Web Application Programming Interface (API)

Tahaluf Training Center 2021

Chapter 02

1. Overview of Database Design
2. Create a Class Diagram
3. Overview of Package
4. Overview of Stored Procedure
5. Create a Stored Procedure

Overview of Database Design

شركة تحالف الإمارات للحلول التقنية ذ.م.م.
TAHALUF AL EMARAT TECHNICAL SOLUTIONS L.L.C.

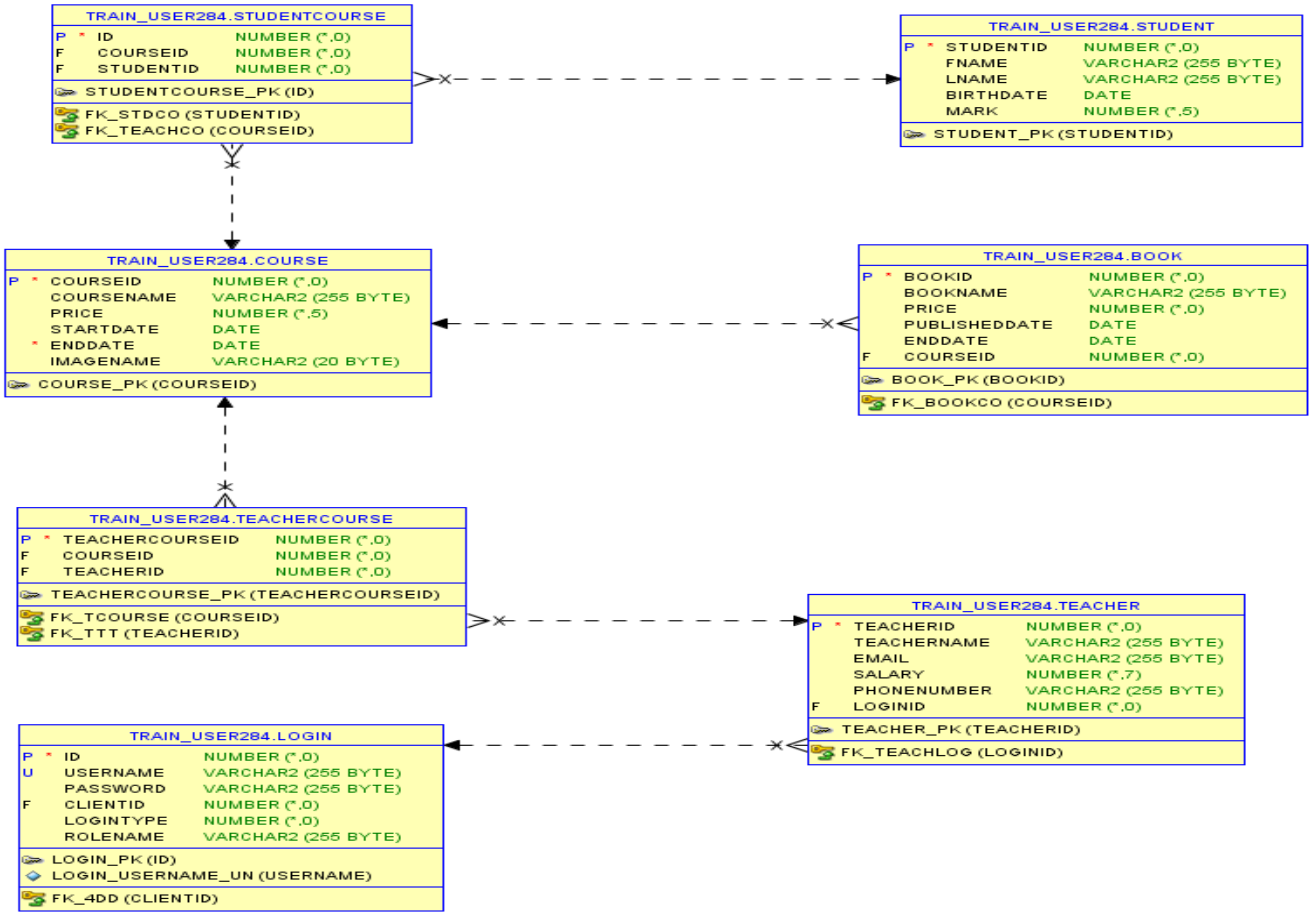
A Database is a collection of related data organized in a way that can make the process of data access, manage and update easier.

Database can be hardware based or software based in order to store data.

Exercise

Create LMS System that contains many teachers teach a many courses. Each student study many courses using a different books.

