

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**Khoa Hệ Thống Thông Tin**



**Sinh viên thực hiện:**  
Nguyễn Ngọc Công - 15520069

Nguyễn Xuân Hội – 15520268

Nguyễn Xuân Phóng - 1552

**Giảng viên hướng dẫn:**  
ThS. Đỗ Thị Minh Phụng

**ĐỒ ÁN MÔN   
HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU**

ĐỀ TÀI: ỨNG DỤNG QUẢN LÝ KHO CUNG CẤP VẬT LIỆU XÂY DỰNG

**TP. Hồ Chí Minh, Tháng 06 năm 2020**

**LỜI CẢM ƠN**

Đầu tiên, em xin gửi lời cảm ơn chân thành đến quý Thầy cô giảng viên Trường Đại học Công nghệ thông tin – Đại học Quốc gia TP.HCM và quý thầy cô khoa Hệ thống Thông tin đã giúp cho em có những kiến thức cơ bản làm nền tảng để thực hiện đồ án này.

Đặc biệt, em xin gửi lời cảm ơn và lòng biết ơn sâu sắc nhất tới cô – ThS. Đỗ Thị Minh Phụng, người đã hướng dẫn cho em trong suốt thời gian làm đồ án. Cô đã trực tiếp hướng dẫn tận tình, sửa chữa và đóng góp nhiều ý kiến quý báu giúp nhóm chúng em hoàn thành tốt báo cáo môn học của mình. Một lần nữa em chân thành cảm ơn cô và chúc cô dồi dào sức khoẻ.

Trong thời gian một học kỳ thực hiện đồ án, chúng em đã vận dụng những kiến thức nền tảng đã tích lũy đồng thời kết hợp với việc học hỏi và nghiên cứu những kiến thức mới từ thầy cô, bạn bè cũng như nhiều nguồn tài liệu tham khảo. Từ đó, nhóm chúng em đã vận dụng tối đa những gì đã thu thập được để hoàn thành một báo cáo đồ án tốt nhất. Tuy nhiên, vì kiến thức chuyên môn còn hạn chế và bản thân còn thiếu nhiều kinh nghiệm thực tiễn nên nội dung của báo cáo không tránh khỏi những thiếu sót, em rất mong nhận được sự góp ý, chỉ bảo thêm của quý thầy cô nhằm hoàn thiện những kiến thức của mình để em có thể dùng làm hành trang thực hiện tiếp các đề tài khác trong tương lai cũng như là trong việc học tập và làm việc sau này.

Một lần nữa xin gửi đến thầy cô, bạn bè lời cảm ơn chân thành và tốt đẹp nhất.

Thành phố Hồ Chí Minh, tháng 06 năm 2020

Nhóm sinh viên thực hiện

Nguyễn Ngọc Công

Nguyễn Xuân Hội

**NHẬN XÉT CỦA GIẢNG VIÊN**

**……………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………**

# TỔNG QUAN ĐỀ TÀI

## Đặt vấn đề

Ngày nay Công nghệ thông tin đã phát triển với tốc độ nhanh chóng. CNTT đã ứng dụng hầu như vào mọi hoạt động của con người từ nguyên cứu khoa học, phát triển kinh tế, quân sự, nghệ thuật, kinh doanh… Cùng với sự phát triển không ngừng về kỹ thuật máy tính và mạng điện tử, công nghệ thông tin cũng được những công nghệ có đẳng cấp cao và lần lượt chinh phục hết đỉnh cao này đến đỉnh cao khác. Mạng Internet là một trong những sản phẩm có giá trị hết sức lớn lao và ngày càng trở nên một công cụ không thể thiếu, là nền tảng chính cho sự truyền tải, trao đổi thông tin trên toàn cầu.

Giờ đây, mọi việc liên quan đến thông tin trở nên thật dễ dàng cho người sử dụng: chỉ cần có một máy tính kết nối Internet và một dòng dữ liệu truy tìm thì gần như lập tức… cả thế giới về vấn đề mà bạn đang quan tâm sẽ hiện ra, có đầy đủ thông tin, hình ảnh và thậm chí đôi lúc có cả những âm thanh nếu bạn cần…

Bằng Internet, chúng ta đã thực hiện được nhiều công việc với tốc độ nhanh hơn và chi phí thấp hơn nhiều so với cách thức truyền thống. Chính điều này, đã thúc đẩy sự khai sinh và phát triển của thương mại điện tử và chính phủ điện tử trên khắp thế giới, làm biến đổi đáng kể bộ mặt văn hóa, nâng cao chất lượng cuộc sống con người.

Trong hoạt động sản xuất, kinh doanh, giờ đây, việc công nghệ thông tin hóa quy trình kinh doanh đã khẳng định được vai trò xúc tiến và thúc đẩy sự phát triển của doanh nghiệp. Quản lý kho là một công việc rất vất vả cho người chủ shop và nhân viên bán hàng. Khi cửa hàng, doanh nghiệp ngày càng phát triển thì số lượng hàng hóa trong kho cũng ngày càng tăng cao. Quản lý kho thế nào để hàng hóa bán ra – nhập vào không bị nhầm lẫn, thất thoát là bài toán nan giải được các doanh nghiệp đặt ra. Nhận thấy việc quản lý kho bãi thủ công bằng giấy tờ hiện tại gặp nhiều bất cập và sai sót, gây ảnh hưởng lớn đến doanh nghiệp. Nhóm em quyết định lên ý tưởng thực hiện đề tài Quản trị cơ sở dữ liệu Quản lý kho cho doanh nghiệp. Với đề tài này, ứng dụng Quản lý kho có thể áp dụng cho những doanh nghiệp có hình thức kinh doanh có kho hàng. Đối với phạm vi đề tài, nhóm chúng em chọn triển khai ví dụ cho doanh nghiệp Kinh doanh Vật liệu Xây Dựng.

## Mục tiêu

Dùng những kiến thức đã học ở trường, đặc biệt là kiến thức môn Hệ quản trị cơ sở dữ liệu để xây dựng một ứng dụng Quản lý kho. Ứng dụng được xây dựng đáp ứng đầy đủ các nhu cầu cơ bản cho nghiệp vụ quản lý kho hàng.

## Công cụ sử dụng

Trong quá trình thực hiện, nhóm sử dụng một số công cụ sau:

* SQL Developer
* Oracle SQL Developer Data Modeler
* Draw.io
* IntelliJ IDEA
* Github

# PHÂN TÍCH VÀ THIẾT KẾ CƠ SỞ DỮ LIỆU

## Phân tích nghiệp vụ:

Lĩnh vực kinh doanh: Kinh doanh vật liệu xây dựng

### Những nghiệp vụ chung

* Nghiệp vụ quản lý sản phẩm: Mục đích quản lý các sản phẩm, nhà sản cung cấp, giá cả, doanh số bán hàng của từng loại sản phẩm.
* Nghiệp vụ quản lý tồn kho: Mục đích quản lý tồn kho sản phẩm, xem xét số lượng tồn, để lên phương án giảm giá thúc đẩy bán sản phẩm tồn hoặc liên hệ nhà cung cấp để mua hàng mới.
* Nghiệp vụ nhập hàng: Mục đích quản lý nhập hàng, giá cả, nhà cung cấp, quản lý các chừng từ liên quan.
* Nghiệp vụ xuất hàng: Mục đích quản lý hàng hóa xuất đi, khách hàng, quản lý các chứng từ liên quan.
* Nghiệp vụ báo cáo, thống kê: Mục đích báo cáo/ thống kê tổng quát hiệu suất công việc bản hàng, kho bãi, lượng hàng tồn, để người quản lý có các chiến lược phương án báng hàng, nhập hàng phù hợp.

### Nghiệp vụ riêng của người quản lý

* Nghiệp vụ quản lý nhân viên: Mục đích quản lý danh sách nhân viên đang làm việc, tính lương, tính hiệu xuất cũng như đánh giá nhân viên.

## Phân tích yêu cầu

### Yêu cầu chức năng

Chức năng phải đáp ứng tốt những nhu cầu cơ bản của một doanh nghiệp kinh doanh có kho hàng. Người dùng bao gồm: Người quản lý và nhân viên, trong đó người quản lý có toàn quyền với mọi chức năng của ứng dụng, nhân viên bị hạn chế một số chức năng nhất định, từ đây đến cuối đồ án, đề cập đến tác nhân người dùng có nghĩa là bao gồm cả nhân viên và người quản lý.

* **Chức năng hệ thống**

1. Chức năng đăng nhập / đăng xuất: Người dùng sau khi đăng nhập, dựa vào cơ sở dữ liệu hệ thống sẽ xác định quyền của người dùng là quyền nhân viên hay quản lý. Từ đó danh sách menu trong bảng điều khiển sẽ hiển thị những menu thích hợp.

* **Những chức năng chung**

1. Chức năng quản lý sản phẩm: Chức năng này quản lý thông tin về sản phẩm. Người dùng có thể xem danh sách sản phẩm, thực hiện các chức năng thêm, sửa, xóa, tìm kiếm sản phẩm.
2. Chức năng quản lý loại sản phẩm: Chức năng này quản lý thông tin về loại sản phẩm. Người dùng có thể xem danh sách loại sản phẩm, thực hiện các chức năng thêm, sửa, xóa, tìm kiếm loại sản phẩm.
3. Chức năng quản lý hàng tồn kho: Chức năng này quản lý số lượng hàng đang tồn trong kho. Người dùng có thể xem chi tiết số lượng còn tồn trong kho với mỗi mặt hàng.
4. Chức năng quản lý nhập hàng: Chức năng này quản lý quá trình nhập hàng của người dùng. Ở chức năng này, hóa đơn ghi lại thông tin, chi tiết nhập hàng.
5. Chức năng xuất hàng: Chức năng này quản lý quá trình xuất hàng của người dùng. Ở chức năng này, hóa đơn ghi lại thông tin, chi tiết xuất hàng.
6. Chức năng báo cáo / thống kê: Chức năng hỗ trợ người dùng có thể xuất báo cáo, thống kê doanh thu về các mặt hàng riêng biệt, cũng như doanh số xuất / nhập theo thời gian.

* **Những chức năng riêng cho người quản lý**

1. Chức năng quản lý nhân viên: Chức năng này quản lý thông tin về nhân viên. Người quản có thể xem danh sách loại sản phẩm, thực hiện các chức năng thêm, sửa, xóa, tìm kiếm nhân viên.
2. Chức năng quản lý phân quyền: Chức năng này giúp người quản lý có thể phân quyền quân viên của họ. Cho phép cấp và thu hồi quyền của mỗi nhân viên, từ đó nâng cao hiệu suất, gia tăng tính bảo mật và hiệu quả của hệ thống.

### Yêu cầu phi chức năng

**Yêu cầu về giao diện:** giao diện ứng dụng gần gũi, thân thiện. Người dùng có thể sử dụng dễ dàng, thuận tiện và có tính tương tác cao. Thao tác trên ứng dụng nhanh gọn và dễ sử dụng.

**Yêu cầu về chất lượng:**

Tính tiến hóa: Dễ dàng trong việc nâng cấp hay thêm các modules tiện ích khác.

**Tính tiện dụng:**

* Ứng dụng có giao diện trực quan, thân thiện và dễ sử dụng.
* Các đặc tả và hướng dẫn sử dụng rõ ràng.
* Thao tác đơn giản và nhanh gọn, đáp ứng nhu cầu của học sinh.

