**BỘ CÂU TRẢ LỜI AUDIT MODEL 3**

|  |  |
| --- | --- |
| 1**.Cơ sở dữ liệu là gì?** | Cơ sở dữ liệu là một hệ thống các thông tin có cấu trúc, được lưu trữ trên các thiết bị lưu trữ nhằm thỏa mãn yêu 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. **Phân biệt các câu lệnh DML & DDL?** | **\*** Phân biệt các câu lệnh DML & DDL?  **DDL giúp thay đổi cấu trúc của cơ sở dữ liệu trong khi DML giúp quản lý dữ liệu trong cơ sở dữ liệu.**  \* Phân biệt các câu lệnh DML & DDL,DML, DDL bao gồm các câu lệnh nào?   * DDL (Data Definition Language) :Ngôn ngữ định nghĩa dữ liệu. * Create : để tạo cơ sở dữ liệu và các đối tượng của nó như (bảng, chỉ mục, dạng xem, thủ tục lưu trữ, hàm và trình kích hoạt) * Alter : thay đổi cấu trúc của cơ sở dữ liệu hiện có * Drop : xóa các đối tượng khỏi cơ sở dữ liệu * Truncate : xóa tất cả các bản ghi khỏi một bảng, bao gồm tất cả các không gian được phân bổ cho các bản ghi được xóa * Comment : thêm ý kiến vào từ điển dữ liệu * Rename : đổi tên một đối tượng * DML(Data Manipulation Language): Ngôn ngữ thao tác dữ liệu. * Select – lấy dữ liệu từ cơ sở dữ liệu * Insert – chèn dữ liệu vào bảng * Update – cập nhật dữ liệu hiện có trong một bảng * Delete – Xóa tất cả các bản ghi khỏi bảng cơ sở dữ liệu * DCL( Data Control Language): Ngôn ngữ điều khiển dữ liệu. * Grant: Lệnh này được sử dụng để cung cấp cho phép người sử dụng cụ thể trên các đối tượng cơ sở dữ liệu cụ thể như bảng, xem vv * Revoke: lệnh này được sử dụng để đưa ra sự cho phép của người dùng cụ thể trên các đối tượng cơ sở dữ liệu cụ thể như bảng, xem vv * TCL(Transaction Control Language): Ngôn ngữ điều khiển giao dịch. * Commit – Chuyển các thay đổi xuống database. * Rollback: Quay lại trạng thái ban đầu trước khi bị transaction chỉnh sửa dữ liệu do xảy ra lỗi trong bên trong xử lý của transaction. * Savepoint – Lưu trạng thái dữ liệu ở một thời điểm bất kỳ của một transaction, chúng ta có thể roll-back về trạng thái này bất cứ khi nào. |
| 3**.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?** | ERD (Entity-Relationship Diagram: sơ đồ quan hệ thực thể) thể hiện mối quan hệ giữa các thực thể có trong database.  *Mục tiêu của mô hình ERD là phân tích dữ liệu rồi chuyển hóa thành 1 bản thiết kế thể hiện các mối quan hệ giữa các thực thể*  Các thành phần có trong ERD:   1. Thực thể(Entity): là đối tượng trong thế giới thực, tồn tại độc lập và phân biệt được với các đối tượng khác 2. Thuộc tính(Atrribute): là những đặc điểm thông tin riieng của đối tượng. Có 2 loại thuộc tính:  * Thuộc tính đơn: k thể chia nhỏ( kí hiệu elip) * Thuộc tính phực hợp: có thể tách ra từ nhiều phần (kí hiệu elip)  1. Mối quan hệ và tập mối quan hệ: là sự liên kết giữa 2 hay nhiều thực thể   Có 3 mối quan hệ chính:   * One to One (1 – 1) * One to many (1 – n) * Many to many (n – n) |
| 4. **Các bước chuyển đổi từ ERD -> Table?** | B1: Biến đổi thực thể thành table theo nguyên tắc:  -Thuộc tính khóa của thực thể chính là khóa chính của table  - Các thuộc tính thông thường của thực thể là thuộc tính của table  B2: Biến đổi thuộc tính đa trị   * Tách thuộc tính đa trị thành bảng mới, xác định mối quan hệ giữa bảng mới và bảng hiện tại, đặt khóa ngoại   B3: Biến đổi quan hệ 1:1   * C1: Đặt thuộc tính khóa ở phía tùy chọn sang bắt buộc * C2: Thêm 1 table mới xác định bởi các thuộc tính khóa của các thực thể liên quan   B4: Biến đổi mối quan hệ 1-n:   * Đặt khóa chính bên 1 là khóa ngoại bên n   B5: Biến đổi mối quan hệ n-n:   * Tạo 1 table mới có khóa chính là tập hợp của 2 khóa chính của các thực thể tham gia vào mối quan hệ |
| 5. **Các bước chuẩn hóa dữ liệu?** | * **Có 4 bước chuẩn hóa dữ liệu nhưng thông thường chỉ cần 3 bước là đủ.**   **+ Dạng chuẩn 1NF:**   * Xác định thuộc tính khóa chính khóa ngoại và thuộc tính bthg.( loại bỏ các thuộc tính đa trị)   **+ Dạng chuẩn 2NF:**   * Phải đạt đc 1NF,loại bỏ các thuộc tính trùng nhau, dư thừa ( tách các thuộc tính không phải là khóa mà không phụ thuộc vào khóa chính.)   **+ Dạng chuẩn 3NF:**   * Phải đạt đc 2NF, loại bỏ các thuộc tính bắt cầu.   \*Đưa ví dụ cụ thể ?  \* Mục đích của chuẩn hóa dl? |
| 6. **Ràng buộc (constraint) là gì? Các loại ràng buộc trong MySQL?** | Constraint là những quy tắc được áp dụng trên các cột dữ liệu, trên bảng. Được sử dụng để kiểm tra tính hợp lệ của dữ liệu vào, đảm bảo tính chính xác, tính toàn vẹn của dữ liệu.  Các loại ràng buộc:  **- NOT NULL** : Đảm bảo dữ liệu của cột không được nhận giá trị NULL.  **- DEFAULT** : Cung cấp một giá trị mặc định cho một cột trong trường hợp dữ liệu của cột không được nhập vào hay không được xác định.  **- UNIQUE** : Đảm bảo rằng dữ liệu của cột là duy nhất, tất cả các giá trị trong một cột là khác nhau không trùng lặp.  - **PRIMARY Key** : Dùng để thiết lập khóa chính trên bảng, giá trị của cột làm khóa chính phải là duy nhất, không được trùng lặp. Việc khai báo ràng buộc khóa chính yêu cầu các cột phải NOT NULL. Có 2 loại PK (PK đơn ; PK phức hợp: được tạo từ 2 cột trở lên)  **- FOREIGN Key** : Dùng để thiết lập khóa ngoại trên bảng, tham chiếu đến bảng khác thông qua giá trị của cột được liên kết. Giá trị của cột được liên kết phải là duy nhất trong bảng kia.  - **CHECK** : Đảm bảo tất cả các giá trị trong một cột thỏa mãn một số điều kiện. |
| 7.Khóa chính (Primary key) là gì? “Khóa chính chứa một hoặc nhiều thuộc tính khóa” nghĩa là gì? | Khóa chính (Primary Key, hay ràng buộc khóa chính) được sử dụng để định danh duy nhất mỗi bản ghi (dòng) trong bảng của cơ sở dữ liệu.   * “Khóa chính chứa một hoặc nhiều thuộc tính khóa” nghĩa là gì?   Khóa chính chia làm 2 loại   * PK đơn * Pk phức hợp: được tạo ra từ 2 cột trở lên. |
