**TRƯỜNG ĐẠI HỌC SÀI GÒN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**Tiểu luận học phần: Cơ sở dữ liệu nâng cao**

NGÔN NGỮ ĐẠI SỐ QUAN HỆ NÂNG CAO

**Giảng viên hướng dẫn: Nguyễn Hòa**

**Sinh viên thực hiện:**

**NguyễnVăn Cường\_3123411045**

**Đặng Thành Sơn\_ 3123411258**

**Nguyễn Sĩ Huy\_ 3123411122**

**Trương Văn Tuấn\_3123411328**

Thành phố Hồ Chí Minh, tháng 9 năm 2025

# LỜI CAM ĐOAN

Chúng tôi là nhóm 2 , sinh viên lớp DCT123C6, xin cam đoan: Tiểu luận “Ngôn ngữ đại số quan hệ nâng cao” là quá trình nghiên cứu của riêng nhóm tôi dưới sự hướng dẫn của thầy Nguyễn Hòa. Các tài liệu, dữ liệu và thông tin tham khảo trong tiểu luận đều có nguồn gốc rõ ràng và được trích dẫn đầy đủ theo quy định. Chúng tôi không sao chép sử dụng bài làm của người khác một cách trái phép. Nếu vi phạm tôi xin chấp nhận mọi hình thức kỷ luật.

Thành phố Hồ Chí Minh, ngày 29 tháng 9 năm 2025

Nhóm thực hiện

Nguyễn Văn Cường\_3123411045

Đặng Thành Sơn\_ 3123411258

Nguyễn Sĩ Huy\_ 3123411122

Trương Văn Tuấn\_3123411328

# MỤC LỤC

[LỜI CAM ĐOAN i](#_bookmark0)

[MỤC LỤC](#_bookmark1) ii

[DANH MỤC CÁC HÌNH VẼ iii](#_bookmark2)

[DANH MỤC CÁC BẢNG… iv](#_TOC_250001)

[LỜI MỞ ĐẦU 1](#_bookmark3)

[CHƯƠNG 1. TỔNG QUAN VỀ HỆ ĐIỀU HÀNH DI ĐỘNG 3](#_bookmark4)

* 1. [Khái niệm về hệ điều hành di động 3](#_bookmark5)
     1. [Vai trò của hệ điều hành trong điện thoại di động 4](#_bookmark6)
     2. [Lịch sử phát triển của hệ điều hành di động 5](#_bookmark7)
  2. [Thị trường và sự phát triển của các hệ điều hành hiện nay 8](#_bookmark8)
  3. [Các yếu tố ảnh hưởng đến việc lựa chọn hệ điều hành của người dùng: 10](#_bookmark9)
  4. [Tóm tắt chương 1 13](#_bookmark10)

[CHƯƠNG 2. PHÂN TÍCH CÁC HỆ ĐIỀU HÀNH DI ĐỘNG PHỔ BIẾN 14](#_bookmark11)

* 1. [Android 14](#_bookmark12)
     1. [Tổng quan về Android 14](#_bookmark13)
     2. [Ưu điểm và nhược điểm của Android: 16](#_bookmark14)
  2. [IOS 17](#_bookmark15)
     1. [Tổng quan về IOS 17](#_bookmark16)
     2. [Ưu điểm và nhược điểm của iOS 19](#_bookmark17)
  3. [BlackBerryOS 22](#_bookmark18)
     1. [Tổng quan về BlackBerryOS 22](#_bookmark19)
     2. [Ưu điểm và nhược điểm của BlackBerryOS. 24](#_bookmark20)
  4. [Windows Phone 25](#_bookmark21)
     1. [Tổng quan về Windows Phone 25](#_bookmark22)
     2. Ưu và nhược điểm của Windows Phone… 26
  5. [Tóm tắt chương 2 27](#_bookmark23)

[CHƯƠNG 3. SO SÁNH VÀ XU HƯƠNG PHÁT TRIỂN CỦA CÁC HỆ ĐIỀU](#_bookmark24) [HÀNH DI ĐỘNG 29](#_bookmark24)

* 1. [So sánh về khả năng tùy chỉnh, bảo mật và hiệu năng giữa các hệ điều hành 29](#_bookmark25)
     1. [Khả năng tùy chỉnh 29](#_bookmark26)
     2. [Bảo mật 30](#_bookmark27)
     3. [Hiệu năng 31](#_bookmark28)
  2. [Thị phần của các hệ điều hành trong tương lai 33](#_bookmark29)
     1. [Android : thống trị thị trường 33](#_bookmark30)
     2. [iOS: Sự bền bỉ trong phân khúc cao cấp 34](#_TOC_250000)
     3. BlackBerry: Từ Đỉnh Cao đến Sự Chuyển Hướng 35
     4. Window Phone : sự sụp đổ 36
  3. [Xu hướng phát triển của hệ điều hành di động và tác động của chúng đến thị](#_bookmark32)

[trường. 37](#_bookmark32)

* 1. [Tóm tắt chương 3 38](#_bookmark31)

[KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 39](#_bookmark33)

[TÀI LIỆU THAM KHẢO 41](#_bookmark34)

**DANH MỤC CÁC HÌNH VẼ**

# DANH MỤC CÁC BẢNG

# LỜI MỞ ĐẦU

# CHƯƠNG 1. GIỚI THIỆU VÀ CƠ SỞ LÝ THUYẾT

## Bối cảnh và tầm quan trọng của Hệ Quản trị cơ Sở Dữ liệu

Trong các tổ chức hiện đại, dữ liệu đóng vai trò trung tâm trong mọi hoạt động — từ quản lý khách hàng, giao dịch tài chính, quản trị nhân sự cho đến sản xuất và nghiên cứu. Sự gia tăng nhanh chóng về khối lượng và độ phức tạp của dữ liệu khiến các phương pháp lưu trữ truyền thống bằng tệp (file-processing systems) trở nên lỗi thời và khó kiểm soát.

Các hệ thống tệp độc lập thường gặp nhiều hạn chế: dữ liệu bị **trùng lặp và không nhất quán**, khó chia sẻ giữa các ứng dụng, cấu trúc dữ liệu phụ thuộc chặt chẽ vào chương trình, thiếu **cơ chế kiểm soát đồng thời và phục hồi** khi xảy ra lỗi. Ngoài ra, việc duy trì **tính toàn vẹn và bảo mật dữ liệu** cũng rất khó khăn khi không có cơ chế quản lý tập trung.

Để khắc phục những vấn đề này, hệ quản trị cơ sở dữ liệu (DBMS) được phát triển như một lớp phần mềm trung gian giữa người dùng, ứng dụng và dữ liệu. DBMS cung cấp môi trường có cấu trúc giúp lưu trữ, truy xuất và cập nhật dữ liệu một cách hiệu quả, đồng thời đảm bảo các yêu cầu về an toàn, nhất quán và khả năng mở rộng.