Exercise Solution



A package is a schema object used to collect logically related PL/SQL variables, types and subprograms.

Packages have two parts, a specification (header) and a body.

The specification is the interface.

The body used to define the code for the subprograms and the queries for the cursors.

Overview of Stored Procedure

Stored procedures are similar to functions.

Stored procedure is created once and can be executed more than one time.

A stored procedure is created with a CREATE PROCEDURE statement and is executed with a CALL statement.

Benefits of Stored Procedures

- Procedural logic (looping and branching), which straight SQL does not support.
- Dynamically creating a SQL command and execute it.
- Error handling.

Example

Create a stored procedure to display all courses in the database.

Example Solution

```
CREATE OR REPLACE PACKAGE Course_Package AS
PROCEDURE GetAllCourse;
END Course_Package;

CREATE OR REPLACE PACKAGE  Body Course_Package AS
PROCEDURE GetAllCourse
AS
c_all sys_refcursor;
BEGIN
open c_all for
select * from Course;
DBMS_SQL.RETURN_RESULT(c_all);
END GetAllCourse;
END Course_Package;
```

Example

Create a stored procedure to create a course in the database.

Example Solution

Update on Course_Package header => Add Create Procedure.

```
PROCEDURE CreateCourse(CourseName IN VARCHAR, Price IN
NUMBER,StartDate IN DATE,EndDate IN DATE);
```

Example Solution

Update on Course_Package body => Add Create Procedure.

```
PROCEDURE CreateCourse(CourseName IN VARCHAR, Price IN
NUMBER,StartDate IN DATE,EndDate IN DATE)
IS
BEGIN
INSERT INTO Course (CourseName,Price,StartDate,EndDate)
VALUES(CourseName, Price, TO_DATE(StartDate,'yyyy-mm-
dd'), TO_DATE(EndDate,'yyyy-mm-dd'));
COMMIT;
END CreateCourse;
```

Example

Create a stored procedure to update a course in the database.

Example Solution

Update on Course_Package header => Add Update Procedure.

```
PROCEDURE UpdateCourse(CourseId IN NUMBER, CourseName IN
VARCHAR, Price IN NUMBER, StartDate IN DATE, EndDate IN
DATE);
```

Example Solution

Update on Course_Package body => Add Update Procedure.

```
PROCEDURE UpdateCourse(CourseId IN NUMBER, CourseName IN
VARCHAR, Price IN NUMBER, StartDate IN DATE, EndDate IN
DATE)
IS
BEGIN
Update  Course  SET CourseName=CourseName, Price=Price,
StartDate=StartDate, EndDate=EndDate
WHERE CourseId=CourseId;
COMMIT;
END UpdateCourse;
```

Example

Create a stored procedure to delete a course from the database.

Example Solution

Update on Course_Package header => Add Delete Procedure.

```
PROCEDURE DeleteCourse(CourseId IN NUMBER);
```

Update on Course_Package body => Add Delete Procedure.

```
PROCEDURE DeleteCourse(CourseId IN NUMBER)
IS
BEGIN
DELETE Course WHERE CourseId=CourseId;
COMMIT;
END DeleteCourse;
```

Exercise

- ✓ Create a stored procedure to display course name and price in the database.
- ✓ Create a stored procedure to display course by course name in the database.
- ✓ Create a stored procedure to display course by price in the database.
- ✓ Create a stored procedure to display top three of cheapest course in terms of price in the database.

Exercise Solution

```
PROCEDURE GetCourseNameAndPrice;
```

```
PROCEDURE GetCourseNameAndPrice
AS
c_all sys_refcursor;
BEGIN
OPEN c_all FOR
SELECT CourseName,Price FROM Course;
DBMS_SQL.RETURN_RESULT(c_all);
END GetCourseNameAndPrice;
```

Exercise Solution

```
PROCEDURE GetByCourseName(CourseName IN VARCHAR);

PROCEDURE GetByCourseName(CourseName IN VARCHAR)
AS
c_all sys_refcursor;
BEGIN
OPEN c_all for
SELECT * FROM Course WHERE CourseName=CourseName;
END GetByCourseName;
```

Exercise Solution

```
PROCEDURE GetByCoursePrice(Price IN NUMBER);

PROCEDURE GetByCoursePrice(Price IN NUMBER)
AS
c_all sys_refcursor;
BEGIN
OPEN c_all FOR
SELECT * FROM Course
WHERE Price=price;
DBMS_SQL.RETURN_RESULT(c_all);
COMMIT;
END GetByCoursePrice;
```

Exercise Solution

```
PROCEDURE GetCheapestCourse;
```

```
PROCEDURE GetCheapestCourse
AS
c_all sys_refcursor;
BEGIN
OPEN c_all FOR
SELECT * FROM Course
WHERE ROWNUM<3
ORDER BY Price ASC;
DBMS_SQL.RETURN_RESULT(c_all);
```

Exercise

- ✓ Create a stored procedure to display a course name in the interval in the database.
- ✓ Create a stored procedure to display course by start date in the database.
- ✓ Create a stored procedure to display course by end date in the database.

