SEMINAR

CHỦ ĐỀ CÁC MỨC CÔ LẬP VÀ CHẾ ĐỘ KHÓA

I. Giới thiệu về " Dữ liệu đồng thời và nhất quán"

Trong cơ sở dữ liệu một người dùng, người dùng có thể sửa đổi dữ liệu mà không cần quan tâm đến người dùng khác sửa đổi cùng một dữ liệu cùng một lúc. Tuy nhiên, trong cơ sở dữ liệu nhiều người dùng, các câu lệnh trong nhiều transaction đồng thời có thể cập nhật cùng một dữ liệu. Các transaction thực hiện đồng thời phải tạo ra kết quả nhất quán.

Cơ sở dữ liệu nhiều người dùng phải cung cấp các thông tin sau:

- Đảm bảo rằng người dùng có thể truy cập dữ liệu cùng một lúc (đồng thời dữ liệu).
- Đảm bảo rằng mỗi người dùng sẽ thấy một chế độ xem thống nhất về dữ liệu (tính nhất quán của dữ liệu), bao gồm các thay đổi có thể nhìn thấy được thực hiện bởi các transaction của chính người dùng và các transaction đã cam kết của những người dùng khác.

II. Tổng quan về các mức cô lập

Các mức cô lập này được xác định theo các hiện tượng phải được ngăn chặn giữa các transaction thực hiện đồng thời. Các hiện tượng có thể phòng ngừa là:

- Dirty reads: Một transaction đọc dữ liệu đã được update bởi một transaction khác mà chưa được commit.
- Nonrepeatable read: Một transaction đọc lại dữ liệu mà nó đã đọc trước đó và thấy rằng dữ liệu đã bị thay đổi do một transaction khác đã sửa đổi hoặc xóa dữ liệu (đã commit). Ví dụ: người dùng truy vấn một dòng và sau đó truy vấn cùng một dòng và phát hiện ra rằng dữ liệu đã thay đổi.
- Phantom reads: Một transaction chạy lại một truy vấn trả về một tập hợp các dòng và thấy rằng số dòng đã được thêm bởi một transaction khác (đã chèn thêm vào và đã commit). Ví dụ, một transaction truy vấn số lượng nhân viên. Năm phút sau, nó thực hiện cùng một truy vấn, nhưng bây giờ số lượng đã tăng lên một vì một người dùng khác đã chèn một dòng dữ liệu

mới vào. Nhiều dữ liệu hơn đã được truy vấn, nhưng không giống như trong Nonrepeatable read, dữ liệu đọc trước đó không thay đổi.

II.2 Các hiện tượng được ngăn ngừa bởi các mức cô lập

Isolation Level	Dirty Read	Nonrepeatable Read	Phantom Read	Lost Update
Read uncommitted	Có xảy ra	Có xảy ra	Có xảy ra	Có xảy ra
Read committed	Không xảy ra	Có xảy ra	Có xảy ra	Có xảy ra
Repeatable read	Không xảy ra	Không xảy ra	Có xảy ra	Có xảy ra
Serializable	Không xảy ra	Không xảy ra	Không xảy ra	Không xảy ra

III. Tổng quan về các mức cô lập trong Oracle

III.1 Read Committed Isolation Level

Trong Read Committed , mọi truy vấn được thực hiện bởi một Transaction chỉ nhìn thấy dữ liệu đã được commit. Mức cô lập này là mặc định. Nó phù hợp với môi trường cơ sở dữ liệu trong đó ít Transaction có khả năng xung đột.

Xung đột ghi trong Read Committed:

- Trong Read Committed, một xung đột ghi xảy ra khi transaction cố gắng thay đổi một dòng đã được cập nhật bởi một transaction khác mà chưa được commit hay rollback.
- Transaction ngăn chặn sửa đổi dòng được gọi là blocking transaction. Transaction Read Committed chờ blocking transaction kết thúc và giải phóng khóa rồi transaction Read Committed mới lấy được khóa và thực hiện thay đổi.