Một trong những nguyên lý cốt lõi của DBMS là **sự trừu tượng hóa dữ liệu (data abstraction)**. Hệ thống phân tách dữ liệu thành ba mức mô tả chính:

* **Mức vật lý (physical level)**: mô tả cách dữ liệu được lưu trữ thực tế trong bộ nhớ hoặc trên đĩa.
* **Mức logic (logical level)**: biểu diễn cấu trúc của toàn bộ cơ sở dữ liệu dưới dạng các bảng, thuộc tính và mối quan hệ.
* **Mức nhìn (view level)**: thể hiện dữ liệu dưới góc nhìn của người dùng hoặc ứng dụng cụ thể.

Cấu trúc ba lớp này cho phép **độc lập dữ liệu** – thay đổi ở một mức không ảnh hưởng tới mức khác. Nhờ đó, các ứng dụng có thể phát triển hoặc bảo trì mà không cần chỉnh sửa cách dữ liệu được lưu trữ.

Ngoài ra, DBMS đảm nhận nhiều chức năng quan trọng khác:

* **Quản lý tính toàn vẹn dữ liệu (data integrity)** thông qua các ràng buộc như khóa chính, khóa ngoại và miền giá trị.
* **Quản lý truy cập đồng thời (concurrency control)** để nhiều người dùng có thể thao tác cùng lúc mà không gây xung đột.
* **Đảm bảo tính bền vững (durability)** và **khả năng phục hồi (recovery)** thông qua cơ chế ghi log và khôi phục sau lỗi.
* **Cung cấp bảo mật (security)** bằng các cơ chế phân quyền và xác thực người dùng.
* **Hỗ trợ ngôn ngữ truy vấn cấp cao (SQL)** giúp người dùng tương tác với dữ liệu một cách linh hoạt mà không cần thao tác trực tiếp ở tầng vật lý.

Nhờ các chức năng này, DBMS trở thành nền tảng không thể thiếu trong mọi hệ thống thông tin hiện đại, từ quy mô nhỏ đến các tập đoàn toàn cầu. Nó không chỉ giúp tăng hiệu quả xử lý và giảm lỗi dữ liệu, mà còn cho phép tổ chức khai thác giá trị từ dữ liệu để ra quyết định chiến lược.

**Nguồn :** [1] Abraham Silberschatz, Henry F. Korth, S. Sudarshan, Database System Concepts, 7th Edition, McGraw-Hill Education, 2019.

## Giới thiệu về Mô hình quan hệ và Ngôn ngữ Truy vấn

Thị trường hệ điều hành di động hiện nay chủ yếu bao gồm Android, iOS, BlackBerry OS, và Windows Phon**e,** mỗi hệ điều hành mang đến những đặc điểm và mục tiêu phục vụ khác nhau, tạo nên sự đa dạng cho thị trường di động toàn cầu. Sự cạnh tranh giữa các hệ điều hành này ảnh hưởng mạnh mẽ đến sự phát triển của các thiết bị di động và dịch vụ đi kèm.

## Đại số quan hệ : Vai trò và đặc điểm

Việc lựa chọn hệ điều hành (HĐH) của người dùng bị ảnh hưởng bởi nhiều yếu tố khác nhau, bao gồm tính dễ sử dụng, bảo mật, tính năng hỗ trợ, và khả năng tương thích với các thiết bị khác. Mỗi yếu tố có tác động riêng, tạo ra những khác biệt trong trải nghiệm người dùng và góp phần định hình thị trường hệ điều hành di động.

## Tóm tắt chương 1

điều hành di động hiện nay.

# CHƯƠNG 2. CÁC TOÁN TỬ CƠ BẢN CỦA ĐẠI SỐ QUAN HỆ

## Nhóm toán tử tập hợp

Hệ điều hành android

## Phép hợp

Hệ điều hành Android được phát triển bởi Android Inc., một công ty khởi nghiệp

## Phép hiệu

Hệ điều hành Android đã trở thành một trong những nền tảng di động phổ biến nhất trên thế giới, nhờ vào những ưu điểm vượt trội của nó.

**2.1.3 Phép tích đề-các**

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

## Nhóm các toán tử quan hệ cơ bản

aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa

## Phép chọn

iCloud, tạo ra một hệ sinh thái tích hợp mà người dùng khó có thể từ chối.

## Phép chiếu

aaaaaaaaaaaaaaa

## Phép đổi tên

aaaaaaaaaaaaa

## Tóm tắt chương 2

Chương này chủ yếu tập trung phân tích các hệ điều hành di động phổ biến hiện nay, bao gồm Android, iOS, BlackBerryOS và Windows Phone, qua đó làm nổi bật đặc trưng và ưu nhược điểm của từng nền tảng. Android nổi bật nhờ vào mã nguồn mở, khả năng tùy chỉnh cao, nhưng gặp khó khăn về vấn đề bảo mật. Trong

khi đó iOS ghi điểm nhờ trải nghiệm mượt mà, bảo mật mạnh mẽ, song hạn chế về tùy biến và giá thành. BlackBerryOS giữ vị thế trong bảo mật doanh nghiệp, nhưng dần tụt lại bởi giao diện lỗi thời và hệ sinh thái kém phong phú. Ngược lại Windows Phone gây ấn tượng với giao diện hiện đại nhưng không đủ ứng dụng để cạnh tranh. Chương này không chỉ giúp người đọc hiểu sâu hơn về thị trường hệ điều hành mà còn gợi mở cách lựa chọn tối ưu cho từng nhu cầu.

# CHƯƠNG 3. CÁC TOÁN TỬ DẪN XUẤT NÂNG CAO

## Phép giao (intersection)

### 3.1.1. Định nghĩa và khái niệm

Trong Đại số Quan hệ, phép Giao (INTERSECTION) là một toán tử nhị phân (binary operator) được sử dụng để tìm ra các bộ (tuple) xuất hiện trong cả hai quan hệ. Phép giao thể hiện phần dữ liệu chung giữa hai tập quan hệ, nghĩa là chỉ những bộ có mặt đồng thời trong cả hai mới được đưa vào kết quả.

**Nguồn:** [1] Ramez Elmasri, Shamkant B. Navathe, Fundamentals of Database Systems, 6th Edition, Addison-Wesley, 2011.

Phép giao là một trong các phép toán tập hợp cơ bản, cùng với phép hợp (UNION) và phép hiệu (SET DIFFERENCE). Các phép này có nguồn gốc từ toán học và được mở rộng để áp dụng trong mô hình dữ liệu quan hệ.

Khi áp dụng phép giao, hai quan hệ R và S phải tương thích hợp (union-compatible). Điều này có nghĩa là:

* + - * Hai quan hệ phải có cùng số lượng thuộc tính.
      * Mỗi cặp thuộc tính tương ứng giữa hai quan hệ phải có cùng miền giá trị (domain).

