**`CÂU HỎI AUDIT MODULE 3**

|  |  |  |
| --- | --- | --- |
| **STT** | | **Q&A** |
| 1 | | **Q:** Cơ sở dữ liệu (database) là gì ?  🡺 **A:** Là một hệ thống thông tin có cấu trúc được lưu trữ trên các hệ quản trị cơ sử dữ liệu nhằm thỏa mãn nhu cầu khai thác thông tin đồng thời của nhiều người sử dụng hay nhiều chương trình ứng dụng chạy cùng một lúc với những mục đích khác nhau |
| 2 | | **Q:** Phân biệt các câu lệnh DML & DDL? DML, DDL bao gồm các câu lệnh nào?  🡺 **A:**   |  |  | | --- | --- | | DDL | DML | | - Nhóm các câu lệnh liên quan đến việc định nghĩa dữ liệu  - Các lệnh: Create, Alter, Drop | - Nhóm các câu lệnh liên quan tới thao tác dữ liệu  - Các lệnh: Insert (into), Update (...set), Delete | |
| 3 | | **Q:** Trình bày về mô hình thực thể mối quan hệ (ERD - Entity-Relationship Diagram)? Trong mô hình thực thể mối quan hệ có những thành phần nào?  🡺 **A:**  - Mô hình thực thể mối quan hệ là 1 sơ đồ hiển thị mối quan hệ của các tập thực thể lưu trữ trong CSDL, giúp giải thích cấu trúc logic của CSDL  - Các thành phần: Thực thể và tập thực thể, thuộc tính, mối quan hệ giữa các tập thực thể, tập mối quan hệ |
| 4 | | **Q:** Các bước chuyển đổi từ ERD 🡺 Table?  🡺 **A:**  - Biến đổi thực thể → table  + Thuộc tính khoá của thực thể là khoá chính của bảng  + Thuộc tính thường có của thực thể là thuộc tính thường của bảng  - Biến đổi thuộc tính đa trị  + Tách thuộc tính đa trị ra bảng mới  + Đặt khoá ngoại vào bảng hiện tại  - Biến đổi 1-1  + Đặt thuộc tính khoá ở phía tuỳ chọn → phía bắt buộc  - Biến đổi 1-n  + Đặt thuộc tính khoá ở 1→n  - Biến đổi n-n  + Tạo ra một bảng mới có khoá chính là khoá phức hợp 2 khoá chính của 2 thực thể tham gia vào mqh → khoá composite |
| 5 | | **Q:** Các bước chuẩn hóa dữ liệu?  🡺 **A:** Có 3 dạng chuẩn hóa:  - Dạng chuẩn hóa 1NF: các thuộc tính phải là đơn trị, tách các thuộc tính đa trị thành bảng mới  - Dạng chuẩn 2NF: đạt dạng chuẩn 1NF và tất cả các thuộc tính không khoá đều phụ thuộc đầy đủ vào mọi khoá  - Dạng chuẩn 3NF: đạt chuẩn 2NF và tất cả các thuộc tính không khóa phải phụ thuộc bắc cầu vào thuộc tính khóa |
| 6 | | **Q:** Ràng buộc (constraint) là gì? Các loại ràng buộc trong MySQL?  🡺 **A:**  - Ràng buộc (constraint) là các quy tắc được quy định cho bảng  - Các loại ràng buộc trong MySQL:  + NOT NULL: Không cho phép giá trị null  + UNIQUE: Mỗi giá trị là duy nhất, không cho phép trùng lặp  + PRIMARY KEY: Khóa chính, không được null và không được trùng lặp  + FOREIGN KEY: Khóa ngoại, được phép null  + CHECK: Kiểm tra giá trị theo điều kiện nào đó  + DEFAULT: Giá trị mặc định của cột khi không nhập dữ liệu  + INDEX: Tăng tốc độ truy vấn |
| 7 | | **Q:** Khóa chính (Primary key) là gì? “Khóa chính chứa một hoặc nhiều thuộc tính khóa” nghĩa là gì?  🡺 **A:**  - Khóa chính (Primary key) là một cột (hoặc nhiều cột) được sử dụng để xác định 1 bản ghi duy nhất trong bảng. Mỗi bảng chỉ có thể có 1 khóa chính  - |