| 8. Khóa ngoại (foreign key) là gì? | Khoá ngoại (foreign key) là cơ chế để tạo liên kết giữa các bảng trong cùng một cơ sở dữ liệu. |
| 9.**Phân biệt DELETE và TRUNCATE?** | Cú pháp:  Delete: DELETE FROM table\_name;  Truncate: TRUNCATE TABLE table\_name;  Có một điểm khác nhau giữa lệnh DELETE và lệnh TRUNCATE. Lệnh **DELETE** chỉ xóa các hàng từ bảng dựa trên điều kiện đã được định nghĩa bởi mệnh đề WHERE hoặc xóa tất cả các hàng từ bảng nếu điều kiện này không được xác định. Nhưng nó **không** giải phóng bộ nhớ chứa bảng đó.  Trong khi đó, lệnh TRUNCATE xóa tất cả các hàng từ bảng và **giải phóng bộ nhớ** chứa bảng đó.  Ví dụ: Để xóa tất cả các hàng từ bảng SINHVIEN, truy vấn như sau:   * TRUNCATE TABLE SINHVIEN; * Dữ liệu bị xóa bởi DELETE có thể restored được còn TRUNCATE thì không. * Tùy vào từng trường hợp thì bạn sẽ dùng DELETE hoặc TRUNCATE, TRUNCATE sẽ luôn nhanh hơn vì nó tiêu tốn ít bộ nhớ hơn, các bạn có thể cân nhắc điều này khi cần xóa 1 bảng lớn vs nhiều record. |
| 10.**Các loại JOIN? Phân biệt?** | + Inner join: Lấy phần tử chung giữa 2 bảng  + Outer join:   * Left join: Lấy phần chung và toàn bộ bảng bên trái * Right join: Lấy phần chung và toàn bộ bảng bên phải * Full join: lấy hết |
| 11. **Phân biệt JOIN và UNION?** | |  |  | | --- | --- | | JOIN | UNION | | JOIN kết hợp dữ liệu từ nhiều bảng dựa trên một điều kiện phù hợp giữa chúng. | SQL kết hợp tập hợp kết quả của hai hoặc nhiều câu lệnh SELECT. | | Nó kết hợp dữ liệu thành các cột mới. | Nó kết hợp dữ liệu thành các hàng mới | | Số cột được chọn từ mỗi bảng có thể không giống nhau. | Số cột được chọn từ mỗi bảng phải giống nhau. | | Kiểu dữ liệu của các cột tương ứng được chọn từ mỗi bảng có thể khác nhau. | Kiểu dữ liệu của các cột tương ứng được chọn từ mỗi bảng phải giống nhau. | | Nó có thể không trả về các cột riêng biệt. | Nó trả về các hàng riêng biệt. | |
| 12. **Phân biệt UNION và UNION ALL?** | * UNION có tác dụng kết hợp hai lệnh SELECT và loại bỏ đi những dòng bị trùng lặp chỉ để lại một dòng duy nhất, tuy nhiên không phải lệnh nào cũng gộp được mà phải thoải điều kiện sau:  1. Tên của các column phải giống nhau 2. Thứ tự các column phải giống nhau 3. Tổng các column phải bằng nhau  * UNION ALL có cú pháp và tác dụng tương tự như UNION, chỉ có điều là lệnh UNION ALL sẽ giữ lại tất cả dữ liệu kể cả những dòng bị trùng lặp. |
| 13. Phân biệt EXIST và IN? | |  |  | | --- | --- | | EXISTS | IN | | Nó được sử dụng để kiểm tra sự tồn tại của dữ liệu trong một truy vấn con. Nói cách khác, nó xác định liệu giá trị có được trả về hay không. | Nó được sử dụng để giảm thiểu nhiều điều kiện OR. | | Nó không so sánh các giá trị giữa truy vấn con và truy vấn mẹ. | Nó so sánh các giá trị giữa truy vấn con (truy vấn con) và truy vấn mẹ | | Nó dừng để thực hiện thêm khi đáp ứng điều kiện tích cực duy nhất. | Nó quét tất cả các giá trị bên trong khối IN. | | Nó trả về TRUE hoặc FALSE. Do đó, chúng tôi không thể sử dụng nó để so sánh các giá trị NULL. | Nó có thể trả về TRUE, FALSE hoặc NULL. Do đó, chúng ta có thể sử dụng nó để so sánh các giá trị NULL. | | Chúng ta chỉ có thể sử dụng nó trên các truy vấn con. | Chúng ta có thể sử dụng nó trên các truy vấn con cũng như với các giá trị. | | Nó thực thi nhanh hơn khi kết quả truy vấn con lớn. Nó hiệu quả hơn IN vì nó xử lý các giá trị Boolean hơn là các giá trị của chính nó. | Nó thực thi nhanh hơn khi kết quả truy vấn con ít hơn. | |  |  |   Cả hai từ "EXIST" và "IN" đều liên quan đến kiểm tra sự tồn tại của dữ liệu trong cơ sở dữ liệu hoặc trong một tập hợp dữ liệu nào đó. Tuy nhiên, chúng được sử dụng trong các trường hợp khác nhau.   * "EXIST" được sử dụng để kiểm tra xem một giá trị hoặc một bản ghi có tồn tại trong cơ sở dữ liệu hay không. Ví dụ: "SELECT \* FROM users WHERE email = '[example@gmail.com](mailto:example@gmail.com)' AND EXISTS (SELECT \* FROM orders WHERE orders.user\_id = users.id)" sẽ trả về tất cả các người dùng có địa chỉ email là '[example@gmail.com](mailto:example@gmail.com)' và đã từng đặt hàng. * "IN" được sử dụng để kiểm tra xem một giá trị có nằm trong một danh sách giá trị hay không. Ví dụ: "SELECT \* FROM products WHERE category IN ('book', 'movie', 'music')" sẽ trả về tất cả các sản phẩm thuộc danh mục 'book', 'movie', hoặc 'music'.   Vì vậy, "EXIST" thường được sử dụng để kiểm tra sự tồn tại của các bản ghi trong cơ sở dữ liệu, trong khi "IN" thường được sử dụng để kiểm tra sự tồn tại của các giá trị trong một danh sách giá trị cụ thể. |
| 14. Phân biệt AND và OR? | * AND cho phép sử dụng nhiều điều kiện trong mệnh đề WHERE của câu lệnh SQL. * OR được sử dụng để kết hợp nhiều điều kiện trong mệnh đề WHERE của lệnh SQL. |
| **15.**  **Phân biệt GROUP BY và ORDER BY?** | |  |  | | --- | --- | | GROUP BY | ORDER BY | | Nó được sử dụng để nhóm các hàng có cùng giá trị. | Nó sắp xếp tập kết quả theo thứ tự tăng dần hoặc giảm dần. | | Nó có thể được cho phép trong câu lệnh CREATE VIEW. | Nó không được phép trong câu lệnh CREATE VIEW | | Nó kiểm soát việc trình bày các hàng. | Nó kiểm soát việc trình bày các cột. | | Thuộc tính k được nằm dưới hàm tổng hợp trong câu lệnh GROUP BY. | Thuộc tính có thể nằm dưới hàm tổng hợp trong câu lệnh ORDER BY. | | Nó luôn được sử dụng trước mệnh đề ORDER BY trong câu lệnh SELECT. | Nó luôn được sử dụng sau mệnh đề GROUP BY trong câu lệnh SELECT | | Bắt buộc phải sử dụng các hàm tổng hợp trong GROUP BY. | Không bắt buộc phải sử dụng các hàm tổng hợp trong ORDER BY | |  |  |   Cả hai từ "GROUP BY" và "ORDER BY" đều được sử dụng trong câu lệnh SQL để sắp xếp dữ liệu, tuy nhiên, chúng có mục đích khác nhau.   * "GROUP BY" được sử dụng để nhóm các bản ghi dữ liệu theo một hoặc nhiều cột dữ liệu, tạo thành các nhóm dữ liệu dựa trên các giá trị trong các cột đó. Ví dụ: "SELECT category, COUNT(\*) FROM products GROUP BY category" sẽ trả về số lượng sản phẩm trong mỗi danh mục. * "ORDER BY" được sử dụng để sắp xếp các bản ghi dữ liệu theo một hoặc nhiều cột dữ liệu. Ví dụ: "SELECT \* FROM products ORDER BY price DESC" sẽ trả về tất cả các sản phẩm được sắp xếp theo giá từ cao đến thấp.   Tóm lại, "GROUP BY" được sử dụng để nhóm các bản ghi dữ liệu thành các nhóm dựa trên các giá trị trong các cột đó, trong khi "ORDER BY" được sử dụng để sắp xếp các bản ghi dữ liệu theo một hoặc nhiều cột dữ liệu.   * Mệnh đề Group By được sử dụng để nhóm dữ liệu dựa trên cùng một giá trị trong một cột cụ thể. Mặt khác, mệnh đề ORDER BY sắp xếp kết quả và hiển thị nó theo thứ tự tăng dần hoặc giảm dần. * Bắt buộc phải sử dụng chức năng tổng hợp để sử dụng Nhóm Theo. Mặt khác, không bắt buộc phải sử dụng hàm tổng hợp để sử dụng Order By. * Thuộc tính không thể nằm trong câu lệnh GROUP BY trong hàm tổng hợp, trong khi thuộc tính có thể nằm trong câu lệnh ORDER BY trong hàm tổng hợp. * Mệnh đề Group By kiểm soát việc trình bày các bộ giá trị có nghĩa là việc nhóm được thực hiện dựa trên sự giống nhau giữa các giá trị thuộc tính của hàng. Ngược lại, mệnh đề ORDER BY kiểm soát việc trình bày các cột có nghĩa là việc sắp xếp hoặc sắp xếp được thực hiện dựa trên các giá trị thuộc tính của cột theo thứ tự tăng dần hoặc giảm dần. * GROUP BY luôn được đặt sau mệnh đề WHERE nhưng trước mệnh đề ORDER BY. Mặt khác, ORDER BY luôn được sử dụng sau câu lệnh GROUP BY. |
| 16. Muốn nối chuỗi trong MySQL thì sử dụng gì? | Hàm concat có tác dụng là nối các chuỗi lại với nhau với cú pháp như sau:   |  |  | | --- | --- | |  | concat(str1, str2, str2, ... , strn) | |
| 17. | \* Yêu cầu tính điểm trung bình của tất cả các sinh viên thì làm thế nào?  **Sử dụng AVG ở cột điểm số**. Vd:  SELECT  AVG(ten\_cot\_diem) ten\_bang |
| 18. | \* Yêu cầu tính điểm trung bình của từng bạn sinh viên thì làm thế nào?  **Sử dụng AVG ở cột điểm số và Group By theo ID hoặc tên**. Vd:  SELECT name\_student, avg(mark\_column) FROM student GROUP BY name\_student; |
| 19. | \* 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?  **Dùng Having, Group By, AVG**. Vd:  SELECT name\_student, avg(mark\_column) dtb FROM student  HAVING dtb > 5  GROUP BY name\_student; |
| 20. | \* Hiển thị danh sách sinh viên và lớp học tương ứng thì làm thế nào (hai bảng Student và Class có cột chung ClassID)?  **Sử dụng Join**.Vd:  SELECT st.name\_student , cls.class\_name FROM student st  JOIN class cls ON cls.class\_id = st.class\_id; |
| 21 | Hiển thị danh sách sinh viên theo thứ tự tăng dần của điểm thi?  **Select OrderBy cột điểm thi**. Vd:  SELECT \* FROM student ORDER BY mark; |
| 22 | 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?  **Select OrderBy cột tuổi và sd Desc để sx giảm dần**. Vd:  SELECT \* FROM student ORDER BY name\_student DESC, age; |
| 23 | Hiển thị các sinh viên có họ là ‘Le’ và tên là ‘Hai’?  SELECT \* FROM student WHERE student\_name LIKE ‘Le%’ AND student\_name LIKE ‘%Hai’; |
| 24 | Hiển thị các sinh viên có tên bắt đầu là ‘C’?  SELECT \* FROM student  HAVING (REVERSE(SUBSTRING(REVERSE(student\_name), 1, INSTR(REVERSE(student\_name,' ')) LIKE ‘C%’; |
| 25. **Làm sao để có thể tăng tốc độ truy vấn? Nhược điểm là gì?** | - Có thể tăng tốc độ truy vấn nhờ câu lệnh INDEX(chỉ mục)  - Nhược điểm :  Index làm *tăng hiệu năng* của lệnh SELECT nhưng lại *làm giảm hiệu năng của lệnh INSERT, UPDATE và DELETE* bởi vì khi database thay đổi thì chúng ta cần thực hiện lại việc đánh index (Ví A có index là 0, B có index là 1, khi xóa A đi thì B có index là 0, các trường phía sau cũng thay đổi)  \*Chỉ nên đánh index với table ít thêm sửa xóa và với column được sd để timg kiếm nhiều   1. Tốn nhiều tài nguyên: Chỉ mục index có thể tốn nhiều không gian lưu trữ và tài nguyên của hệ thống. 2. Tốc độ cập nhật: Khi thêm, sửa hoặc xóa các bản ghi, chỉ mục index phải được cập nhật, điều này có thể làm chậm quá trình cập nhật dữ liệu. |
| 26. Các dạng tham số ở SP? | * **IN**: Đây là chế độ mặc định (*nghĩa là nếu bạn không định nghĩa loại nào thì nó sẽ hiểu là IN*). Khi bạn sử dụng mức này thì nó sẽ được bảo vệ an toàn, có nghĩa là sẽ không bị thay đổi nếu như trong Procedure có tác động đến *(như truyền tham trị)* * **OUT**: Chế độ này nếu như trong Procedure có tác động thay đổi thì nó sẽ thay đổi theo. Hoạt động giống tham chiếu nên biến truyền vào dạng OUT không cần định nghĩa trước, chính vì vậy khởi đầu nó có giá trị NULL *(như truyền tham chiếu)* * **INOUT**: Đây là sự kết hợp giữa **IN** và **OUT**. Nghĩa là **có thể gán giá trị trước** và **có thể bị thay đổi** nếu trong Procedure có tác động tới |
| 27. **Phân biệt SP, Trigger, Function?** | * Strored procedure: Thủ tục lưu trữ có thể trả về giá trị zero, một hoặc nhiều giá trị. Trong khi hàm phải trả về một giá trị duy nhất (có thể là bảng). * Trigger: không có tham số và k có kết quả trả về. tự động thực thi 1 nv nào đó.   \*Tại sao Trigger k có tham số?   * Funcition: chỉ trả về 1 giá trị duy nhất không thể goi SP.Giá trị có thể là 1 table.   SP, Trigger và Function là các đối tượng trong cơ sở dữ liệu.   * SP (Stored Procedure) là một khối mã lệnh được lưu trữ trong cơ sở dữ liệu và được sử dụng để thực thi các hoạt động trên cơ sở dữ liệu. SP thường được sử dụng để thực hiện các tác vụ phức tạp, đặc biệt là khi có nhiều thao tác phải thực hiện hoặc khi phải xử lý nhiều dòng dữ liệu. * Trigger là một đối tượng trong cơ sở dữ liệu được sử dụng để tự động thực thi một hành động nào đó khi có sự kiện xảy ra trên cơ sở dữ liệu. Trigger thường được sử dụng để kiểm tra tính hợp lệ của dữ liệu trước khi nó được chèn hoặc cập nhật vào cơ sở dữ liệu. * Function là một khối mã lệnh được sử dụng để thực hiện một tác vụ cụ thể và trả về một giá trị. Function thường được sử dụng để tính toán và trả về các giá trị phức tạp, giúp giảm thiểu sự trùng lặp mã lệnh và tăng tính tái sử dụng của mã.   Tóm lại, SP được sử dụng để thực hiện các tác vụ phức tạp trên cơ sở dữ liệu, Trigger được sử dụng để tự động thực thi các hành động khi có sự kiện xảy ra trên cơ sở dữ liệu và Function được sử dụng để tính toán và trả về giá trị. |