Exercise Solution

```
PROCEDURE GetCourseBetweenDate(DateFrom IN DATE, DateTo
IN DATE);

PROCEDURE GetCourseBetweenDate(DateFrom IN DATE, DateTo
IN DATE)
AS
c_all sys_refcursor;
BEGIN
OPEN c_all FOR
SELECT * FROM Course WHERE StartDate>=DateFrom AND
EndDate<DateTo;
DBMS_SQL.RETURN_RESULT(c_all);
END GetCourseBetweenDate;
```

Exercise Solution

```
PROCEDURE GetCourseByDateFrom(StartAt IN DATE);

PROCEDURE GetCourseByDateFrom(StartAt IN DATE)
AS
c_all sys_refcursor;
BEGIN
OPEN c_all FOR
SELECT * FROM Course WHERE StartDate=StartAt;
DBMS_SQL.RETURN_RESULT(c_all);
END GetCourseByDateFrom;
```

Exercise Solution

```
PROCEDURE GetCourseByDateTo(EndAt IN DATE);

PROCEDURE GetCourseByDateTo(EndAt IN DATE)
AS
c_all sys_refcursor;
BEGIN
OPEN c_all FOR
SELECT * FROM Course WHERE EndDate=EndAt;
DBMS_SQL.RETURN_RESULT(c_all);
END GetCourseByDateTo;
```

Exercise

- ✓ Create a stored procedure to display a list of teacher in the database.
- ✓ Create a stored procedure to create a teacher.
- ✓ Create a stored procedure to update a teacher.
- ✓ Create a stored procedure to delete a teacher.

Example Solution

Create Teacher_Package header => Add Procedures.

```
CREATE OR REPLACE PACKAGE Teacher_Package AS
PROCEDURE CreateTeacher(TeacherName IN VARCHAR, Email IN
VARCHAR, Salary IN VARCHAR, PhoneNumber IN VARCHAR,
LoginId IN NUMBER);
PROCEDURE UpdateTeacher(TeacherId IN NUMBER,TeacherName
IN VARCHAR, Email IN VARCHAR, Salary IN VARCHAR,
PhoneNumber IN VARCHAR, LoginId IN NUMBER);
PROCEDURE GetAllTeacher;
PROCEDURE DeleteTeacher(TeacherId IN NUMBER);
END Teacher_Package;
```

Example Solution

Create Teacher_Package body => Add Procedures.

```
CREATE OR REPLACE PACKAGE BODY Teacher_Package AS
PROCEDURE CreateTeacher(TeacherName IN VARCHAR, Email IN
VARCHAR, Salary IN VARCHAR, PhoneNumber IN VARCHAR,
LoginId IN NUMBER)
IS
BEGIN
INSERT INTO Teacher(TeacherName, Email, Salary,
PhoneNumber, LoginId) VALUES (TeacherName, Email, Salary,
PhoneNumber, LoginId);
COMMIT;
END CreateTeacher;
```

Example Solution

```
PROCEDURE GetAllTeacher
AS
c_all sys_refcursor;
BEGIN
OPEN c_all FOR
SELECT * FROM Teacher;
DBMS_SQL.RETURN_RESULT(c_all);
END GetAllTeacher;
```

Example Solution

```
PROCEDURE UpdateTeacher(TeacherId IN NUMBER, TeacherName
IN VARCHAR, Email IN VARCHAR, Salary IN VARCHAR,
PhoneNumber IN VARCHAR, LoginId IN NUMBER)
IS
BEGIN
UPDATE Teacher SET TeacherName=TeacherName, Email=Email,
Salary=Salary, PhoneNumber=PhoneNumber, LoginId=LoginId
WHERE TeacherId=TeacherId;
END UpdateTeacher;
```

Example Solution

```
PROCEDURE DeleteTeacher(TeacherId IN NUMBER)
IS
BEGIN
DELETE Teacher WHERE TeacherId=TeacherId;
END DeleteTeacher;
```

Exercise

- ✓ Create a stored procedure to display teacher by id in the database.
- ✓ Create a stored procedure to display teacher name by email in the database.
- ✓ Create a stored procedure to display teacher phone and email in the database.

