**Thực Tập Sinh: Nguyễn Quang Dương**

**Đề bài**

**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?
2. Có mấy loại Join, ý nghĩa của từng loại, lấy ví dụ minh hoạ.
3. 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.
4. Tìm hiểu về Nested Query, Correlated Query. Phân biệt, lấy ví dụ minh hoạ.
5. 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.
6. Tìm hiểu về partition của bảng, nên tạo partition cho bảng trong trường hợp nào? Lấy ví dụ minh hoạ?
7. Phân biệt câu lệnh delete và truncate? Cách sử dụng?

**Lời giải**

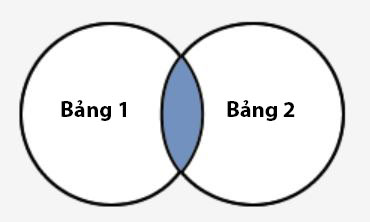
1. Trong SQL thì gồm 2 khóa là Khóa chính (Primary Key) và Khóa ngoại (Foreign Key). Khóa chính dùng để định danh một dòng dữ liệu duy nhất trong một bảng. Khóa chính chứa các giá trị duy nhất hay cơ sở dữ liệu duy nhất được lưu thông xuyên suốt ở nhiều bảng với nhau. Giá trị của khóa chính có thể dung để tham chiếu tới toàn bộ các giá trị khác trên bản ghi.

Khóa ngoại được coi là con trỏ tới khóa chính của các bảng khác. Khóa ngoại tạo ra một mối liên kết giữa 2 bảng, nếu người dung vô tình hoặc cố ý xóa các liên kết này thì sẽ bị ngăn chặn, đồng thời người dùng không thể nhập vào khóa ngoại một giá trị mà khóa chính (được khóa ngoại trỏ tới) không tồn tại giá trị đó

Một bảng có thể có nhiều loại khóa, một bảng chỉ có 1 khóa chính tuy nhiên một khóa chính có thể có nhiều cột, 1 bảng có thể có nhiều khóa phụ, giá trị của khóa chính bắt buộc không được null còn khóa ngoại thì ngược lại, nó có thể nhận giá trị null

1. Có 4 loại join cơ bản trong sql bao gồm Inner Join, Left outer join, Right outer join và Full outer join

* **INNER JOIN**: Dùng để lấy phần chung giữa 2 bảng



Kết quả trả về của inner join

Ví dụ có 2 bảng như sau:

SinhVien

|  |  |
| --- | --- |
| MaSv | Name |
| 001 | A |
| 002 | B |
| 003 | C |
| 004 | D |

HocPhi

|  |  |  |
| --- | --- | --- |
| MaHoaDon | MaSv | NgayNop |
| Hd1 | 001 | 06/08/2022 |
| Hd2 | 003 | 05/08/2022 |
| Hd3 | 007 | 02/08/2022 |

Chạy câu lệnh

SELECT SinhVien.MaSv, SinhVien.Name, HocPhi.NgayNop

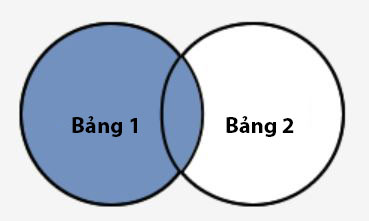
FROM SinhVien INNER JOIN HocPhi ON SinhVien.MaSv = HocPhi.MaSv;

Kết quả trả về

|  |  |  |
| --- | --- | --- |
| MaSv | Name | NgayNop |
| 001 | A | 06/08/2022 |
| 003 | C | 05/08/2022 |

Kết quả trả về là sinh viên đã thanh toán học phí và có trong bảng SinhVien

* **LEFT OUTER JOIN:** lấy dữ liệu thuộc bảng bên trái và phần chung giữa 2 bảng



Kết quả trả về của left outer join

Ví dụ cho 2 bảng như sau:

SinhVien

|  |  |
| --- | --- |
| MaSv | Name |
| 001 | A |
| 002 | B |
| 003 | C |
| 004 | D |

HocPhi

|  |  |  |
| --- | --- | --- |
| MaHoaDon | MaSv | NgayNop |
| Hd1 | 001 | 06/08/2022 |
| Hd2 | 003 | 05/08/2022 |
| Hd3 | 007 | 02/08/2022 |