Table 1 Xung đột ghi and Lost Updates trong READ COMMITTED Transaction

Session 1	Session 2	Explanation
SQL>SELECT empNo,salary FROM EMPLOYEE WHERE empNo in (1,2,11); EMPNO SALARY ---------- ---------- 1 6200 2 9500	No action.	Session 1 truy vấn lương của nhân viên 1, 2, và 11. Không có tìm được nhân viên có mã số nhân viên là 11.
SQL>UPDATE EMPLOYEE SET salary = 7000 WHERE empNo = 1;	No action.	Session 1 Bắt đầu transaction 1 bằng câu lệnh cập nhật lương cho nhân viên 1. Mức cô lập mặc định cho transaction này là READ COMMITTED.
No action.	SQL> SET TRANSACTION ISOLATION LEVEL READ COMMITTED;	Session 2 bắt đầu transaction 2 và đặt mức cô lập là READ COMMITTED.
No action.	SQL>SELECT empNo,salary FROM EMPLOYEE WHERE empNo in (1,2,11); EMPNO SALARY ---------- ---------- 1 6200 2 9500	Transaction 2 truy vấn lương của nhân viên 1, 2 và 11. Oracle Database sử dụng tính nhất quán để hiển thị mức lương cho nhân viên 1 lúc trước khi được cập nhật bởi transaction 1 (chưa được commit)
No action.	SQL> UPDATE employees SET salary = 9900 WHERE empNo = 2;	Transaction 2 cập nhật lương của nhân viên 2 thành công vì lúc này transaction 1 chỉ khóa dòng nhân viên 1.
SQL> INSERT INTO Employee VALUES (11,'Name11', '1/1/1998', ,8,3,'1/1/2000'), NULL,1,1,'Note1', 'mail11@com.vn');	No action.	Transaction 1 inserts 1 dòng dữ liệu nhân viên 11 vào bảng EMPLOYEE, nhưng vẫn chưa được commit.

Table 1 (tiếp tục) Xung đột ghi and Lost Updates trong READ COMMITTED Transaction

Session 1	Session 2	Explanation
No action.	SQL>SELECT empNo,salary FROM EMPLOYEE WHERE empNo in (1,2,11); EMPNO SALARY ---------- ---------- 1 6200 2 9900	Transaction 2 truy vấn lương của nhân viên 1, 2 và 11. Transaction 2 chỉ thấy cập nhật riêng mức lương cho nhân viên 2 của mình. Transaction 2 không thấy được cập nhật mức lương cho nhân viên 1 hoặc dòng dữ liệu nhân viên 11 đã thêm vào bởi transaction 1 bởi vì chưa được commit.
No action.	SQL>UPDATE EMPLOYEE SET salary = 6300 WHERE empNo = 1; -- prompt does not return	Transaction 2 cập nhật lương cho nhân viên 1, hiện transaction 1 đang giữ khóa, việc này tạo ra một xung đột ghi. Transaction 2 chờ đến khi transaction 1 kết thúc và nhả khóa.
SQL> COMMIT;	No action.	Transaction 1 commit, kết thúc transaction 1.
No action.	1 row updated. SQL>	Lúc này transaction 2 đã xin được khóa nên transaction 2 tiến hành cập nhật mức lương cho nhân viên 1.
No action.	SQL>SELECT empNo,salary FROM EMPLOYEE WHERE empNo in (1,2,11); EMPNO SALARY ---------- ---------- 1 6300 2 9900 11	Transaction 2 truy vấn lương của nhân viên 1, 2 và 11. Nhân viên 11 được thêm vào (đã được commit) bởi transaction 1 có thể nhìn thấy đối với transaction 2. Transaction 2 thấy cập nhật riêng của mình về mức lương của nhân viên 1.
No action.	COMMIT;	Transaction 2 commit, kết thúc transaction 2.

SQL>SELECT empNo,salary FROM EMPLOYEE WHERE empNo in (1,2,11); EMPNO SALARY ---------- ---------- 1 6300 2 9900 11	No action.	Mức lương của nhân viên 1 lúc này là 6300 đây là bản cập nhật được thực hiện bởi Transaction 2. Việc cập nhật mức lương của nhân viên 1 lên 7000 được thực hiện bởi transaction 1 đã bị mất đây là tình trạng Lost Update.

III.2 Serializable Isolation Level

- Ở mức cô lập tuần tự (serializable isolation level), transaction chỉ thấy những thay đổi đã được commit trước thời điểm transaction có mức cô lập tuần tự này bắt đầu và các thay đổi được thực hiện bởi chính transaction này.
- Một transaction tuần tự hoạt động trong một môi trường chỉ một mình nó như thể không có người dùng nào khác đang sửa đổi dữ liệu trong cơ sở dữ liệu. mức cô lập tuần tự phù hợp với các môi trường:
  - Với cơ sở dữ liệu lớn và các transaction ngắn chỉ cập nhật một vài dòng.
  - Trường hợp có 2 transaction đồng thời sửa đổi cùng 1 dòng là tương đối thấp.
  - Trường hợp các transaction tương đối dài chủ yếu chỉ đọc.
- Ở mức cô lập tuần tự, Bất kỳ dòng nào được đọc bởi transaction tuần tự được đảm bảo giống nhau khi đọc lại. Bất kỳ truy vấn nào cũng được đảm bảo trả về cùng kết quả trong suất thời gian transaction đó hoạt động, do đó, những thay đổi được thực hiện bởi transaction khác sẽ không hiển thị đối với truy vấn của transaction tuần tự bất kể những thay đổi của transaction khác đã chạy được bao lâu. Transaction tuần tự (serializable) ngăn ngừa được Dirty reads, Nonrepeatable reads, Phantom reads.
- Trong cơ sở dữ liệu Oracle, nếu transaction Tuần tự thực hiện thay đổi 1 dòng nào đó mà dòng này đã được thay đổi bởi một transaction khác **đã commit sau khi** transaction tuần bắt đầu thực thi thì sẽ xuất hiện 1 lỗi đó là :

```
ORA-08177: Cannot serialize access for this transaction
```

