Ôn Tập

Phần 2. Bài tập

```
--2. Bài tập tổng hợp (??t tên các objects có prefix là "BT-")
--    a. Thiết k? CSDL (ràng bu?c, bieu thuc chinh quy…) g?m ít nh?t 5 b?ng, nh?p DL ít
nh?t 3 dòng m?i b?ng
--    b. Viết ít nhất 2 thủ tục, 2 hàm, 2 trigger cho CSDL trên
--    c. C?p quy?n cho 1 bạn trong nhóm có th? truy c?p DL, hàm, th? t?c trên

--nguoichoi: *manc, tennc, email, tendangnhap, matkhau
--nhanvat: *manv, tennv, manc(manc), ngaytaonv(sysdate)
--class: *classid, classname
--sword: *swordid, swordname, damage
--information: *manv, classid(classid), gioitinh
--equipment: *manv, swordid(swordid)
--log: *logid, logtime(sysdate), action, oldval, newval
CREATE TABLE BT_nguoichoi (
    manc VARCHAR(10) NOT NULL CHECK (regexp_like(manc, '^NC\d+$')),
    tennc VARCHAR(20) NOT NULL,
    email VARCHAR(30) CHECK (regexp_like(email, '[A-Za-z0-9]+@[a-z]+')),
    tendangnhap VARCHAR(30) NOT NULL UNIQUE,
    matkhau VARCHAR(20) NOT NULL
        CHECK (regexp_like(matkhau, '[A-Za-z0-9]{4,}')),
    CONSTRAINT PK_BT_nguoichoi PRIMARY KEY (manc)
);

alter table bt_nguoichoi
    drop constraint bt_nguoichoi_matkhau_check_regex;
alter table bt_nguoichoi
    add constraint bt_nguoichoi_matkhau_check_regex
        check (regexp_like(matkhau, '[A-Za-z0-9]{4,}?'));

--nhanvat: *manv, tennv, manc(manc), ngaytaonv(sysdate)
CREATE TABLE BT_nhanvat (
    manv VARCHAR(10) NOT NULL CHECK (regexp_like(manv, '^NV\d+$')),
    tennv VARCHAR(20) NOT NULL UNIQUE,
    manc  VARCHAR(10) NOT NULL CHECK (regexp_like(manc, '^NC\d+$')),
    ngaytaonv DATE DEFAULT sysdate,
    CONSTRAINT PK_BT_nhanvat PRIMARY KEY (manv, manc),
    CONSTRAINT FK_BT_nguoichoi_nhanvat FOREIGN KEY (manc)
            REFERENCES BT_nguoichoi (manc)
);

alter table bt_nhanvat
    drop constraint PK_BT_nhanvat;

alter table bt_nhanvat
   add constraint PK_BT_nhanvat PRIMARY KEY (manv, manc);

--class: *classid, classname
CREATE TABLE BT_class (
    classid VARCHAR(10) NOT NULL CHECK (regexp_like(classid, '^CL\d+$')),
    classname VARCHAR(20) NOT NULL UNIQUE,
    CONSTRAINT PK_BT_class PRIMARY KEY (classid)
);

--sword: *swordid, swordname, damage
CREATE TABLE BT_sword (
```

```
    swordid VARCHAR(10) NOT NULL CHECK (regexp_like(swordid, '^SO\d+$')),
    swordname VARCHAR(20) NOT NULL UNIQUE,
    damage INT not null check(damage > 0),
    CONSTRAINT PK_BT_sword PRIMARY KEY (swordid)
);

alter table BT_sword
   add damage INT not null check(damage > 0);

--information: *manv, classid(classid), gioitinh
CREATE TABLE BT_information (
    manc VARCHAR(10) NOT NULL CHECK (regexp_like(manc, '^NC\d+$')),
    manv VARCHAR(10) NOT NULL CHECK (regexp_like(manv, '^NV\d+$')),
    classid VARCHAR(10) NOT NULL CHECK (regexp_like(classid, '^CL\d+$')),
    gioitinh VARCHAR(5) DEFAULT('NAM') CHECK(gioitinh IN ('NAM', 'NU')),

    CONSTRAINT PK_BT_information PRIMARY KEY (manv, manc),
    CONSTRAINT FK_BT_class_information FOREIGN KEY (classid)
            REFERENCES BT_class (classid)
);

alter table BT_information
   add manc VARCHAR(10) NOT NULL CHECK (regexp_like(manc, '^NC\d+$'));

alter table BT_information