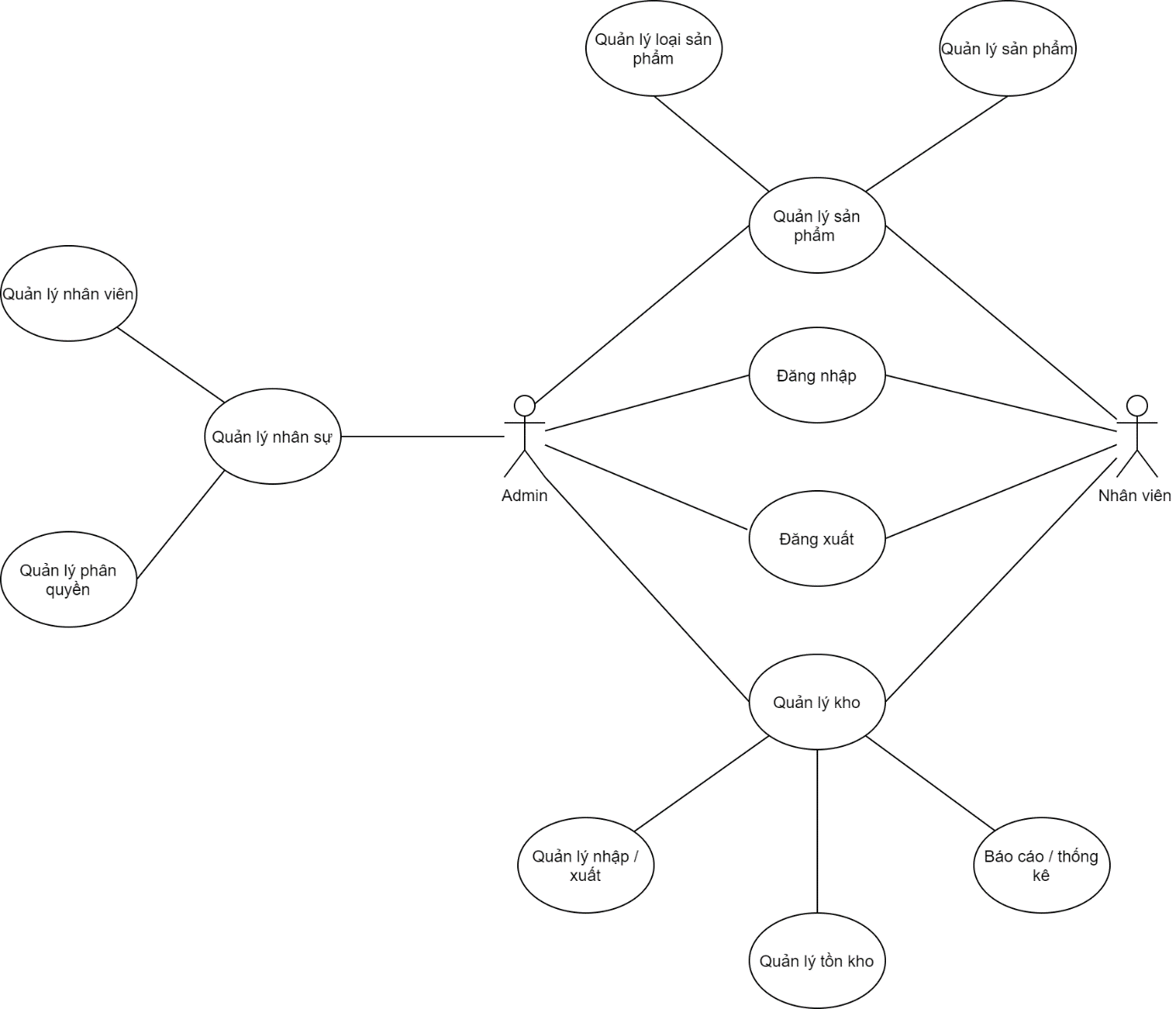
**Tính tương thích**: Hoạt động tốt trên mọi nền tảng, trên ứng dụng, mobile, tablet.

**Tính hiệu quả:**

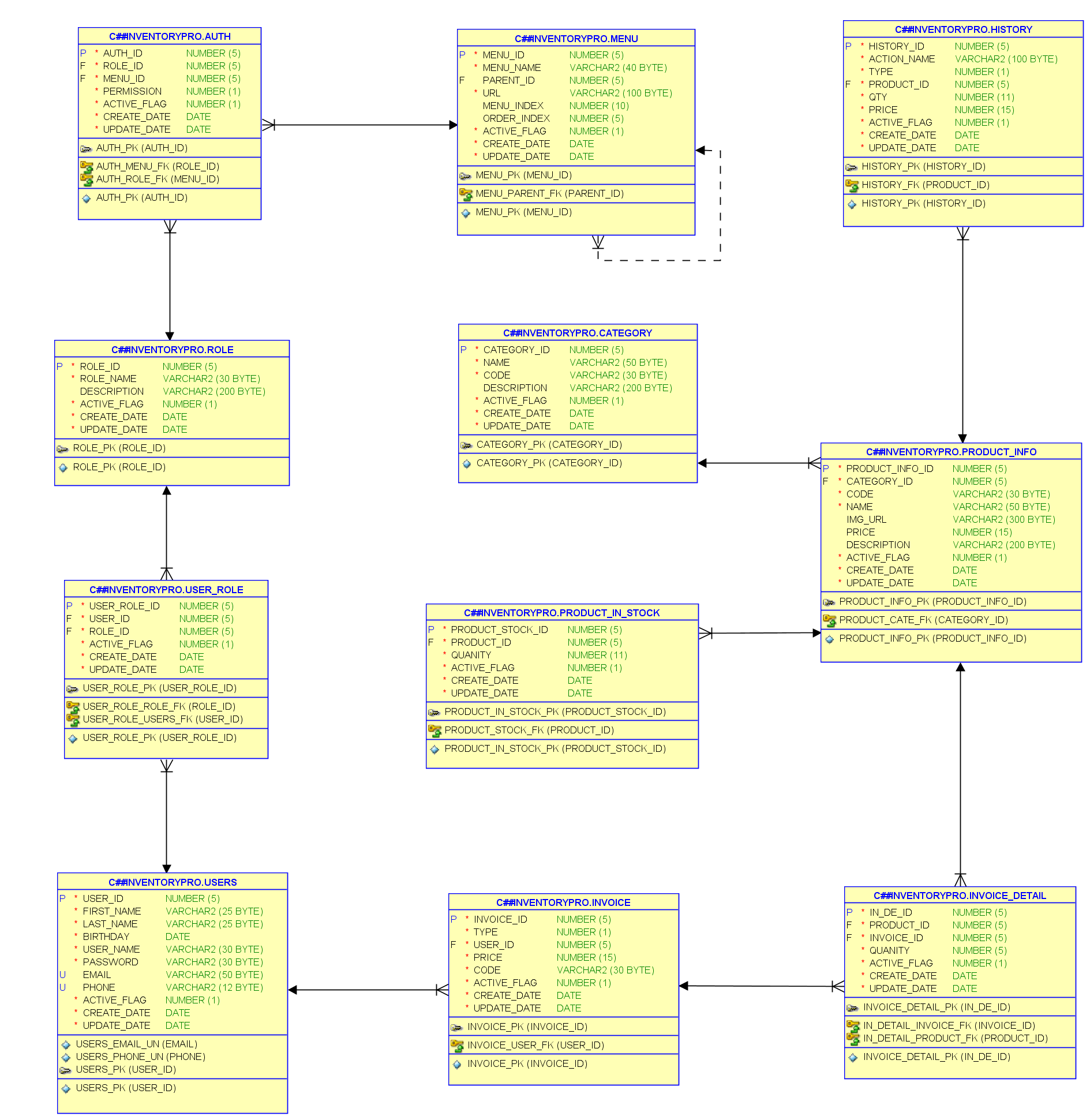
* Ứng dụng hoạt động tốt và đầy đủ những yêu cầu chức năng cơ bản của nghiệp vụ quản lý kho.
* Trang ứng dụng hoạt động ổn định, và đáng tin cậy, có thể thể truy cập và xử lý đồng thời nhiều hành động

### Thiết kế mô hình quan hệ

1. Sơ đồ Use-case



1. Mô hình dữ liệu quan hệ



1. Mô tả thành phần dữ liệu

Bảng AUTH:

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Diễn giải** |
| 1 | AUTH\_ID | NUMBER(5,0) |  |
| 2 | ROLE\_ID | NUMBER(5,0) |  |
| 3 | MENU\_ID | NUMBER(5,0) |  |
| 4 | PERMISSION | NUMBER(1,0) |  |
| 5 | ACTIVE\_FLAG | NUMBER(1,0) |  |
| 6 | CREATE\_DATE | DATE |  |
| 7 | UPDATE\_DATE | DATE |  |

Bảng CATEGORY

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Diễn giải** |
| 1 | CATEGORY\_ID | NUMBER(5,0) |  |
| 2 | NAME | VARCHAR2(50 BYTE) |  |
| 3 | CODE | VARCHAR2(30 BYTE) |  |
| 4 | DESCRIPTION | VARCHAR2(200 BYTE) |  |
| 5 | ACTIVE\_FLAG | NUMBER(1,0) |  |
| 6 | CREATE\_DATE | DATE |  |
| 7 | UPDATE\_DATE | DATE |  |

Bảng HISTORY

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Diễn giải** |
| 1 | HISTORY\_ID | NUMBER(5,0) |  |
| 2 | ACTION\_NAME | VARCHAR2(50 BYTE) |  |
| 3 | TYPE | NUMBER(1,0) |  |
| 4 | PRODUCT\_ID | NUMBER(5,0) |  |
| 5 | QTY | NUMBER(11,0) |  |
| 6 | PRICE | NUMBER(15,0) |  |
| 7 | ACTIVE\_FLAG | NUMBER(1,0) |  |
| 8 | CREATE\_DATE | DATE |  |
| 9 | UPDATE\_DATE | DATE |  |

Bảng INVOICE

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Diễn giải** |
| 1 | INVOICE\_ID | NUMBER(5,0) |  |
| 2 | TYPE | NUMBER(1,0) |  |
| 3 | USER\_ID | NUMBER(5,0) |  |
| 4 | PRICE | NUMBER(15,0) |  |
| 5 | CODE | VARCHAR2(30 BYTE) |  |
| 6 | ACTIVE\_FLAG | NUMBER(1,0) |  |
| 7 | CREATE\_DATE | DATE |  |
| 8 | UPDATE\_DATE | DATE |  |

Bảng INVOICE\_DETAIL

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Diễn giải** |
| 1 | IN\_DE\_ID | NUMBER(5,0) |  |
| 2 | PRODUCT\_ID | NUMBER(5,0) |  |
| 3 | INVOICE\_ID | NUMBER(5,0) |  |
| 4 | QUANITY | NUMBER(5,0) |  |
| 5 | ACTIVE\_FLAG | NUMBER(1,0) |  |
| 6 | CREATE\_DATE | DATE |  |
| 7 | UPDATE\_DATE | DATE |  |