Bảng 2 Serializable Transaction

Session 1	Session 2	Explanation
SQL>SELECT empNo,salary FROM EMPLOYEE WHERE empNo in (1,2,11); EMPNO SALARY ---------- ---------- 1 6200 2 9500	No action.	Session 1 truy vấn lương của nhân viên 1, 2, và 11. Không có tìm được nhân viên có mã số nhân viên là 11.
SQL>UPDATE EMPLOYEE SET salary = 7000 WHERE empNo = 1;	No action.	Session 1 Bắt đầu transaction 1 bằng câu lệnh cập nhật lương cho nhân viên 1. Mức cô lập mặc định cho transaction này là READ COMMITTED.
No action.	SQL> SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;	Session 2 bắt đầu transaction 2 và đặt mức cô lập là SERIALIZABLE.
No action.	SQL>SELECT empNo,salary FROM EMPLOYEE WHERE empNo in (1,2,11); EMPNO SALARY ---------- ---------- 1 6200 2 9500	Transaction 2 truy vấn lương của nhân viên 1, 2 và 11. Oracle Database sử dụng tính nhất quán để hiển thị mức lương cho nhân viên 1 lúc trước khi được cập nhật bởi transaction 1 (chưa được commit)
No action.	SQL> UPDATE employees SET salary = 9900 WHERE empNo = 2;	Transaction 2 cập nhật lương của nhân viên 2 thành công vì lúc này transaction 1 chỉ khóa dòng nhân viên 1.
SQL> INSERT INTO Employee VALUES (11,'Name11', '1/1/1998', ,8,3,'1/1/2000'), NULL,1,1,'Note1', 'mail11@com.vn');	No action.	Transaction 1 inserts 1 dòng dữ liệu nhân viên 11 vào bảng EMPLOYEE, nhưng vẫn chưa được commit.

Bảng 2(tiếp tục) Serializable Transaction

Session 1	Session 2	Explanation
SQL> COMMIT;	No action.	Transaction 1 commit, kết thúc transaction 1.
SQL>SELECT empNo,salary FROM EMPLOYEE WHERE empNo in (1,2,11); EMPNO SALARY ---------- ---------- 1 7000 2 9500 11	SQL>SELECT empNo,salary FROM EMPLOYEE WHERE empNo in (1,2,11); EMPNO SALARY ---------- ---------- 1 6200 2 9900	Session 1 truy vấn mức lương của nhân viên 1, 2, 11 và xem các thay đổi được cam kết bởi transaction 1. Transaction 1 không thấy bản cập nhật lương của nhân viên 2 bởi transaction 2 vì chưa được commit Session 2 truy vấn mức lương của nhân viên 1, 2, 11. Tính nhất quán của cơ sở dữ liệu Oracle đảm bảo rằng phần insert nhân viên 11 và bản cập nhật lương của nhân viên 1 đã được commit bởi transaction 1 không hiển thị cho transaction 2. Trasaction 2 thấy bản cập nhật của chính nó đối với mức lương nhân viên 2.
No action.	COMMIT;	Transaction 2 commit, kết thúc transaction 2.
SQL>SELECT empNo,salary FROM EMPLOYEE WHERE empNo in (1,2,11); EMPNO SALARY ---------- ---------- 1 7000 2 9900 11	SQL>SELECT empNo,salary FROM EMPLOYEE WHERE empNo in (1,2,11); EMPNO SALARY ---------- ---------- 1 7000 2 9900 11	Cả hai session truy vấn mức lương của nhân viên 1, 2 và 11. Mỗi session đều thấy tất cả các thay đổi đã cam kết được thực hiện bởi transaction 1 và transaction 2.
SQL>UPDATE EMPLOYEE SET salary = 7100 WHERE empNo = 11;	No action.	Phần 1 bắt đầu transaction 3 bằng cách cập nhật mức lương của nhân viên 11. Mức cô lập mặc định cho transaction 3 là READ COMMITTED.