    drop constraint PK_BT_information;

alter table BT_information
   add constraint PK_BT_information PRIMARY KEY (manv, manc);

--equipment: *manv, swordid(swordid)
CREATE TABLE BT_equipment (
    manc VARCHAR(10) NOT NULL CHECK (regexp_like(manc, '^NC\d+$')),
    manv VARCHAR(10) NOT NULL CHECK (regexp_like(manv, '^NV\d+$')),
    swordid VARCHAR(10) NOT NULL CHECK (regexp_like(swordid, '^SO\d+$')),

    CONSTRAINT PK_BT_equipment PRIMARY KEY (manv, manc),
    CONSTRAINT FK_BT_sword_equipment FOREIGN KEY (swordid)
            REFERENCES BT_sword (swordid)
);

alter table BT_equipment
   add manc VARCHAR(10) NOT NULL CHECK (regexp_like(manc, '^NC\d+$'));

alter table BT_equipment
    drop constraint PK_BT_equipment;

alter table BT_equipment
   add constraint PK_BT_equipment PRIMARY KEY (manv, manc);

--log: *logid, logtime(sysdate), action, oldval, newval, note
CREATE TABLE BT_log (
    logid INT NOT NULL,
    logtime DATE DEFAULT sysdate,
    action VARCHAR(10) NOT NULL
        CHECK(action in ('UPDATE', 'INSERT', 'DELETE', 'PROCEDURE', 'FUNCTION')),
    oldval VARCHAR(20),
    newval  VARCHAR(20),
    note VARCHAR(20),

    CONSTRAINT PK_BT_log PRIMARY KEY (logid)
);
```

```
CREATE SEQUENCE id_log_seq
    START WITH 0
    INCREMENT BY 1
    MINVALUE 0
    NOCACHE
   NOCYCLE;

--################INSERT##############

INSERT INTO BT_nguoichoi (manc, tennc, email, tendangnhap, matkhau)
    VALUES ('NC001', 'Chevrolet', 'chevrolet@gmail.com', 'Chevrolet', 'Chev113');
INSERT INTO BT_nguoichoi (manc, tennc, email, tendangnhap, matkhau)
    VALUES ('NC002', 'Ssangyong', 'ssangyong@gmail.com', 'Ssangyong', 'Ssan113');
INSERT INTO BT_nguoichoi (manc, tennc, email, tendangnhap, matkhau)
    VALUES ('NC003', 'Aderayevans', 'aderayevans@gmail.com', 'Aderayevans', 'Ader113');
INSERT INTO BT_nguoichoi (manc, tennc, email, tendangnhap, matkhau)
    VALUES ('NC004', 'Tester', 'tester@gmail.com', 'MrTester', 'Tes1');

select * from B1510210.BT_nguoichoi;

SELECT * FROM user_cons_columns
    where constraint_name = 'SYS_C00122650';

INSERT INTO BT_nhanvat (manv, tennv, manc)
    VALUES ('NV001', 'Chevrolet001', 'NC001');
INSERT INTO BT_nhanvat (manv, tennv, manc)
    VALUES ('NV001', 'Ssangyong001', 'NC002');
INSERT INTO BT_nhanvat (manv, tennv, manc)
   VALUES ('NV001', 'Aderayevans001', 'NC003');

select * from B1510210.BT_nhanvat;

INSERT INTO BT_class (classid, classname)
    VALUES ('CL001', 'Knight');
INSERT INTO BT_class (classid, classname)
    VALUES ('CL002', 'Hunter');
INSERT INTO BT_class (classid, classname)
    VALUES ('CL003', 'Sorcerer');

select * from B1510210.BT_class;

INSERT INTO BT_sword (swordid, swordname, damage)
    VALUES ('SO001', 'Claymore', '25');
INSERT INTO BT_sword (swordid, swordname, damage)
    VALUES ('SO002', 'Crystal Greatsword', '55');
INSERT INTO BT_sword (swordid, swordname, damage)
   VALUES ('SO003', 'Uchigatana', '45');

select * from B1510210.BT_sword;

INSERT INTO BT_information (manc, manv, classid, gioitinh)
    VALUES ('NC001', 'NV001', 'CL001', 'NU');

INSERT INTO BT_information (manc, manv, classid, gioitinh)