| 8 | | **Q:** Khóa ngoại (foreign key) là gì?  🡺 **A:**  - Khóa ngoại là cơ chế để tạo liên kết giữa 2 bảng trong cùng CSDL  - Khóa ngoại được đặt trên 1 cột của bảng này và tham chiếu đến khóa chính của 1 bảng khác  - Kiểu dữ liệu của khóa chính và khóa ngoại phải giống nhau |
| 9 | | **Q:** Phân biệt DELETE và TRUNCATE?  🡺 **A:**   |  |  | | --- | --- | | DELETE | TRUNCATE | | - Cú pháp: DELETE FROM table\_name WHERE conditions;  - Nếu muốn xóa cả bảng: DELETE FROM table\_name;  - Khi chạy lệnh DELETE thì SQL sẽ log lại từng dòng đã xóa vào transaction log, vì thế nên khi tạo 1 record mới, giá trị của id sẽ không bắt đầu từ 1 mà sẽ có giá trị n+1 với n là giá trị của record cuối cùng được tạo.  - Xoá bằng delete thì sẽ restored được  - Chậm hơn truncate | - Cú pháp: TRUNCATE TABLE table\_name;  - Đối với TRUNCATE chỉ có thể dùng cho bảng chứ không thể dùng cho từng record.  - Khi chạy lệnh TRUNCATE thì SQL sẽ xóa hết dữ liệu của bảng và reset transaction log, vì thế khi tạo 1 record mới, giá trị của id sẽ bắt đầu từ 1  - Xoá bằng truncate thì không restored được  - Nhanh hơn vì nó tiêu tốn ít bộ nhớ hơn | |
| ưii10 | | **Q:** Các loại JOIN? Phân biệt?  🡺 **A:**  - Inner join: Nối bảng lấy những record chung giữa 2 bảng  - Outer join: Nối bảng lấy phần tử chung và phần tử riêng  + Left outer join (left join): phần tử chung + phân tử riêng của bảng bên trái  + Right outer join (right join): phần tử chung + phần tử riêng của bảng bên phải  - Cross join: 1 phần tử của bảng này sẽ nối với tất cả các phần tử của bảng kia  - Self join: Tự nối với chính mình, trong phép self join có thể có inner join, outer join hoặc cross join  - Excluding join: Join 2 bảng lại với nhau và lấy phần tử riêng  + Left excluding join: left join loại đi các phần tử chung chỉ lấy phần tử riêng của bảng bên trái  + Right excluding join: right join loại đi các phần tử chung chỉ lấy phần tử riêng của bảng bên phải  + Full excluding join: full chỉ lấy phần tử riêng của 2 bảng bên trái và bên phải |
| 11 | | **Q:** Phân biệt JOIN và UNION?  🡺 **A:**   |  |  | | --- | --- | | JOIN | UNION | | - Cho phép kết nối dữ liệu liên quan trong các bảng khác nhau | - Hợp dữ liệu của 2 bảng mà có cột loại data giống nhau | |
| 12 | | **Q:** Phân biệt UNION và UNION ALL?  🡺 **A:**   |  |  |  | | --- | --- | --- | |  | UNION | UNION ALL | | Trả lại giá trị | - Lệnh union trả về các giá trị khác biệt với nhau | - Hiển thị tất cả các giá trị cụ thể cho truy vấn, bao gồm các giá trị đang được sao chép | | Loại bỏ sự trùng lặp | - Luôn loại bỏ các giá trị trùng lặp và trình bày các giá trị riêng | - Không loại bỏ các giá trị trùng lặp | | Tốc độ | - Chậm hơn | - Tương đối nhanh | | Đề xuất | - Các giá trị cần được sắp xếp và thống nhất | - Các giá trị cần phải được thống nhất, nhưng không cần phải sắp xếp | |