Nếu điều kiện này được thỏa mãn, phép giao giữa hai quan hệ được định nghĩa như sau:

R ∩ S ={t ∣ t ∈ R và t ∈ S}

Kết quả: là một quan hệ mới bao gồm các bộ có mặt trong cả R và S. Theo quy ước, kết quả sẽ giữ nguyên tên thuộc tính của quan hệ thứ nhất R. Nếu cần, người thiết kế có thể dùng phép đổi tên (Rename – ρ) để điều chỉnh lại tên thuộc tính cho phù hợp với ngữ cảnh truy vấn.

### 3.1.2. Biểu diễn bằng các toán tử cơ bản

Phép giao không phải là một toán tử cơ bản trong Đại số Quan hệ vì nó có thể được biểu diễn lại bằng các phép toán khác như phép hợp (UNION) và phép hiệu (SET DIFFERENCE).

Cụ thể, phép giao có thể được viết lại dưới dạng:

R ∩ S = R − ( R – S )

Điều này có nghĩa là phần giao của hai quan hệ chính là tập hợp các bộ thuộc R sau khi đã loại bỏ những bộ không có trong S. Ngoài ra, trong một số cách biểu diễn khác, phép giao cũng có thể được mô tả dưới dạng kết hợp của phép hợp và phép hiệu như sau:

R ∩ S = (( R ∪ S ) − ( R – S )) − ( S – R )

Như vậy, phép giao là một toán tử dẫn xuất (derived operator), vì có thể được định nghĩa dựa trên các phép toán cơ bản. Tuy nhiên, trong thực tế, việc dùng trực tiếp ký hiệu ∩ giúp các biểu thức truy vấn trở nên ngắn gọn và trực quan hơn nhiều.

**Nguồn:** [1] Ramez Elmasri, Shamkant B. Navathe, Fundamentals of Database Systems, 6th Edition, Addison-Wesley, 2011.

### 3.1.3. Ví dụ minh họa

Xét hai quan hệ có cùng cấu trúc thuộc tính sau:

**STUDENT**

|  |  |
| --- | --- |
| **Fname** | **Fname** |
| Susan | Yao |
| Ramesh | Shah |
| Johnny | Kohler |
| Barbara | Jones |
| Amy | Ford |
| Jimmy | Wang |
| Ernest | Gilbert |

**INSTRUCTOR**

|  |  |
| --- | --- |
| Fname | Lname |
| John | Smith |
| Ricardo | Browne |
| Susan | Yao |
| Francis | Johnson |
| Ramesh | Shah |

Hai quan hệ này có cùng cấu trúc thuộc tính (Fname, Lname), do đó chúng tương thích hợp và có thể áp dụng các phép toán tập hợp như UNION, INTERSECTION và MINUS.

Khi thực hiện phép giao:

RESULT ← STUDENT ∩ INSTRUCTOR

Kết quả thu được là:

|  |  |
| --- | --- |
| **Fname** | **Fname** |
| Susan | Yao |
| Ramesh | Shah |

Kết quả này thể hiện rằng hai cá nhân **Susan Yao** và **Ramesh Shah** đồng thời có trong cả hai quan hệ, nghĩa là họ vừa là sinh viên vừa là giảng viên.

Phép giao cho phép loại bỏ các bản ghi trùng lặp và chỉ giữ lại các bộ thực sự xuất hiện trong cả hai quan hệ, qua đó giúp nhận diện và xử lý các phần dữ liệu chung giữa các bảng trong cơ sở dữ liệu.

### 3.1.4. Tính chất đại số của phép giao

Phép giao có một số tính chất quan trọng trong Đại số Quan hệ, phản ánh trực tiếp các quy tắc trong lý thuyết tập hợp:

**Tính giao hoán :**

Phép Giao là một phép toán giao hoán, nghĩa là thứ tự của hai quan hệ không ảnh hưởng đến kết quả cuối cùng.

R ∩ S = S ∩ R

Ví dụ: Nếu R = {1, 2, 3} và S={2,3,4} thì cả R ∩ S và S ∩ R đều cho kết quả {2,3}.

Điều này thể hiện rằng khi hai quan hệ R và S có cùng cấu trúc (union-compatible), việc hoán đổi vị trí của chúng trong biểu thức không làm thay đổi tập kết quả.

**Tính kết hợp :**

Tương tự, phép Giao còn có tính kết hợp, nghĩa là khi áp dụng phép toán này cho ba hay nhiều quan hệ khác nhau, ta có thể nhóm các phép toán theo bất kỳ thứ tự nào mà không ảnh hưởng đến kết quả.

( R ∩ S ) ∩ T = R ∩ ( S ∩ T)

Ví dụ: Với 𝑅 = {1, 2, 3} , 𝑆 = {2, 3, 4}, và 𝑇 = {3, 4, 5}, kết quả ở cả hai vế đều là {3}.

Tính chất này giúp cho việc biểu diễn và xử lý các truy vấn phức hợp trong hệ cơ sở dữ liệu trở nên linh hoạt hơn.

**Nguồn:** [1] Ramez Elmasri, Shamkant B. Navathe, Fundamentals of Database Systems, 6th Edition, Addison-Wesley, 2011.

## Phép kết (join)

### 3.2.1. Kết theta (theta JOIN)

Phép kết nối là một trong những phép toán quan trọng và phổ biến trong đại số quan hệ. Mục đích của phép kết nối là kết hợp dữ liệu từ hai hay nhiều quan hệ khác nhau dựa trên một điều kiện xác định, nhằm tạo ra một quan hệ mới chứa thông tin có ý nghĩa hơn. Trong đó, phép kết nối Theta (Theta Join) là dạng tổng quát nhất của phép kết nối, cho phép sử dụng bất kỳ toán tử so sánh nào trong điều kiện nối, thay vì chỉ giới hạn ở phép bằng.

Phép kết nối Theta được ký hiệu là ⋈θ​, trong đó “θ” là toán tử so sánh được sử dụng trong điều kiện nối. Giả sử có hai quan hệ R(A1,A2,…,An) và S(B1, B2,…,Bm) , khi đó điều kiện nối có thể được biểu diễn dưới dạng:

C = (Ai​θBj​)

với θ ∈ {=,≠,<,>,≤,≥}. Khi thực hiện phép kết nối Theta, chỉ những cặp bộ (tr, ts) trong đó tr ∈ R và tr ∈ S thỏa mãn điều kiện C mới được đưa vào kết quả. Kết quả của phép toán là một quan hệ mới có lược đồ bao gồm toàn bộ thuộc tính của hai quan hệ ban đầu.

Phép kết nối Theta không phải là một phép toán cơ bản, mà là phép toán dẫn xuất, được định nghĩa dựa trên sự kết hợp giữa phép chọn (σ) và phép tích Descartes (×). Cụ thể, phép kết nối Theta có thể được viết lại dưới dạng:

R ⋈C​ S = σC​(R × S)

Trong đó, phép tích Descartes R × S tạo ra tất cả các cặp bộ có thể giữa hai quan hệ, còn phép chọn σC​ được sử dụng để lọc các bộ thỏa mãn điều kiện C. Điều này cho thấy phép kết nối Theta có thể được hiểu như một bước rút gọn có điều kiện trên kết quả của phép tích Descartes, giúp giảm đáng kể số lượng bộ được xét so với việc kết hợp toàn bộ các giá trị.