    VALUES ('NC002', 'NV001', 'CL003', 'NAM');

INSERT INTO BT_information (manc, manv, classid, gioitinh)
    VALUES ('NC003', 'NV001', 'CL002', 'NAM');

select * from B1510210.BT_information;
```

```sql
INSERT INTO BT_equipment (manc, manv, swordid)
    VALUES ('NC001', 'NV001', 'SO001');
INSERT INTO BT_equipment (manc, manv, swordid)
    VALUES ('NC002', 'NV001', 'SO002');
INSERT INTO BT_equipment (manc, manv, swordid)
   VALUES ('NC003', 'NV001', 'SO003');

select * from B1510210.BT_equipment;

--###################PROCEDURE#########################
CREATE OR REPLACE PROCEDURE add_class
    (newclassid BT_class.classid%TYPE, newclassname BT_class.classname%TYPE)
IS
BEGIN
    INSERT INTO BT_class (classid, classname)
        VALUES (newclassid, newclassname);
    DBMS_OUTPUT.PUT_LINE('Add a new class ' || newclassname || ' succeeded');
END;

SET SERVEROUTPUT ON
EXECUTE add_class('CL004', 'Thief');

select * from B1510210.BT_class;

CREATE OR REPLACE PROCEDURE add_sword(
        newswordid      BT_sword.swordid%TYPE,
        newswordname    BT_sword.swordname%TYPE,
        newdamage    BT_sword.damage%TYPE
    )
IS
BEGIN
    INSERT INTO BT_sword (swordid, swordname, damage)
        VALUES (newswordid, newswordname, newdamage);
    DBMS_OUTPUT.PUT_LINE('Add a new sword ' || newswordname || ' (' || newdamage || ')'
|| ' succeeded');
END;

SET SERVEROUTPUT ON
EXECUTE add_sword('SO004', 'Zweihander', 50);

select * from B1510210.BT_sword;

--###################FUNCTION#########################
CREATE OR REPLACE FUNCTION get_nhanvat (
    input_manc  BT_information.manc%TYPE,
    input_manv  BT_information.manv%TYPE
)
RETURN
    VARCHAR2
IS
    thongtin VARCHAR2(100);
    newline VARCHAR(20);
    tennv   BT_nhanvat.tennv%TYPE;
    gioitinh    BT_information.gioitinh%TYPE;
    classname   BT_class.classname%TYPE;
    swordname   BT_sword.swordname%TYPE;
    damage  BT_sword.damage%TYPE;
BEGIN
--    have: manc, manv
--    essentials: tennv(BT_nhanvat), gioitinh(BT_information),
--                classname(BT_class), swordname(BT_sword), damage(BT_sword)
--    manc + manv -> BT_information + BT_equipment
```

```
--    BT_information ->(classid) BT_class
--    BT_equipment ->(swordid) BT_sword
    DBMS_OUTPUT.PUT_LINE(input_manc);
    DBMS_OUTPUT.PUT_LINE(input_manv);
    select tt.tennv, tt.gioitinh, class.classname, sword.swordname, sword.damage into
tennv, gioitinh, classname, swordname, damage
    from
    (
        select nv.tennv, info.gioitinh, info.classid, equip.swordid
        from BT_nhanvat nv
            left join BT_information info
                on (nv.manc = info.manc and nv.manv = info.manv)
            left join BT_equipment equip
                on (nv.manc = equip.manc and nv.manv = equip.manv)
        where nv.manc = input_manc and nv.manv = input_manv
    ) tt
        left join BT_class class
            on tt.classid = class.classid
        left join BT_sword sword
            on tt.swordid = sword.swordid;

--    Nhân vật Chevrolet001
--    Giới tính: Nữ
--    Class: Knight
--    Sword: Claymore (25 damages)