| 28. **CSS là gì? Các cách nhúng CSS vào tài liệu html? Độ ưu tiên?** | - CSS là chữ viết tắt của **Cascading Style Sheets**, nó là một ngôn ngữ được sử dụng để tìm và định dạng lại các phần tử được tạo ra bởi các ngôn ngữ đánh dấu (HTML). Nói ngắn gọn hơn là ngôn ngữ tạo phong cách cho trang web  - Các cách nhúng CSS vào tài liệu html  + Inline style: khai báo trực tiếp trong thuộc tính stylr của HTML  + Internal stylesheet: sd thẻ <style> ở trong <head>  + External stylesheet: sd thẻ <link> để nhúng 1 file css từ bên ngoài vào.  - Độ ưu tiên trong CSS:  + Inline  + Internal ~ External: Với 2 cách internal và external thì cách nào đc sử dụng gần thẻ hơn thì sẽ ưu tiên cách đó   * *Để phá vỡ sự ưu tiên thì sd !important*  1. Inline styles (nhúng trực tiếp vào phần tử HTML) có độ ưu tiên cao nhất. 2. Internal styles (nhúng trong phần đầu của tài liệu HTML) có độ ưu tiên cao hơn so với external styles (nhúng từ tệp bên ngoài). 3. Nếu có nhiều internal hoặc external styles, trình duyệt sẽ áp dụng theo thứ tự xuất hiện trong tài liệu HTML, với styles xuất hiện sau có độ ưu tiên cao hơn. 4. Nếu các styles cùng nhau có cùng độ ưu tiên, trình duyệt sẽ áp dụng theo độ ưu tiên của các selectors (các phần tử, class hoặc ID được áp dụng styles). Các selector có độ ưu tiên cao hơn sẽ được áp dụng trước. |
| 29. Phân biệt giữa bộ chọn class và bộ chọn id? | |  |  | | --- | --- | | **ID** | **Class** | | - Khai báo bằng “#” | - Khai báo bằng “.” | | - Là duy nhất, không được trùng nhau, chỉ sử dụng cho một phần tử duy nhất | - Có thể trùng nhau, có thể sử dụng cho nhiều class | |
| 30**. Trình bày mô hình hộp (box model) trong CSS** | - **Box model** là hộp bao quanh 1 thẻ html  - Có 4 thành phần:  + **content**: nội dung của thẻ HTML  + **padding**: khoảng cách từ content => border  + **border**: độ dày đường viền của thẻ HTML  + **margin**: khoảng cách từ box model này đến box model khác. |
| **31. Trình bày thuộc tính float, clear?** | - Float:  **Thuộc tính Float** sử dụng để chuyển một phần tử sang góc trái hoặc phải của không gian bao quanh nó, rất cần thiết trong việc định dạng bố cục trang.  Theo mặc định, tất cả các phần tử HTML là *không float*.  *Thuộc tính Float có thể có một trong các giá trị sau:*   * ***left****: Cố định phần tử về bên trái.* * ***right:****Cố định phần tử về bên phải.* * ***none:****Nằm tại chính vị trí của nó (trạng thái bình thường).* * ***inherit****(kế thừa): Phần tử kế thừa giá trị từ float cha.*   - Clear:  **Clear** gần như là ngược lại với *float.* Thuộc tính *Clear* ngăn chặn thành phần A chiếm vùng không gian của thành phần B (với thành phần B là thành phần sử dụng *float).* Đôi khi không muốn float ở một số tình huống nào đó ta sẽ dùng clear để khắc chế.  Nói một cách đơn giản, *clear* sử dụng để giải quyết vấn đề trong mục **chú ý** của thuộc tính *float* phía trên.  *Thuộc tính float có thể có một trong các giá trị sau:*   * ***left:****Tràn về phía bên trái.* * ***right:****Tràn về phía bên phải.* * ***none:****Cho phép tràn lên cả hai phía (mặc định).* * ***both:****Không cho phép tràn về bên nào*   ***inherit****(kế thừa): Phần tử kế thừa giá trị từ float cha.* |
| **32. Thuộc tính position dùng để làm gì? Phân biệt các giá trị: static, absolute, relative và fixed?** | - Position trong Css được sử dụng để định vị vị trí hiển thị của các phần tử HTML  \* Phân biệt các giá trị: static, absolute, relative và fixed?   |  |  |  |  | | --- | --- | --- | --- | | **Static** | **Absolute** | **Relative** | **Fixed** | | Đây được xem là giá trị hiển thị Position trong css một cách mặc định, ko bị ảnh hưởng bởi top, left, bottom, right | Căn chỉnh so với thẻ cha chứa nó (chú ý thẻ cha cần có position khác static) | Căn chỉnh so vs static, có thẻ dùng kết hợp vs top, left, bottom, right | Cố định so với màn hình, ko bị ảnh hưởng bởi scroll | |
| 33. Responsive Web Design là gì? | **Responsive web design (RWD) hay còn gọi: Thiết kế web thích ứng**là kỹ thuật đc sd để trang web có thẻ hiển thị tốt (k bị mất nd ) trên nhiều thiết bị với kích thước và trình duyệt khác nhau  \**Grid-View: Áp dụng để thiết kế 1 trang web theo dạng lưới -> có nhiều dòng và nhiều cột.* |
| 34. Làm thế nào để tạo 1 trang web responsive? | Có 2 cách :   1. Khai báo meta viewport   Meta viewport nghĩa là một thẻ thiết lập cho trình duyệt hiển thị tương ứng với kích thước màn hình   1. SD gird view để chia dạng 12 cột và sd @media query   - Media Queries cho phép bạn định kiểu các phần tử dựa trên chiều rộng  khung nhìn  - Media Queries là một phần quan trọng của responsive web, thường được sử dụng cho bố cục lưới, font-size, margin và padding khác nhau giữa kích thước và hướng màn hình. |
| 35. Bootstrap là gì? | * Bootstrap là một front-end framework miễn phí để phát triển web nhanh hơn và dễ dàng hơn. * Bootstrap bao gồm các mẫu thiết kế dựa trên html, css và javascipt * Bootstrap cung cấp khả năng tạo các thiết kế responsive. |