Phép kết nối Theta có tính tổng quát cao, vì nó cho phép sử dụng nhiều loại toán tử khác nhau để mô tả mối quan hệ giữa các thuộc tính của hai quan hệ. Nhờ đó, người dùng có thể biểu diễn linh hoạt các truy vấn mang tính điều kiện phức tạp, chẳng hạn như so sánh giá trị thuộc tính, giới hạn miền dữ liệu, hay kiểm tra chênh lệch giữa các cột trong hai bảng. Trong đại số quan hệ, đây được xem là phép toán trung gian quan trọng để hình thành các phép kết nối cụ thể khác như Equi-Join hoặc Natural Join.

**Xét ví dụ hai quan hệ sau:**

SINHVIEN(MSSV, HoTen, MaLop, DiemTB)

LOP(MaLop, TenLop, Khoa)

Mục tiêu là tìm tất cả các cặp sinh viên và lớp học sao cho mã lớp trong quan hệ SINHVIEN trùng với mã lớp trong quan hệ LOP. Khi đó điều kiện nối là:

C = (SINHVIEN.MaLop = LOP.MaLop)

Phép kết nối Theta tương ứng được viết:

SINHVIEN⋈SINHVIEN.MaLop=LOP.MaLop​LOP

và có thể biểu diễn bằng các phép cơ bản như sau:

σSINHVIEN.MaLop=LOP.MaLop​(SINHVIEN × LOP)

Kết quả thu được là một quan hệ mới chứa các thuộc tính của cả hai bảng:

(MSSV, HoTen, SINHVIEN.MaLop, DiemTB, LOP.MaLop, TenLop, Khoa)

Mỗi bộ trong quan hệ kết quả thể hiện thông tin đầy đủ về sinh viên và lớp mà họ đang theo học.

Ngoài việc dùng toán tử bằng “=”, phép kết nối Theta còn có thể sử dụng các toán tử khác để mở rộng khả năng biểu diễn truy vấn. Chẳng hạn, nếu cần tìm các sinh viên có điểm trung bình cao hơn điểm trung bình chung của lớp mà họ học, điều kiện kết nối có thể viết:

C = (SINHVIEN.DiemTB > LOP.DiemTBtb)

Khi đó, phép kết nối Theta trở thành:

SINHVIEN⋈SINHVIEN.DiemTB>LOP.DiemTBtb​LOP

Phép toán này giúp xác định những sinh viên nổi bật hơn mức trung bình của lớp, minh họa rõ khả năng linh hoạt của phép kết nối Theta trong việc mô tả các mối quan hệ logic giữa các thuộc tính dữ liệu.

Phép kết nối Theta cũng có các tính chất đại số quan trọng tương tự như các phép nối khác trong đại số quan hệ, bao gồm tính giao hoán và tính kết hợp. Nghĩa là, với ba quan hệ R, S và T, ta có thể thực hiện các phép nối theo bất kỳ thứ tự nào mà không làm thay đổi kết quả cuối cùng:

R ⋈ S=S ⋈ R và (R ⋈ S) ⋈ T = R ⋈ (S ⋈ T)

Những tính chất này giữ vai trò quan trọng trong tối ưu hóa truy vấn, giúp hệ quản trị cơ sở dữ liệu có thể lựa chọn thứ tự thực hiện phép nối hiệu quả nhất để giảm chi phí xử lý.

Phép kết nối Theta cũng đóng vai trò nền tảng cho nhiều phép toán nâng cao trong đại số quan hệ. Các phép như Equi-Join (khi θ ==), Natural Join (loại bỏ các cột trùng lặp sau khi Equi-Join), hay Outer Join (bảo toàn các bộ không khớp) đều có thể xem như các trường hợp đặc biệt hoặc mở rộng của phép kết nối Theta. Nhờ vậy, phép nối này được xem là cầu nối giữa lý thuyết đại số quan hệ thuần túy và các thao tác truy vấn phức tạp trong hệ quản trị cơ sở dữ liệu.

**3.2.2. Kết bằng (equi JOIN)**

Phép kết nối bằng là một trường hợp đặc biệt của phép kết nối Theta, trong đó toán tử so sánh được sử dụng là dấu bằng (=). Mục tiêu của phép kết nối bằng là kết hợp các bộ dữ liệu từ hai quan hệ khác nhau khi các giá trị của một hay nhiều thuộc tính trong hai quan hệ đó bằng nhau. Đây là một trong những phép toán phổ biến nhất trong đại số quan hệ, vì trong hầu hết các cơ sở dữ liệu, các quan hệ thường được liên kết thông qua các khóa có cùng giá trị định danh.

Giả sử có hai quan hệ R(A1, A2,…, An) và S(B1​, B2​,…, Bm​), trong đó tồn tại cặp thuộc tính Ai​ và Bj ​ có cùng miền giá trị. Khi đó, phép kết nối bằng được ký hiệu là:

R ⋈ Ai​=Bj​​ S

và được định nghĩa là tập hợp tất cả các bộ kết hợp (tr, ts) , trong đó tr ∈ R và ts ∈ S thỏa mãn điều kiện tr[Ai] = ts[Bj]. Kết quả của phép toán là một quan hệ mới có lược đồ bao gồm tất cả các thuộc tính của hai quan hệ đầu vào, trong đó các cặp bộ có giá trị thuộc tính bằng nhau được kết hợp lại.

Phép kết nối bằng, tương tự như các phép nối khác, có thể được biểu diễn bằng các phép toán cơ bản trong đại số quan hệ. Cụ thể, phép này được xác định bởi công thức:

R⋈Ai​=Bj​​ S = σAi​=Bj​​(R × S)

Điều này có nghĩa rằng phép kết nối bằng được hình thành bằng cách thực hiện phép tích Descartes giữa hai quan hệ R và S, sau đó áp dụng phép chọn để giữ lại các bộ có giá trị thuộc tính bằng nhau. Phương pháp này giúp làm rõ bản chất của Equi-Join như là một phép chọn có điều kiện bằng trên tích Descartes.

Để minh họa, xét hai quan hệ sau:

NHANVIEN(MaNV, TenNV, MaPB, Luong)

PHONGBAN(MaPB, TenPB, DiaChi)

Điều kiện nối là :

C = (NHANVIEN.MaPB = PHONGBAN.MaPB)

Trong phép kết nối bằng, điều kiện nối được chỉ định tường minh thông qua dấu bằng (=) giữa các thuộc tính tương ứng của hai quan hệ, nhằm xác định các cặp bộ có giá trị thuộc tính bằng nhau.