    newline := CHR(13) || CHR(10);
    thongtin := 'Nhân vật ' || tennv || newline ||
                    'Giới tính: ' || gioitinh || newline ||
                    'Class: ' || classname || newline ||
                    'Sword: ' || swordname || ' (' || damage || ' damages)';
    RETURN thongtin;
END;

SET SERVEROUTPUT ON
DECLARE
    result VARCHAR(100);
BEGIN
    result := get_nhanvat('NC003', 'NV001');
    DBMS_OUTPUT.PUT_LINE(result);
END;

--select tt.tennv, tt.gioitinh, class.classname, sword.swordname, sword.damage
--from
--(
--    select nv.tennv, info.gioitinh, info.classid, equip.swordid
--    from BT_nhanvat nv
--        left join BT_information info
--            on (nv.manc = info.manc and nv.manv = info.manv)
--        left join BT_equipment equip
--            on (nv.manc = equip.manc and nv.manv = equip.manv)
--    where nv.manc = 'NC001' and nv.manv = 'NV001'
--) tt
--    left join BT_class class
--        on tt.classid = class.classid
--    left join BT_sword sword
--        on tt.swordid = sword.swordid;

CREATE OR REPLACE FUNCTION get_damage (
    input_swordname  BT_sword.swordname%TYPE
)
RETURN
    BT_sword.damage%TYPE
```

```
IS
    result BT_sword.damage%TYPE;
BEGIN
    select BT_sword.damage into result
    from BT_sword
    where BT_sword.swordname = input_swordname;
    RETURN result;
END;

select * from BT_sword;

SET SERVEROUTPUT ON
DECLARE
    result BT_sword.damage%TYPE;
BEGIN
    result := get_damage('Uchigatana');
    DBMS_OUTPUT.PUT_LINE('Uchigatana: ' || result);
END;

--###################TRIGGER#########################

CREATE OR REPLACE TRIGGER check_before_update_damage
     BEFORE UPDATE OF damage on BT_SWORD
     FOR EACH ROW
BEGIN
    IF :new.damage = :old.damage THEN
        raise_application_error(-20225, 'Damage is the same as before');
    END IF;
END;

select * from BT_sword;

update Bt_sword
set damage = 50
where swordname = 'Zweihander';

CREATE OR REPLACE TRIGGER write_log
     AFTER UPDATE OF damage on BT_SWORD
     FOR EACH ROW
BEGIN
    INSERT INTO BT_log (logid, logtime, action, oldval, newval)
        VALUES (id_log_seq.nextval, sysdate, 'UPDATE', :old.damage, :new.damage);
END;

select * from BT_sword;

update Bt_sword
set damage = 52
where swordname = 'Zweihander';

select * from BT_nguoichoi;
select * from BT_nhanvat;
select * from BT_class;
select * from BT_sword;
select * from BT_information;
select * from BT_equipment;
select * from BT_log;

###################################
--Cấp quyền cho 1 bạn trong nhóm có thể truy cập DL, hàm, thủ tục trên
GRANT SELECT, UPDATE ON BT_nguoichoi TO B1609509;
GRANT SELECT, UPDATE ON BT_nhanvat TO B1609509;
```

```
GRANT SELECT, UPDATE ON BT_class TO B1609509;
GRANT SELECT, UPDATE ON BT_sword TO B1609509;
GRANT SELECT, UPDATE ON BT_information TO B1609509;
GRANT SELECT, UPDATE ON BT_equipment TO B1609509;
GRANT SELECT, UPDATE ON BT_log TO B1609509;

GRANT EXECUTE ON add_class TO B1609509;
GRANT EXECUTE ON add_sword TO B1609509;
GRANT EXECUTE ON get_nhanvat TO B1609509;
GRANT EXECUTE ON get_damage TO B1609509;

COMMIT WORK;
```