| 13 | | **Q:** Phân biệt EXIST và IN?  🡺 **A:**   |  |  | | --- | --- | | EXIST | IN | | - Kiểm tra một giá trị có xuất trong tập giá trị của truy vấn con hay không  - Quét từng bộ ở câu truy vấn con và so sánh trực tiếp với yêu cầu mẹ, nếu không thỏa sẽ loại trừ ngay | - Kiểm tra xem một truy vấn con có trả về kết quả nào không, trả về true nếu có ít nhất 1 dòng và trả về false nếu không có dòng nào  - Sẽ tìm ra tất cả các bộ thỏa yêu cầu truy vấn con rồi mới so sánh với câu truy vấn mẹ | |
| 14 | | **Q:** Phân biệt AND và OR?  🡺 **A:**   |  |  |  | | --- | --- | --- | |  | AND | OR | | Cú pháp | WHERE [DIEU\_KIEN1] AND [DIEU\_KIEN2] ... AND [DIEU\_KIENN] | WHERE [DIEU\_KIEN1] OR [DIEU\_KIEN2] ... OR [DIEU\_KIENN] | | Dữ liệu lấy ra | - Phải thỏa mãn cùng lúc tất cả các điều kiện có trong mệnh đề WHERE | - Chỉ cần thỏa mãn 1 trong các điều kiện được liệt kê | |
| 15 | | **Q:** Phân biệt GROUP BY và ORDER BY?  🡺 **A:**   |  |  | | --- | --- | | GROUP BY | ORDER BY | | - Để sắp xếp dữ liệu giống nhau thành các nhóm  - Mệnh đề GROUP BY tuân theo mệnh đề WHERE trong câu lệnh SELECT và đứng trước mệnh đề ORDER BY | - Được sử dụng để sắp xếp dữ liệu theo thứ tự tăng dần hoặc giảm dần, dựa trên một hoặc nhiều cột  - Lệnh ASC được sử dụng để sắp xếp tăng dần và DESC được sử dụng để sắp xếp giảm dần | |
| 16 | | **Q:** Muốn nối chuỗi trong MySQL thì sử dụng gì?  🡺 **A:** Sử dụng CONCAT()  - Cú pháp: CONCAT(str1, str2, ..., strn) |
| 17 | | **Q:** Yêu cầu tính điểm trung bình của tất cả các sinh viên thì làm thế nào?  🡺 **A:**  SELECT AVG(Diem) FROM SINHVIEN |
| 18 | | **Q:** Yêu cầu tính điểm trung bình của từng bạn sinh viên thì làm thế nào?  🡺 **A:**  SELECT id, name, AVG(Diem) as Diem  FROM SINHVIEN; |
| 19 | | **Q:** Yêu cầu hiển thị tên sinh viên và điểm trung bình tương ứng, chỉ hiển thị các bạn có điểm trung bình lớn hơn 5 thì làm thế nào?  🡺 **A:**  SELECT id, name, AVG(Diem) as Diem  FROM SINHVIEN  GROUPBY id, name;  HAVING AVG(Diem) > 5; |
| 20 | | **Q:** Hiển thị danh sách sinh viên và lớp học tương ứng thì làm thế nào (hai bảng Student và Class có cột chung ClassID)?  🡺 **A:**  SELECT studentId, studentName, className  FROM STUDENT s  JOIN CLASS c ON s.classId = c.classId; |
| 21 | | **Q:** Hiển thị danh sách sinh viên theo thứ tự tăng dần của điểm thi?  🡺 **A:**  SELECT \* FROM SINHVIEN ORDER BY DIEMTHI  (tăng: asc, giảm: desc) |
| 22 | | **Q:** Hiển thị danh sách sinh viên theo thứ tự giảm dần của họ tên, nếu họ tên trùng nhau thì sắp xếp theo tăng dần của tuổi?  🡺 **A:**  SELECT id, name, age FROM SINHVIEN  ORDER BY name DESC, age; |
| 23 | | **Q:** Hiển thị các sinh viên có họ là ‘Le’ và tên là ‘Hai’?  🡺 **A:**  SELECT id, ho, ten FROM SINHVIEN  WHERE ho = “Le” and ten = “Hai” |
| 24 | | **Q:** Hiển thị các sinh viên có tên bắt đầu là ‘C’?  🡺 **A:**  SELECT id, name FROM SINHVIEN  Where name like “C%” |
| 25 | | **Q:** Làm sao để có thể tăng tốc độ truy vấn? Nhược điểm là gì?  🡺 **A:**  - Dùng INDEX để tăng tốc độ truy vấn  - Nhược điểm: thêm, xóa, sửa dữ liệu bị chậm và chiếm dụng bộ nhớ |