Chạy câu lệnh

SELECT SinhVien.MaSv, SinhVien.Name, HocPhi.NgayNop

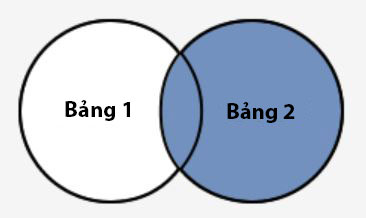
FROM SinhVien LEFT OUTER JOIN HocPhi ON SinhVien.MaSv = HocPhi.MaSv;

Kết quả trả về như sau

|  |  |  |
| --- | --- | --- |
| MaSv | Name | NgayNop |
| 001 | A | 06/08/2022 |
| 002 | B | Null |
| 003 | C | 05/08/2022 |
| 004 | D | Null |

Kết quả trả về sẽ vẫn là bảng SinhVien do dùng LEFT OUTER JOIN và những sinh viên trong bảng sinh viên nếu chưa có tên trong bảng HocPhi thì phần NgayNop sẽ để giá trị Null.

* **RIGHT OUTER JOIN**: Ngược lại với LEFT OUTER JOIN nó sẽ lấy dữ liệu của bảng bên phải và phần dữ liệu chung giữa 2 bảng



Kết quả trả về của right outer join

Ví dụ cho 2 bảng như sau

SinhVien

|  |  |
| --- | --- |
| MaSv | Name |
| 001 | A |
| 002 | B |
| 003 | C |
| 004 | D |

HocPhi

|  |  |  |
| --- | --- | --- |
| MaHoaDon | MaSv | NgayNop |
| Hd1 | 001 | 06/08/2022 |
| Hd2 | 003 | 05/08/2022 |
| Hd3 | 007 | 02/08/2022 |

Chạy câu lệnh

SELECT HocPhi.MaHoaDon, HocPhi.NgayNop, SinhVien.Name

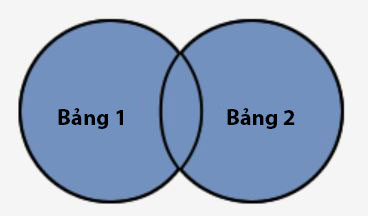
FROM SinhVien RIGHT OUTER JOIN HocPhi ON SinhVien.MaSv = HocPhi.MaSv;

Kết quả trả về như sau

|  |  |  |
| --- | --- | --- |
| MaHoaDon | NgayNop | Name |
| Hd1 | 06/08/2022 | A |
| Hd2 | 05/08/2022 | C |
| Hd3 | 02/08/2022 | Null |

Kết quả trả về gồm toàn bộ sinh viên đã nộp học phí, sinh viên nào đã nộp học phí nhưng không có tên trong bảng SinhVien sẽ có giá trị Null

* **FULL OUTER JOIN:** Sẽ lấy dữ liệu của cả 2 bảng



Kết quả trả về của full outer join

Ví dụ cho 2 bảng như sau:

SinhVien

|  |  |
| --- | --- |
| MaSv | Name |
| 001 | A |
| 002 | B |
| 003 | C |
| 004 | D |

HocPhi

|  |  |  |
| --- | --- | --- |
| MaHoaDon | MaSv | NgayNop |
| Hd1 | 001 | 06/08/2022 |
| Hd2 | 003 | 05/08/2022 |
| Hd3 | 007 | 02/08/2022 |

Chạy câu lệnh như sau:

SELECT SinhVien.MaSv, SinhVien.Name, HocPhi.NgayNop

FROM SinhVien FULL OUTER JOIN HocPhi ON SinhVien.MaSv = HocPhi.MaSv;

Kết quả trả về như sau

|  |  |  |
| --- | --- | --- |
| MaSv | Name | NgayNop |
| 001 | A | 06/08/2022 |
| 002 | B | Null |
| 003 | C | 05/08/2022 |
| 004 | D | Null |
| 007 | Null | 02/08/2022 |

Kết quả trả về gồm toàn bộ sinh viên và ngày nộp, sinh viên nào chưa nộp thì NgayNop sẽ nhận giá trị Null, sinh viên nào đã nộp nhưng chưa có tên trong SinhVien thì Name sẽ nhận giá Null

1. Có 2 loại UNION trong sql là UINION thường và UNION ALL

* Mệnh đề/Toán tử UNION được sử dụng để kết hợp kết quả của 2 hoặc nhiều lệnh SELECT mà không trả về bất cứ bản sao nào của bản ghi.