Phần 1. Lý thuyết

```
--##Tóm tắt câu trả lời cho các vấn đề dưới đây:
--HQTCSDL là gì? Các HQTCSDL hiện nay
- Hệ quản trị CSDL (HQTCSDL) là phần mềm cung cấp môi trường thuận lợi và hiệu quả để
tạo lập, lưu trữ và khai thác thông tin của CSDL
- Các HQTCSDL phổ biến hiện nay bao gồm Oracle, MySQL, Microsoft SQL Server, PostgreSQL,
MongoDB, DB2, ...
--Khác nhau giữa HQTCSDL và bảng tính (vd, Excel) là gì?
- Về định nghĩa: Bảng tính là một ứng dụng máy tính tương tác để tổ chức, phân
tích và lưu trữ dữ liệu ở dạng bảng. Nhưng, cơ sở dữ liệu là một tập hợp dữ liệu có tổ
chức, thường được lưu trữ và truy cập từ một hệ thống máy tính. Đây là sự khác biệt
chính giữa bảng tính và cơ sở dữ liệu.
- Về truy cập: Một bảng tính được người dùng truy cập trực tiếp, trong khi cơ sở
dữ liệu được người dùng hoặc ứng dụng truy cập để nhập hoặc sửa đổi dữ liệu.
- Về sử dụng: Bảng tính được sử dụng cho các nhiệm vụ kế toán, trong khi cơ sở
dữ liệu được sử dụng trong các doanh nghiệp lớn để lưu trữ nhiều dữ liệu (do cơ sở dữ
liệu lưu trữ được nhiều dữ liệu hơn).
Tóm lại, sự khác biệt chính giữa bảng tính và cơ sở dữ liệu là bảng tính là một
ứng dụng máy tính giúp sắp xếp, quản lý và tính toán dữ liệu, trong khi cơ sở dữ liệu
là tập hợp các dữ liệu liên quan được tổ chức theo cách để truy cập dữ liệu dễ dàng.
--DBA là ai? Nhiệm vụ?
- DBA là nhà quản trị CSDL (Database administrator) chịu trách nhiệm về việc
bảo đảm cho sự hoạt động thông suốt của các chức năng, tính hiệu quả của các CSDL
và ứng dụng truy cập vào các CSDL của tổ chức.
- Nhiệm vụ:
● DBA với vai trò là người am hiểu nhất về dữ liệu của tổ chức, tham gia vào tất
cả các giai đoạn của đề án từ đặc tả yêu cầu, phân tích, thiết kế, cài đặt và kiểm
thử ứng dụng nhằm đảm bảo cho ứng dụng có được sự truy cập chính xác, hiệu
quả đến dữ liệu của tổ chức.
● DBA phải đảm bảo rằng HQTCSDL đã được chuẩn bị sẵn sàng cho tải mới, cài
đặt các biện pháp an ninh, đo đạc, điều chỉnh nhu cầu về lưu trữ và bộ nhớ của
ứng dụng mới đồng thời dự đoán sự ảnh hưởng của tải mới này đối với CSDL
và các ứng dụng trong hệ thống đang vận hành.
● DBA phải đảm bảo tính sẵn sàng, an ninh, toàn vẹn của hệ thống; giám sát hiệu
suất và điều chỉnh hệ thống, sao lưu và phục hồi hệ thống.
● DBA phải giúp xác định tình trạng cuối cùng của dữ liệu sử dụng bởi ứng
dụng. Liệu dữ liệu còn cần cho ứng dụng nào khác, hay cần phải lưu trữ theo
một quy định nào đó, v.v...
--Các pp bảo vệ dữ liệu
- Cấp quyền (authorization)
- Khung nhìn (Views)
- Sao lưu và phục hồi (Backup and restore)
- Toàn vẹn dữ liệu (Integrity)
- Mật hóa (Encryption)
--Giao dịch là gì?
- Giao dịch là một hành động hay một chuỗi các hành động được thực hiện bởi 1
người dùng hoặc 1 chương trình ứng dụng, trong đó có truy cập hoặc thay đổi nội
```