Bảng MENU

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Diễn giải** |
| 1 | MENU\_ID | NUMBER(5,0) | Mã số duy nhất của menu |
| 2 | MENU\_NAME | VARCHAR2(40 BYTE) | Tên menu |
| 3 | PARENT\_ID | NUMBER(5,0) | Menu cha |
| 4 | URL | VARCHAR2(100 BYTE) | Link menu |
| 5 | ORDER\_INDEX | NUMBER(5,0) | Thứ tự sắp xếp menu |
| 6 | ACTIVE\_FLAG | NUMBER(1,0) | Trạng thái hoạt động: 0 – inactive, 1 – active |
| 7 | CREATE\_DATE | DATE | Ngày tạo |
| 8 | UPDATE\_DATE | DATE | Ngày cập nhật |

Bảng PRODUCT\_INFO

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Diễn giải** |
| 1 | PRODUCT\_INFO\_ID | NUMBER(5,0) | Mã số duy nhất của sản phẩm |
| 2 | CATEGORY\_ID | NUMBER(5,0) | Mã số thể loại |
| 3 | CODE | VARCHAR2(30 BYTE) | Mã định danh sản phẩm |
| 4 | NAME | VARCHAR2(50 BYTE) | Tên sản phẩm |
| 5 | IMG\_URL | VARCHAR2(300 BYTE) | Link hình ảnh sản phẩm |
| 6 | PRICE | NUMBER(15,0) | Giá sản phẩm |
| 7 | DESCRIPTION | VARCHAR2(200 BYTE) | Mô tả |
| 8 | ACTIVE\_FLAG | NUMBER(1,0) | Trạng thái hoạt động: 0 – inactive, 1 – active |
| 9 | CREATE\_DATE | DATE | Ngày tạo |
| 10 | UPDATE\_DATE | DATE | Ngày cập nhật |

Bảng PRODUCT\_IN\_STOCK

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Diễn giải** |
| 1 | PRODUCT\_STOCK\_ID | NUMBER(5,0) | Mã số duy nhất của bảng |
| 2 | PRODUCT\_ID | NUMBER(5,0) | Mã số sản phẩm |
| 3 | QUANITY | NUMBER(11,0) | Số lượng |
| 4 | ACTIVE\_FLAG | NUMBER(1,0) | Trạng thái hoạt động: 0 – inactive, 1 – active |
| 5 | CREATE\_DATE | DATE | Ngày tạo |
| 6 | UPDATE\_DATE | DATE | Ngày cập nhật |

Bảng ROLE

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Diễn giải** |
| 1 | ROLE\_ID | NUMBER(5,0) | Mã số duy nhất của phân quyền |
| 2 | ROLE\_NAME | VARCHAR2(30 BYTE) | Tên quyền |
| 3 | DESCRIPTION | VARCHAR2(200 BYTE) | Mô tả |
| 4 | ACTIVE\_FLAG | NUMBER(1,0) | Trạng thái hoạt động: 0 – inactive, 1 – active |
| 5 | CREATE\_DATE | DATE | Ngày tạo |
| 6 | UPDATE\_DATE | DATE | Ngày cập nhật |

Bảng USERS

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Diễn giải** |
| 1 | USER\_ID | NUMBER(5,0) | Mã số duy nhất của nhân viên |
| 2 | FIRST\_NAME | VARCHAR2(25 BYTE) | Tên người dùng |
| 3 | LAST\_NAME | VARCHAR2(25 BYTE) | Họ người dùng |
| 4 | BIRTHDAY | DATE | Ngày sinh |
| 5 | USER\_NAME | VARCHAR2(30 BYTE) | Tên đăng nhập |
| 6 | PASSWORD | VARCHAR2(30 BYTE) | Mật khẩu |
| 7 | EMAIL | VARCHAR2(50 BYTE) | Địa chỉ email |
| 8 | PHONE | VARCHAR2(12 BYTE) | Điện thoại |
| 9 | ACTIVE\_FLAG | NUMBER(1,0) | Trạng thái hoạt động: 0 – inactive, 1 – active |
| 10 | CREATE\_DATE | DATE | Ngày tạo |
| 11 | UPDATE\_DATE | DATE | Ngày cập nhật |

Bảng USER\_ROLE

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Diễn giải** |
| 1 | USER\_ROLE\_ID | NUMBER(5,0) | Mã số duy nhất của bảng |
| 2 | USER\_ID | NUMBER(5,0) | Mã người dùng |
| 3 | ROLE\_ID | NUMBER(5,0) | Mã phân quyền |
| 4 | ACTIVE\_FLAG | NUMBER(1,0) | Trạng thái hoạt động: 0 – inactive, 1 – active |
| 5 | CREATE\_DATE | DATE | Ngày tạo |
| 6 | UPDATE\_DATE | DATE | Ngày cập nhật |

# CÔNG NGHỆ SỬ DỤNG

## Oracle

* + 1. Giới thiệu Oracle

Oracle Database hay còn gọi là Oracle RDBMS hoặc đơn giản là Oracle (do đây có lẽ là sản phẩm nổi tiếng nhất của hãng), là 1 hệ quản trị cơ sở dữ liệu quan hệ, được phát triển và phân phối bởi tập đoàn Oracle.

Là hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mang tính mềm dẻo, linh động, thích ứng cao với các quy mô xử lý giao dịch, an toàn hệ thống. Cung cấp các công cụ xây dựng và quản lý cơ sở dữ liệu.

Giống như các phần mềm RDBMS khác, Oracle Database được xây dựng dựa trên ngôn ngữ SQL. Phần mềm Oracle được bổ sung PL/SQL, được Oracle phát triển nhằm bổ sung một số extension độc quyền cho SQL chuẩn – khá phổ biến trong các nhà cung cấp RDBMS.

Cơ sở dữ liệu Oracle cũng hỗ trợ lập trình bằng Java, tích hợp web (kết nối ứng dụng với công nghệ Web được tích hợp trong Oracle WebServer)

* + 1. Các phiên bản

- Phiên bản 1 (năm 1977), Phiên bản 2 (năm 1979)

- Phiên bản 3 (năm 1983), Phiên bản 4 (1984)

- Phiên bản 5 phát hành năm 1985 (SQLNet: hệ thống khách/chủ (client/server)).

- Phiên bản 6 phát hành năm 1988 (Sequence, thao tác ghi trễ).

- Oracle7 được phát hành năm 1992 (SQL\*DBA).

- Năm 1999 Oracle giới thiệu Oracle8i (i:internet).

- Năm 2001-2002: 2 phiên bản Oracle9i (Release 1&2).

- Năm 2004-2005: 2 phiên bản Oracle10g (g:Grid) (Release 1&2).

- Năm 2008 – 2009: Phiên bản 11g

- Năm 2013: Phiên bản 12c (c: cloud)

- Năm 2018: Phiên bản 18c, Cơ sở Dữ liệu Tự động của Oracle (Oracle Autonomous Databas, cloud database có khả năng tự quản lý, tự bảo vệ và tự sửa chữa nhờ công nghệ học máy (machine learning) giúp cung cấp hiệu suất cao, khả năng bảo mật mà không cần sự can thiệp của con người, với chi phí thấp hơn một nửa so với Amazon Web Services.

* + 1. Oracle 12c

Để theo kịp xu hướng Cloud Computing, PaaS (Platform as a Service), từ phiên bản Oracle 12c thì Oracle đã giới thiệu tính năng mới là Pluggable Database. Tính năng này cho phép 1 Database độc lập có thể được di chuyển dễ dàng qua các hệ thống khác nhau hoặc qua các nền tảng khác nhau hoặc giữa các bản phân phối (release) Database khác nhau.

Multitenant là 1 khái niệm mới được sử dụng trong phiên bản Oracle 12c và là 1 bước thay đổi lớn trong lịch sử kiến trúc của Oracle. Multitenant bao gồm 2 khái niệm nhỏ là Container Database (CDB) and Pluggable Database (PDB).

Khái niệm database trong 11g tương ứng với khái niệm Container Database (CDB) trong 12c. Cụ thể sau khi cài đặt phần mềmOracle 12c, có thể tạo 1 hoặc nhiều Container Database (CDB). (Thực tế là chỉ cần 1).

Trong Oracle 12C có một khái niệm mới là CDB$ROOT (Hoặc gọi là CDB Root) là một đối tượng nằm trong CDB. Các SCHEMA có thể gắn vào trên CDB$ROOT. Hoặc có các Pluggable Database (PDB) thông thường khác có gắn vào CDB$ROOT. Mỗi Pluggable Database chứa 0 hoặc nhiều SCHEMA.

PDB$SEED là một Pluggable database mẫu (Template) nó được sử dụng để làm mẫu (mặc định) để tạo ra một Pluggable Database mới. Tất nhiên có thể lấy một Pluggable Database bất kỳ nào đó làm mẫu để tạo ra một Pluggable Database mới.

Mối quan hệ giữa CDB và PDB là mối quan hệ 1-nhiều. Trong phiên bản Oracle 12c R1 thì 1 CDB có thể chứa tối đa là 250 PDB.

## Hibernate

* + 1. ORM framework

ORM (Object Relational Mapping) framework là một cơ chế cho phép người lập trình thao tác với database một cách hoàn toàn tự nhiên thông qua các đối tượng. Lập trình viên hoàn toàn không quan tâm đến loại database sử dụng SQL Server, Oracle, MySQL, PostgreSQL, …

ORM giúp đơn giản hoá việc tạo ra dữ liệu, thao tác dữ liệu và truy cập dữ liệu. Đó là một kỹ thuật lập trình để ánh xạ đối tượng vào dữ liệu được lưu trữ trong cơ sở dữ liệu.

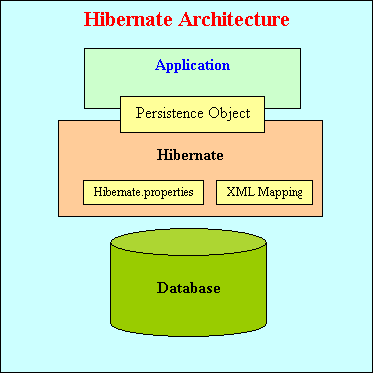
* + 1. Hibernate framework

Hibernate là một trong những ORM Framework. Hibernate framework là một framework cho persistence layer. Như vậy, nhờ có Hibernate framework mà giờ đây khi bạn phát triển ứng dụng bạn chỉ còn chú tâm vào những layer khác mà không phải bận tâm nhiều về persistence layer nữa.

Hibernate giúp lập trình viên viết ứng dụng Java có thể map các object (POJO) với hệ quản trị cơ sở dữ liệu quan hệ (database), và hỗ trợ thực hiện các khái niệm lập trình hướng đối tượng với cơ dữ liệu quan hệ.

Hibernate giúp lưu trữ và truy vấn dữ liệu quan hệ mạnh mẽ và nhanh. Hibernate cho phép bạn truy vẫn dữ liệu thông qua Java Persistence API (JPA) hoặc bằng ngôn ngữ SQL mở rộng của Hibernate (HQL) hoặc bằng SQL thuần (Native SQL).

Kiến trúc Hibernate bao gồm nhiều đối tượng như đối tượng persistent D, session factory, transaction factory, connection factory, session, transaction, …



## JPA

JPA là viết tắt của Java Persistence API, nó là một đặc tả Java cho việc ánh xạ giữa các đối tượng Java với cơ sở dữ liệu quan hệ sử dụng công nghệ phổ biến là ORM (Object Relational Mapping).