| 26 | | **Q:** Các dạng tham số ở SP?  🡺 **A:**   |  |  |  | | --- | --- | --- | | IN | OUT | INOUT | | - Tham số tương đương truyền tham trị (mặc định)  - Lấy tham số đầu vào và không bị thay đổi bên ngoài | - Tham số có mục đích lấy giá trị trả về  - Tham số nhận vào giá trị null mặc dù ta truyền có giá trị, nếu bên trong procedure làm thay đổi biến đó thì bên ngoài cũng thay đổi | - Tham số tương đương truyền tham chiếu  - Kết hợp IN và OUT, nhận tham số đầu vào và nếu bên trong procedure làm thay đổi biến đó thì bên ngoài cũng thay đổi | |
| 27 | | **Q:** Phân biệt SP, Trigger, Function?  🡺 **A:**   |  |  |  | | --- | --- | --- | | SP | Trigger | Function | | - Là tập hợp các câu lệnh SQL để thực hiện 1 nghiệp vụ nào đó  - Khi khai báo SP thì sẽ có 3 loại tham số truyền vào:  + IN: Tham số tương đương truyền tham trị (mặc định)  + OUT: Tham số có mục đích lấy giá trị trả về  +INOUT: Tham số tương đương truyền tham chiếu | - Là tập hợp các câu lệnh SQL để thực hiện 1 nghiệp vụ nào đó, được gắn liền trên table nhất định (trigger chỉ áp dụng cho 1 table)  - Trigger sẽ được tự động chạy khi có sự kiện thay đổi dữ liệu trên table (insert, update, delete) | - Là tập hợp các câu lệnh SQL để thực hiện 1 nghiệp vụ nào đó nhưng luôn luôn trả về giá trị  - Sử dụng chung được với các câu Query | |
| 28 | | **Q:** CSS là gì? Các cách nhúng CSS vào tài liệu html? Độ ưu tiên?  🡺 **A:**  - CSS là viết tắt của cụm từ Cascading Style Sheets, giúp mô tả cách hiển thị của các thẻ HTML trên các thiết bị  - Có 3 cách để nhúng CSS vào HTML  + Inline: sử dụng thuộc tính style ở HTML  + Internal: sử dụng thẻ style trong thẻ head  + External: sử dụng thẻ link để nhúng 1 file css từ ngoài vào  - Độ ưu tiên: Inline 🡺 Internal & External tùy thuộc vào khai báo, cái nào khai báo sau thì độ ưu tiên cao hơn |
| 29 | | **Q:** Phân biệt giữa bộ chọn class và bộ chọn id?  🡺 **A:**   |  |  | | --- | --- | | Class | Id | | - Khai báo một class, các thẻ sẽ có cùng chung thuộc tính trong CSS  - Có dấu **.** ở đầu | - Khi khai báo id cho 1 thẻ nào đó thì tên id đó không được dùng lại nx  - Có dấu # ở đầu | |
| 30 | | **Q:** Trình bày mô hình hộp (box model) trong CSS?  🡺 **A:**  - Content: là phần text hoặc hình ảnh ở trên màn hình  - Padding: khoảng trống nằm giữa content và border, bọc xung quanh content  - Border: phần khung bao bọc xung quanh padding và content  - Margin: phần ngoài cùng của Box Model |
| 31 | | **Q:** Trình bày thuộc tính float, clear?  🡺 **A:**   |  |  | | --- | --- | | Float | Clear | | - Sử dụng để chuyển một phần tử sang góc trái hoặc phải của không gian bao quanh nó, cần thiết trong việc đinh dạng bố cục trang | - Ngăn chặn thành phần này chiếm vùng không gian của thành phần khác. Đôi khi không muốn float ở một số tình huống nào đó, sẽ sử dụng clear để khắc chế. | |