dung của một CSDL.
- Một giao dịch là một đơn vị luận lý của công việc trên CSDL. Nó có thể là toàn bộ chương trình, một phần của chương trình, hoặc một lệnh đơn lẻ như INSERT hay UPDATE, và nó có thể bao gồm nhiều thao tác trên CSDL.
--Các trạng thái của GD
- Hoạt động (Active): Trạng thái khởi đầu; giao dịch giữ trong trạng thái này trong khi nó đang thực hiện.
- Hoàn tất một phần (Partially Committed): Sau khi lệnh cuối cùng được thực hiện.
- Thất bại (Failed): Sau khi phát hiện rằng sự thực hiện không thể tiếp tục được nữa.
- Hủy bỏ (Aborted): Sau khi giao dịch đã bị cuộn lại (rolled back) và CSDL đã phục hồi lại trạng thái của nó trước khi khởi động giao dịch.
- Hoàn tất (Committed): Sau khi hoàn thành thành công giao dịch CSDL đạt tới trạng thái nhất quán mới.
--Thuộc tính ACID là gì?
- Tính nguyên tử (Atomicity) hay tính ’Tất cả hoặc không có gì’
- Tính nhất quán (consistency)
- Tính cô lập (Isolation)
- Tính bền vững (Duration)
--Các vấn đề cạnh tranh trong môi trường đa người dùng là gì? Giải pháp khắc phục?
- Vấn đề về mất dữ liệu đã cập nhật (lost update)
Giải pháp: ngăn không cho GD khác đọc giá trị của mục dữ liệu đang được cập nhật cho đến khi việc cập nhật hoàn tất.
- Vấn đề về sự phụ thuộc vào các GD không hoàn tất (uncommitted denpendency)
Giải pháp: ngăn không cho GD khác đọc giá trị đang được cập nhật bởi một giao dịch chưa kết thúc khác (chưa hoàn tất/chưa hủy bỏ).
- Vấn đề phân tích không nhất quán (inconsistent analysis).
Giải pháp: ngăn không cho GD khác đọc giá trị đang được cập nhật bởi một GD chưa hoàn tất khác.
--Lịch trình là gì? Lịch trình tuần tự và không tuần tự
- Lịch trình là một chuỗi các thao tác thực hiện bởi một tập hợp các GD cạnh tranh mà vẫn đảm bảo thứ tự của các thao tác trong từng GD đơn lẻ.
● Lịch trình tuần tự (serial schedule) là một lịch trình trong đó các thao tác của một GD được thực hiện liên tiếp nhau, không có bất kỳ thao tác nào của các GD khác xen vào giữa.
● Lịch trình không tuần tự (nonserial schedule) là một lịch trình trong đó các thao tác từ một tập hợp các giao dịch cạnh tranh đan xen lẫn nhau.
--Các kỹ thuật quản lý cạnh tranh
--Bi quan (lock)
- Bất kì một GD nào cần truy cập một mục dữ liệu, trước hết khóa mục đó lại bằng cách yêu cầu 1 khóa đọc (khóa chia sẻ-shared lock) cho truy cập chỉ đọc hoặc khóa ghi (khóa độc quyền-exclusive lock) cho truy cập có cả đọc và ghi.
- Nếu mục dữ liệu này chưa bị khóa bởi 1 GD khác ⇒ khóa được cấp.
- Nếu mục dữ liệu này bị khóa, HQTCSDL kiểm tra sự tương thích giữa khóa đang yêu cầu và khóa đã có.
- Nếu khóa đọc đang được yêu cầu trên 1 mục đang bị khóa bằng 1 khóa đọc ⇒ Yêu cầu được đáp ứng.
- Ngược lại, đợi đến khi khóa đã có được giải phóng.
- Một GD tiếp tục giữ khóa đến khi nó giải phóng khóa tường minh lúc thực thi/kết thúc(hủy bỏ hoặc hoàn tất). Khi nào giải phóng khóa ghi thì hiệu ứng của các thao tác ghi mới được nhìn thấy bởi các GD khác.
--Sử dụng Giao thức khóa 2 kỳ để giải quyết cạnh tranh
- Mỗi GD phải đạt được khóa (đọc hoặc ghi) trên một mục trước khi nó xử lý mục đó.
- Khi GD giải phóng 1 khóa ⇒ Không thể yêu cầu thêm khóa mới nào.
- Nếu nâng cấp khóa được cho phép ⇒ chỉ diễn ra trong kỳ phát triển, GD phải đợi đến khi GD khác giải phóng khóa đọc trên mục đó.
- Việc giáng cấp diễn ra trong kỳ co lại.
--Các vấn đề khi sd 2PL