JPA cung cấp đầy đủ các công cụ cho phép chúng ta có thể thao tác với cơ sở dữ liệu một cách đơn giản và nhanh chóng. JPA có thể dùng để persist một đối tượng Java (POJO – Plain Old Java Object) vào trong cơ sở dữ liệu hoặc lấy dữ liệu từ cơ sở dữ liệu và ánh xạ (mapping) ra các đối tượng Java một cách đơn giản.

JPA hoạt động như một cầu nối giữa các table/ các mối quan hệ giữa các table trong database và các class/ mối quan hệ giữa các object. Ví dụ: table USER với các column (Id, username, password) sẽ tương ứng với class User.java với các field Id, username, password. Từ đó mỗi khi truy vấn table hay các column ta sẽ gọi trực tiếp các phương thức trên các class, các field của class mà không cần quan tâm tới việc đang dùng loại database nào, kiểu dữ liệu database ra sao, …

## Java Spring

Spring là framework phát triển ứng dụng phổ biến nhất dành cho Java Enterprise. Ban đầu nó được viết bởi Rod Johnson và lần đầu tiên được phát hành theo giấy phép Apache 2.0 vào tháng 6 năm 2003. Spring có kích thướng nhẹ, phiên bản cơ bản của Spring framework có kích thước khoảng 2MB.

Spring framework là một Java Platform mã nguồn mở, một giải pháp gọn nhẹ dành cho Java Enterprise. Với Spring Framework các nhà phát triển có thể tạo ra các mã có hiệu suất cao, dễ kiểm thử và có thể sử dụng lại được.

Các tính năng core của Spring Framework có thể được sử dụng trong việc phát triển bất kỳ ứng dụng Java nào. Bên cạnh đó, phần mở rộng được sử dụng để xây dựng các ứng dụng web trên nền tảng Java EE. Mục tiêu của Spring Framework là làm cho việc phát triển ứng dụng J2EE dễ dàng hơn và thúc đẩy việc lập trình tốt hơn bằng mô hình POJO-based.



## Github

* + 1. Git

Git là tên gọi là một hệ thống quản lý phiên bản phân tán phổ biến nhất hiện nay (Distributed Version Control System – DVCS). Git sẽ giúp người dùng lưu lại các phiên bản của những lần thay đổi vào mã nguồn và sẽ dễ dàng cho việc khôi phục lại mà không cần phải thủ công copy rồi paste vào đâu đó, phiên bản đó đã được sao lưu. Khi chúng ta phát hiện ra lỗi ở đâu đó và muốn backup lại phiên làm việc trước khi bị lỗi xảy ra thì sẽ thật đơn giản khi chúng ta sử dụng Git. Một điểm đặc biệt nữa là một thành viên trong cùng một team khi làm việc với nhau hoàn toàn có thể theo dõi online được các thay đổi của các thành viên khác ở từng phiên bản làm việc mà không nhất thiết phải ngồi ngay cạnh nhau, họ cũng có thể đối chiếu được những thay đổi đó để rồi gộp phiên bản của thành viên khác vào phiên bản của họ. Cuối cùng là tất cả có thể đưa các thay đổi vào mã nguồn của mình lên một kho chứa mã nguồn.

* + 1. Github

Github được cho ra đời với mục đích cung cấp dịch vụ Git server miễn phí. Ngoài ra trang này cũng cung cấp các giao diện để người dùng có thể dễ dàng theo dõi các sự thay đổi trên trình duyệt mà không cần phải cài phần mềm Git trên máy tính.

## IntelliJ IDEA

IntelliJ IDEA là một IDE Java thông minh cung cấp một sự kết hợp mạnh mẽ của các công cụ phát triển phần mềm. IntelliJ IDEA là công cụ nhằm tạo ra những dự án lập trình cho điện thoại hoặc cho khả năng mã hóa sâu sắc và điều hướng nhanh, phần mềm này còn cung cấp cho người dùng một danh sách các biểu tượng và ký hiệu phục vụ trong công việc lập trình của người dùng. Chức năng của IntelliJ IDEA được tiếp tục mở rộng bởi người dùng và bên thứ ba thông qua các plugin. IntelliJ IDEA cung cấp hỗ trợ cho Java EE, Spring / Hibernate và các ngăn xếp công nghệ khác.

# XÂY DỰNG VÀ QUẢN LÝ GIAO TÁC

## Trigger

* + 1. Trigger trong Oracle

Trigger là một đơn vị chương trình lưu trữ trong database và thực thi (fire) để đáp ứng một sự kiện nào đó.

Sự kiện này được kết hợp với một table, view, schema, hoặc database, và là một trong những sự kiện sau:

* Một câu lệnh DML (DELETE, INSERT, hoặc UPDATE).
* Một câu lệnh DDL (CREATE, ALTER, DROP)
* Một tác vụ trên database (SERVERERROR, LOGON, LOGOFF, STARTUP, hoặc SHUTDOWN).

**Cú pháp tạo Trigger:**

|  |
| --- |
| CREATE [OR RELACE] TRIGGER trigger\_name              {BEFORE | AFTER}              {DELETE, INSERT, UPDATE [OF column\_name….]}        ON table\_name              [REFERENCING {OLD AS old, NEW AS new}]              [FOR EACH ROW [WHEN condition]]              DECLARE                          Variable declaration;                          Constant declaration;   BEGIN               PL/SQL subprogram body;              [EXCEPTION                          exception PL/SQL block;   END; |

**Giải thích:**

CREATE [OR REPLACE ] TRIGGER trigger\_name: tạo hoặc thay thế một trigger đã tồn tại thành trigger\_name.

{BEFORE | AFTER }: cái này chỉ định khi nào trigger được thực thi.

{DELETE, INSERT, UPDATE [OF column\_name….]}chỉ định cụ thể lệnh nào thuộc loại lệnh DML được thực thi. [OF col\_name]: chỉ định cụ thể cột nào sẽ được cập nhật.

ON table\_name: chỉ định trigger sẽ được thực thi trên bảng nào.

[REFERENCING {OLD AS old, NEW AS new}]: cho phép dùng giá trị new hay old cho các lệnh DML, như là Insert, Update, Delete.

[FOR EACH ROW [WHEN condition]]: cho biết cụ thể trigger sẽ thực thi trên từng dòng. WHEN condition: cho biết dòng đó có điều kiện gì thì trigger mới thực thi được.

* + 1. Danh sách các Trigger

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên | Thao tác | Bảng | Nội dung | Ghi chú |
| 1 | update\_quantity | INSERT | INVOICE\_DETAIL | Câp nhập số hàng tồn kho khi thêm một hóa đơn nhập hàng hoặc hóa đơn xuất hàng bất kì. |  |
| 2 | check\_quanity\_in\_stock | INSERT, UPDATE | PRODUCT\_IN\_STOCK | Kiểm tra số hàng tồn kho khi thêm mới hoặc sửa một dòng có vượt quá 150. |  |
| 3 | validator\_email | INSERT, UPDATE | USERS | Kiểm tra Email khi thêm mới hoặc sửa thông tin một User có đúng chuẩn hay chưa. |  |
| 4 | update\_price\_on\_invoice | INSERT, UPDATE, DELTE | INVOICE\_DETAIL | Cập nhập tổng số tiền của hóa đơn khi thêm, sửa hoặc xóa một dòng invoice\_detail |  |
| 5 |  |  |  |  |  |

* + 1. Mô tả một số Trigger

1. Cập nhập số lượng sản phẩm khi thêm một hóa đơn

Tên: update\_quantity

Thao tác: INSERT

Trên bảng: INVOICE\_DETAIL

**Mã PL/SQL:**

|  |
| --- |
| CREATE OR REPLACE TRIGGER update\_quantity  BEFORE INSERT ON INVOICE\_DETAIL  FOR EACH ROW  DECLARE  v\_type INVOICE.TYPE%TYPE;  v\_quantity PRODUCT\_IN\_STOCK.QUANITY%TYPE;  BEGIN  SELECT TYPE INTO v\_type  FROM INVOICE  WHERE INVOICE\_ID = :NEW.INVOICE\_ID AND ACTIVE\_FLAG = 1;    SELECT QUANITY INTO v\_quantity  FROM PRODUCT\_IN\_STOCK  WHERE PRODUCT\_ID = :NEW.PRODUCT\_ID;    IF v\_type = 1 THEN  --NHAP HANG  UPDATE PRODUCT\_IN\_STOCK  SET QUANITY = QUANITY + :NEW.QUANITY  WHERE PRODUCT\_ID = :NEW.PRODUCT\_ID AND ACTIVE\_FLAG = 1;  ELSE  --XUAT HANG  IF v\_quantity - :NEW.QUANITY > 0 THEN  UPDATE PRODUCT\_IN\_STOCK  SET QUANITY = QUANITY - :NEW.QUANITY  WHERE PRODUCT\_ID = :NEW.PRODUCT\_ID AND ACTIVE\_FLAG = 1;  ELSE  raise\_application\_error(-20001, 'Quantity is not enough to export!');  END IF;  END IF;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('INSERT NEW PRODUCT\_IN\_STOCK!');  INSERT INTO PRODUCT\_IN\_STOCK(PRODUCT\_ID, QUANITY) VALUES (:NEW.PRODUCT\_ID, :NEW.QUANITY);    WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('OTHERS EXCEPTION!');  END; |

**Các bước thực hiện:**

[1]: Lấy ra loại của hóa đơn thêm vào

[2]: Lấy ra QUANITY tương ứng với product\_id thêm vào. Nếu QUANITY khác rỗng có nghĩa là product\_id đã có trong kho hàng thì ta sẽ thực hiện cập nhập số lượng, ngược lại ta sẽ thêm mới một dòng dữ liệu vào kho hàng.

[3]:Kiểm tra giá trị type.

[3.1]: Nếu type = 1 thì ta sẽ cập nhập số lượng mới sẽ bằng số lượng cũ cộng cho số lượng đơn nhập hàng.

[3.2]: Ngược lại, nếu type = 0 thì ta sẽ cập nhập số lượng mới sẽ bằng số lượng cũ trừ cho số lượng đơn nhập hàng với kết quả sau khi trừ phải lớn hơn 0.

1. Kiểm tra Email có đúng định dạng

Tên: validator\_email

Thao tác: INSERT, UPDATE

Trên bảng: USERS

**Mã PL/SQL:**

|  |
| --- |
| CREATE OR REPLACE TRIGGER validator\_email  BEFORE INSERT OR UPDATE ON USERS  FOR EACH ROW  BEGIN  IF (REGEXP\_LIKE (:NEW.EMAIL , '^[A-Za-z]+[A-Za-z0-9.]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,4}$')) THEN  DBMS\_OUTPUT.PUT('Email validate success!');  ELSE  raise\_application\_error(-20001,'Email incorrect!');  END IF;  END; |

[1]: Kiểm tra email của User khi insert hay update vào bảng USERS có đúng định dạng hay chưa:

[1.1]: Nếu đúng thì không làm gì cả

[1.2]: Nếu sai thì raise ra lỗi và dòng không được thêm hay cập nhập vào bảng.

## Store Procedure

* + 1. Store procedure trong Oracle

Một chương trình con (PL/SQL subprogram) chính là một khối lệnh PL/SQL được đặt tên và được gọi với một tập các đối số. Một chương trình con có thể là một thủ tục (procedure) hoặc là một hàm (function).