Để sử dụng UNION, mỗi lệnh SELECT phải có cùng số cột đã chọn, cùng số biểu thức của các cột, cùng kiểu dữ liệu và chúng có cùng thứ tự nhưng chúng không cần cùng độ dài.

Ví dụ: Cho 2 bảng:

SinhVien

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Ten | Tuoi | KhoaHoc | HocPhi |
| 1 | Hoang | 20 | CNTT | 4000000 |
| 2 | Viet | 18 | TKĐH | 5000000 |
| 3 | Thanh | 19 | MAR | 5000000 |
| 4 | Nhan | 19 | CNTT | 6000000 |
| 5 | Huong | 21 | MAR | 4000000 |

MuonSach

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Ngay | SinhVienID | SoTien |
| 100 | 05-08-2022 | 3 | 100 |
| 101 | 07-08-2022 | 3 | 100 |
| 102 | 07-08-2022 | 2 | 100 |
| 103 | 08-08-2022 | 4 | 100 |

Kết hợp 2 select bằng union như sau:

SELECT ID, Ten, SoTien, Ngay

FROM SinhVien

LEFT JOIN MuonSach

ON SinhVien.ID = MuonSach.SinhVienID

UNION

SELECT ID, Ten, SoTien, Ngay

FROM SinhVien

RIGHT JOIN MuonSach

ON SinhVienID = MuonSach.SinhVienID;

Kết quả trả về sẽ như sau:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Ten | SoTien | Ngay |
| 1 | Hoang | Null | Null |
| 2 | Viet | 100 | 07-08-2022 |
| 3 | Thanh | 100 | 05-08-2022 |
| 3 | Thanh | 100 | 07-08-2022 |
| 4 | Nhanh | 100 | 08-08-2022 |
| 5 | Huong | Null | Null |

* Toán tử UNION ALL được sử dụng để kết hợp nhiều kết quả của 2 lệnh SELECT bao gồm bản sao của tất cả các bảng

Các quy tắc áp dụng cho UNION cũng áp dụng cho UNION ALL

Ví dụ: Vẫn là 2 bảng SinhVien và MuonSach như trên ta chạy UNION ALL như sau:

SELECT ID, Ten, SoTien, Ngay

FROM SinhVien

LEFT JOIN MuonSach

ON SinhVien.ID = MuonSach.SinhVienID

UNION ALL

SELECT ID, Ten, SoTien, Ngay

FROM SinhVien

RIGHT JOIN MuonSach

ON SinhVienID = MuonSach.SinhVienID;

Kết quả trả về sẽ như sau

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Ten | SoTien | Ngay |
| 1 | Hoang | Null | Null |
| 2 | Viet | 100 | 07-08-2022 |
| 3 | Thanh | 100 | 05-08-2022 |
| 3 | Thanh | 100 | 07-08-2022 |
| 4 | Nhanh | 100 | 08-08-2022 |
| 5 | Huong | Null | Null |
| 2 | Viet | 100 | 07-08-2022 |
| 3 | Thanh | 100 | 05-08-2022 |
| 3 | Thanh | 100 | 07-08-2022 |
| 4 | Nhanh | 100 | 08-08-2022 |

1. Nested Query: (Truy vấn lồng nhau) Truy vấn bên trong chạy trước và chỉ một lần, truy vấn bên ngoài được thực hiện dựa trên kết quả của truy vấn bên trong hay nói cách khác truy vấn bên trong được sử dụng để thực hiện truy vấn bên ngoài.

