**Bài tập lý thuyết về SQL:**

1. Có mấy loại khoá của bảng? Định nghĩa và ý nghĩa của từng loại? Một bảng có thể có nhiều loại khoá không? Một bảng có thể có nhiều khoá cùng 1 loại hay không? Các giá trị của khoá có thể là NULL hay không?

* Có 3 loại khóa của bảng
* PRIMARY KEY: Khoá chính dùng để xác định, phân biệt các dòng dữ liệu khác nhau trong table
* FOREIGN KEY: Ràng buộc dữ liệu giữa hai tables khác nhau. Đảm bảo dữ liệu thuộc table này phải tương ứng với dữ liệu trong một table khác
* UNIQUE KEY: chỉ ràng buộc duy nhất, các giá trị của column chỉ trong mệnh đề UNIQUE trong các row của table phải có giá trị khác biệt. Giá trị null là cho phép nêu UNIQUE dựa trên một cột.
* Một bảng có thể có nhiều loại khóa
* Một bảng chỉ có 1 khóa chính và có thể có nhiều khóa ngoại
* Các giá trị của
* PRIMARY KEY không thể null, giá trị nhân bản
* FOREIGN KEY ó thể chấp nhận nhiều null, giá trị nhân bản
* UNIQUE có thể chấp nhận chỉ có một giá trị **null** và nó không thể có giá trị nhân bản.

1. Có mấy loại Join, ý nghĩa của từng loại, lấy ví dụ minh hoạ.

* Có 7 loại join

**CROSS JOIN**: Là lệnh join tham gia chéo, là hình thức tham gia đơn giản nhất nó khớp với từng hàng từ một bảng cơ sở dữ liệu với tất cả các hàng khác. Hay nói cách khác, nó cung cấp cho chúng ta sự kết hợp của từng hàng của bảng đầu tiên với tất cả các bản ghi trong bảng thứ 2

Select e.ename, d.dname

From emp e cross join dept d;

**NATURAL JOIN**

Select e.ename, d.dname

From emp e natural join dept d;

**USING**: cũng có thể được sử dụng cho cùng 1 mục đích. Sự khác biệt là nó cần phải có tên giống hệt nhau giữa các cột trong cả hai bảng.

Select e.ename, d.dname

From emp e join dept d using (deptno);

**ON**: dùng để khớp các bản ghi giữa bảng

Select e.ename, d.dname

From emp e join dept d on (e.deptno = d.deptno);

**INNER JOIN**: được sử dụng để kết hợp các hàng từ cả hai bảng thỏa mãn điều kiện đã cho.

Select e.ename, d.dname

From emp e inner join dept d on (e.deptno = d.deptno);

**LEFT OUTER JOIN**: cho kết quả tất cả các hàng từ bảng bên trái ngay cả khi không tìm thấy hàng phù hợp trong bảng bên phải. Trường hợp không tìm thấy kết quả phù hợp trong bảng bên phải, kết quả sẽ là null.

Select e.ename, d.dname

From emp e left outer join dept d on (e.deptno = d.deptno);

**RIGHT OUTER JOIN**: cho kết quả tất cả các hàng từ bảng bên phải ngay cả khi không tìm thấy hàng phù hợp trong bảng bên trái. Trường hợp không tìm thấy kết quả phù hợp trong bảng bên trái, kết quả sẽ là null.

Select e.ename, d.dname

From emp e right outer join dept d on (e.deptno = d.deptno);

**FULL OUTER JOIN**: trả về tất cả các dòng đúng với 1 trong các bảng, là sự kết hợp của left join và right join

Select e.ename, d.dname

From emp e full outer join dept d on (e.deptno = d.deptno);

1. Tìm hiểu về từ khoá UNION trong truy vấn dữ liệu? Có mấy loại UNION, cách sử dụng và ví dụ minh hoạ cho từng loại.

* UNION: Kết hợp kết quả của nhiều câu hỏi với nhau, chỉ giữ lại một đại diện cho các mẫu tin trùng nhau.
* Mỗi lệnh SELECT trong toán tử UNION phải có cùng số cột trong bộ kết quả với kiểu dữ liệu tương ứng.
* Có 2 loại UNION
* UNION: được dùng để kết hợp 2 bộ kết quả từ 2 hoặc nhiều lệnh SELECT. Nó sẽ xóa các hàng trùng trong các lệnh SELECT này.