No action.	SQL> SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;	Session 2 bắt đầu transaction 4 và đặt mức cô lập là SERIALIZABLE.

Bảng 2(tiếp tục) Serializable Transaction

Session 1	Session 2	Explanation
No action.	SQL>UPDATE EMPLOYEE SET salary = 7200 WHERE empNo = 11; -- prompt does not return	Transaction 4 cố gắng cập nhật mức lương cho nhân viên 11, nhưng bị chặn vì transaction 3 đã khóa dòng nhân viên 11. transaction 4 phải đợi transaction 3 kết thúc và nhả khóa.
SQL> COMMIT;	No action.	Transaction 3 commit cập nhật mức lương của nhân viên 11.
No action.	UPDATE EMPLOYEE SET salary = 7200 WHERE empNo = 11; * ERROR at line 1: ORA-08177: can't serialize access for this transaction	Commit kết thúc của transaction 3 khiến bản cập nhật mức lương của nhân 11 thực hiện bởi transaction 4 bị lỗi ORA-08177. Lỗi xảy ra do transaction 3 đã commit cập nhật lương của nhân viên 11 sau khi transaction 4 bắt đầu (transaction 4 cũng cập nhật mức lương của nhân viên 11)
No action.	SQL> ROLLBACK;	Session 2 rolls back transaction 4, kết thúc transaction 4
No action.	SQL> SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;	Session 2 bắt đầu transaction 5 và đặt mức cô lập là SERIALIZABLE.
No action.	SQL>SELECT empNo,salary FROM EMPLOYEE WHERE empNo in (1,2,11); EMPNO SALARY ---------- ---------- 1 7000 2 9900 11 7100	Transaction 5 truy vấn mức lương của nhân viên 1,2 và 11. Bản cập nhật lương cho nhân viên 11 đã thực hiện bởi transaction 3 (đã commit) được hiển thị.

Bảng 2(tiếp tục) Serializable Transaction

Session 1	Session 2	Explanation
No action.	SQL>UPDATEEMPLOYEE SET salary = 7200 WHERE empNo = 11; 1 row updated.	Transaction 5 cập nhật mức lương khác cho nhân viên 11.
No action.	SQL> COMMIT;	Session 2 commit bản cập nhật và kết thúc transaction 5.

III.3 Read-only isolation level

Mức cô lập Read-only tương tự như mức cô lập Serializable, nhưng các transaction có mức cô lập Read-only chỉ được đọc dữ liệu, không được sửa đổi dữ liệu trong transaction (trừ trường hợp là người dùng là SYS)

IV. Mô tả trong đồ án môn học

IV.1 Lost update

- Mô tả tình huống: Khi thêm 2 hoặc nhiều sinh viên cùng một lớp tại cùng một thời điểm thì có khả năng xảy ra mất dữ liệu khi cập nhật sỉ số lớp.

```
CREATE OR REPLACE PROCEDURE sleep (in_time number)
AS
    v_now date;
BEGIN
    SELECT SYSDATE
    INTO v_now
    FROM DUAL;

    LOOP
        EXIT WHEN v_now + (in_time * (1/86400)) <= SYSDATE;
    END LOOP;
end;
create or replace procedure addStudent(in_studentID
```

```
Student.studentid%type, in_classID Class.ClassID%type)
as
    v_classSizes number;
begin
    set transaction isolation level read committed;
    INSERT INTO Student VALUES(in_studentID,in_classID);
    select classSizes
    into v_classSizes
    from class
    where classID = in_classID;

    update class
    set classsizes = v_classSizes + 1
    where classID = in_classID;

    sleep(10);
    commit;
end;
```

Bảng 3 Mô tả Lost Update

Session 1	Session 2	Explanation

`select classsizes`
`from class`
`where classID = 'CLA01';`
 	
`CLASSSIZES`
`----------`
`7`
		Session 1 truy vấn sỉ số của lớp CLA01 có 7 sinh viên trong lớp này.