| **36.Một số thành phần thông dụng ở Boopstrap** | - Navbar: thanh điều hướng thường dùng để làm header  - Dropdown menu: thường dùng để làm menu tùy chọn  - Carousel: dùng để hiển thị dạng slide show  - Card: thường dùng cho chức năng danh sách.  - Container/container-fluid: dùng để thiết kế phần thân body chứa thông tin  - Spinner: loading  Bootstrap là một framework phổ biến được sử dụng để phát triển các trang web đáp ứng (responsive) và thân thiện với người dùng. Dưới đây là một số thành phần thông dụng của Bootstrap:   1. Grid System: Hệ thống lưới được Bootstrap sử dụng để xây dựng cấu trúc trang web. Hệ thống lưới giúp cho các phần tử trên trang web được sắp xếp theo cột và hàng để tạo ra một bố cục linh hoạt và đáp ứng. 2. Navigation Bar: Thanh điều hướng cho phép người dùng dễ dàng truy cập đến các trang web khác nhau và các chức năng trên trang web. Thanh điều hướng có thể được tùy chỉnh để phù hợp với phong cách và mục đích của trang web. 3. Forms: Bootstrap cung cấp một loạt các thành phần để tạo ra các biểu mẫu trên trang web, bao gồm các trường đầu vào (input), các nút (button), các ô chọn (select), và nhiều hơn nữa. 4. Buttons: Bootstrap cung cấp các nút được thiết kế sẵn với nhiều kiểu dáng và màu sắc khác nhau. Các nút này có thể được sử dụng để thực hiện các hành động trên trang web, ví dụ như chuyển trang, đăng ký, hoặc mua hàng. 5. Icons: Bootstrap cung cấp một bộ các biểu tượng (icon) để sử dụng trên trang web. Các biểu tượng này giúp cho các phần tử trên trang web trở nên rõ ràng và dễ hiểu hơn. 6. Typography: Bootstrap cung cấp các kiểu chữ và định dạng để tạo ra các phần tử văn bản trên trang web, bao gồm các tiêu đề và đoạn văn bản. 7. Modal: Modal là một cửa sổ nhỏ hiển thị trên trang web để yêu cầu người dùng thực hiện một hành động nhất định trước khi tiếp tục truy cập trang web. Bootstrap cung cấp các thành phần để tạo ra các modal trên trang web. 8. Carousel: Carousel là một thành phần cho phép hiển thị các hình ảnh hoặc nội dung khác nhau trên trang web. Bootstrap cung cấp các thành phần để tạo ra các carousel trên trang web.   Tất cả các thành phần này đều được thiết kế để giúp các nhà phát triển tạo ra các trang web đáp ứng và thân thiện với người dùng một cách nhanh chóng và dễ dàng. |
| 37. **Phân biệt Static web và Dynamic web?** | |  |  | | --- | --- | | **Static web** | **Dynamic web** | | Là loại website mà nd k thay đổi theo thời gian thường được viết bằng HTML, CSS, JS | Là loại website mà nd có thể thay đổi theo thời gian ngữ cảnh hoặc sự tương tác của người dùng. Thường được viết bằng nn Back-end như Java, PHP, C#  \*Thông thường được lưu trữ ở 1 máy chủ web (web server) | |
| 38.**Phân biệt get và post** | |  |  | | --- | --- | | **Get** | **Post** | | Data hiển thị ở URL | Ko hiển thị ở URL | | Kém bảo mật | Bảo mật ( được mã hóa) | | Nhanh hơn | Chậm hơn | | Chỉ gửi dc dạng text | Gửi dc text, hình ảnh, video vvv( nhị phân ) | | Dung lượng giới hạn là 2048 kí tự ( tùy vào trình duyệt) | Ko giới hạn ( tùy vào trình duyệt) | | Không bao giờ sử dụng phương thức GET nếu gửi password hoặc thông tin nhay cảm lên Server. | Thường dùng để gửi thông tin bảo mật | | Đối với dữ liệu ít thay đổi thường dùng phương thức GET để truy xuất và xử lý nhanh hơn. | Đối với những dữ liệu luôn được thay đổi thì thường sử dụng phương thức POST | |
| 39. Phân biệt request và response | \* Request và response  Request (yêu cầu) và Response (phản hồi) là hai khái niệm quan trọng trong các ứng dụng web. Dưới đây là các khái niệm này:   * Request (yêu cầu) là một thông điệp được gửi từ một client tới một server, yêu cầu server thực hiện một hoặc nhiều hành động nhất định. Request bao gồm các thông tin như URL, phương thức HTTP (GET, POST, PUT, DELETE,..), các tham số và dữ liệu được gửi lên server, và các header đi kèm. * Response (phản hồi) là một thông điệp được gửi từ server tới client, trả về kết quả của yêu cầu từ client. Response bao gồm các thông tin như mã trạng thái HTTP (200 OK, 404 Not Found,..), các header đi kèm, và nội dung phản hồi, ví dụ như HTML, JSON, XML,..  |  |  | | --- | --- | | **Request** | **Response** | | Request có thể hiểu nhanh là thông tin gửi từ client lên server, lúc này phía server sẽ phân tích yêu cầu và sẽ gửi luồng xử lý tới vị trí vị trí lưu trữ của mã nguồn PHP (hoặc mã nguồn bất kì) và nhiệm vụ của các mã nguồn là tiếp nhận yêu cầu, phân tích request đó và trả kết quả lại cho client. | Reponse là dữ liệu mà server trả về cho client. Ví dụ khi bạn nhập vào địa chỉ domain thì kết quả trả về (*response*) chính là giao diện của website và các thông tin của header. Như vậy dữ liệu mà server trả về là những đoạn mã HTML kèm theo các thông tin của header | |
| 40. **Phân biệt forward và redirect** | \*Forward và redirect  Forward và Redirect là hai khái niệm quan trọng trong phát triển ứng dụng web. Dưới đây là sự khác nhau giữa Forward và Redirect:   * Forward là quá trình chuyển hướng yêu cầu từ một servlet hoặc JSP (JavaServer Pages) tới một servlet hoặc JSP khác trong cùng một ứng dụng web. Khi sử dụng Forward, yêu cầu từ client được chuyển hướng trực tiếp tới servlet hoặc JSP mới, và kết quả trả về từ servlet hoặc JSP mới sẽ được gửi trực tiếp tới client. Trong quá trình Forward, URL trên trình duyệt của client không thay đổi. * Redirect là quá trình chuyển hướng yêu cầu từ một URL của một ứng dụng web tới một URL khác trong cùng hoặc khác ứng dụng web. Khi sử dụng Redirect, ứng dụng web sẽ trả về một mã trạng thái HTTP (HTTP status code) cho client và yêu cầu client thực hiện một yêu cầu mới tới URL mới được chỉ định. Trong quá trình Redirect, URL trên trình duyệt của client sẽ thay đổi theo URL mới được chỉ định.   