|  |  |
| --- | --- |
| 1 | Cơ sở dữ liệu (database) 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 thõa 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? DML, DDL bao gồm các câu lệnh nào?  => các nhóm câu lệnh sql  - DML (Data manipulation language) – dùng để thao tác truy vấn dữ liệu  + Insert: them mới  + Update: cập nhật  + Select: truy vấn đọc, chọn  + Delete:Xóa   * DDL (Data definition language) – dùng để định nghĩa cấu trúc dữ liệu   + Create: tạo  +Drop: Xóa  + Alter: chỉnh sửa   * Ngoài ra còn có   + DCL (data control language) – dùng để phân quyền  + TCL (Transaction control language)- dùng cho tiến trình |
| 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): là sơ đồ quan hệ giữa các thực thể * Với 1 dự án đầu tiên sẽ nhận được yêu cầu của khách hàng (SRS – software requirement specifications) – tuy nhiên chỉ như vậy sẽ rất khó tạo database. Vì vậy từ SRS cần thiết kế nên ERD rồi mới tạo database * Các bước thiết kế ERD   + Từ SRS xác định các thực thể (Entity)  + Xác định thuộc tính cho các thực thể (Entity)  +Xác định mối quan hệ giữa các thực thể: 1-1, 1-n, n-n |
| 4 | Các bước chuyển đổi từ ERD -> Table?  =>  - Chuyển đổi thực thể thành table dựa trên nguyên tắc:  + Thuộc tính khóa là khóa chính (PRIMARY KEY) của table  + Thuộc tính thường của entity sẽ là thuộc tính thường của table   * Biến đổi thuộc tính đa trị (ví dụ bằng cấp là đa trị)   + Tách thuộc tính đa trị thành 1 bảng mới -> xác định mối quan hệ giữa bảng mới và bảng hiện tại   * Biến đổi mối quan hệ 1-1   + Đặt thuộc tính khóa ở phía tùy chọn sang phía bắt buộc   * Biến đổi 1-n   + Đặt khóa ngoại (foregin key) bên nhiều (n)  +Đặt primary key ở bên (1)   * Biến đổi mối quan hệ n-n   + Tách thành bảng mới. Có khóa chính là tập hơn 2 khóa của 2 thực thể tham gia vào mối quan hệ => khóa phức hợp (composite key) |
| 5 | Các bước chuẩn hóa dữ liệu?  =>  Chuẩn hóa dữ liệu 1NF -> 2NF -> 3NF   * Chuẩn 1NF   + Loại bỏ thuộc tính đa trị  + Loại bỏ các thuộc tính tính toán hoặc suy diễn   * Chuẩn 2NF   + Đạt chuẩn 1NF  + Loại bỏ các phụ thuộc bộ phận (1 trường không phải khóa chính nhưng chỉ phụ thuộc 1 phần khóa chính) -> loại bỏ các thuộc tính không khóa và có phụ thuộc 1 phần khóa chính   * Chuẩn 3NF   + Đạt chuẩn 2NF  + Loại bỏ các thuộc tính phụ thuộc bắc cầu (tức là thuộc tính phụ thuộc thuộc tính không khóa khác) |
| 6 | Ràng buộc (constraint) là gì? Các loại ràng buộc trong MySQL?   * Constraint là những quy tắc được quy định cho bảng, sử dụng để kiểm tra tính hợp lệ của dữ liệu vào  |  |  | | --- | --- | | NOT NULL | Không được để trống | | DEFAULT | Gán giá trị mặc định trong trường hợp dữ liệu của cột không được nhập vào hay không được xác định. | | UNIQUE | duy nhất, không trùng lặp giá trị trên cùng 1 cột. | | PRIMARY KEY | Dùng để thiết lập khóa chính trên bảng, auto là duy nhất, không trùng lặp | | FOREIGN KEY (Khóa ngoại) | Dùng để thiết lập khóa ngoại trên bảng, dùng liên kết với khóa chính (primary key của bảng khác). Giá trị của cột được liên kết phải là duy nhất trong bảng kia. | | CHECK  Index | Kiểm tra điều kiện  Chỉ mục giúp tang tốc độ tìm kiếm | |
| 7 | Khóa chính (Primary key) là gì? “Khóa chính chứa một hoặc nhiều thuộc tính khóa” nghĩa là gì?   * khóa chính là một trường hoặc kết hợp nhiều trường và được xác định là một bản ghi duy nhất. Không trường nào trong khóa chính được chứa giá trị NULL. Một bảng chỉ có duy nhất một khóa chính * Khóa chính chứa một hoặc nhiều thuộc tính khóa nghĩa là với trường hợp bảng được tạo ra từ 2 bảng có mối quan hệ n-n thì khóa chính của bảng mới này còn được gọi là khóa phức hợp, là tổ hợp của 2 khóa chính của 2 bảng kia |
| 8 | Khóa ngoại (foreign key) là gì?  =>  Dùng để thiết lập khóa ngoại trên bảng, dùng liên kết với khóa chính (primary key của bảng khác). Khóa ngoại nghĩa là giá trị trong bảng này phải xuất hiện trong bảng khác. |