Khi thực hiện phép kết nối bằng, ta thu được:

NHANVIEN ⋈NHANVIEN.MaPB=PHONGBAN.MaPB​ PHONGBAN

Kết quả của phép toán là một quan hệ mới chứa các thuộc tính:

(MaNV, TenNV, MaPB(NHANVIEN)​, Luong, MaPB(PHONGBAN)​, TenPB, DiaChi)

Trong kết quả, ta nhận thấy rằng thuộc tính MaPB xuất hiện hai lần, một lần từ quan hệ NHANVIEN và một lần từ PHONGBAN. Đây là đặc điểm tiêu biểu của phép Equi-Join: phép nối này không loại bỏ các thuộc tính trùng lặp, mà giữ nguyên toàn bộ cấu trúc của hai quan hệ ban đầu. Việc trùng thuộc tính phản ánh đúng bản chất của phép toán này, khi nó chỉ đảm bảo điều kiện “bằng nhau” mà không thực hiện thao tác gộp hay loại bỏ.

Đặc điểm này cũng là cơ sở để hình thành phép kết nối tự nhiên (Natural Join), một biến thể của Equi-Join, trong đó các thuộc tính trùng tên sẽ được tự động loại bỏ khỏi kết quả để tạo ra một quan hệ ngắn gọn, dễ đọc và trực quan hơn. Điều đó giúp Natural Join thể hiện cùng một mối quan hệ dữ liệu như Equi-Join, nhưng dưới dạng cô đọng và tối ưu hơn.

Phép kết nối bằng cũng có các tính chất đại số quan trọng tương tự như các phép nối khác trong đại số quan hệ, bao gồm tính giao hoán và tính kết hợp. Nghĩa là, với ba quan hệ R, S và T, ta có thể thực hiện các phép nối theo bất kỳ thứ tự nào mà không làm thay đổi kết quả cuối cùng:

R ⋈ S= S ⋈ R và (R ⋈ S) ⋈ T = R ⋈ (S ⋈ T)

Những tính chất này giữ vai trò quan trọng trong tối ưu hóa truy vấn, giúp hệ quản trị cơ sở dữ liệu có thể lựa chọn thứ tự thực hiện phép nối hiệu quả nhất để giảm chi phí xử lý.

Tóm lại, phép kết nối bằng là một phép nối nền tảng trong đại số quan hệ, thường được sử dụng để liên kết dữ liệu giữa các bảng có khóa chung. Nó thể hiện mối quan hệ logic giữa các thực thể có cùng giá trị thuộc tính, đồng thời là nền tảng cho các phép nối mở rộng như Natural Join hay Outer Join. Hiểu rõ đặc điểm của phép Equi Join đặc biệt là hiện tượng trùng thuộc tính giúp nắm bắt sâu hơn bản chất của các phép nối trong cơ sở dữ liệu quan hệ.

**3.2.3. Phép kết tự nhiên ( natural join )**

Phép kết nối tự nhiên là một biến thể đặc biệt của phép kết nối bằng (Equi-Join), được sử dụng phổ biến trong đại số quan hệ. Điểm khác biệt của phép kết nối tự nhiên là nó tự động loại bỏ các thuộc tính trùng tên trong kết quả, nhờ đó giúp tạo ra một quan hệ mới ngắn gọn, trực quan và dễ hiểu hơn. Phép toán này được gọi là “tự nhiên” vì nó dựa trên sự tương đồng tên thuộc tính giữa các quan hệ, chứ không cần chỉ định điều kiện nối một cách tường minh.

Giả sử có hai quan hệ R(A1, A2,…, An) và S(B1, B2,…, Bm). Nếu hai quan hệ này có một hoặc nhiều thuộc tính trùng tên, khi đó phép kết nối tự nhiên giữa chúng được ký hiệu là:

R ⋈ S

và được định nghĩa là phép nối giữa R và S trên tất cả các cặp thuộc tính có cùng tên và cùng miền giá trị. Kết quả của phép nối là một quan hệ mới, trong đó:

* + - * Chỉ giữ lại một bản duy nhất của mỗi thuộc tính trùng tên.
      * Chứa tất cả các thuộc tính không trùng tên từ cả hai quan hệ.

Phép kết nối tự nhiên có thể được xem như một phép Equi-Join kèm theo bước loại bỏ các thuộc tính trùng lặp. Vì vậy, nó có thể biểu diễn bằng công thức sau:

R ⋈ S = ΠA,B​(σAi​=Bi​​(R × S))

trong đó phép chọn σAi=Biσ\_{A\_i = B\_i}σAi​=Bi​​ thực hiện điều kiện bằng nhau giữa các thuộc tính trùng tên, và phép chiếu ΠA,B loại bỏ các cột trùng lặp trong kết quả.

Để minh họa, xét lại ví dụ hai quan hệ:

NHANVIEN(MaNV, TenNV, MaPB, Luong)

PHONGBAN(MaPB, TenPB, DiaChi)

Điều kiện nối là:

Điều khác biệt là ở phép kết nối tự nhiên, điều kiện này không cần chỉ định tường minh như trong Equi-Join, mà được tự động xác định dựa trên các thuộc tính trùng tên và cùng miền giá trị giữa hai quan hệ ở đây là MaPB.

Khi thực hiện phép kết nối tự nhiên, ta có:

NHANVIEN ⋈ PHONGBAN

Kết quả của phép toán là một quan hệ mới chứa các thuộc tính:

(MaNV, TenNV, MaPB, Luong, TenPB, DiaChi)

Kết quả thu được là một quan hệ thể hiện mối liên hệ giữa mỗi nhân viên và phòng ban mà họ thuộc về, trong đó thuộc tính MaPB chỉ xuất hiện một lần nhờ đặc tính tự động loại bỏ trùng lặp của phép Natural Join. Kết quả này vì vậy ngắn gọn hơn và dễ quan sát hơn so với kết quả của phép Equi Join.

Phép kết nối tự nhiên cũng kế thừa các tính chất đại số của phép nối tổng quát, bao gồm tính giao hoán và tính kết hợp. Nghĩa là, với ba quan hệ R, S và T, ta có thể thực hiện các phép nối tự nhiên theo bất kỳ thứ tự nào mà không làm thay đổi kết quả cuối cùng:

R ⋈ S = S ⋈ R và (R ⋈ S) ⋈ T = R ⋈ (S ⋈ T)

Những tính chất này rất quan trọng trong tối ưu hóa truy vấn, giúp hệ quản trị cơ sở dữ liệu lựa chọn thứ tự thực hiện phép nối sao cho hiệu quả nhất.

Tóm lại, phép kết nối tự nhiên là một phép nối mở rộng từ Equi Join, vừa giữ được bản chất logic của phép nối bằng, vừa loại bỏ sự dư thừa thuộc tính. Phép Natural Join giúp biểu diễn các mối quan hệ giữa các bảng dữ liệu một cách ngắn gọn, rõ ràng và dễ xử lý hơn. Nhờ ưu điểm này, nó trở thành một trong những phép toán được sử dụng rộng rãi nhất trong thực tế khi mô hình hóa và truy vấn cơ sở dữ liệu quan hệ.