Example Solution

Update on Teacher_Package header => Add Procedures.

```
PROCEDURE GetTeacherById(TeacherId IN NUMBER);
PROCEDURE GetTeacherNameByEmail;
PROCEDURE GetTeacherPhoneAndEmail;
```

Example Solution

Update on Teacher_Package body => Add Procedures.

```
PROCEDURE GetTeacherById(TeacherId IN NUMBER)
AS
c_all sys_refcursor;
BEGIN
OPEN c_all FOR
SELECT * FROM Teacher WHERE TeacherId=TeacherId;
DBMS_SQL.RETURN_RESULT(c_all);
END GetTeacherById;
```

Example Solution

```
PROCEDURE GetTeacherNameByEmail
AS
c_all sys_refcursor;
BEGIN
OPEN c_all FOR
SELECT * FROM Teacher WHERE LIKE '%com';
DBMS_SQL.RETURN_RESULT(c_all);
END GetTeacherNameByEmail;
```

Example Solution

```
PROCEDURE GetTeacherPhoneAndEmail
AS
c_all sys_refcursor;
BEGIN
OPEN c_all FOR
SELECT  PhoneNumber, Email FROM Teacher;
```

Exercise

- ✓ Create a stored procedure to display list of books in the database.
- ✓ Create a stored procedure to create a book in the database.
- ✓ Create a stored procedure to update a book in the database.
- ✓ Create a stored procedure to delete a book from the database.
- ✓ Create a stored procedure to retrieve a list of book depend on price.
- ✓ Create a stored procedure to retrieve most expensive book.

Example Solution

Create Book_Package header => Add Procedures.

```
REATE OR REPLACE PACKAGE Book_Package AS
PROCEDURE CreateBook(BookName IN VARCHAR, PublishedDate IN
DATE, EndDate IN DATE, Price IN NUMBER, CourseId IN NUMBER);
PROCEDURE UpdateBook (BookId IN NUMBER, BookName IN VARCHAR,
PublishedDate IN DATE, EndDate IN DATE, Price IN NUMBER,
CourseId IN NUMBER);
PROCEDURE GetAllBook;
PROCEDURE DeleteBook(BookId IN NUMBER);
PROCEDURE GetBookByAscendingPrice;
PROCEDURE GetMostExpensiveBook;
END Book_Package;
```

Example Solution

Create Book_Package body => Add Procedures.

```
CREATE OR REPLACE PACKAGE  Body Book_Package AS
PROCEDURE CreateBook
(BookName IN VARCHAR, PublishedDate IN DATE, EndDate IN
DATE, Price IN NUMBER, CourseId IN NUMBER)
IS
BEGIN
INSERT INTO Book(BookName, Price, PublishedDate, EndDate,
CourseId) VALUES(BookName, Price,
TO_DATE(PublishedDate,'yyyy-mm-dd'), TO_DATE(EndDate,'yyyy-
mm-dd'), CourseId);
COMMIT;
END CreateBook;
```

Example Solution

```
PROCEDURE UpdateBook (BookId IN NUMBER, BookName IN VARCHAR,
PublishedDate IN DATE, EndDate IN DATE, Price IN NUMBER,
CourseId IN NUMBER)
IS
BEGIN
UPDATE Book SET
BookName=BookName,vPublishedDate=PublishedDate,vEndDate=EndD
ate, Price=Price, CourseId=CourseId
WHERE BookId=BookId;
END UpdateBook;
```

Example Solution

```
PROCEDURE GetBookByAscendingPrice
AS
c_all sys_refcursor;
BEGIN
OPEN c_all FOR
SELECT * FROM Book
ORDER BY Price ASC;
DBMS_SQL.RETURN_RESULT(c_all);
END GetBookByAscendingPrice;
```

Example Solution

```
PROCEDURE GetMostExpensiveBook
AS
c_all sys_refcursor;
BEGIN
OPEN c_all FOR
SELECT * FROM Book
ORDER BY Price DESC;
DBMS_SQL.RETURN_RESULT(c_all);
END GetMostExpensiveBook;
END Book_Package;
```

References

[1]. https://docs.snowflake.com/en/sql-reference/stored-procedures-overview.html#:~:text=Stored%20procedures%20are%20loosely%20similar,executed%20with%20a%20CALL%20command

[2].https://docs.oracle.com/cd/B19306_01/appdev.102/b14261/packages.htm