| 9 | Phân biệt DELETE và TRUNCATE?  =>  Delete có thể xóa dòng trong bảng hoặc xóa cả 1 bảng, tuy nhiên bản chất vẫn sẽ xóa từng dòng  Truncate chỉ có thể xóa 1 bảng nhưng bản chất là xóa luôn 1 bảng không quan tâm số lượng dòng, như vậy xóa bảng bằng truncate sẽ nhanh hơn  Dữ liệu xóa bởi delete thì có thể restore nhưng truncate thì không |
| 10 | Các loại JOIN? Phân biệt?  =>  Có 2 loại join là inner join (lấy dữ liệu chung của 2 bảng) và outer join   * Outer join có các loại sau:   + Left join: lấy phần chung + phần bảng bên trái  + Right join: lấy phần chung + phần bảng bên phải  + Full join: lấy cả bên trái + chung + bên phải (tuy nhiên my sql không có full join) |
| 11 | Phân biệt JOIN và UNION?   * Union là gộp kết quả 2 hay nhiều câu truy vấn select lại với nhau theo điều kiện số cột truy vấn bằng nhau. Ở mysql cũ thì có them điều kiện cũng kiểu dữ liệu còn ở phiên bản mới nhất của mysql thì không cần điều kiện này. Nếu dùng union (từ left join và right join của 2 bảng ta sẽ được full join) * Join là lấy dữ liệu từ 2 bảng theo 1 điều kiện |
| 12 | Phân biệt UNION và UNION ALL?   * Union all khác union ở chỗ union all sẽ thể hiện toàn bộ nội dung bất chấp trùng lặp trong khi union sẽ bỏ những nội dung trùng lặp |
| 13 | Phân biệt EXIST và IN?   * Bản chất kết quả nhận được là như nhau * **Toán tử IN** * Nó so sánh các giá trị giữa truy vấn con (truy vấn con) và truy vấn mẹ. * Nó quét tất cả các giá trị bên trong khối IN. * 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 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 ít hơn.   Toán tử EXISTS   * Nó được sử dụng để kiểm tra sự EXISTS của dữ liệu trong một truy vấn con. Nói cách khác, nó xác định liệu giá trị có được trả về hay không. * Nó không so sánh các giá trị giữa truy vấn con và truy vấn mẹ. * Nó trả về TRUE hoặc FALSE. Do đó, chúng tôi không thể sử dụng nó để so sánh các giá trị NULL. * Chúng ta chỉ có thể sử dụng nó trên các truy vấn con. * Nó thực thi nhanh hơn khi kết quả truy vấn con lớn. |
| 14 | Phân biệt AND và OR?  =>  - And: nếu tất cả điều kiện đúng sẽ trả về kết quả  - Or: nếu một trong các điều kiện đúng sẽ trả về kết quả |
| 15 | Phân biệt GROUP BY và ORDER BY?   * Group by: dùng để nhóm dữ liệu trên 1 cột, thường kết hợp với các hàm (count, sum, min, max, avg,…) * Order by: Sử dụng để sắp xếp theo 1 cột hoặc nhiều cột   + Mặc định ASC: tăng dần  + DESC: giảm dần |
| 16 | Muốn nối chuỗi trong MySQL thì sử dụng gì?   * Sử dụng hàm concat(String1, string2,…); |
| 17 | Yêu cầu tính điểm trung bình của tất cả các sinh viên thì làm thế nào?   * Select avg(point) as avgP, customerName from student group by customerName |
| 18 | Yêu cầu tính điểm trung bình của từng bạn sinh viên thì làm thế nào? |
| 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?   * Select studentName, avg(studentPoint) avgS from student group by studentName having avgS>=5; |
| 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)?   * Select studentName, className from student join class on student.studentClassId = class. ClassId |
| 21 | Hiển thị danh sách sinh viên theo thứ tự tăng dần của điểm thi?   * Select studentName, studentPoint from student order by studentPoint; |
| 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 \* from student order by studentName DESC, studentAge ASC |
| 23 | Hiển thị các sinh viên có họ là ‘Le’ và tên là ‘Hai’?   * Select \* from student where studentName like ‘Le%’ and studentName like ‘%Hai’ |
| 24 | Hiển thị các sinh viên có tên bắt đầu là ‘C’?   * Select \* from student where studentName like ‘C%’; |
| 25 | Làm sao để có thể tăng tốc độ truy vấn? Nhược điểm là gì?   * Sử dụng index để tang tốc độ truy vấn. Nhược điểm của index là giảm tốc độ insert, update, delete vì sau mỗi lần này hệ thống phải đánh lại chỉ số index. Lưu ý vì lý do trên nên không nên tạo index cho các bảng nhỏ, các bảng thường xuyên update, insert, delete, cột có nhiều giá trị null, chỉ những cột hay tìm kiếm thì mới tạo Index |