- Vấn đề cuộn nhiều tầng
⇒ Giải pháp: giải phóng mọi khóa ở cuối GD.
- Vấn đề khóa sống livelock: bị bỏ ở tình trạng chờ mãi mãi, không thêm được khóa mới, dù HQTCSDL không bị khóa chết.
Lý do: giải thuật chờ không công bằng, không xem xét thời gian GD đợi.
⇒ Giải pháp: sử dụng hệ thống ưu tiên với giao thức càng chờ lâu hệ số ưu tiên càng cao. Hoặc sử dụng hàng đợi, đến trước được phục vụ trước.

--Deadlock

- Vấn đề khóa chết Deadlock: hai hay nhiều GD đang chờ lẫn nhau để có được các khóa đang giữ bởi đối phương.
Phát hiện khóa chết: sử dụng đồ thị chờ WFG.

--Xu ly deadlock

Ngăn chặn khóa chết hoặc phát hiện-phục hồi khóa chết (được sử dụng nhiều). Để ngăn chặn khóa chết, ta có 2 giải thuật:
● Giải thuật Wait-Die: GD cũ hơn đợi GD mới hơn, nếu không GD sẽ bị hủy và khởi động lại với cùng nhãn thời gian, cuối cùng trở thành GD đang hoạt động cũ nhất và không chết nữa.
● Giải thuật Wound-Wait: GD mới đợi GD cũ. GD cũ yêu cầu khóa đang giữ bởi GD mới ⇒ GD mới sẽ bị hủy.

--Sử dụng nhãn thời gian

Các GD xung đột vì bị cuộn lại và được khởi động lại.
- Nhãn thời gian: thời điểm bắt đầu tương đối của GD, tạo ra bằng cách sử dụng đồng hồ hệ thống (hoặc tăng số đếm luận lý) khi GD bắt đầu.
● Trường hợp GD T phát ra lệnh read(x):
o GD T yêu cầu đọc mục dữ liệu (x) được cập nhật bởi 1 GD mới hơn
ts(T) < write_timestamp(x) ⇒ GD T bị hủy, khởi động lại với TG mới
o Ngược lại, thao tác đọc được tiến hành
read_timestamp(x) = max(ts(T),read_timestamp(x))