| 32 | | **Q:** Thuộc tính position dùng để làm gì? Phân biệt các giá trị: static, absolute, relative và fixed?  🡺 **A:**  - Thuộc tính position quy định cách thức xếp đặt vị trí của 1 thành phần trong trang Web  - Phân biệt:   |  |  |  |  | | --- | --- | --- | --- | | Static | Absolute | Relative | Fixed | | - Phần tử sẽ được sắp xếp đúng theo vị trí tự nhiên của nó | - Xác định vị trí tuyệt đối so với thành phần bên ngoài hoặc cửa số trình duyệt | - Xác định vị trí tương đối theo thành phần bên ngoài, bị ảnh hưởng bởi left, top,... | - Giúp thành phần cố định 1 chỗ, không ảnh hưởng khi scrall màn hình | |
| 33 | | **Q:** Responsive Web Design là gì?  🡺 **A:**  - Là khái niệm thể hiện trang Web trên tất cả các loại thiết bị với đầy đủ nội dung và không xuất hiện trên thanh scrollbar nằm ngang |
| 34 | | **Q:** Làm thế nào để tạo 1 trang web responsive?  🡺 **A:**  - Dùng media queries để chỉnh sửa css cho thiết bị khác nhau  - Dùng grid view: 12 cột |
| 35 | | **Q:** Bootstrap là gì?  🡺 **A:**  - **L**à một bộ công cụ mạnh mẽ – một tập hợp các công cụ HTML, CSS và JavaScript để tạo và xây dựng các trang web và ứng dụng web. |
| 36 | | **Q:** Một số thành phần thông dụng ở Bootstrap?  🡺 **A:**  - Bao gồm:  + Carousel: dùng hiển thị slide ảnh dạng quảng cáo  + Model: hiển thị hộp thoại thông báo – xác nhận  + Tables: dùng để hiển thị danh sách của đối tượng  + Form: dùng để hiển thị form input nhập dữ liệu vào |
| 37 | | **Q:** Phân biệt Static web và Dynamic web?  🡺 **A:**   |  |  | | --- | --- | | Static web | Dynamic web | | - Web tĩnh với nội dung cố định  - Chỉ xác định dựa trên các ngôn ngữ Front-end, HTML + CSS + JS | - Web động với nội dung thay đổi theo tương tác của người dùng  - Các trang web được xây dựng trên 2 dòng ngôn ngữ:  + Front – end: HTML + CSS + JS  + Back – end: JV (PhP, C#, Python, ...) | |
| 38 | | **Q:** Phân biệt 2 phương thức GET/POST  🡺 **A:**   |  |  | | --- | --- | | GET | POST | | - Kém bảo mật | - Bảo mật hơn | | - Data sẽ hiển thị trên thanh URL | - Data không hiển thị trên thanh URL | | - Các hành động ở phương thức GET có thể access hiển thị trên URL | - Các hành động ở phương thức POST chỉ có thể submit từ form | | - Độ dài dữ liệu tối đa của GET là 2048 | - Độ dài dữ liệu không hạn chế | | - Data được gửi dưới dạng text | - Data có thể gửi dưới dạng (âm thanh, hình ảnh, ...) | |
| 39 | | **Q:** Phân biệt request/response  🡺 **A:**   |  |  | | --- | --- | | Request | Response | | - Là yêu cầu gửi từ Client 🡺 Server | - Là phản hồi từ Server 🡺 Client | |
| 40 | | **Q:** Phân biệt forward/redirect  🡺 **A:**   |  |  | | --- | --- | | Forward | Redirect | | - Là chuyển tiếp từ Servelet sang trang JSP kèm data | - Là chuyển hướng từ servelet này sang servelet khác hoặc có thể ra ngoài website nhưng không kèm data | |
