"POSITION " & NATURAL' IMAGE(KEY_POSITION) & ";" &
"LENGTH " & POSITIVE' IMAGE(KEY_LENGTH) & ";" &
"TYPE STRING;"
```

4. OPEN connects *i* in mode INOUT_FILE to an existing physical file named NAME.

5. CLOSE closes *i*.

6. ADD writes the record NEW_ITEM into *i*.

7. DELETE deletes the record of *i* whose key is KEY.

8. RETRIEVE returns the record of *i* whose key is KEY.

9. UPDATE place the record NEW_ITEM in *i*, replacing the existing record whose key is KEY(NEW_ITEM).

10. ADD_SEQUENTIAL_FILE takes records from the sequential file named NAME and adds them to *i* by calling ADD.

11. The following table specifies when exceptions are raised:

Raise exception...	from within operation...	when...
DATABASE_ALREADY_OPEN	CREATE	handling STATUS_ERROR
DATABASE_ALREADY_OPEN	OPEN	handling STATUS_ERROR
DATABASE_NAME_ERROR	CREATE	handling NAME_ERROR
DATABASE_CREATE_ERROR	CREATE	handling USE_ERROR
DATABASE_NOT_FOUND	OPEN	handling NAME_ERROR
DATABASE_OPEN_ERROR	OPEN	handling USE_ERROR
DATABASE_NOT_OPEN	CLOSE	handling STATUS_ERROR
DATABASE_NOT_OPEN	ADD	handling STATUS_ERROR
DATABASE_NOT_OPEN	DELETE	handling STATUS_ERROR
DATABASE_NOT_OPEN	RETRIEVE	handling STATUS_ERROR
DATABASE_NOT_OPEN	UPDATE	handling STATUS_ERROR
SEQ_FILE_NOT_FOUND	ADD_SEQUENTIAL_FILE	handling NAME_ERROR
SEQ_FILE_OPEN_ERROR	ADD_SEQUENTIAL_FILE	handling USE_ERROR
RECORD_ALREADY_EXISTS	ADD	handling USE_ERROR or KEY_ERROR
RECORD_NOT_FOUND	DELETE	handling EXISTENCE_ERROR
RECORD_NOT_FOUND	RETRIEVE	handling EXISTENCE_ERROR
RECORD_NOT_FOUND	UPDATE	handling EXISTENCE_ERROR

Construct an interactive procedure named PROGRAM_4 which takes queries from a user and manages the database program_4.dat, which is a file having indexed organization containing fixed-length records defined by the Ada record type STUDENT:

```
subtype TEST_SCORE is NATURAL range 0..100;
type STUDENT is record
    NAME: STRING(1..30) := (others => ' ');
    SCORE: TEST_SCORE;
end record;
```

There are several queries which can be made:

- **New**—Create a new database named `program_4.dat`.
- **Open**—Open an existing database named `program_4.dat`.
- **Close**—Close the database.
- **File**—Add the records in the sequential file named `[cs.carich]program_2.dat` to the database.
- **Add**—Prompt the user for a name and score and add the new record to the database.
- **Delete**—Prompt the user for a name and delete the record in the database.
- **Retrieve**—Prompt the user for a name, retrieve the record from the database, and output the record information to the standard output file.
- **Update**—Prompt the user for a name and score, and update the record in the database.
- **Quit**—Quit the procedure.

For live examples of the interaction between the procedure and a user you may run my procedure `PROGRAM_4`:

```
$ run [cs.carich]program_4
```

PROGRAM_4 Implementation Notes

1. The procedure should declare functions `GET_NAME` and `GET_SCORE` which return values of type `STRING` and `TEST_SCORE`, respectively. These procedures should prompt the user for the appropriate value, then read the value and return it. These functions are useful in many queries.
2. When an exception is raised by an `INDEXED_DATABASE` operation, give an appropriate error message and loop back to the query prompt.