Thông thường, procedure được sử dụng để thực hiện một tác vụ nào đó còn function được sử dụng để tính toán và trả về kết quả.

Một thủ tục (procedure) và một hàm (function) có cùng cấu trúc, ngoại trừ:

Phần đầu của function phải chứa mệnh đề RETURN (return clause) xác định kiểu dữ liệu trả về. Còn procedure không chứa mệnh đề RETURN này.

Một function phải chứa ít nhất một câu lệnh RETURN (return statement) trong phần thực thi. Trong procedure, câu lệnh RETURN không bắt buộc.

**Cú pháp:**

|  |
| --- |
| CREATE [OR REPLACE] PROCEDURE procedure\_name  [ (parameter [,parameter]) ]  IS  [declaration\_section]  BEGIN  executable\_section  [EXCEPTION  exception\_section]  END [procedure\_name]; |

**Giải thích:**

* procedure\_name : Tên của procedure
* [OR REPLACE]: Thay thế những procedure đang tồn tại.
* IS: [declaration\_section] Những biến muốn khai báo thì khai báo ở đây
* executable\_section: chứa những câu lệnh của procedure.
* [EXCEPTION exception\_section]: Xữ lý những ngoại lệ khi xãy ra.
  + 1. Transaction trong Oracle

1. Định nghĩa

Một giao tác là một đơn vị công việc nguyên tố chứa một hoặc nhiều câu lệnh SQL.

Nguyên tố: không thể phân chia được nữa. Hoặc là thực hiện thành công (committed) hoặc là bị hủy bỏ (rolled back)

Tính (ACID) của giao tác: giao tác phải đảm bảo tính nguyên tố (Atomic), nhất quán (Consistency), cô lập (Isolation), bền vững (Durability).

Một session bắt đầu một transaction khi gặp câu lệnh DML và kết thúc khi gặp một câu lệnh COMMIT hoặc ROLLBACK.

1. Cấu trúc Transaction

Một transaction có bắt đầu và kết thúc.

Bắt đầu của một Transaction:

Một Transaction bắt đầu khi câu lệnh SQL đầu tiên được thực thi, bao gồm câu lệnh DDL, DML và SET TRANSACTION.

|  |
| --- |
| SET TRANSACTION  [ ISOLATION LEVEL [ SERIALIZE | READ COMMITED ]  [ NAME 'transaction\_name' ]; |

* ISOLATION LEVEL: mức cô lập của transaction sẽ có hai lựa chọn là SERIALIZE hoặc READ COMMITED.
* NAME 'transaction\_name' đặt tên cho transaction.
* Khi 1 transaction mới bắt đầu , hệ quản trị oracle sẽ gán nó vào undo data segment ( ghi nhận lại các thao tác của transaction trước khi commit, để có thể rollback khi có lỗi).

Kết thúc một Transaction:

Một Transaction kết thúc khi có bất kì hành động nào dưới đây:

* Gặp lệnh COMMIT hoặc ROLLBACK mà không có SAVEPOINT.
* Chạy một câu lệnh DDL như CREATE, DROP, RENAME hoặc ALTER.
* User ngắt kết nối đến hệ quản trị đột ngột, transaction sẽ tự động commit.
* Các ứng dụng đang kết nối đến hệ quản trị bị dừng đột ngột, transaction sẽ tự động rollback.

1. Transaction control

Gồm các lệnh để quản lý sự thay đổi của DML lên database, gồm 1 số lệnh chính:

* SAVEPOINT: Xác định 1 điểm trong transaction để rollback về khi có sự cố.
* COMMIT: Kết thúc transaction, lưu thay đổi vĩnh viễn, xóa tất cả SAVEPOINT, mở transaction locks.
* ROLLBACK: phục hồi lại dữ liệu trước khi thay đổi.

ROLLBACK:

* Hoàn tác mọi thay đổi.
* Mở tất cả khóa.
* Xóa toàn bộ savepoints.
* Kết thúc transaction.
  + 1. Danh sách Store Procedure

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| STT | Tên | Tham số đầu vào | Tham số đầu ra | Ý nghĩa | Ghi chú |
| 1 | get\_amount | Table of an Obejct with 2 properties (1\*) | NUMBER | Lấy tổng số tiền của một danh sách các mặt hàng sắp được thêm vào hóa đơn |  |
| 2 | add\_goods\_receipt\_invoice | (1\*), v\_type NUMBER, v\_user\_id NUMBER, v\_code VARCHAR2 |  | Với đầu vào là một danh sách mặt hàng để thêm vào invoice. Khi thêm invoice thì procedure này sẽ thêm danh sách invoice\_detail tương ứng. |  |
| 3 | create\_category | (v\_cate\_id NUMBER, v\_name VARCHAR2, v\_description VARCHAR2, v\_code VARCHAR2 |  | Nếu tham số đầu vào là một category đã có trong CSDL thì ta cập nhập giá trị, ngược lại nếu là category chưa có trong CLDL thì ta thêm vào. |  |
| 4 | check\_invoice\_price | v\_invoice\_id INVOICE.INVOICE\_ID%TYPE | NUMBER | Kiểm tra tổng tiền trong hóa đơn có bằng tổng tiền của các chi tiết hóa đơn |  |
| 5 | get\_quantity\_product\_in\_stock |  |  | Lấy ra quantity hiện tại của một Product |  |
| 6 | get\_info\_user | v\_id\_user USERS.USER\_ID%TYPE |  | Lấy và in ra thông tin của User với user\_id |  |
| 7 | get\_product\_price | v\_product\_id PRODUCT\_INFO.PRODUCT\_INFO\_ID%TYPE | PRODUCT\_INFO.PRICE%TYPE | Lấy giá tiền của sản phẩm với product\_info\_id |  |
| 8 | get\_all\_menu |  |  |  |  |
| 9 |  |  |  |  |  |

* + 1. Mô tả một số Store Procedure

1. Lấy tổng số tiền của một hóa đơn

Tên: get\_amount

Nội dung: Lấy tổng số tiền của một danh sách các mặt hàng sắp được thêm vào hóa đơn

Tham số đầu vào: v\_l\_info\_gr list\_info\_goods\_receipt

Tham số đầu ra: NUMBER

**Mã PL/SQL:**

|  |
| --- |
| CREATE OR REPLACE TYPE info\_goods\_receipt AS OBJECT (v\_product\_id NUMBER(5,0), v\_quantity NUMBER(5,0));  CREATE OR REPLACE TYPE list\_info\_goods\_receipt IS TABLE OF info\_goods\_receipt;  CREATE OR REPLACE FUNCTION get\_amount(v\_l\_info\_gr list\_info\_goods\_receipt)  RETURN NUMBER  AS  v\_amount NUMBER;  v\_price PRODUCT\_INFO.PRICE%TYPE;  BEGIN  v\_amount := 0;  FOR ind IN v\_l\_info\_gr.FIRST..v\_l\_info\_gr.LAST  LOOP  --GET PRICE FROM PRODUCT ID  SELECT PRICE INTO v\_price  FROM PRODUCT\_INFO  WHERE PRODUCT\_INFO\_ID = v\_l\_info\_gr(ind).v\_product\_id;  v\_amount := v\_amount + v\_l\_info\_gr(ind).v\_quantity \* v\_price;  END LOOP;    RETURN v\_amount;    EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('NO\_DATA\_FOUND!');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('OTHERS EXCEPTION!');  END; |

**Các bước thực hiện:**

Bước nằm ngoài function: tạo 2 TYPE là tham số đầu vào của function là info\_goods\_receipt và list\_info\_goods\_receipt.

[1]: Duyệt danh sách v\_l\_info\_gr là tham số đầu vào

[2]: Lấy ra giá trị PRICE của sản phẩm tương ứng với product\_id

[3]: Giá trị tổng tiền(v\_amount) của hóa đơn bằng tổng tiền trước đó cộng cho số lượng sản phẩm nhân cho số lượng mà được thêm vào.

[4]: Trả về giá trị tổng tiền

1. Thêm danh sách chi tiết hóa đơn khi thêm hóa đơn

Tên: add\_goods\_receipt\_invoice

Nội dung: Với đầu vào là một danh sách mặt hàng để thêm vào invoice. Khi thêm invoice thì procedure này sẽ thêm danh sách invoice\_detail tương ứng.

Tham số đầu vào: v\_l\_info\_gr list\_info\_goods\_receipt, v\_invoice\_id, v\_type NUMBER, v\_user\_id NUMBER, v\_code VARCHAR2

Tham số đầu ra:

**Mã PL/SQL:**

|  |
| --- |
| CREATE OR REPLACE PROCEDURE add\_goods\_receipt\_invoice(v\_l\_info\_gr list\_info\_goods\_receipt,  v\_invoice\_id NUMBER, v\_type NUMBER, v\_user\_id NUMBER, v\_code VARCHAR2)  AS  v\_mount NUMBER;  BEGIN  --INSERT INTO INVOICE  v\_mount := get\_amount(v\_l\_info\_gr);  INSERT INTO INVOICE(INVOICE\_ID, TYPE, USER\_ID, PRICE, CODE) VALUES(v\_invoice\_id, v\_type, v\_user\_id, v\_mount, v\_code);  FOR ind IN v\_l\_info\_gr.FIRST..v\_l\_info\_gr.LAST  LOOP  --INSERT INTO INVOICE DETAIL  INSERT INTO INVOICE\_DETAIL(INVOICE\_ID, PRODUCT\_ID, QUANITY)  VALUES (v\_invoice\_id, v\_l\_info\_gr(ind).v\_product\_id, v\_l\_info\_gr(ind).v\_quantity);    END LOOP;  END; |

**Các bước thực hiện:**

[1]: Tính tổng tiền cho hóa đơn

[2]: Thêm một dòng dữ liệu cho bảng INVOICE với các tham số là v\_invoice\_id, v\_type, v\_user\_id, v\_mount, v\_code.

[3]: Duyệt từng phần tử trong danh sách list\_info\_goods\_receipt và tương ứng với mỗi phần tử thì ta sẽ thêm một dòng cho bảng INVOICE\_DETAIL.

1. Tạo một category

Tên: create\_category

Nội dung: Nếu tham số đầu vào là một category đã có trong CSDL thì ta cập nhập giá trị, ngược lại nếu là category chưa có trong CLDL thì ta thêm vào.

Tham số đầu vào: v\_cate\_id NUMBER, v\_name VARCHAR2, v\_description VARCHAR2, v\_code VARCHAR2

Tham số đầu ra:

**Mã PL/SQL:**

|  |
| --- |
| CREATE OR REPLACE PROCEDURE create\_category(v\_cate\_id NUMBER, v\_name VARCHAR2,  v\_description VARCHAR2, v\_code VARCHAR2)  AS  v\_id NUMBER;  BEGIN  SELECT CATEGORY\_ID INTO v\_id  FROM CATEGORY  WHERE CATEGORY\_ID = v\_cate\_id;    IF SQL%ROWCOUNT = 1 THEN  DBMS\_OUTPUT.PUT\_LINE('UPDATE CATEGORY!');  UPDATE CATEGORY  SET NAME = v\_name, DESCRIPTION = v\_description, CODE = v\_code  WHERE CATEGORY\_ID = v\_cate\_id;  END IF;    EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('INSERT CATEGORY!');  INSERT INTO CATEGORY(CATEGORY\_ID, NAME, DESCRIPTION, CODE)  VALUES(v\_cate\_id, v\_name, v\_description, v\_code);  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('OTHERS EXCEPTION!');  END; |

**Các bước thực hiện:**

[1]: Lấy ra category\_id tương ứng với giá trị v\_cate\_id tham số đầu vào để kiểm tra giá trị v\_cate\_id đã có trong bảng CATEGORY hay chưa

[1.1]: Nếu đã có giá category\_id trong bảng CATEGORY thì ta sẽ cập nhập giá trị mới cho category đó.