Sự khác nhau chính giữa Forward và Redirect là ở cách thức xử lý yêu cầu từ client. Forward được sử dụng để chuyển hướng yêu cầu giữa các servlet hoặc JSP trong cùng một ứng dụng web, trong khi Redirect được sử dụng để chuyển hướng yêu cầu giữa các URL khác nhau, có thể là trong cùng hoặc khác ứng dụng web. Khi sử dụng Forward, dữ liệu yêu cầu và phản hồi được giữ nguyên và được xử lý bởi một servlet hoặc JSP mới. Khi sử dụng Redirect, dữ liệu yêu cầu và phản hồi sẽ được xử lý bởi hai servlet hoặc JSP khác nhau.   |  |  | | --- | --- | | **Forward** | **Redirect** | | Forward là chuyển tiếp từ webserver sang trang khác kèm theo dữ liệu | Redirect là chuyển hướng sang trang khác(không kèm theo dữ liệu) | | Khi một yêu cầu (request) của trình duyệt gửi tới một **Servlet**, nó có thể chuyển tiếp yêu cầu tới một trang khác (hoặc một servlet khác). Địa chỉ trên trình duyệt của người dùng vẫn là đường dẫn của trang đầu tiên, nhưng nội dung của trang do trang được chuyển tiếp tới tạo ra. | Khi một yêu cầu (request) từ phía người dùng tới một Servlet (Trang A), servlet này có thể chuyển yêu cầu này tới một trang khác (Trang B), và kết thúc nhiệm vụ của nó.  Trang được chuyển hướng tới có thể là trang trong ứng dụng của bạn,  hoặc có thể là một trang bất kỳ. Địa chỉ trên trình duyệt của người dùng  lúc này sẽ hiển thị đường dẫn của trang B. | |
| 41. **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** | Đúng, về bản chất JSP cũng chính là Servlet. JSP (JavaServer Pages) và Servlet đều là các công nghệ của Java EE (Java Enterprise Edition) được sử dụng để phát triển các ứng dụng web.  JSP được thiết kế để tạo ra các trang web động bằng cách sử dụng các thẻ HTML và các đoạn mã Java được nhúng vào giữa các thẻ HTML. Khi JSP được tải lên server, nó được biên dịch thành một Servlet, và Servlet này sẽ thực hiện các hành động tương tự như khi phát triển các Servlet truyền thống.  Giống nhau:   * Cả JSP và Servlet đều được sử dụng để phát triển các ứng dụng web sử dụng Java EE. * Cả JSP và Servlet đều có thể truy cập các đối tượng Java và các API của Java EE. * Cả JSP và Servlet đều có thể tương tác với database để lưu trữ và truy xuất dữ liệu.   Khác nhau:   * JSP cho phép lập trình viên tự do sử dụng các thẻ HTML để thiết kế giao diện người dùng, trong khi Servlet tập trung vào xử lý logic của ứng dụng web. * JSP có thể sử dụng các thư viện tag (tag library) để đơn giản hóa việc sử dụng các thẻ HTML, trong khi Servlet không có chức năng này. * JSP được biên dịch thành Servlet trước khi được chạy trên server, trong khi Servlet được biên dịch thành bytecode trực tiếp trên server. * JSP thường được sử dụng để hiển thị dữ liệu động, trong khi Servlet thường được sử dụng để xử lý logic phức tạp của ứng dụng web.   Tóm lại, JSP và Servlet là hai công nghệ quan trọng trong phát triển ứng dụng web sử dụng Java EE. JSP cũng chính là Servlet vì nó được biên dịch thành Servlet trước khi thực thi trên server. Cả JSP và Servlet đều có những đặc điểm giống nhau và khác nhau, tùy thuộc vào mục đích sử dụng của ứng dụng web và nhu cầu của lập trình viên.  JSP( Java Server Page) là nn kịch bản ở phía sever, cho phép người dùng tạo ra các trang web động.  \*“Về bản chất JSP cũng chính là Servlet đúng hay sai? Giải thích?  Về bản chất JSP cũng chính là Servlet là đúng vì JSP sẽ phải biên dịch ra Servlet rồi mới chạy được.  \* So sánh giống và khác nhau giữa JSP và Servlet  - Giống:  Cả 2 đều dùng đc cả HTML và Java. Đều nằm ở phía server và xử lý dl động. Đều chạy với web container.Đều là những công nghệ quan trọng của Java EE.  - Khác:  +Servlet: mạnh ở xử lý logic nghiệp vụ, yếu về giao diện  +JSP: mạnh ở xử lý giao diện, yếu về logic nghiệp vụ  +Ở Servlet, mã HTML nằm trong mã Java. Còn ở JSP, mã Java nằm trong mã HTML. |
| 42. JSTL là gì? Giới thiệu một số thẻ bạn đã sử dụng ? | JSTL (JSP standard tag library) là bộ thư viện thẻ bổ sung cho JSP với mục đích là tối ưu lập trình logic cho JSP(Vì nếu viết như hiện tại, file JSP chứa cùng lúc 2 thành phần là HTML và Java, sẽ gây khó khăn cho việc phát triển => JSTL ra đời  để hỗ trợ và khắc phục hạn chế này.)   |  |  | | --- | --- | | **Thẻ** | **Mô tả** | | <c:out> | Để viết một cái gì đó trong trang JSP, bạn có thể sử dụng EL cũng có thẻ này | | <c:redirect> | Chuyển hướng (redirect) yêu cầu tới một nguồn dữ liệu khác. | | <c:set> | Sét đặt giá trị biến cho bởi phạm vi. | | <c:remove> | Loại bỏ biến ra khỏi phạm vi đã cho. | | <c:catch> | Bắt ngoại lệ và gói vào một đối tượng. | | <c:if> | Điều kiện logic đơn giản, sử dụng với EL và bạn có thể sử dụng nó để xử lý các trường hợp ngoại lệ  từ *<c:catch>* | | <c:choose> | Tag có điều kiện đơn giản mà thiết lập một bối cảnh cho các trường hợp loại trừ lẫn nhau có điều kiện,  đánh dấu bằng *<c:when>* và *<c:otherwise>* | | <c:when> | Thẻ con của *<c:choose>*, khi một điều kiện tại when là đúng. | | <c:otherwise> | Thẻ con của *<c:choose>*, khi tất cả các điều kiện *<c:when>* là sai. | | <c:forEach> | Dùng để lặp trên một tập hợp. | | <c:url> | Để tạo một URL với các tham số tùy chọn (optional query string parameters). | |
| 43. **Trình bày kiến trúc MVC? Vai trò của các thành phần trong kiến trúc MVC?** | MVC: là viết tắt của Model – View – Controller  -Kiến trúc MVC đc chia làm 3 tầng:  + Model: biểu diễn dl và thực hiện logic nghiệp vụ chương trình.  + View: hiển thị dl, cung cấp giao diện tương tác vs người dùng.  + Controller: đón nhận request của ng dùng, kết nối Model và View |