| 41 | | **Q:** JSP là gì? Nói: “Về bản chất JSP cũng chính là Servlet” đúng hay sai? Giải thích?  So sánh giống và khác nhau giữa JSP và Servlet ?  🡺 **A:**  - **Là viết tắt của JavaServer Pages**là một công nghệ để phát triển các trang web động. JSP giúp các nhà phát triển chèn java code vào các trang HTML bằng cách sử dụng các thẻ JSP đặc biệt.  - Về bản chất, JSP cũng chính là Servlet. Vì trong quá trình biên dịch, JSP sẽ được chuyển thành Servlet rồi Servlet này sẽ được biên dịch. Servlet của trang JSP có vòng đời giống như các Servlet thông thường.  - Phân biệt JSP và Servlet   |  |  | | --- | --- | | JSP | Servlet | | - Là mã dựa trên HTML | - Là mã Java | | - Viết dễ | - Viết khó hơn | | - Bước đầu tiên, nó sẽ dịch mã Java rồi biên dịch nên làm việc chậm hơn | - Làm việc nhanh hơn | | - Chỉ chấp nhận duy nhất yêu cầu của giao thức HTTP | - Chấp nhận được tất cả yêu cầu của các giao thức | | - Bật tự động quản lý phiên | - Người dùng phải thực hiện thao tác bật quản lý phiên | | - Không cho phép người dùng ghi đè lên phương thức service() | - Cho phép | | - Có khả năng sửa lỗi nhanh | - Sửa lỗi chậm vì phải trải qua các hoạt động như tải lại, tái biên dịch và tái khởi động máy chủ | |
| 42 | | **Q:** JSTL là gì? Giới thiệu một số thẻ bạn đã sử dụng ?  🡺 **A:**  - Là bộ thư viện thẻ được xây dựng dựa trên ngôn ngữ Java dùng trên JSP  - Các thẻ thông dụng:  + <c:out>: in dữ liệu ra màn hình dạng text  + <c:set>: để tạo 1 biến chứa dữ liệu trong jsp  + <c:remove>: xóa biến đã tạo ở <c:set>  + <c:forEach>: lặp để thực hiện hiển thị dữ liệu dạng list  + <c:if>: (rẽ nhánh) hiển thị dựa trên điều kiện trong if  + <c:choose>, <c:when>, <c:otherwise>: sử dụng giống như switch – case |
| 43 | | **Q:** Trình bày kiến trúc MVC? Vai trò của các thành phần trong kiến trúc MVC?  🡺 **A:**  - Gồm 3 phần chính: Model, View, Controller  + Model: là thành phần thực hiện xử lý nghiệp vụ và tương tác trực tiếp với database  + View: là thành phần giao diện để người dùng tương tác với hệ thống  + Controller: là thành phần điều hướng, nhận, gửi dữ liệu từ View và Model |
| 44 | | **Q:** Sự khác nhau giữa MVC1 và MVC2  🡺 **A:**   |  |  | | --- | --- | | MVC1 | MVC2 | | - Controller và View đều là jsp. Yêu cầu từ Client sẽ được gửi tới trang JSP ban đầu được coi là controller xử lý, sau đó truyền lệnh thực thi đến các JavaBeans lấy dữ liệu từ database trả về trang JSP ban đầu theo yêu cầu phía Client  - Là thế hệ đầu tiên sử dụng JSP/JavaBeans để thực hiện mô hình MVC | - Các controller là các Servlet còn Model là các class của java, view là trang JSP  - Là mô hình kiến trúc MVC trên nền web  - Làm việc hiệu quả, logic hơn | | |  |  |  | | --- | --- | --- | | Đặc điểm | MVC1 | MVC2 | | Controller | JSP | Servlet | | Model | Không được xác định rõ ràng | Lớp Java | | View | JSP | JSP | | Logic nghiệp vụ | Được xử lý trong Controller và View | Được xử lý chủ yếu trong Controller | | Truy cập dữ liệu | Được thực hiện trực tiếp trong Controller | Được thực hiện thông qua Model | | Khả năng mở rộng | Thấp | Cao | | Khả năng bảo trì | Thấp | Cao | | | |
| MN | **Q:** Mô tả cách thực hiện tạo và lấy kết quả 1 câu query bất kỳ bằng JDBC?  🡺 **A:**  - B1: Kết nối với database  - B2: Tạo đối tượng statement  - B3: Thực thi câu lệnh SQL:  + Với câu lệnh select thì executeQuery 🡺 trả về đối tượng ResultSet  + Với insert, update, delete thì executeUpdate 🡺 trả về số record bị ảnh hưởng | |