[1.2]: Ngược lại nếu chưa có giá category\_id trong bảng CATEGORY thì ta sẽ thêm một dòng vàobảng CATEGORY.

# XỬ LÝ TRUY XUẤT ĐỒNG THỜI

## Các mức cô lập trong Oracle

* + 1. Read committed

a. 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.

b. Xung đột 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.

c. Ưu điểm và khuyết điểm

* Ưu điểm:
  + Giải quyết vấn đề Dirty Reads.
  + Shared Lock được giải phóng ngay, không cần phải giữ cho đến hết giao tác nên không cản trở nhiều đến thao tác cập nhật của các giao tác khác.
* Khuyết điểm:
  + Chưa giải quyết được vấn đề Unrepeatable Reads, Phantoms, Lost Updates.
  + Phải chờ nếu đơn vị dữ liệu cần đọc đang được giữ khoá ghi.
    1. Serializable

1. 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.

1. Ưu điểm và khuyết điểm:

* Ưu điểm:
  + Giải quyết thêm được vấn đề Phantoms.
* Khuyết điểm:
  + Phải chờ nếu đơn vị dữ liệu cần đọc đang được giữ khoá ghi.
  + Cản trở nhiều đến việc cập nhật dữ liệu của các giao tác khác.

1. Một ví dụ cho serializable

|  |  |  |
| --- | --- | --- |
| Session 1 | Session 2 | Explanation |
| SELECT last\_name, salary  FROM employees WHERE  last\_name IN ('Banda',  'Greene', 'Hintz');  LAST\_NAME SALARY  ------------------------- ----------  Banda 6300  Greene 9500 | No action | Session 1 thực hiện truy vấn lấy ra họ và lương của nhân viên có họ là Banda, Greene, Hintz. |
| UPDATE employees  SET salary = 7000  WHERE last\_name='Banda'; | No action | Session 1 thực hiện update lương cho nhân viên có họ là Banda |
| No action | SET TRANSACTION  ISOLATION LEVEL SERIALIZABLE; | Session 2 bắt đầu transaction và thiết lập mức cô lập là SERIALIZABLE |
| No action | SELECT last\_name, salary  FROM employees  WHERE last\_name IN  ('Banda', 'Greene', 'Hintz');  LAST\_NAME SALARY  ------------------------- ----------  Banda 6300  Greene 9500 | Session 2 thực hiện truy vấn lấy ra họ và lương của nhân viên có họ là Banda, Greene, Hintz. |
|  | UPDATE employees  SET salary = 9900  WHERE last\_name = 'Greene'; | Session 2 thực hiện update lương cho nhân viên có họ là Greene. |
| INSERT INTO employees  (employee\_id, last\_name,  email, hire\_date, job\_id)  VALUES (210, 'Hintz',  'JHINTZ', SYSDATE,'SH\_CLERK'); |  | Session 1 thực hiện insert một nhân viên có họ là Hintz |
| COMMIT; |  | Session 1 thực hiện COMMIT |
| SELECT last\_name, salary  FROM employees  WHERE last\_name IN  ('Banda', 'Greene', 'Hintz');  LAST\_NAME SALARY  ------------------------- ----------  Banda 7000  Greene 9500  Hintz | SELECT last\_name, salary  FROM employees  WHERE last\_name IN  ('Banda', 'Greene', 'Hintz');  LAST\_NAME SALARY  ------------------------- ----------  Banda 6300  Greene 9900 | Session 1 và Session 2 thực hiện truy vấn lấy ra họ và lương của nhân viên có họ là Banda, Greene, Hintz.  Kết quả hiển thị ở Session 1 cho thấy chỉ có lương của nhân viên có họ là Banda được cập nhập.  Tương tự ở Session 2 cũng chỉ thấy lương của nhân viên có họ là Greene được cập nhập mặc dù Session 1 đã COMMIT. |
|  | COMMIT; | Session 2 thực hiện COMMIT |

* + 1. Read-only

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).

## Cơ chế khóa

Khóa là một cơ chế để ngăn chặn các tương tác phá hoại.

Tương tác phá hoại là khi nó cập nhập dữ liệu sai hoặc thay đổi không chính xác 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.

Tóm tắt các hành vi khóa:

* Cơ sở dữ liệu(CSDL) có một số khóa khác nhau, tùy thuộc vào loại hoạt động lên dữ liệu. Nhìn chung, CSDL sử dụng hai loại khóa chính: exclusive locks (khóa độc quyền) và share locks (khóa chia sẽ).
* 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.
* Khóa ảnh hưởng tới tương tác giữa việc đọc và ghi dữ liệu:
  + Một dòng chỉ bị lock khi có một hành động ghi dữ liệu.
  + Một hành động ghi lên một dòng dữ liệu sẽ chặn những hành động khác cùng ghi lên dòng dữ liệu đó.
  + Một người đọc dữ liệu sẽ không chặn một người ghi lên dữ liệu. 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 chặn hành động đọc dữ liệu. 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.

Sử dụng Locks:

Khi có nhiều người dùng đang truy cập và sửa đổi dữ liệu , CSDL phải cung cấp một cách để ngăn chặn sửa đổi đồng thời của một dữ liệu.

Cơ chế khóa phải thỏa mãn các yêu cầu quan trọng của cơ sở dữ liệu: tính nhất quán và toàn vẹn dữ liệu.

**Mô tả một ví dụ khóa dòng:**

|  |  |  |
| --- | --- | --- |
| Session 1 | Session 2 | Explanation |
| SELECT PRODUCT\_INFO\_ID AS PRO\_ID, CODE, PRICE  FROM PRODUCT\_INFO  WHERE PRODUCT\_INFO\_ID = 26;  PRO\_ID CODE PRICE  ---------- ------------------------------ ----------  26 XMXT11 170000 | No action | Session 1 thực hiện truy vấn lấy ra id, code và price của sản phẩm có id là 26. |
|  | SELECT PRODUCT\_INFO\_ID AS PRO\_ID, CODE, PRICE  FROM PRODUCT\_INFO  WHERE PRODUCT\_INFO\_ID = 26;  PRO\_ID CODE PRICE  ---------- ------------------------------ ----------  26 XMXT11 170000 | Session 2 thực hiện truy vấn lấy ra id, code và price của sản phẩm có id là 26. Có cùng kết quả với session 1. |
| UPDATE PRODUCT\_INFO  SET CODE = 'XMXT00'  WHERE PRODUCT\_INFO\_ID = 26 AND CODE = 'XMXT11';  1 row updated. |  | Session 1 thực hiện update code = 'XMXT00' cho sản phẩm có id là 26 và code = 'XMXT11'. |
| No action | UPDATE PRODUCT\_INFO  SET CODE = 'XMXT22'  WHERE PRODUCT\_INFO\_ID = 26 AND CODE = 'XMXT11'; | Session 2 thực hiện update code = 'XMXT22' cho sản phẩm có id là 26 và code = 'XMXT11'. Vì session 2 cập nhập cùng một dòng với session 1 nên session 2 phải chờ cho đến khi session 1 hoản tất. |
| COMMIT;  Commit complete. |  | Session 1 thực hiện COMMIT, hoàn tất Transaction. |
|  | 0 rows updated. | Session 2 lấy được khóa nhưng lúc này không còn sản phẩm với id là 26 và code = 'XMXT11' nên session 2 không cập nhập dòng dữ liệu nào. |

## Deadlock

Deadlock là tình huống trong đó hai hoặc nhiều người đang chờ dữ liệu bị khóa bởi nhau. Deadlock ngăn cản một số transaction tiếp tục hoạt động.

Cơ sở dữ liệu Oracle tự động phát hiện các deadlock và giải quyết chúng bằng cách roll back một transaction gây deadlock và giải phóng một dòng đang bị khóa. Cơ sở dữ liệu sẽ trả về thông báo cho transaction bị rollback.

Mô tả một ví dụ cho deadlock:

|  |  |  |
| --- | --- | --- |
| Session 1 | Session 2 | Explanation |
| UPDATE PRODUCT\_INFO  SET PRICE = 180000  WHERE PRODUCT\_INFO\_ID = 25; | UPDATE PRODUCT\_INFO  SET PRICE = 160000  WHERE PRODUCT\_INFO\_ID = 26; | Session 1 thực hiện update price cho sản phẩm có id là 25.  Session 2 thực hiện update price cho sản phẩm có id là 26. |
| UPDATE PRODUCT\_INFO  SET PRICE = 150000  WHERE PRODUCT\_INFO\_ID = 26; | UPDATE PRODUCT\_INFO  SET PRICE = 170000  WHERE PRODUCT\_INFO\_ID = 25; | Session 1 thực hiện update price cho sản phẩm có id là 26.  Session 2 thực hiện update price cho sản phẩm có id là 25. |
| SQL Error: ORA-00060: deadlock detected while waiting for resource |  | Session 1 báo lỗi là xãy ra deadlock và roll back câu lênh update sản phẩm với id là 26. |
| COMMIT;  Commit complete. |  | Session 1 thực hiện COMMIT, hoàn tất Transaction. |
|  | 1 row updated. | Session nhận được khóa và thực hiện update price cho sản phẩm với id là 25. |
|  | COMMIT; | Session 2 thực hiện COMMIT, hoàn tất Transaction. |

## Mô tả trong đồ án môn học

* + 1. Non-repeatable read

Khái niệm: Một transaction đọc lại dữ liệu mà nó đã đọc trước đó và thấy rằng một transaction khác đã commit dữ liệu đã được thay đổi hoặc xóa. Ví dụ: người dùng truy vấn một hàng và sau đó truy vấn cùng một hàng, chỉ để phát hiện ra rằng dữ liệu đã thay đổi.

Ví dụ, một giao dịch truy vấn số lượng nhân viên. Năm phút sau thực hiện cùng một truy vấn, nhưng bây giờ số lượng đã tăng lên một vì một người dùng khác đã thêm một nhân viên mới.

1. Mô tả tình huống

Nhân viên thứ nhất thêm một hóa đơn nhập hàng cho mặt hàng với product\_id là 21. Sau khi thêm xong nhân viên 1 lấy ra danh sách sản phẩm trong kho và kiểm tra số lượng mặt hàng mới thêm vào. Trong lúc đó thì nhân viên thứ 2 đang thêm một hóa đơn xuất hàng cũng cho mặt hàng với product\_id là 21, giao dịch thành công. Cuối cùng nhân viên 1 lấy ra danh sách sản phẩm trong kho và kiểm tra số lượng mặt hàng với product\_id là 21 thì thấy dữ liệu đã khác so với ban đầu.