| 26 | Các dạng tham số ở SP?   * Có 3 kiểu tham số in (truyền tham trị), out (truyền tham chiếu) và inout (kết hợp in và out) |
| 27 | Phân biệt SP, Trigger, Function?   |  |  |  | | --- | --- | --- | | SP | Function | Trigger | | Không trả về giá trị | Trả về giá trị | Không trả về giá trị | | Có hoặc không có tham số | Có hoặc không có tham số | Không có tham số | | Có 3 kiểu tham số  +In  +Out  + InOut | Có 1 kiểu tham số mặc định là In | Không có tham số | | Gọi SP dùng từ “Call Spname”. Gọi thủ công | Gọi function thì chỉ cần ten function. Gọi thủ công | Gọi tự động bởi hệ thống | | Mục đích tận dụng soure code thực thi | Mục đích tận dụng 1 function dùng để truy vấn tượng tự các function thong dụng | Mục đích để theo dõi sự thay đổi của 1 bảng (insert, update, delete). Mỗi 1 chức năng muốn theo dõi phải làm 1 trigger | |  |  | Theo sau trigger event là trigger time và câu lệnh muốn trigger thực hiện | |
| 28 | CSS là gì? Các cách nhúng CSS vào tài liệu html? Độ ưu tiên?   * CSS viết tắt là cascading style sheets. * Có 3 cách nhúng CSS:   + Inline: Sử dụng thuộc tính style của thẻ HTML viết css  + Internal: sử dụng cặp thẻ <style> </style> để định nghĩa CSS (nên viết trong thẻ head)  +External: tạo 1 file CSS và sử dụng thẻ link để nhúng vào html   * Độ ưu tiên: Inline đầu tiên -> internal hoặc external tùy thuộc vào cái nào nằm gần thẻ body sẽ ưu tiên |
| 29 | Phân biệt giữa bộ chọn class và bộ chọn id?   * Bộ chọn class sử dụng .class name để định nghĩa CSS * Bộ chọn ID sử dụng #idname để định nghĩa CSS |
| 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  + **padding**: khoảng cách từ content => border  + **border**: độ dày đường viền  + **margin**: khoảng cách từ border của thẻ đến border của thẻ 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:  Thuộc tính clear xác định 2 bên của phần tử (left, right), nơi mà phần tử [float](https://hocwebchuan.com/reference/cssSection/pr_float.php) không được cho phép (ngăn cản thành phần không được float trái, phải hay cả hai).  Thuộc tính Clear có những giá trị sau tương tự như Float CSS:  Left/Right: tràn phần tử về bên trái/phải.  None: Cho phép tràn lên cả hai phía của phần tử.  Both: Không cho phép tràn về phía nào.  Inherit: Kế thừa từ phẩn tử cha. |
| 32 | Thuộc tính position dùng để làm gì? Phân biệt các giá trị: static, absolute, relative và fixed?  =>  - **Position** là thuộc tính của CSS để xác định cách thức hiện thị vị trí của thẻ HTML  **+ Static:** Mặc định**.**Không bị ảnh hưởng bởi thuộc tính top, left, right, bottom  **+ Relative:**(tương đối) lấy chính nó làm gốc. Bị ảnh hưởng bởi thuộc tính top, left, right, bottom  **+ Absolute: (tuyệt đối)** Định vị theo thành phần bao ngoài (thẻ cha), hoặc ít nhất là theo cửa sổ trình duyệt. **Fixed:**Định vị trí tương đối cho thành phần theo cửa sổ trình duyệt.  **+ Fixed:**Định vị và giúp cho phần tử luôn cố định một chỗ, ví dụ như khi bạn scroll trình duyệt chẳng hạn, phần tử sẽ không thay đổi.  **+ Inherit:**Xác định thừa hưởng thuộc tính từ thành phần cha (thành phần bao ngoài). |
| 33 | Responsive Web Design là gì? |
| 34 | Làm thế nào để tạo 1 trang web responsive? |
| 35 | Bootstrap là gì? |
| 36 | Một số thành phần thông dụng ở Bootstrap? |
| 37 | Phân biệt Static web và Dynamic web? |
| 38 | Phân biệt 2 phương thức GET/POST |
| 39 | Phân biệt request/response |
| 40 | Phân biệt forward/redirect |
| 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 ? |
| 42 | JSTL là gì? Giới thiệu một số thẻ bạn đã sử dụng ? |
| 43 | Trình bày kiến trúc MVC? Vai trò của các thành phần trong kiến trúc MVC? |
| 44 | Sự khác nhau giữa MVC1 và MVC2 |
| 45 | Mô tả cách thực hiện tạo và lấy kết quả 1 câu query bất kỳ bằng JDBC? |
| 46 | Phân biệt các loại Statement ? |
| 47 | Kỹ thuật sử dụng transaction bằng JDBC là như thế nào? |
| 48 | Các tính chất của Transaction? |
| 49 | Phương pháp gọi 1 SP ở JDBC? |
| 50 | Phân biệt Ant, Maven, Gradle? |