Ví dụ:

Select job from emp where deptno =10

Union

Select job from emp where deptno = 30

* UNION ALL: Toán tử UNION chỉ chọn các giá trị riêng biệt theo mặc định. Để cho phép các giá trị trùng lặp, hãy sử dụng UNION ALL

Ví dụ: Select job from emp where deptno =10

Union

Select job from emp where deptno = 30

1. Tìm hiểu về Nested Query, Correlated Query. Phân biệt, lấy ví dụ minh hoạ.

* Nested Query là truy vấn trong 1 truy vấn khác và được nhúng trong mệnh đề where

Chúng ta có thể có một subquery mà gọi subquery khác

Ví dụ:

select ename**,** job**,** deptno**,** sal from emp

where sal in **(**select max**(**sal**)** from emp group by deptno**)**

order by deptno

* Correlated Query là 1 truy vấn con(một truy vấn trong 1 truy vấn khác) sử dụng kết quả của truy vấn cha.

Khi một subquery lấy các tham số từ parent query

Ví dụ:

select ename**,** to\_char**(**hiredate**,**'dd-MM-yyyy'**)** hiredate**,** deptno from emp

where hiredate in**(**select min**(**hiredate**)** from emp group by deptno**)**

1. Tìm hiểu về Index? Phân biệt global index và local index? Khi viết câu lệnh truy vấn, muốn các index được sử dụng phải chú ý gì? Lấy ví dụ minh hoạ.

|  |  |
| --- | --- |
| Global Index | Local Index |
| * Linh hoạt ở mức độ phân vùng và partition key độc lập với phương pháp partition của bảng. * Điểm cao nhất là điểm giới hạn của partition, tất cả đều có giá trị MAXVALUE * Không thể drop partition cao nhất trong global index. * Các lần thêm mới index chỉ xảy ra ở cạnh bên phải của một index, được trải đều trên N partition đã hash cho một blobal hash partition index * Index vẫn avaiable và online trong suốt quá trình hoạt động. Do đó không có ứng dụng nào khác bị ảnh hưởng bởi thao tác này * Index không phải rebuild sau các hoạt động trên * Việc duy trì global index cho drop và truncate được thực hiện dưới dạng metadata | * Dễ quản lý hơn global index * Cung cấp tính khả dụng cao hơn và phổ biến trong môi trường DSS * Hỗ trợ tính khả dụng hơn khi có các hoạt động bảo trì partion hay subpartition. * Khi thêm mới, drop, merge, split, hash partition hay subpartition được thêm mới thì index partition sẽ được tự động tạo theo. * Local index có thể là duy nhất |

* Khi viết câu lệnh truy vấn, muốn các index được sử dụng phải chú ý
* Nếu có quá nhiều index sẽ gây ảnh hưởng đến số liệu khi mà dữ liệu trong table bị thay đổi thứ tự theo index

Ví dụ: Thêm một row vào bảng tất cả các Index sẽ được update

* Nên chọn lựa giữa yêu cầu query, và insert, update để có một index hợp lý. Đối với các khóa primary key và unique key từ khóa unique được tự động thêm khi tạo index.