1. Mã PL/SQL:

Function: get\_amount()

|  |
| --- |
| CREATE OR REPLACE TYPE info\_goods\_receipt AS OBJECT (v\_product\_id NUMBER(5,0), v\_quantity NUMBER(5,0));  CREATE OR REPLACE TYPE list\_info\_goods\_receipt IS TABLE OF info\_goods\_receipt;  CREATE OR REPLACE FUNCTION get\_amount(v\_l\_info\_gr list\_info\_goods\_receipt)  RETURN NUMBER  AS  v\_amount NUMBER;  v\_price PRODUCT\_INFO.PRICE%TYPE;  BEGIN  v\_amount := 0;  FOR ind IN v\_l\_info\_gr.FIRST..v\_l\_info\_gr.LAST  LOOP  --GET PRICE FROM PRODUCT ID  SELECT PRICE INTO v\_price  FROM PRODUCT\_INFO  WHERE PRODUCT\_INFO\_ID = v\_l\_info\_gr(ind).v\_product\_id;  v\_amount := v\_amount + v\_l\_info\_gr(ind).v\_quantity \* v\_price;  END LOOP;    RETURN v\_amount;    EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('NO\_DATA\_FOUND!');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('OTHERS EXCEPTION!');  END; |

Store procedure: add\_goods\_receipt\_invoice()

|  |
| --- |
| CREATE OR REPLACE PROCEDURE add\_goods\_receipt\_invoice(v\_l\_info\_gr list\_info\_goods\_receipt,  v\_invoice\_id NUMBER, v\_type NUMBER, v\_user\_id NUMBER, v\_code VARCHAR2)  AS  v\_mount NUMBER;  BEGIN  --INSERT INTO INVOICE  v\_mount := get\_amount(v\_l\_info\_gr);  INSERT INTO INVOICE(INVOICE\_ID, TYPE, USER\_ID, PRICE, CODE) VALUES(v\_invoice\_id, v\_type, v\_user\_id, v\_mount, v\_code);  FOR ind IN v\_l\_info\_gr.FIRST..v\_l\_info\_gr.LAST  LOOP  --INSERT INTO INVOICE DETAIL  INSERT INTO INVOICE\_DETAIL(INVOICE\_ID, PRODUCT\_ID, QUANITY)  VALUES (v\_invoice\_id, v\_l\_info\_gr(ind).v\_product\_id, v\_l\_info\_gr(ind).v\_quantity);    END LOOP;  END; |

Trigger: update\_quantity

|  |
| --- |
| CREATE OR REPLACE TRIGGER update\_quantity  BEFORE INSERT ON INVOICE\_DETAIL  FOR EACH ROW  DECLARE  v\_type INVOICE.TYPE%TYPE;  v\_product\_id PRODUCT\_IN\_STOCK.PRODUCT\_ID%TYPE;  BEGIN  SELECT TYPE INTO v\_type  FROM INVOICE  WHERE INVOICE\_ID = :NEW.INVOICE\_ID AND ACTIVE\_FLAG = 1;    SELECT PRODUCT\_ID INTO v\_product\_id  FROM PRODUCT\_IN\_STOCK  WHERE PRODUCT\_ID = :NEW.PRODUCT\_ID;    IF v\_type = 1 THEN  --NHAP HANG  UPDATE PRODUCT\_IN\_STOCK  SET QUANITY = QUANITY + :NEW.QUANITY  WHERE PRODUCT\_ID = :NEW.PRODUCT\_ID AND ACTIVE\_FLAG = 1;  ELSE  --XUAT HANG  UPDATE PRODUCT\_IN\_STOCK  SET QUANITY = QUANITY - :NEW.QUANITY  WHERE PRODUCT\_ID = :NEW.PRODUCT\_ID AND ACTIVE\_FLAG = 1;  END IF;    EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('INSERT NEW PRODUCT\_IN\_STOCK!');  INSERT INTO PRODUCT\_IN\_STOCK(PRODUCT\_ID, QUANITY) VALUES (:NEW.PRODUCT\_ID, :NEW.QUANITY);    WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('OTHERS EXCEPTION!');  END; |

c. Bảng mô tả Non-repeatable read

|  |  |  |
| --- | --- | --- |
| Session 1 | Session 2 | Explanation |
| SELECT PRODUCT\_STOCK\_ID,PRODUCT\_ID, QUANITY  FROM PRODUCT\_IN\_STOCK  WHERE ACTIVE\_FLAG = 1;  PRODUCT\_STOCK\_ID PRODUCT\_ID QUANITY  ---------------- ---------- ----------  61 21 75  81 22 5 |  | Giá trị Quantity ban đầu của sản phẩm với PRODUCT\_ID = 21 là 75 |
| DECLARE  v\_l\_info\_gr list\_info\_goods\_receipt := list\_info\_goods\_receipt(  info\_goods\_receipt( 21, 15)  );  v\_type NUMBER;  v\_user\_id NUMBER;  v\_code VARCHAR2(30);  v\_invoice\_id NUMBER;  BEGIN  v\_type := 1;  v\_user\_id := 1;  v\_code := 'HD015';  v\_invoice\_id := 38;    add\_goods\_receipt\_invoice(v\_l\_info\_gr, v\_invoice\_id, v\_type, v\_user\_id, v\_code);  END;  PL/SQL procedure successfully completed. |  | Session 1 tương ứng với nhân viên thứ nhất thêm mới một hóa đơn nhập hàng với product\_id = 21 và  quantity = 15 sử dụng  add\_goods\_receipt\_invoice() |
| SELECT PRODUCT\_STOCK\_ID,PRODUCT\_ID, QUANITY  FROM PRODUCT\_IN\_STOCK  WHERE ACTIVE\_FLAG = 1;  PRODUCT\_STOCK\_ID PRODUCT\_ID QUANITY  ---------------- ---------- ----------  61 21 90  81 22 5 |  | Nhân viên thứ nhất lấy ra danh sách hàng tồn kho và kiểm tra số lượng của sản phẩm với product\_id = 21 lúc này có giá trị là 90. |
|  | SET TRANSACTION  ISOLATION LEVEL  READ COMMITTED;  Transaction ISOLATION succeeded. | Session 2 thiết lập mức cô lập là read committed. |
|  | SELECT PRODUCT\_STOCK\_ID,PRODUCT\_ID, QUANITY  FROM PRODUCT\_IN\_STOCK  WHERE ACTIVE\_FLAG = 1;  PRODUCT\_STOCK\_ID PRODUCT\_ID QUANITY  ---------------- ---------- ----------  61 21 75  81 22 5 | Session 2 lấy ra danh sách hàng tồn kho, kiểm tra giá trị Quantity của sản phẩm với product\_id lúc này vẫn là 75 vì Session 1 chưa COMMIT |
|  | DECLARE  v\_l\_info\_gr list\_info\_goods\_receipt := list\_info\_goods\_receipt(  info\_goods\_receipt( 21, 5)  );  v\_type NUMBER;  v\_user\_id NUMBER;  v\_code VARCHAR2(30);  v\_invoice\_id NUMBER;  BEGIN  v\_type := 0;  v\_user\_id := 1;  v\_code := 'HD014';  v\_invoice\_id := 37;  add\_goods\_receipt\_invoice(v\_l\_info\_gr, v\_invoice\_id, v\_type, v\_user\_id, v\_code);  END; | Session 2 Thêm mới một hóa đơn xuất hàng với cùng một product\_id với session 1 product\_id = 21 và  quantity = 5 sử dụng  add\_goods\_receipt\_invoice().  Lúc này Session vẫn đang chạy vì Session 1 vẫn đang giữ khóa trên dòng với product\_id = 21 của TABLE PRODUCT\_IN\_STOCK |
| COMMIT;  Commit complete. |  | Session 1 COMMIT |
|  | PL/SQL procedure successfully completed. | Session 2 lúc này nhận được khóa và hiển thị kết quả là đã thực hiện xong thao tác thêm hóa đơn xuất hàng. |
|  | COMMIT;  Commit complete. | Session 2 COMMIT |
| SELECT PRODUCT\_STOCK\_ID,PRODUCT\_ID, QUANITY  FROM PRODUCT\_IN\_STOCK  WHERE ACTIVE\_FLAG = 1;  PRODUCT\_STOCK\_ID PRODUCT\_ID QUANITY  ---------------- ---------- ----------  61 21 85  81 22 5 |  | Lúc này Session 1 lấy lại danh sách hàng tồn kho và nhận ra số lượng tồn kho của sản phẩm với product\_id = 21 đã thay đổi. Số lượng lúc này đã thay đổi thành 85(giá trị trước đó là 90). |

1. Nguyên nhân và giải pháp

Vấn đề xảy ra: Transaction 1 thêm một hóa đơn nhập hàng và thực hiện truy vấn lấy ra danh sách hàng tồn kho để kiểm tra số lượng mới của sản phẩm được nhập. Ngay lúc đó Transaction 2 thực hiện thêm một hóa đơn xuất hàng cho cùng một sản phẩm ở Transaction1. Cuối cùng Transaction 1 thực hiện truy vấn lấy lại danh sách thì thấy dữ liệu đã bị thay đổi.

Nguyên nhân: Transaction 1 thiết lập mức cô lập là READ COMMITED nên mỗi lần truy vấn nó sẽ đọc dữ liệu đã được COMMIT từ CSDL mặc dù những dữ liệu này tác động trên cùng một đơn vị dữ liệu.

Giải pháp: Thiết lập mức cô lập là SERIALIZABLE cho Transaction 1 thay cho mức cô lập là READ COMMITED sử dụng câu lệnh “set transaction isolation level serializable”.

* + 1. Phantom read

Khái niệm: Một transaction chạy một truy vấn để lấy danh sách các hàng với điều kiện (Where) và thấy rằng một transaction khác đã thêm một hàng thỏa với điều kiện như trên.

a. Mô tả tình huống: Trong khi nhân viên thứ nhất đang xem có bao nhiêu đơn nhập hàng hôm nay thì nhân viên thứ 2 thêm một đơn nhập hàng và cập nhập vào hệ thống, lúc này nhân viên thứ nhất lập tổng hóa đơn nhập hàng trong ngày thị thấy dữ liệu đã bị thay đổi.