**3.2.4. Phép kết ngoài ( outer join )**

Phép kết nối ngoài (Outer Join) là một mở rộng của các phép kết nối thông thường trong đại số quan hệ, chẳng hạn như Equi Join hoặc Natural Join. Điểm khác biệt của phép nối này là nó không chỉ giữ lại các bộ thỏa mãn điều kiện nối giữa hai quan hệ, mà còn bảo toàn những bộ không có giá trị khớp ở một trong hai quan hệ. Nhờ vậy, kết quả của phép kết nối ngoài luôn bao gồm toàn bộ dữ liệu từ một hoặc cả hai quan hệ, với các giá trị thiếu được điền bằng NULL.

Mục đích của phép kết nối ngoài là giúp tránh mất dữ liệu trong các tình huống mà một trong hai bảng có các bộ không có giá trị khớp, chẳng hạn như khi muốn liệt kê toàn bộ nhân viên kể cả những người chưa được phân công phòng ban, hoặc toàn bộ phòng ban kể cả những phòng chưa có nhân viên.

#### 3.2.4.1. Phân loại phép kết nối ngoài

Phép kết nối ngoài bao gồm ba biến thể chính:

* + - * **Left Outer Join (⋈⟕)** – Giữ lại tất cả các bộ của quan hệ bên trái
      * **Right Outer Join (⋈⟖)** – Giữ lại tất cả các bộ của quan hệ bên phải.
      * **Full Outer Join (⋈⟗)** – Giữ lại tất cả các bộ của cả hai quan hệ.

Các phép này chỉ khác nhau ở phạm vi giữ lại các bộ không khớp, còn bản chất xử lý phần “nối” vẫn tương tự như Equi-Join hoặc Natural Join.

#### 3.2.4.2. Ký hiệu và công thức

Giả sử có hai quan hệ R và S cùng điều kiện nối C:

**Left Outer Join:**

R⟕C​S

Bao gồm tất cả các bộ từ R⋈CS và các bộ còn lại của R không có bộ tương ứng trong S, với các giá trị thiếu được điền bằng NULL.

**Right Outer Join:**

R⟖C​S

Giữ lại toàn bộ các bộ từ S, kể cả những bộ không có đối tượng tương ứng trong R.

**Full Outer Join:**

R⟗C​S

Kết hợp kết quả của cả Left và Right Outer Join, đảm bảo rằng tất cả các bộ từ cả hai quan hệ đều được đưa vào kết quả cuối cùng.

#### 3.2.4.3. Ví dụ minh họa

Xét hai quan hệ:

NHANVIEN(MaNV, TenNV, MaPB)

PHONGBAN(MaPB, TenPB)

Giả sử điều kiện nối là:

C=(NHANVIEN.MaPB=PHONGBAN.MaPB)

* **Left Outer Join:**

NHANVIEN⟕NHANVIEN.MaPB=PHONGBAN.MaPB​PHONGBAN

Kết quả bao gồm tất cả nhân viên, kể cả những người chưa được phân vào phòng ban nào.

Các giá trị thuộc tính của PHONGBAN sẽ được điền NULL nếu không có giá trị khớp.

|  |  |  |  |
| --- | --- | --- | --- |
| **MaNV** | **TenNV** | **MaPB** | **TenPB** |
| NV01 | An | P01 | Kinh doanh |
| NV02 | Bình | P02 | Kế toán |
| NV03 | Cường | NULL | NULL |

* **Right Outer Join:**

NHANVIEN⟖NHANVIEN.MaPB=PHONGBAN.MaPB​PHONGBAN

Kết quả bao gồm tất cả phòng ban, kể cả những phòng chưa có nhân viên nào.

Các giá trị của NHANVIEN sẽ là NULL nếu không có bộ tương ứng.

|  |  |  |  |
| --- | --- | --- | --- |
| **MaNV** | **MaNV** | **MaPB** | **TenPB** |
| NV01 | An | P01 | Kinh doanh |
| NV02 | Bình | P02 | Kế toán |
| NULL | NULL | P03 | Nhân sự |

* **Full Outer Join:**

NHANVIEN⟗NHANVIEN.MaPB=PHONGBAN.MaPB​PHONGBAN

Kết quả bao gồm tất cả các bộ từ cả hai quan hệ, dù có hay không có giá trị khớp.

|  |  |  |  |
| --- | --- | --- | --- |
| **MaNV** | **MaNV** | **MaPB** | **TenPB** |
| NV01 | An | P01 | Kinh doanh |
| NV02 | Bình | P02 | Kế toán |
| NV03 | Cường | NULL | NULL |
| NULL | NULL | P03 | Nhân sự |

#### 3.2.4.4. Tính chất của phép kết nối ngoài

Phép kết nối ngoài mở rộng tính chất giao hoán và kết hợp của phép nối thông thường, tuy nhiên có một số điểm cần lưu ý:

* Left Outer Join và Right Outer Join không hoàn toàn giao hoán, do vai trò “bên trái – bên phải” khác nhau. Tuy nhiên, chúng có mối quan hệ đối xứng:

R ⟕ S = S ⟖ R

* **Full Outer Join** có tính giao hoán và kết hợp tương tự phép nối trong:

R ⟗ S =S ⟗ R, (R ⟗ S) ⟗ T = R⟗ (S ⟗ T)

#### 3.2.4.5. Ý nghĩa và ứng dụng

Phép kết nối ngoài đặc biệt hữu ích trong các tình huống dữ liệu không đầy đủ hoặc thiếu quan hệ ràng buộc. Nó cho phép người dùng:

* Giữ lại toàn bộ thông tin của một bảng chính mà không bị mất dữ liệu.
* Thực hiện các truy vấn tổng hợp, thống kê, và báo cáo đầy đủ hơn.
* Hỗ trợ các bài toán thực tế như “liệt kê tất cả nhân viên và phòng ban, kể cả khi chưa có sự phân công”.

Phép Outer Join do đó được xem là sự mở rộng linh hoạt và thực tiễn của các phép nối cơ bản trong đại số quan hệ, đồng thời là nền tảng quan trọng trong các hệ quản trị cơ sở dữ liệu hiện đại.

## 3.3.Phép chia ( Division )

**3.3.1. Khái niệm và ý nghĩa**

Phép Chia (Division – ÷) là một trong những phép toán nhị phân quan trọng trong đại số quan hệ, được sử dụng cho những truy vấn có chứa cụm từ “for all” (cho tất cả).

Phép chia giúp ta tìm ra các giá trị của một thuộc tính trong quan hệ R(X, Y) sao cho giá trị đó liên kết với tất cả các giá trị của thuộc tính trong một quan hệ khác S(Y).

Nói một cách cụ thể hơn, nếu ta có hai quan hệ R(X, Y) và S(Y), thì kết quả của phép chia R ÷ S là một quan hệ T(X) gồm các giá trị X mà đối với mọi giá trị Y trong S, cặp (X, Y) đều tồn tại trong R. Điều này có nghĩa là phép chia dùng để biểu diễn mối quan hệ dạng “X có liên quan đến tất cả các Y trong S”.