`begin`
`addstudent('SV015','CLA01');`
`end;`
	No action.	Session 1 gọi thủ tục `addstudent` thêm sinh viên có mã số 'SV015' vào lớp 'CLA01'. Bắt đầu transaction 1

No action.	`begin`
	`addstudent('SV016','CLA01');`
	`end;`
		Session 2 gọi thủ tục `addstudent` thêm sinh viên có mã số 'SV016' vào lớp 'CLA01'. Bắt đầu transaction 2

PL/SQL procedure successfully completed.		Transaction 1 thực hiện thành công và đã kết thúc.
	PL/SQL procedure successfully completed.	Transaction 2 thực hiện thành công và đã kết thúc.
select * from student where studentID in ('SV015','SV016'); StudentID ClassID ---------- ---------- SV015 CLA01 SV016 CLA01	select * from student where studentID in ('SV015','SV016'); StudentID ClassID ---------- ---------- SV015 CLA01 SV016 CLA01	Cả 2 Session thực hiện truy vấn dữ liệu của sinh viên SV015 và SV016, cả 2 truy vấn đều có kết quả giống nhau vì vậy sinh viên SV015 và SV016 đã được thêm vào thành công.
select classsizes from class where classID = 'CLA01'; CLASSSIZES ---------- 8	No action	Session 1 truy vấn sỉ số lớp CLA01 chỉ có 8 sinh viên trong lớp này. Vậy dữ liệu đã bị sai, kết quả đúng phải là 9 sinh viên vì đã thêm thành công 2 sinh viên SV015 và SV016.

- Vấn đề xảy ra: Sỉ số lớp do transaction 1 cập nhật bị mất do transaction 2 ghi đè lên. Sỉ số lớp thấp hơn so với thực tế.
- Nguyên nhân: Transaction 1 và transaction 2 đều đọc giá trị sỉ số lớp ban đầu. transaction 2 không nhận biết được transaction 1 đã cập nhật sỉ số lớp mới trong khi đó transaction 2 vẫn sử dụng giá trị sỉ số ban đầu. Vì vậy sỉ số lớp đã bị cập nhật sai.
- Giải pháp: sử dụng câu lệnh "`set transaction isolation level serializable`" thay cho câu lệnh "`set transaction isolation level read committed`".
- Kết quả: Nếu transaction 1 thực hiện thêm sinh viên mới vào cùng một lớp rồi cập nhật sỉ số lớp và commit sau khi transaction 2 đã bắt đầu thì transaction 2 sẽ báo lỗi "`ORA-08177`" và rollback. Sau đó transaction 2 phải được thực hiện lại thì lúc này sỉ số lớp sẽ được cập nhật đúng với thực tế.

IV.2 Non repeatable read

- Mô tả tình huống: Trong khi một phiên đang xem số tín chỉ, môn học đã đăng ký được và chuẩn bị xem học phí của kỳ này là bao nhiêu, thì một phiên khác đăng ký thêm môn học. Xảy ra tình huống Unrepeatable read.

```
CREATE OR REPLACE PROCEDURE register_Subject (in_studentID
STUDENT.STUDENTID%TYPE,in_PDT_id PHONGDAOTAO.PDT_ID%TYPE,
in_subjectID SUBJECT.SUBJECTID%TYPE,in_semester
SUBJECT_REGISTRATION.SEMESTER%TYPE)
AS
    v_studentID STUDENT.STUDENTID%TYPE;
    v_semester SUBJECT_REGISTRATION.SEMESTER%TYPE;
    v_hocphi number;
    v_tinchi number;
    v_status number := 0;
BEGIN
    set TRANSACTION ISOLATION LEVEL READ COMMITTED;
    INSERT INTO SUBJECT_REGISTRATION VALUES
(REGISTRATIONID_SEQ.NEXTVAL,in_studentID,in_PDT_id,in_subje
ctID,in_semester);

    SELECT SUBJECT.NUMBEROFCREDITS
    INTO v_tinchi
    FROM SUBJECT
    WHERE SUBJECT.SUBJECTID = in_subjectID;
    v_hocphi := v_tinchi * 500000;
    BEGIN
        SELECT FEE.STUDENTID,FEE.STUDENTID
        INTO v_studentID,v_semester
        FROM FEE
        WHERE FEE.STUDENTID = in_studentID AND FEE.SEMESTER
= in_semester;
        EXCEPTION
        WHEN NO_data_found
        THEN
            v_status := 1;
    END;
    IF(v_status = 1)
    THEN
```

```
        INSERT INTO FEE VALUES
(FEE_ID_SEQ.NEXTVAL,v_hocphi,in_semester,in_studentID,0,NUL
L);
    ELSE
        UPDATE fee
        set MONEY = MONEY + v_hocphi
        WHERE fee.STUDENTID = in_studentID AND fee.SEMESTER
= in_semester;
    END IF;
    COMMIT;
END;
```

Bảng 4 Mô tả Non repeatable read

Session 1	Session 2	Explanation
`set transaction isolation level read committed;`		Session 1 thiết lập transaction 1 với mức cô lập `read committed`

`SQL>select registeredBy as StudentID,subjectName as Monhoc,numberOfCredits as tinchi from subject s,SUBJECT_REGISTRATION SR where sr.registeredBy = 'SV001' and sr.subjectID = s.subjectID;`
 	