Correlated Query: (Truy vấn tương quan) Truy vấn bên ngoài chạy trước và đối với mỗi một hàng ở truy vấn bên ngoài thì truy vấn bên trong được thực thi

Ví dụ: Cho 2 bảng sau

SinhVien

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Ten | Tuoi | KhoaHoc | HocPhi |
| 1 | Hoang | 20 | CNTT | 4000000 |
| 2 | Viet | 18 | TKĐH | 5000000 |
| 3 | Thanh | 19 | MAR | 5000000 |
| 4 | Nhan | 19 | CNTT | 6000000 |
| 5 | Huong | 21 | MAR | 4000000 |

MuonSach

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Ngay | SinhVienID | SoTien |
| 100 | 05-08-2022 | 3 | 100 |
| 101 | 07-08-2022 | 3 | 100 |
| 102 | 07-08-2022 | 2 | 100 |
| 103 | 08-08-2022 | 4 | 100 |

Nested Query:

SELECT \* FROM SinhVien

WHERE ID in (SELECT SinhVienID FROM MuonSach);

Correlated Query:

SELECT \* FROM SinhVien

WHERE EXISTS (SELECT SinhVienID FROM MuonSach

WHERE MuonSach.SinhVienID = SinhVien.ID);

2 câu lệnh này có giá trị trả về như nhau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Ten | Tuoi | KhoaHoc | HocPhi |
| 2 | Viet | 18 | TKĐH | 5000000 |
| 3 | Thanh | 19 | MAR | 5000000 |
| 4 | Nhan | 19 | CNTT | 6000000 |

1. Chỉ mục (Index) là bảng tra cứu đặc biệt mà Database Search Engine có thể sử dụng để tăng nhanh thời gian và hiệu suất thu thập dữ liệu. Hiểu đơn giản, một chỉ mục là một con trỏ tới dữ liệu trong một bảng. Một chỉ mục trong một Database là tương tự như một chỉ mục trong Mục lục của cuốn sách.

* Global Index là các index có mối quan hệ 1-Nhiều giữa các bảng được phân vùng. Chúng ta có thể áp dụng phân vùng cho global table hoặc các table đã được phân vùng trong đó phân vùng sẽ là global hoặc local, tạo global index sẽ lâu hơn tạo local index, global index được tạo cho mọi phân vùng của bảng đã được phân vùng, global index khó quản lý hơn local index.
* Local Index là các index được tạo có mối quan hệ 1-1 giữa các bảng được phân vùng và các phân vùng. Cách sử dụng local index sử dụng phương pháp Chia để trị trong đó các index được áp dụng cho các local partition của bảng. Các local index được tạo ra khi bạn tạo local partition và khởi tạo local index nhanh hơn tạo global index. Key của local partition và value của index giống hệt nhau trong trường hợp của Local index. Bạn chỉ có thể tạo index cho phân vùng cụ thể chứ không phải toàn bộ table.
* Khi viết lệnh truy vấn cần chú ý nếu dùng function đối với các trường hoặc các bảng đã được đánh index thì index sẽ bị mất tác dụng, ảnh hưởng rất lớn đến database nên khi viết cần hết sức lưu ý. Ví dụ:

Select to\_char(HIREDATE,’yyyy-dd-mm’) from EMP;

Nếu trường HIREDATE được đánh index thì index này lập tức sẽ mất tác dụng là database chứa trường này có thể bị lỗi.

1. Partition là việc phân chia một table thành những phần nhỏ theo một logic nhất định, được phân biệt bằng key, key này thường là tên column trong table

Cú pháp :

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl\_name

(create\_definition,...)

[table\_options]

[partition\_options]

Nên dùng partition khi truy vấn DB ở các bảng dữ liệu lớn, thay vì phải duyệt qua toàn bộ dữ liệu trong bảng ta sẽ chỉ cần duyệt dữ liệu ở các vùng partition nhất định

Ví dụ:

Cho một table chưa được tạo partition:

Table

Description automatically generated

Shape, arrow

Description automatically generated

Khi bảng này được tạo ra nó sẽ mặc định được lưu trữ thành một chunk trong file system

Khi sử dụng partition, table sẽ được chia nhỏ ra thành nhiều chunk với key mà chúng ta đã định nghĩa. Ví dụ chia theo tuổi, ta lấy trường age làm key

Diagram

Description automatically generated

Test độ hiệu quả của partition bằng cách đơn giản sau:

Chạy câu truy vấn: select \* from persons where age = 24

Trong trường hợp chưa dùng partition thời gian thực hiện truy vấn là 0.00064 sec.