● Trường hợp GD T phát ra lệnh write(x)
o GD T yêu cầu ghi mục dữ liệu (x) mà giá trị của nó đã được đọc bởi 1 GD mới hơn.
ts(T) < write_timestamp(x) ⇒ cuộn GD T, khởi động lại với TG mới.
o GD T yêu cầu ghi mục dữ liệu (x) mà giá trị của nó đã được ghi bởi 1 GD mới hơn
o ts(T) < write_timestamp(x) ⇒ cuộn GD T, khởi động lại với TG mới.
o Ngược lại, thao tác ghi được tiến hành, write_stamp(x)=ts(T)

--Lạc quan: 3 kỳ (đọc, kiểm tra, ghi)

- Kỳ đọc: bắt đầu GD ⇒ hành động commit
- Kỳ kiểm tra:sau kỳ đọc
- Kỳ ghi: theo sau kỳ kiểm tra thành công
● GD chỉ đọc trải qua 2 kỳ: đọc và kiểm tra. Kiểm tra các dữ liệu đọc vào các biến.
o Nếu vẫn còn là giá trị hiện hành trong các mục dữ liệu tương ứng ⇒ GD hoàn tất.
o Ngược lại, GD bị hủy và khởi động lại.
● GD cập nhật trải qua 3 kỳ: đọc, kiểm tra và ghi. Kiểm tra các dữ liệu đọc vào các biến.
o Nếu sau khi cập nhật: dữ liệu được đảm bảo tính nhất quán, khả tuần tự được duy trì ⇒ GD hoàn tất
o Ngược lại, GD bị hủy.

--Độ mịn của mục dữ liệu

- Độ mịn của mục dữ liệu là kích thước của mục dữ liệu được chọn như là một đơn vị bảo vệ bởi giao thức điều khiển cạnh tranh.
● Một số mục dữ liệu điển hình:
Toàn Bộ CSDL.
Một tập tin.
Một trang page.
Một mẫu tin.
Một giá trị của một trường trong mẫu tin.

● Sự phân cấp của độ mịn: Biểu diễn độ mịn của khóa trên một cấu trúc phân cấp, nơi mà mỗi nút sẽ biểu diễn cho các mục dữ liệu với các kích cỡ khác nhau, mỗi khi một nút bị khóa thì tất cả các nút con cháu của nó cũng bị khóa.

--Phục hồi dữ liệu

--Tại sao cần phục hồi DL

- Phục hồi CSDL để khôi phục CSDL về trạng thái đúng khi có lỗi xảy ra.

Điểm kiểm tra là gì (check point)

--Điểm kiểm tra là gì (check point)

- Điểm kiểm tra là một điểm mà tại đó sự đồng bộ giữa CSDL và tập tin nhật ký GD được ghi nhận. Khi đó, tất cả các vùng đệm phải được ghi ép buộc ra bộ lưu thứ cấp.
- Các điểm kiểm tra được lập lịch tại các khoảng thời gian định trước và tại điểm kiểm tra, các thao tác sau sẽ được tiến hành:

● Ghi tất cả các mẫu tin nhật ký có trong bộ nhớ trong ra bộ lưu trữ thứ cấp

● Ghi các khối đã được sửa đổi trong vùng đệm CSDL ra bộ lưu trữ thứ cấp

● Ghi một mẫu tin kiểm tra vào tập tin nhật ký. Mẫu tin này chứa các định danh của các GD đang hoạt động tại thời điểm kiểm tra

--Các kỹ thuật phục hồi DL

- Cập nhật trì hoãn (NO-UNDO/REDO): trì hoãn các cập nhật thực sự lên CSDL cho đến khi GD kết thúc thành công và đạt đến điểm hoàn tất, CSDL không bao giờ được cập nhật cho đến khi GD hoàn tất.
- Cập nhật tức thì: CSDL có thể được cập nhật ngay bởi các thao tác của một GD, trước khi GD tiến đến điểm hoàn tất.
- Tạo trang bóng: Trong quá trình thực hiện GD cơ chế này tạo ra các bảng hai trang , một bảng hiện tại và một bảng trang bóng, bảng trang bóng không bao giờ thay đổi, vì vậy được dùng để phục hồi CSDL.