`StudentID MonHoc TinChi`
`---------- ---------- ----------`
SV001 Subject Name 1 3
SV001 Subject Name 2 4
	No action.	Session 1 truy vấn tên môn học và số tín chỉ mà sinh viên SV001 đã đăng ký. Hiện tổng số tín chỉ của sinh viên SV001 là 7 (tương ứng với học phí là 3500000)

No action.	`BEGIN REGISTER_SUBJECT('SV001', 'PDT01','MH003' ,'2018-2019'); END;`	Session 2 bắt đầu transaction 2, gọi thủ tục `REGISTER_SUBJECT` đăng ký thêm môn học MH003 cho sinh viên SV001.
select studentID, money from fee where studentid = 'SV001'; StudentID Money ---------- ----------	No action.	Session 1 truy vấn học phí của sinh viên SV001. Nhận thấy số học phí bị sai, vì chỉ có 7 tín chỉ mà học phí lại lên đến 5000000.

SV001	5000000		

- Vấn đề xảy ra: T1 thực hiện truy vấn môn học, tín chỉ của sinh viên SV001. Vào lúc đó T2 thực hiện đăng ký môn học MH003 cho sinh viên SV001, việc đăng ký môn học này đã thay đổi thông tin học phí của SV001. Tiếp đó, T1 thực hiện truy vấn số học phí mà mình phải đóng thì thấy số học phí lại cao hơn so với số học phí của 7 tín chỉ (3.500.000).
- Nguyên nhân: T1 thiết lập mức cô lập read committed nên mỗi lần SELECT trong cùng 1 thao tác dữ liệu, nó sẽ đọc lại dữ liệu từ cơ sở dữ liệu(cơ sở dữ liệu lúc này có thể bị thay đổi) mắc dù những câu lệnh này đọc trên những đơn vị dữ liệu giống nhau.
- Giải pháp: sử dụng câu lệnh "`set transaction isolation level serializable`" thay cho câu lệnh "`set transaction isolation level read committed`".
- Kết quả: Số học phí sẽ được giữ nguyên tương ứng với số tín chỉ mà đã SELECT ban đầu. vì lúc này T1 sẽ dùng dữ liệu trước khi T2 thực hiện đăng ký thêm môn học.

IV.3 Phantom read

- Mô tả tình huống: Khi một nhân viên phòng đào tạo thực hiện thống kê danh sách sinh viên thì một nhân viên khác thực hiện thêm mới 1 sinh viên

```
create or replace procedure thongke(in_classID
Class.ClassID%type)
as
    tong_sv_students number;
    siso number;
begin
    set transaction isolation level read committed;
    select count(*) into tong_sv_students from student
where classID = in_classID;
    DBMS_OUTPUT.PUT_LINE('tong so sinh vien:' ||
tong_sv_students);
    Sleep(20);
    Select classsizes into siso from class where
classID=in_classID;
    DBMS_OUTPUT.PUT_LINE('si so:' ||siso);
    commit;
end;
```

Bảng 4 Mô tả Phantom read

Session 1	Session 2	Explanation

`Begin`
`Thongke('cla01');`
`End;`
	No action.	Session 1 gọi thủ tục thongke, bắt đầu transaction 1

No action.	`begin`
	`addstudent('st04','CLA01');`
	`end;`
		Session 2 gọi thủ tục `addstudent` thêm sinh viên có mã số 'st04' vào lớp 'CLA01'. Bắt đầu transaction 2

`PL/SQL procedure successfully completed.`	No action.	Transaction 1 thực hiện thành công và đã kết thúc.
	`PL/SQL procedure successfully completed.`	Transaction 2 thực hiện thành công và đã kết thúc.

- Vấn đề xảy ra: Session 1 đếm số sinh viên có mã lớp 'cla01' trong bảng students, kết quả ra 3. Sau đó session 2 thực hiện thêm mới sinh viên vào lớp 'cla01'. Session 1 lại truy vấn in ra sĩ số lớp 'cla01', kết quả ra 4.
- Nguyên nhân: T1 thiết lập mức cô lập read committed nên lần đếm thứ 2 sẽ tính cả sinh viên mới được thêm vào.
- Giải pháp: sử dụng câu lệnh "`set transaction isolation level serializable`" thay cho câu lệnh "`set transaction isolation level read committed`".