| 44. **Sự khác nhau giữa MVC1 và MVC2**  *// Sự khác, giống nhau, ưu, nhược điểm của MVC1 & MVC2.* | MVC1 (Model-View-Controller 1) và MVC2 (Model-View-Controller 2) là hai kiểu kiến trúc (architecture) được sử dụng để phát triển các ứng dụng web. Dưới đây là sự khác nhau giữa MVC1 và MVC2:   1. Kiến trúc MVC1:  * Kiến trúc MVC1 bao gồm một Servlet đóng vai trò là Controller, xử lý yêu cầu từ client và tạo ra các đối tượng Java để hiển thị nội dung trên trang web. * Thông thường, View (giao diện người dùng) được tạo ra bằng cách sử dụng các thẻ HTML và Java code được nhúng vào trong các trang JSP (JavaServer Pages). * Model (mô hình dữ liệu) được thực hiện bởi các đối tượng Java và được sử dụng để truy xuất và lưu trữ dữ liệu.  1. Kiến trúc MVC2:  * Kiến trúc MVC2 bao gồm một Servlet đóng vai trò là Controller, xử lý yêu cầu từ client và điều hướng tới các đối tượng Java để xử lý logic và trả về kết quả. * View được tạo ra bằng cách sử dụng các trang JSP hoặc các file HTML và CSS để hiển thị nội dung trên trang web. * Model được thực hiện bởi các đối tượng Java và được sử dụng để truy xuất và lưu trữ dữ liệu.   Sự khác nhau chính giữa MVC1 và MVC2 là cách thức xử lý yêu cầu từ client và tạo ra giao diện người dùng. Trong kiến trúc MVC1, View được tạo ra bằng cách sử dụng các trang JSP, trong khi trong kiến trúc MVC2, View được tạo ra bằng cách sử dụng các trang JSP hoặc các file HTML và CSS. Thêm vào đó, kiến trúc MVC2 còn có sự phân chia rõ ràng hơn giữa Model, View và Controller, đảm bảo tính phân tách (separation of concerns) giữa các thành phần của ứng dụng web.  Tóm lại, MVC1 và MVC2 đều là các kiểu kiến trúc được sử dụng để phát triển các ứng dụng web, với những đặc điểm giống và khác nhau. Tùy thuộc vào nhu cầu của ứng dụng web và tính linh hoạt của lập trình viên mà sẽ lựa chọn sử dụng kiểu kiến trúc phù hợp.   |  |  | | --- | --- | | MVC1 | MVC2 | | - Trong MVC1, controller và view đều là Jsp. MVC1 là thế hệ đầu tiên sử dụng JSP/ JavaBeans để thực hiện mô hình MVC, yêu cầu từ phía client sẽ được gửi tới trang JSP ban đầu được coi là controller xử lý, sau đó truyền lệnh thực thi đến các JavaBeans, các JavaBeans lấy dữ liệu từ database trả về cho trang JSP ban đầu theo yêu cầu phía client. | MVC2 là một mô hình mô tả kiến trúc MVC trên nền web do Sun phát minh, khác với MVC1 trong MVC2 các Controller là các Servlet còn Model là các class của Java, view là các trang JSP. Ở MVC2 ta thấy các logic phức tạp được viết trong Controller (Hard working), các servlet nhận yêu cầu từ client qua sử lý được gửi cho Model và View để tương tác trả kết quả cho client. |   Ta thấy MVC2 không kế thừa từ MVC1 nhưng làm việc hiệu quả và logic hơn.  MVC2 tốt cho các dự án lớn nhưng MVC1 thì không.  Với thành được phân chia theo chức năng, MVC2 có thể dễ dàng tái sử dụng.  //MVC2 phức tạp hơn MVC1 |
| **45. Mô tả cách thực hiện tạo và lấy kết quả 1 câu query bất kỳ bằng JDBC?** | B1: Khai báo driver phù hợp  B2: Tạo connection bởi các thông tin: hostname, database name, user name, pass, port,…  B3: Sd statement, prepastatement, callablestatement để tạo câu lệnh truy vấn  B4: Sd Result để xử lý kết quả trả về ở b3  B5: Dống connection |
| **46. Phân biệt các loại Statement ?** | |  |  |  | | --- | --- | --- | | Statement | Prepastatement | Callablestatement | | Dùng để tạo câu lệnh querry tĩnh | Dùng để tạo câu lệnh truy vấn SQL động (truyền thông qua tham số ?) | Thường dùng để gọi thủ tục Strored procedure | | K bảo mật, có thể bị tấn công SQL Injection, hiệu suất thấp | Bảo mật và hiệu suất cao | Bảo mật và hiệu suất cao | |
| 47. **Kỹ thuật sử dụng transaction bằng JDBC là như thế nào?** | - Transaction là 1 tiến trình có điểm bắt đầu và kết thúc.  - Gồm nhiều phép thực thi nhỏ bên trong, trong đó mỗi phép thực thi đc thực hiện tuần tự độc lập theo nguyên tắc: tất cả thành công thì transaction mới thành công. Chỉ cần 1 phép thực thi thất bại thì transaction thất bại |
| **48.Các tính chất của Transaction?** | * Atomicity( tính nguyên tử ) nghĩa là tất cả thành công hoặc không. * Consistency(tính nhất quán ) bảo đảm rằng tính đồng nhất của dữ liệu. * Isolation(tính độc lập) bảo đảm rằng Transaction này là độc lập với Transaction khác.   Durability (tính bền vững) nghĩa là khi một Transaction đã được ký thác thì nó sẽ vẫn tồn tại như thế cho dù xảy ra các lỗi, … |
| **49. Phương pháp gọi 1 SP ở JDBC?** | - Chúng ta thao tác với các StoredProcedure trong java với đối tượng CallableStatement.  - CallableStatement là đối tượng kế thừa từ đối tượng PreparedStatement, như vậy có các tính chất của 1 PreparedStatement. Chúng ta tạo 1 CallableStatement bằng cú pháp sau:  CallableStatement prepareCall(String sql) throws SQLException  *hoặc*  CallableStatement prepareCall(String sql, int resultSetType, int resultSetConcurrency) throws SQLException.   * Sau đó, chúng ta triệu gọi các stored procedure bằng các cú pháp sau:   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 (function): {?=call procedure\_name(?,?,…)}. |
| 50**.Phân biệt Ant, Maven, Gradle?** | - Ant: Muốn sử dụng 1 thư viện nào đó thì phải import thủ công (down về tự import vào project)  - Maven: Ra đời nhằm khắc phục các nhược điểm của Ant. Sử dụng file pom.xml để quản lý thư viện, tuy nhiên file này còn dài dòng do sử dụng xml bao gồm rất nhiều thẻ.  - Gradle: Khắc phục đc nhược điểm của Ant và Maven, cũng như hỗ trợ nhiều ngôn ngữ. Gradle sử dụng file build.gradle để cấu hình (sử dụng groovy nên ngắn gọn hơn) |
| 51. Phân biệt  Stored Procedure và hàm | |  |  | | --- | --- | | Stored Procedure | Hàm | | Được sử dụng để thực hiện một trình logic nào đó | Được sử dụng để thực hiện phép tính toán | | Không bắt buộc phải có kiểu trả về | Phải có kiểu trả về | | Có thể trả về 0 hoặc nhiều giá trị | Có thể trả về chỉ một giá trị | | Chúng ta có thể gọi hàm từ procedure | Procedure không thể được gọi từ hàm | | Procedure hỗ trợ các tham số input và output | Hàm chỉ hỗ trợ các tham số input | | Xử lý ngoại lệ với khối try/catch có thể được sử dụng trong procedure | Xử lý ngoại lệ với try/catch không thể được sử dụng trong các hàm do người dùng tự định nghĩa | |