Phép chia rất hữu ích trong các tình huống thực tế, khi ta cần tìm các thực thể thỏa mãn tất cả các điều kiện trong một tập xác định.

Ví dụ:

* + - * Tìm các **sinh viên đã đăng ký tất cả các khóa học** trong chương trình đào tạo
      * Tìm các **nhân viên làm việc trong mọi dự án** do một phòng ban quản lý.

Những loại truy vấn này thường khó biểu diễn bằng các phép toán đơn giản, nhưng phép chia giúp mô tả chúng một cách tự nhiên và chính xác hơn.

**3.3.2. Biểu diễn bằng các phép toán cơ bản**

Phép chia có thể được biểu diễn lại bằng các phép toán cơ bản của đại số quan hệ như phép chiếu (π), phép hiệu (−) và phép tích Descartes (×).

Ta có công thức tổng quát như sau:

T(X) = πX​(R) − πX​((πX​(R) × S) − R)

Trong đó:

* + - * R(X, Y): quan hệ ban đầu chứa cặp giá trị (X, Y).
      * S(Y): quan hệ chứa tập giá trị Y mà ta muốn kiểm tra điều kiện “cho tất cả”.
      * T(X): kết quả phép chia, bao gồm các giá trị X thỏa mãn rằng với mọi Y trong S, cặp (X, Y) đều có trong R.
      * X là tập thuộc tính còn lại của R không có trong S

Ý nghĩa của từng phần trong công thức được giải thích như sau:

* + - * πX​(R) × S: tạo ra tất cả các cặp khả dĩ giữa X trong R và Y trong S
      * (πX(R) × S) − R: loại bỏ những cặp (X, Y) không thực sự tồn tại trong R
      * πX((πX(R) × S) − R): xác định các X thiếu ít nhất một Y trong S
      * Cuối cùng, πX− phần trên sẽ cho ra các X không bị thiếu bất kỳ Y nào trong S, tức là các X liên kết với toàn bộ các Y — đó chính là kết quả của phép chia.

Như vậy, phép chia có thể được xem là sự lọc các phần tử X đầy đủ, loại bỏ những phần tử không có quan hệ với toàn bộ các giá trị Y trong S.

**3.3.3. Ví dụ minh họa**

Xét ví dụ trong cơ sở dữ liệu quản lý nhân sự: “Tìm tên các nhân viên làm việc trong tất cả các dự án được quản lý bởi phòng ban số 5.”

Giả sử ta có các quan hệ sau:

* + - * WORKS\_ON(Essn, Pno): chứa thông tin về các nhân viên (Essn) và các dự án (Pno) mà họ đang tham gia
      * PROJECT(Pnumber, Dnum): chứa thông tin về mã dự án và phòng ban quản lý dự án đó.
* **EMPLOYEE(Ssn, Fname, Lname, Dno, …)**: chứa thông tin nhân viên.

**Bước 1:** Chọn tất cả các dự án do **phòng ban số 5** quản lý:

S←πPnumber​(σDnum=5​(PROJECT))

**Bước 2:** Xác định quan hệ **R**, thể hiện mối quan hệ giữa nhân viên và dự án mà họ tham gia:

R←WORKS\_ON(Essn,Pno)

**Bước 3:** Áp dụng **phép chia (÷)** để tìm ra các nhân viên làm việc trong **tất cả các dự án** thuộc **S**:

T(Essn)← R ÷ S

**Bước 4:** Liên kết kết quả với quan hệ **EMPLOYEE** để lấy tên nhân viên:

RESULT←πFname,Lname​(T ⋈ EMPLOYEE)

Kết quả RESULT sẽ chứa tên các nhân viên làm việc trong mọi dự án do phòng ban số 5 quản lý.

Qua ví dụ này, ta thấy phép chia (÷) đặc biệt hữu ích trong các truy vấn có chứa điều kiện “cho tất cả”, chẳng hạn như “làm việc trong mọi dự án”, “đăng ký mọi môn học”, v.v.  
Những dạng truy vấn này khó thể hiện bằng các phép toán đơn giản như chọn, chiếu, hoặc kết nối, nhưng có thể được biểu diễn ngắn gọn và rõ ràng bằng phép chia.

**Chương 4. ÁNH XẠ ĐẠI SỐ QUAN HỆ VÀO NGÔN NGỮ SQL**

## 4.1. Ánh xạ các toán tử cơ bản

Đại số Quan hệ (Relational Algebra – RA) cung cấp một tập hợp các phép toán được thiết kế để thao tác trên các quan hệ . Tính chất quan trọng nhất của Đại số Quan hệ là **tính đóng (closure)**: đầu vào của một phép toán là một hoặc hai quan hệ, và **kết quả đầu ra luôn là một quan hệ mới**. Nhờ tính chất này, các phép toán đại số quan hệ có thể được **kết hợp với nhau** **(composed)** để tạo thành các biểu thức, tương tự như việc kết hợp các phép toán số học như cộng, trừ, nhân, chia. Các biểu thức này cho phép định nghĩa một số lượng lớn các truy vấn cho cơ sở dữ liệu quan hệ.

Tập hợp các phép toán cơ bản (fundamental operations) trong Đại số Quan hệ bao gồm:

* Phép Chọn (Selection, *σ*)
* Phép Chiếu (Projection, *π*)
* Phép Tích Descartes (Cartesian product, ×)
* Phép Hợp (Union, ∪)
* Phép Trừ (Difference, − hoặc \)
* Phép Giao (Intersection, ∩)

Các phép toán này được phân loại dựa trên số lượng quan hệ đầu vào:

* **Các Toán tử Đơn nguyên (Unary Operations):** Chỉ hoạt động trên **một quan hệ** duy nhất. Chúng bao gồm Phép Chọn (*σ*), Phép Chiếu (*π*), và Phép Đổi tên (*ρ*).
* **Các Toán tử Nhị nguyên (Binary Operations):** Hoạt động trên **hai quan hệ**. Chúng bao gồm Phép Hợp (∪), Phép Trừ (− hoặc ∖), Phép Tích Descartes (×), và các biến thể của Phép Kết (Join)

## 4.1.1. Ánh xạ của các phép chọn (σ) và phép chiếu (π)

Các phép toán Chọn (*σ*) và Chiếu (*π*) là các phép toán đơn nguyên (chỉ thao tác trên một quan hệ) và là những phép toán cơ bản nhất để thao tác dữ liệu quan hệ

### A. Phép chọn (*σ*)

#### a. Khái niệm

* + Ký hiệu bằng chữ cái Hy Lạp sigma thường (*σ*).
  + **Mục đích:** **Chọn các bộ (tuples/hàng)** thỏa mãn một điều kiện (vị từ - predicate) đã cho.
  + Phép Chọn hoạt động như một bộ lọc (filter), chỉ giữ lại các hàng thỏa mãn điều kiện. Nó có thể được hình dung là sự phân chia ngang (horizontal partition) của quan hệ.
  + **Ký hiệu:** *σC*​(*R*), trong đó *R* là quan hệ đầu vào và *C* là điều kiện chọn (vị từ) được đặt ở chỉ số dưới.
  + **Kết quả:** Quan hệ mới chứa các bộ từ *R* thỏa mãn *C*. Quan hệ kết quả có lược đồ (số lượng thuộc tính) giống như quan hệ đầu vào *R*.
  + **Điều kiện C:** Là một biểu thức Boolean (logic) kết hợp các phép toán ∧ (AND), ∨ (OR), và ¬ (NOT) trên các điều kiện nguyên tử. Điều kiện nguyên tử có thể là so sánh giữa hai tên cột hoặc so sánh giữa tên cột với một giá trị hằng.
  + **Tính chất:** Phép Chọn có tính giao hoán: *σC*1​(*σC*2​(*R*)) = *σC*2​(*σC*1​(*R*)).