V. Cơ chế khóa

- "Khóa" là một cơ chế giúp ngăn chặn các tương tác "có hại" đến hệ thống.Vậy, thế nào gọi là tương tác "có hại"?
- Tương tác gọi là "có hại" khi nó cập nhật dữ liệu sai hoặc thay đổi sai cấu trúc dữ liệu, khiến dữ liệu không nhất quán giữa, thường xảy ra khi nhiều transaction cùng thao tác lên 1 đơn vị dữ liệu. Vì vậy, cơ chế khóa đóng vai trò cực kì quan trọng để đảm bảo cơ sở dữ liệu đồng thời và nhất quán.

V.1 Điều gì xảy ra bên trong cơ chế khóa?

- Hệ quản trị có thể thực thi một vài cơ chế khóa, tùy vào các thao tác khác nhau lên dữ liệu.
- Nhìn chung, hệ quản trị sử dụng 2 loại chính: **exclusive locks** (khóa độc quyền) và **share locks** (khóa chia sẻ). Tại một thời điểm, chỉ có 1 **excusive lock** được thực thi trên 1 đơn vị dữ liệu, nhưng nhiều **share locks** có thể cùng thực thi trên 1 đơn vị dữ liệu.
- Cơ chế khóa tác động đến sự tương tác giữa việc đọc và ghi dữ liệu:
  - Một dòng dữ liệu chỉ bị khóa khi nó đang được thay đổi bởi việc hành động ghi. Khi có 1 câu truy vấn cập nhật dữ liệu của 1 dòng, hệ quản trị sẽ khóa dòng đó lại.
  - Một hành động ghi lên 1 dòng dữ liệu sẽ ngăn chặn các hành động khác cùng đồng thời ghi lên dòng dữ liệu đó.
  - Hành động đọc dữ liệu không ngăn cản hành động ghi: Vì hành động đọc không khóa đơn vị dữ liệu nó đang đọc, nên hành động ghi có thể thao tác lên đơn vị dữ liệu này. Chỉ có một trường hợp ngoại lệ là câu lệnh **`Select … for update`** sẽ lock đơn vị dữ liệu nó đang đọc.
  - Hành động ghi dữ liệu không ngăn cản hành động đọc: Khi một đơn vị dữ liệu đang được thao tác bởi hành động ghi, hệ quản trị sẽ sử dụng phiên bản trước đó của đơn vị dữ liệu để trả về cho hành động đọc.

V.2 Cơ chế khóa làm được gì?

- Cơ chế khóa phải thỏa các yêu cầu quan trọng của cơ sở dữ liệu:
  - Tính nhất quán
  - Tính toàn vẹn
- Hệ quản trị Oracle đảm bảo dữ liệu đồng thời, nhất quán và toàn vẹn giữa các transactions thông qua cơ chế khóa. Cơ chế khóa diễn ra tự động và không cần sự can thiệp của người dùng.

Lấy ví dụ trong bảng trên trang chủ

T	Session 1	Session 2	Mô tả
T0	SELECT employee_id as ID, email, phone_number FROM hr.employees WHERE last_name='Himuro'; ID EMAIL PHONE_NUMBER --- ------- ------------ 118 GHIMURO 515.127.4565		Trong session 1, user hr1 truy vấn hr.employees với last_name='Himuro' và hiển thị các thuộc tính employee_id (118), email (GHIMURO), phone number (515.127.4565).
T1		SELECT employee_id as ID, email, phone_number FROM hr.employees WHERE last_name='Himuro'; ID EMAIL PHONE_NUMBER --- ------- ------------ 118 GHIMURO 515.127.4565	Trong session 2, user hr2 truy vấn hr.employees với last_name='Himuro' và hiển thị các thuộc tính employee_id (118), email (GHIMURO), phone number (515.127.4565).
T2	UPDATE hr.employees SET phone_number='515.555.1234' WHERE employee_id=118 AND email='GHIMURO' AND phone_number = '515.127.4565'; 1 row updated.		Ở session 1, user hr1 update phone_number thành 515.555.1234, Nó phát ra yếu cầu khóa trên dòng đó.
T3		UPDATE hr.employees SET phone_number='515.555.1235' WHERE employee_id=118 AND email='GHIMURO' AND phone_number = '515.127.4565'; -- SQL*Plus does not show -- a row updated message or -- return the prompt.	Tại session 2, user hr2 đang cố update trên cùng 1 dòng, nhưng đã bị block vì hr1 đang thao tác trên dòng đó.