| 52. So sánh having và where | |  |  | | --- | --- | | Where | Having | | Mệnh đề WHERE dùng để kiểm tra các điều kiện cho các thuộc tính của bảng, trả kết quả đối chiếu với từng dòng.  \* *WHERE nhanh hơn HAVING* | Mệnh đề HAVING được dùng để giới hạn nhóm các hàng trả về trong bảng, trả kết quả đối chiếu cho các nhóm (Sum, Count, Average,…) được tạo bởi Group by. |   Cả "HAVING" và "WHERE" đều được sử dụng để lọc dữ liệu trong câu lệnh SQL, tuy nhiên, chúng có mục đích khác nhau.   * "WHERE" được sử dụng để lọc các bản ghi trong câu lệnh SELECT, UPDATE hoặc DELETE. Nó áp dụng điều kiện lọc trên các cột dữ liệu trong bảng. Ví dụ: "SELECT \* FROM products WHERE price > 100" sẽ trả về tất cả các sản phẩm có giá trị lớn hơn 100. * "HAVING" được sử dụng để lọc các bản ghi dựa trên các giá trị được tính toán trong câu lệnh GROUP BY. Nó áp dụng điều kiện lọc trên các giá trị được tính toán bởi các hàm như SUM, AVG, MAX, MIN. Ví dụ: "SELECT category, COUNT() FROM products GROUP BY category HAVING COUNT() > 10" sẽ trả về tất cả các danh mục có số lượng sản phẩm lớn hơn 10.   Tóm lại, "WHERE" được sử dụng để lọc các bản ghi trước khi chúng được nhóm lại, trong khi "HAVING" được sử dụng để lọc các giá trị được tính toán sau khi chúng đã được nhóm lại. |
| 53. Trình bày CSS display ? | Thuộc tính display trong css cho phép chúng ta xác định kiểu hiển thị của thẻ HTML   * Inline:   + Chiều rộng của thẻ phụ thuộc vào nội dung bên trong của thẻ  + K thể sd đc các thuộc tính weight, height  + Gía trị mặc định của nhóm thẻ HTML inline <span>,<a>,<img>, <button>   * Block:   + Chiều rộng của thẻ chiếm 100% độ dài trang web  + Dùng đc thuộc tính weight, height  + Gía trị mặc định của nhóm HTML block <div>, <h>, <p>   * Inline – block:   +Kết hợp 2 đặc tính của inline và block ( độ rộng vừa vs nd của thẻ, cho phép chỉnh weight, height ); |
| 54. Sự khác giữa kiểu VARCHAR và CHAR | |  |  | | --- | --- | | Char | Varchar | | Char là đại diện cho một chuỗi có độ dài cố định. Nếu độ dài của dữ liệu được chèn vào nhỏ hơn độ dài cố định của char, nó sẽ bị lấp đầy bởi các khoảng trắng.  Vì độ dài cố định nên tốc độ truy cập nhanh hơn varchar rất nhiều, thậm chí nhanh hơn 50%, nhưng vì độ dài cố định nên nó sẽ chiếm thêm dung lượng. Đối với char, số ký tự tối đa có thể được lưu trữ là 255, bất kể mã hóa là gì | Varchar đại diện cho một chuỗi có độ dài thay đổi và độ dài có thể thay đổi. Dữ liệu được chèn vào là bao lâu, nó được lưu trữ tùy theo độ dài của nó, varchar ngược lại với char về mặt truy cập, nó truy cập chậm, vì độ dài không cố định, nhưng vì thế mà nó không chiếm thêm dung lượng. Đối với varchar, số ký tự tối đa có thể được lưu trữ là 65532 | |
| 55. Vòng đời servlet? | Vòng đời Servlet là chuỗi các phương thức được gọi tuần tự bởi container Servlet khi một yêu cầu HTTP được gửi đến Servlet. Vòng đời Servlet bao gồm các phương thức sau đây:   1. init():  * Phương thức này được gọi khi Servlet được khởi tạo bởi container Servlet. * Phương thức này được sử dụng để thực hiện các hoạt động khởi tạo cho Servlet như đọc các tài nguyên cấu hình hoặc thiết lập các kết nối cơ sở dữ liệu.  1. service():  * Phương thức này được gọi mỗi khi một yêu cầu HTTP được gửi đến Servlet. * Phương thức này được sử dụng để xử lý yêu cầu từ client và trả về kết quả.  1. destroy():  * Phương thức này được gọi khi Servlet bị hủy bởi container Servlet. * Phương thức này được sử dụng để giải phóng các tài nguyên đã được cấp phát cho Servlet như kết nối cơ sở dữ liệu hoặc các đối tượng khác.  1. doGet(), doPost(), doPut(), doDelete(), và các phương thức khác:  * Các phương thức này được gọi bởi container Servlet tương ứng với phương thức HTTP được sử dụng trong yêu cầu gửi đến Servlet. * Các phương thức này được sử dụng để xử lý các yêu cầu HTTP cụ thể và trả về kết quả.  1. Các phương thức khác để xử lý các sự kiện trong vòng đời của Servlet:  * Ví dụ: phương thức doGet() và doPost() có thể được sử dụng để xử lý các yêu cầu GET và POST tương ứng, phương thức init() và destroy() được sử dụng để khởi tạo và giải phóng Servlet.   Tóm lại, vòng đời Servlet là chuỗi các phương thức được gọi tuần tự bởi container Servlet khi một yêu cầu HTTP được gửi đến Servlet. Các phương thức này bao gồm init(), service(), destroy(), các phương thức xử lý yêu cầu HTTP (doGet(), doPost(), doPut(), doDelete(), và các phương thức khác), và các phương thức khác để xử lý các sự kiện trong vòng đời của Servlet.  Một vòng đời của servlet có thể được định nghĩa là toàn bộ quá trình từ khi tạo ra đến khi hủy.   * Servlet được khởi tạo bằng cách gọi phương thức init (). * Phương thức servlet service() được gọi để xử lý yêu cầu của khách hàng.   + Phương thức service() được gọi bởi phương thức container và service gọi các phương thức doGet, doPost, doPut, doDelete, vv. Vì vậy, bạn không cần phải làm gì với phương thức service() nhưng bạn phải ghi đè phương thức doGet() hoặc doPost() phụ thuộc vào loại yêu cầu bạn nhận được từ khách hàng.Các phương thức doGet() và doPost() là các phương thức được sử dụng phổ biến nhất để đáp ứng yêu cầu của máy khách   * Servlet được hủy bằng cách gọi phương thức destroy(). * Cuối cùng, servlet được thu thập bởi bộ sưu tập rác của JVM |
| 56.Thủ tục là gì ? cú pháp triển khai ? tham số ? | * Thủ tục ( Strore produce) là 1 tập hợp các câu lệnh sql dùng để thực thi 1 nv nhất định. * Hoạt động gần giống như hàm, method ở các nn lập trình khác   \*Mục đính:  - Hữu ích chho các thao tác lặp đi lặp lại  - Cho phép khai báo và truyền tham số  - Tính bảo mật  \*Cú pháp:  DELIMITER $$  CREATE PRODUCE produce\_name(..)  BEGIN  / nd xử lý/  END $$  DELIMITER;  Tham số: in, out, inout (nd tham số ở câu 26) |