Trong trường hợp đã chia partition thời gian thực hiện truy vấn là 0.0016 sec.



Do DB chưa quá lớn nên khó có thể cảm nhận được thay đổi, tuy nhiên với những dữ liệu chục triệu thậm trí trăm triệu bản ghi thì đây thực sự là một giải pháp tuyệt vời.

1. Câu lệnh DELTE

Lệnh này sẽ giúp chúng ta xóa bỏ các bản ghi – Record khỏi cơ sở dữ liệu theo hàng – Row. Khi 1 bản ghi như vậy được xóa bằng DELETE, toàn bộ thành phần bên trong đều được ghi lại trong phần Transaction Log, các phần ràng buộc được kiểm tra, và bất cứ thành phần nào thừa sẽ bị xóa hoàn toàn. Bên cạnh đó, những bản ghi nếu vô tình bị xóa bằng lệnh DELETE này thì vẫn có thể được phục hồi được. Trong khi DELETE thường được dùng để xóa dữ liệu bản ghi khỏi bảng nhiều hơn khi áp dụng trong một phần dữ liệu nào đó, bên cạnh đó, câu lệnh này thường được dùng đối với những bảng dữ liệu đơn, hoặc trong trường hợp muốn xóa dữ liệu khỏi 1 bảng nào đó khi cần kết hợp với nhiều bảng có quan hệ khác. Mặt khác, việc phân quyền chức năng DELETE trên bảng chỉ được áp dụng trên những tài khoản người dùng khác nhau mà không cần phải gán ownership trên tài khoản đó. Và lệnh DELETE không thay đổi việc tự động tăng hoặc giảm số lượng cột – Column trong bảng chứa dữ liệu đó.

Câu lệnh TRUNCATE

Lệnh TRUNCATE này sẽ thực hiện việc phân bổ lại trang chứa dữ liệu trong bảng, và toàn bộ phần dữ liệu đó sẽ bị xóa bỏ hoàn toàn sau khi thực hiện. Nếu xét về khía cạnh tốc độ thì TRUNCATE nhanh hơn DELETE khá nhiều dựa vào lượng thông tin được lưu lại trong phần Transaction Log, và cũng vì vậy nếu những bản ghi nào đó vô tình bị xóa bỏ thì sẽ không thể khôi phục lại được. Còn về bản chất, TRUNCATE là toán tử Data Definition Language – DDL, cũng có nghĩa rằng chúng ta cần tối thiểu ALTER TABLE hoặc phân quyền cao hơn để thực hiện. Nhưng mức phân quyền TRUNCATE TABLE lại không tồn tại. Nếu 1 bảng dữ liệu sau khi thực hiện TRUNCATE có tính năng tăng số lượng cột thì sẽ tự động được sắp xếp lại theo phần định nghĩa ban đầu. Bên cạnh đó, còn có một số hạn chế với cú pháp TRUNCATE, và không thể được áp dụng trên những bảng cụ thể như tình huống dưới đây:

- Khi bảng dữ liệu được tham chiếu bởi Index View.

- Các khóa liên kết ngoài được rút ngắn.

- Bảng dữ liệu đó được dùng để sao chép.

- Bảng thuộc về cơ sở dữ liệu đang được log.

Tùy vào từng tình huống cụ thể mà các bạn hãy sử dụng lệnh DELETE hoặc TRUNCATE sao cho phù hợp. Trong đó, cú pháp DELETE được sử dụng khá rộng rãi và phổ biến vì cho phép người dùng xác định rõ ràng bản ghi nào cần xóa, được kết hợp cùng với lệnh JOINS và một vài tham số khác. Khả năng ngăn chặn có thể xảy ra khi chúng ta dùng DELETE để xóa bỏ 1 lượng dữ liệu khá lớn, do vậy người dùng cần tuyệt đối cẩn thận khi thao tác. Còn lại, lệnh TRUNCATE sẽ giúp người quản trị xóa toàn bộ 1 bảng dữ liệu nào đó một cách đơn giản và nhanh chóng.