| 46 | **Q:** Phân biệt các loại Statement ?  🡺 **A:**   |  |  |  | | --- | --- | --- | | Statement | PrepareStatement | CallableStatement | | - Dùng tạo câu truy vấn tĩnh 🡺 dễ có lỗ hổng bảo mật | - Kế thừa từ Statement, dùng để tạo câu SQL động, ngăn chặn được SQL injection | - Kế thừa từ Prepare Statement và dùng để gọi tới Store Procedure hoặc Function | | Thực thi các truy vấn SQL tĩnh.  \_Các truy vấn SQL tĩnh là các truy vấn không có tham số. | Thực thi các truy vấn SQL có tham số.  Các truy vấn SQL có tham số là các truy vấn có chứa các giá trị thay đổi được truyền vào khi thực thi truy vấn. | Thực thi các thủ tục được lưu trữ.  Thủ tục được lưu trữ là một chương trình được lưu trữ trong cơ sở dữ liệu và có thể được gọi từ ứng dụng Java. | | |
| 47 | **Q:** Kỹ thuật sử dụng transaction bằng JDBC là như thế nào?  🡺 **A:**  - Thiết lập auto\_commit là false. Khi đã thực hiện các thay đổi với cơ sở dữ liệu và muốn ký thác các thay đổi đó, thì sử dụng phương thức commit(). Hoặc muốn xóa các thay đổi đã được thực hiện trước đó để quay về trạng thái trước khi thực hiện thay đổi khi thấy rằng có lỗi xảy ra, sử dụng phương thức rollback(). | |
| 48 | **Q:** Các tính chất của Transaction?  🡺 **A:**   |  |  | | --- | --- | | Automicity | - Tất cả đều thành công hoặc tất cả đều thất bại | | Consistency | - Bảo đảm tính đồng nhất dữ liệu | | Isolation | - Bảo đảm Transaction này là độc lập với Transaction khác | | Durability | - Khi một Transaction đã được ký thác thì nó sẽ vẫn tồn tại như thế cho dù xảy ra các lỗi | | |
| 49 | **Q:** Phương pháp gọi 1 SP ở JDBC?  🡺 **A:**  - Tạo CallableStatement  + Đối với procedure không có tham số: {call procedure\_name}  + Đối với procedure có tham số: {call procedure\_name(?, ?, ?...)}  + Đối với procedure có giá trị trả về: {?= call procedure\_name(?, ?, ?...)} | |
| 50 | **Q:** Phân biệt Ant, Maven, Gradle?  🡺 **A:**   |  |  |  | | --- | --- | --- | | Ant | Maven | Gradle | | - Là công cụ được sử dụng để quản lý thư viện ở Project JSP Servlet. Cách quản lý của Ant có khả năng thủ công và phức tạp | - Ra đời khắc phục các nhược điểm của Ant, nó sử dụng file cấu hình pom.xml để quản lý thư viện. Flie pom.xml này vẫn còn dài dòng do sử dụng xml | - Ra đời khắc phục các nhược điểm của Ant và Maven, nó hỗ trợ được nhiều ngôn ngữ hơn. Gradle sử dụng cấu hình build.gradle để quản lý thư viện. Flie này sử dụng ngôn ngữ Groovy ngắn gọn hơn Maven |  |  |  |  |  | | --- | --- | --- | --- | | Tính năng | Ant | Maven | Gradle | | Ngôn ngữ định cấu hình | XML | XML | Groovy | | Mục tiêu | Tự do | Có cấu trúc | Tự do | | Quản lý phụ thuộc | Có | Có | Có | | Quản lý vòng đời | Có | Có | Có | | Khả năng mở rộng | Có | Có | Có | | Khả năng tích hợp | Có | Có | Có | | Độ phức tạp | Cao | Trung bình | Trung bình | | Thời gian học tập | Cao | Trung bình | Trung bình | | |