Ví dụ: create index -ename on emp(ename);

Xóa index bằng lệnh: drop index index\_name

1. Tìm hiểu về partition của bảng, nên tạo partition cho bảng trong trường hợp nào? Lấy ví dụ minh hoạ?

* Oracle Partition cung cấp 3 phương thức chia phân vùng cơ bản để kiểm soát việc phân chia dữ liệu
* Các câu truy vấn SQL và câu lệnh DML không cần phải chỉnh sửa để có thể truy xuất phân vùng.
* Giúp tăng hiệu suất, khả năng quản lý và tính sẵn sàng của nhiều ứng dụng

+ Khóa phân vùng:

* Mỗi hàng thuộc bảng đã được phân vùng chỉ thuộc về một phân vùng duy nhất.
* Là tập hợp một hoặc nhiều cột xác định phân vùng nơi từng hàng được lưu trữ.
* Trực tiếp xử lý luồng của các câu lệnh insert, update, delete vào các phân vùng thích hợp thông qua khóa phân vùng.

+ Một khóa phân vùng

* Bao gồm danh sách có thứ tự từ 1 đến 16 cột
* Không thể chứa level, rowid, mslabel pseudocolumn hoặc một cột kiểu rowid
* Không thể chứa cột null

+ Phân vùng bảng

* Tất cả các bảng đều có thể được phân vùng trừ các bảng có chứa các cột kiểu LONG hoặc RAW LONG.
* Có thể sử dụng các bảng có kiểu CLOB, BLOB
* Nên tạo partition cho bảng trong trường hợp

+ Phân vùng bảng

* Bảng lớn hơn 2GB
* Bảng chứa dữ liệu lịch sử, những dữ liệu mới sẽ được thêm vào phân vùng mới.
* Khi nội dung của bảng cần được phân phối trên nhiều loại thiết bị storage khác nhau.

Ví dụ

Ví dụ tạo một bảng PROJECT\_HANH được phân vùng phạm vi theo trường p\_start\_date:

create table PROJECT\_HANH

**(**

PROJID number**(**4**,**0**)** not null constraint PK\_PROJECT\_HANH PRIMARY KEY**,**

P\_DESC varchar2**(**20**),**

P\_START\_DATE date**,**

P\_END\_DATE date**,**

BUDGET\_AMOUNT number**(**7**,**2**),**

MAX\_NO\_STAFF number**(**2**,**0**),**

check**(**P\_END\_DATE **>** P\_START\_DATE**)**

**)**

PARTITION p\_start\_date VALUES LESS THAN(TO\_DATE('09/03/2022,'DD/MM/YYYY')),

PARTITION p\_start\_date VALUES LESS THAN(TO\_DATE('08/04/2022','DD/MM/YYYY')),

PARTITION p\_start\_date VALUES LESS THAN(TO\_DATE('04/01/2022','DD/MM/YYYY')));

+ Phân vùng index

* Tránh việc phải rebuilding toàn bộ dữ liệu index khi dữ liệu bị xóa bỏ
* Thực hiện bảo trì trên một phần dữ liệu mà không gây ảnh hưởng đến toàn bộ index(invalid)
* Giảm ảnh hưởng của lệnh gây ra trên cột giá trị tăng đơn điệu

1. Phân biệt câu lệnh delete và truncate? Cách sử dụng?s

|  |  |
| --- | --- |
| DELETE | TRUNCATE |
| -Xóa cả bảng  -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.  -Dữ liệu bị xóa bởi DELETE có thể restored được còn TRUNCATE thì không.  -Cách sử dụng  Delete from ten\_bang  Where [dieu\_kien] | -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 cua id sẽ bắt đầu từ 1, đây cũng chính là khác biệt lớn nhất của DELETE và TRUNCATE.  -TRUNCATE không thể chạy được khi bảng bạn định xóa có foreign\_key (nếu bảng đó có foreign\_key trỏ đến chính nó thì bạn có thể xóa bình thường)  -Cách sử dụng  TRUNCATE TABLE table\_name [REUSE STORAGE]  Trong đó: REUSE STORAGE giữ lại khung để chứa, chỉ xóa dữ liệu |