Store procedure được sử dụng: add\_goods\_receipt\_invoice()

Function được sử dụng: get\_amount()

Trigger được sử dụng: update\_quantity

b. Mô tả Phantom read

|  |  |  |
| --- | --- | --- |
| Session 1 | Session 2 | Explanation |
| SET TRANSACTION  ISOLATION LEVEL  READ COMMITTED;  Transaction ISOLATION succeeded. | No action | Session 1 thiết lập mức cô lập là READ COMMITTED. |
| SELECT INVOICE\_ID, USER\_ID, PRICE  FROM INVOICE  WHERE TYPE = 1 AND ACTIVE\_FLAG = 1  AND TRUNC(UPDATE\_DATE) = '26-JUN-20';  INVOICE\_ID USER\_ID PRICE  ---------- ---------- ----------  34 1 1350000  38 1 1350000  33 1 1350000 |  | Session 1 thực hiện truy vấn lấy tất cả hóa đơn nhập hàng trong ngày 26-Jun-20 |
| No action | DECLARE  v\_l\_info\_gr list\_info\_goods\_receipt := list\_info\_goods\_receipt(  info\_goods\_receipt( 24, 5),  info\_goods\_receipt( 26, 3 )  );  v\_type NUMBER;  v\_user\_id NUMBER;  v\_code VARCHAR2(30);  v\_invoice\_seq NUMBER(5, 0);  BEGIN  v\_type := 1;  v\_user\_id := 1;  v\_code := 'HD016';  --GET INVOICE\_SEQ NEXT VALUE  SELECT INVOICE\_SEQ.NEXTVAL INTO v\_invoice\_seq FROM DUAL;    add\_goods\_receipt\_invoice(v\_l\_info\_gr, v\_invoice\_seq, v\_type, v\_user\_id, v\_code);  END;  PL/SQL procedure successfully completed. | Session 2 Thêm mới một hóa đơn nhập hàng cũng vào ngày 26-Jun-20 sử dụng add\_goods\_receipt  \_invoice() |
| No action | COMMIT;  Commit complete. | Transaction 2 COMMIT |
| SELECT INVOICE\_ID, USER\_ID, PRICE  FROM INVOICE  WHERE TYPE = 1 AND ACTIVE\_FLAG = 1  AND TRUNC(UPDATE\_DATE) = '26-JUN-20';  INVOICE\_ID USER\_ID PRICE  ---------- ---------- ----------  42 1 1245000  34 1 1350000  38 1 1350000  33 1 1350000 | No action | Transaction 1 thực hiện lại truy vấn lấy tất cả hóa đơn nhập hàng trong ngày 26-Jun-20 thì thấy dữ liệu đã có thêm một dòng dữ liệu mới. |

c. Nguyên nhân và giải pháp

Vấn đề xảy ra: Transaction 1 thực hiện truy vấn lấy ra danh sách đơn hàng trong ngày, ngay sau đó Transaction 2 thực hiện thêm một hóa đơn nhập hàng. Cuối cùng Transaction 1 thực hiện truy vấn lấy lại danh sách đơn hàng nhập thì thấy dữ liệu đã có thêm một hóa đơn mới.

Nguyên nhân: Transaction 1 thiết lập mức cô lập là READ COMMITED nên mỗi lần truy vấn nó sẽ đọc dữ liệu đã được COMMIT từ CSDL.

Giải pháp: Thiết lập mức cô lập là SERIALIZABLE cho Transaction 1 thay cho mức cô lập là READ COMMITED sử dụng câu lệnh “set transaction isolation level serializable”.

* + 1. Lost update

a. Mô tả tình huống: Nhân viên thứ nhất cập nhập giá tiền cho sản phẩm với product\_id là 25. Cùng lúc đó nhân viên thứ hai cũng cập nhập giá tiền cho sản phẩm với product\_id là 25. Giá trị cập nhập của nhân viên thứ nhất sẽ bị mất vì giá trị của sản phẩm lúc này sẽ là giá trị cập nhập của nhân viên thứ 2.

b. Mô tả Lost update

|  |  |  |
| --- | --- | --- |
| Session 1 | Session 2 | Explanation |
| SELECT PRODUCT\_INFO\_ID, CODE, NAME, PRICE  FROM PRODUCT\_INFO  WHERE ACTIVE\_FLAG = 1  AND PRODUCT\_INFO\_ID IN (25, 26, 41);  PRODUCT\_INFO\_ID CODE NAME PRICE  --------------- ------------------------------ -------------------------------------------------- ----------  25 XMNS11 Xi măng Nghi Sơn 150000  26 XMXT11 Xi măng Xuân Thành 140000 | No action | Session 1 truy vấn CODE, NAME, PRICE cho những sản phẩm có ID là 25, 26,41. |
| UPDATE PRODUCT\_INFO  SET PRICE = 145000  WHERE PRODUCT\_INFO\_ID = 25; | No action | Session 1 thực hiện update PRICE cho sản phẩm có ID là 25. Isolaiton level mặc định là READ COMMITTED. |
| No action | SET TRANSACTION  ISOLATION LEVEL  READ COMMITTED; | Bắt đầu Session 2 và thiết lập Isolaiton level là READ COMMITTED. |
| No action | SELECT PRODUCT\_INFO\_ID, CODE, NAME, PRICE  FROM PRODUCT\_INFO  WHERE ACTIVE\_FLAG = 1  AND PRODUCT\_INFO\_ID IN (25, 26, 41);  PRODUCT\_INFO\_ID CODE NAME PRICE  --------------- ------------------------------ -------------------------------------------------- ----------  25 XMNS11 Xi măng Nghi Sơn 150000  26 XMXT11 Xi măng Xuân Thành 140000 | Session 2 truy vấn CODE, NAME, PRICE cho những sản phẩm có ID là 25, 26,41.  Giá trị PRICE của sản phẩm với ID là 25 vẫn chưa được cập nhập do Transaction 1 chưa được COMMIT. |
| No action | UPDATE PRODUCT\_INFO  SET PRICE = 170000  WHERE PRODUCT\_INFO\_ID = 26; | Session 2 thực hiện update PRICE cho sản phẩm có ID là 26 thành công do Transaction 1 chỉ lock dòng có ID là 25. |
| INSERT INTO PRODUCT\_INFO(CATEGORY\_ID, CODE, NAME)  VALUES (62, 'FGH11A', 'Xi mang Hoa Tho'); | No action | Transaction 1 insert một dòng cho PRODUCT\_INFO với ID là 41, nhưng không thực hiện COMMIT. |
| No action | SELECT PRODUCT\_INFO\_ID, CODE, NAME, PRICE  FROM PRODUCT\_INFO  WHERE ACTIVE\_FLAG = 1  AND PRODUCT\_INFO\_ID IN (25, 26, 41);  PRODUCT\_INFO\_ID CODE NAME PRICE  --------------- ------------------------------ -------------------------------------------------- ----------  25 XMNS11 Xi măng Nghi Sơn 150000  26 XMXT11 Xi măng Xuân Thành 170000 | Session 2 truy vấn CODE, NAME, PRICE cho những sản phẩm có ID là 25, 26,41.  Giá trị PRICE của sản phẩm với ID là 26 đã được cập nhập.  Lúc này vẫn chưa thấy dữ liệu cập nhập của sản phẩm với ID là 25 hay ID là 41. |
| No action | UPDATE PRODUCT\_INFO  SET PRICE = 175000  WHERE PRODUCT\_INFO\_ID = 25; | Transaction 2 thực hiện update PRICE cho sản phẩm có ID là 25, dòng này đã bị khóa trên Transaction 1.  Lúc này Transaction 2 phải chờ Transaction 1 thực hiện xong. |
| COMMIT; | No action | Transaction 1 COMMIT thành công, kết thúc transaction. |
| No action | 1 row updated. | Transaction 2 nhận được khóa và thực hiện update PRICE cho sản phẩm có ID là 25. |
| No action | SELECT PRODUCT\_INFO\_ID, CODE, NAME, PRICE  FROM PRODUCT\_INFO  WHERE ACTIVE\_FLAG = 1;  PRODUCT\_INFO\_ID CODE NAME PRICE  --------------- ------------------------------ -------------------------------------------------- ----------  25 XMNS11 Xi măng Nghi Sơn 175000  26 XMXT11 Xi măng Xuân Thành 170000  41 FGH11A Xi mang Hoa Tho | Transaction 2 truy vấn CODE, NAME, PRICE cho những sản phẩm có ID là 25, 26,41.  Transaction 2 thấy được giá trị đã update của sản phẩm với ID là 25 và sản phẩm mới được thêm vào với ID là 41. |
| No action | COMMIT; | Transaction 2 COMMIT thành công, kết thúc transaction. |
| SELECT PRODUCT\_INFO\_ID, CODE, NAME, PRICE  FROM PRODUCT\_INFO  WHERE ACTIVE\_FLAG = 1;  PRODUCT\_INFO\_ID CODE NAME PRICE  --------------- ------------------------------ -------------------------------------------------- ----------  25 XMNS11 Xi măng Nghi Sơn 175000  26 XMXT11 Xi măng Xuân Thành 170000  41 FGH11A Xi mang Hoa Tho |  | Transaction 1 truy vấn CODE, NAME, PRICE cho những sản phẩm có ID là 25, 26,41.  Giá trị PRICE của sản phẩm với ID = 25 bây giờ là 175000.  Việc update PRICE cho sản phẩm với ID = 21 của Transaction 1 đã bị mất. |

c. Nguyên nhân và giải pháp

Vấn đề xảy ra: Transaction 1 thực hiện cập nhập giá tiền cho sản phẩm với product\_id, ngay lúc đó Transaction 2 cũng thực hiện cập nhập giá tiền cho sản phẩm với cùng một sản phẩm ở Transaction 1. Giá tiền cập nhập được cho sản phẩm sẽ là giá trị cập nhập ở Transaction 2 vì thế giá trị cập nhập ở Transaction 1 đã bị mất.

Nguyên nhân: Transaction 1 thực hiện câu lệnh UPDATE lên bảng PRODUCT\_INFO nhưng Transaction 2 vẫn có thể tác động là thực cấu lệnh UPDATE lên bảng PRODUCT\_INFO.

Giải pháp: Thiết lập khóa trên bảng PRODUCT\_INFO sử dụng câu lệnh “LOCK TABLE PRODUCT\_INFO IN SHARE MODE” để khi một transaction thực hiện ghi trên bảng PRODUCT\_INFO thì những transaction khác chỉ được đọc mà không được ghi.

# THIẾT KẾ GIAO DIỆN

# KẾT LUẬN

# Phụ lục

## Phụ lục 1: Bảng phân chia công việc

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Đầu việc** | **Tỷ lệ** | |
| **Công** | **Hội** |
| 1 | Phân tích bài toán, xác định chức năng |  |  |
| 2 | Thiết kế & xây dựng CSDL |  |  |
| 3 | Thiết kế giao diện |  |  |
| 4 | Triển khai ứng dụng bằng Java Spring |  |  |
| 5 | Xử lý vấn đề truy xuất đồng thời |  |  |
| 6 | Kiểm tra chức năng |  |  |
| 7 | Soạn và chỉnh sửa báo cáo |  |  |

## Phụ lục 2: Tài liệu tham khảo

[1] Oracle Database Database Concepts, 12c Release 2 (12.2) - Primary Authors: Lance Ashdown, Tom Kyte

[2] Slide môn Hệ quan trị cơ sở dữ liệu – ThS. Đỗ Thị Minh Phụng, trường ĐH Công Nghệ Thông Tin

[3] Quản lý kho là gì?: <https://www.sapo.vn/blog/kinh-nghiem-quan-ly-kho-dat-hieu-qua/>

[4] Quy trình quản lý kho: <https://sec-warehouse.vn/quy-trinh-quan-ly-kho-hang.html>

[5] Kiến thức Oracle : <https://csc.edu.vn/lap-trinh-va-csdl/tin-tuc/kien-thuc-lap-trinh/hoc-oracle--kien-thuc-co-ban-cho-nguoi-moi-bat-dau-802>

[6] Hibernate framework: <https://hibernate.org/>

[7] Hibernate tutorial: <https://www.tutorialspoint.com/hibernate/hibernate_architecture.htm>

[8] Tổng quan Spring framework: <https://viblo.asia/p/tong-quan-ve-spring-framework-YWOZryEyKQ0>

[9] Spring framework: <https://spring.io/>

[10]Github: <https://github.blog/>

[11]IntelliJ IDEA: <https://www.jetbrains.com/idea/>

[12]

[13]

# 