T4	COMMIT; Commit complete.		Tại session 1, user commit transaction Commit làm thay đổi dữ liệu của dòng đó và nhả khóa cho session 2 đang đợi.
T5		0 rows updated.	Tại session 2, user hr2 nhận thấy dòng mình muốn thao tác đã bị thay đổi, không khớp với điều kiện truy vấn, session 2 không update dòng nào cả.
T6	UPDATE hr.employees SET phone_number='515.555.1235' WHERE employee_id=118 AND email='GHIMURO' AND phone_number='515.555.1234'; 1 row updated.		Tại session 1, user hr1 lại update phone_number của dòng đó thành 515.555.1235, session 1 khóa dòng đang thao tác.
T7		SELECT employee_id as ID, email, phone_number FROM hr.employees WHERE last_name='Himuro'; ID EMAIL PHONE_NUMBER --- ------- ------------ 118 GHIMURO 515.555.1234	Session 2 lúc này select dữ liệu dòng đó, sẽ nhận được kết quả mà session 1 đã commit ở t4.
T8		UPDATE hr.employees SET phone_number='515.555.1235 WHERE employee_id=118 AND email='GHIMURO' AND phone_number = '515.555.1234'; -- SQL*Plus does not show -- a row updated message or -- return the prompt.	Session 2 update cũng tại dòng đó, nhưng bị block vì session 1 đang thao tác.
T9	ROLLBACK; Rollback complete.		Session 1 rollback.

T 1 0		1 row updated.	Session 2 thực hiện update thành công vì session 1 đã rollback.
T 1 1		COMMIT; Commit complete.	Session 2 commit update, kết thúc transaction.

V.3 Các chế độ khóa

- Hệ quản trị Oracle tự động sử dụng chế độ khóa ở cấp cần thiết thấp nhất để đảm bảo khả năng truy cập đồng thời của nhiều tiến trình lên cùng đơn vị dữ liệu.
- Sử dụng chế độ khóa cấp càng thấp, dữ liệu có thể được truy cập bởi càng nhiều user. Ngược lại, nếu sử dụng mức khóa càng cao, dữ liệu càng hạn chế người truy cập.
- Hệ quản trị Oracle sử dụng 2 chế độ khóa chính cho cơ sở dữ liệu nhiều người dùng:
  - Exclusive lock mode: Transaction nhận được exclusive lock khi nó có nhu cầu thay đổi dữ liệu. Các transaction khác sẽ không thể thao tác lên đơn vị dữ liệu này cho đến khi transaction đang giữ exclusive lock nhả khóa.
  - Share lock mode: Chế độ này cho phép chia sẻ cùng 1 đơn vị dữ liệu. Nhiều giao tác đọc có thể chia sẻ dữ liệu cho nhau, mỗi giao tác giữ một share lock để ngăn hành động ghi vào cùng thời điểm của các giao tác ghi. Nhiều giao tác có thể được cấp share lock trên cùng đơn vị dữ liệu.

V.4 Vấn đề deadlock

- Deadlock là tình huống khi một hay nhiều người dùng đang chờ một đơn vị dữ liệu mà đang bị các người dùng khác cũng đang chờ đơn vị dữ liệu khác block.
- Hệ quản trị Oracle tự động phát hiện và xử lý deadlock bằng cách rollback một transaction gây deadlock, giải phóng 1 dòng đang bị khóa. Cơ sở dữ liệu sẽ trả về thông báo cho transaction bị rollback.

Lấy ví dụ trong bảng trên trang chủ

T	Session 1	Session 2	Giải thích
T0	UPDATE employees SET salary = salary*1.1 WHERE employee_id =100; 1 row updated.	UPDATE employees SET salary=salary*1.1 WHERE employee_id = 200; 1 row updated.	Session 1 và 2 cập nhập dữ liệu trên 2 đơn vị khác nhau, không có vấn đề xảy ra.
T1	UPDATE employees SET salary= salary*1.1 WHERE employee_id = 200; -- prompt does not return	UPDATE employees SET salary =salary*1.1 WHERE employee_id = 100; -- prompt does not return	Transaction 1 đang cố update dữ liệu cho nhân viên có id=200(đang bị block bởi transaction 2). Transaction 2 đang cố update dữ liệu cho nhân viên có id=100(đang bị block bởi transaction 1). Deadlock xảy ra vì không có transaction nào nhận được tài nguyên nó đang yêu cầu.
T2	UPDATE employees * ERROR at line 1: ORA-00060: deadlock detected while waiting for resource		Transaction 1 phát tín hiệu deadlock và rollback câu lệnh update tại t1. Tuy nhiên, câu update tại t0 vẫn chưa rollback.
T3	SQL> COMMIT; Commit complete.		Session 1 commit câu update tại t0, kết thúc transaction.
T4		1 row updated. SQL>	Câu update tại t1 của transaction 2(đang bị block bởi transaction 1) được thực thi.

T5		SQL> COMMIT; Commit complete.	Session 2 commits câu update tại t0 và t1, kết thúc transaction.