#### b. Ánh xạ sang SQL

Phép toán Chọn (*σ*) trong Đại số Quan hệ ánh xạ trực tiếp sang **mệnh đề WHERE** trong câu lệnh SELECT của SQL.

*σC*​(*R*) → **WHERE** *C* (trong câu lệnh SELECT)

* + Điều kiện *C* trong *σC*​(*R*) được đặt trong mệnh đề WHERE của SQL.
  + Trong SQL, các phép toán logic tương ứng là AND, OR, và NOT

### B. Phép chiếu (π)

#### a. Khái niệm:

* + Ký hiệu bằng chữ cái Hy Lạp pi hoa (*π* hoặc Π).
  + **Mục đích:** **Chọn các thuộc tính (attributes/cột)** cụ thể từ một quan hệ, loại bỏ các cột không cần thiết.
  + Phép Chiếu hoạt động như một sự phân chia dọc (vertical partition) của quan hệ.
  + **Ký hiệu:** *π*A​(*R*), trong đó *R* là quan hệ đầu vào và *A* là tập hợp các thuộc tính cần chiếu (các cột cần hiển thị).
  + **Tính chất về trùng lặp:** Vì trong mô hình quan hệ toán học, một quan hệ là một **tập hợp** các bộ (set of tuples), phép Chiếu **tự động loại bỏ các bộ trùng lặp**.
  + **Mở rộng:** Phép Chiếu có thể được mở rộng thành **Phép Chiếu Tổng quát (Generalized Projection)** để cho phép sử dụng các biểu thức số học hoặc hàm trên các thuộc tính trong danh sách chiếu (ví dụ: *πTENNV*,(*HSL*+*HSCV*)×*LUONG*\_*CB*​ (*BANGLUONG*))

#### b. Ánh xạ sang SQL

Phép Chiếu (*π*) trong Đại số Quan hệ ánh xạ trực tiếp sang **mệnh đề SELECT** trong SQL:

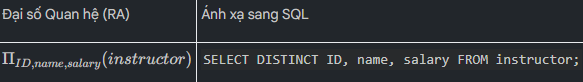
*πA*​(*R*) → **SELECT DISTINCT** *A* (hoặc **SELECT** *A*)

* + Khi ánh xạ *π* (của Đại số Quan hệ thuần túy, vốn tự động loại bỏ trùng lặp) sang SQL, cần sử dụng từ khóa **DISTINCT** trong mệnh đề SELECT để đảm bảo kết quả phù hợp với định nghĩa tập hợp (set version) của Đại số Quan hệ.
  + Nếu từ khóa **DISTINCT** bị bỏ qua, SQL sẽ coi các bảng là đa tập hợp (multiset), cho phép các bộ trùng lặp xuất hiện trong kết quả

### C. Ví dụ chuyển đổi từ ĐSQH sang SQL

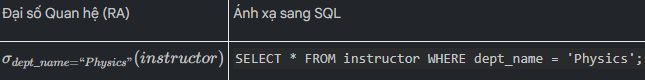
VD: Cho quan hệ *INSTRUCTOR* có lược đồ: *INSTRUCTOR* (*ID*, *name*, *dept*\_*name*, *salary*).

* **Truy vấn 1: Tìm ID, tên và lương của tất cả các giảng viên.**



Hình 4. 1: Tìm thông tin của gv

* **Truy vấn 2: Tìm tất cả các giảng viên thuộc khoa "Physics".**

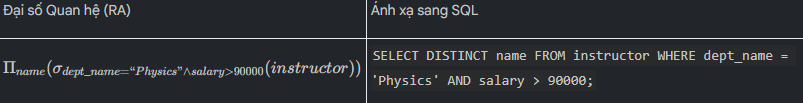


Hình 4. 2: Tìm gv khoa vật lí

* **Truy vấn 3: Tìm tên của các giảng viên thuộc khoa "Physics" có mức lương lớn hơn $90000.**

Bài toán này đòi hỏi sự kết hợp của Phép Chọn và Phép Chiếu.

Phép Chọn được áp dụng trước để giới hạn số lượng bộ (hàng), sau đó Phép Chiếu được áp dụng để chọn các thuộc tính (cột) mong muốn:



Hình 4. 3: Tìm gv khoa vật lí có lương > 90000$

## 4.1.2. Ánh xạ toán tử tập hợp

Trước khi thực hiện bất kỳ phép toán tập hợp cơ bản nào (Hợp, Giao, Trừ), các quan hệ tham gia phải thỏa mãn điều kiện **Tương thích Tập hợp** (Union Compatibility).

Các điều kiện này bao gồm:

* **Cùng số lượng thuộc tính (Arity):** Hai quan hệ phải có cùng số lượng cột.
* **Tương thích miền giá trị (Domain Compatibility):** Các thuộc tính tương ứng (thứ *i*) trong hai quan hệ phải được lấy từ cùng một miền giá trị (domain).

Sự tương thích này đảm bảo rằng kết quả của phép toán cũng là một quan hệ hợp lệ

### A. Phép hợp (∪)

#### a. Khái niệm

Phép Hợp (∪, ký hiệu ĐSQH: *R* ∪ *S*) tạo ra một quan hệ mới bao gồm tất cả các bộ (hàng) có trong quan hệ *R*, hoặc quan hệ *S*, hoặc cả hai.

**Trong SQL, phép Hợp được ánh xạ trực tiếp sang từ khóa UNION**.

**Đặc điểm quan trọng:**

* + Phép toán UNION trong SQL mặc định **tự động loại bỏ các bộ trùng lặp**, phù hợp với định nghĩa toán học rằng một quan hệ là một tập hợp các bộ.
  + Nếu muốn giữ lại tất cả các bộ trùng lặp (tức là làm việc với multisets), SQL có từ khóa UNION ALL. Trong trường hợp sử dụng UNION ALL, số lượng bản sao của một bộ trong kết quả bằng tổng số bản sao của bộ đó trong hai quan hệ đầu vào.
  + Về mặt Đại số, phép toán Hợp có tính chất **giao hoán** (*R* ∪ *S* ≡ *S* ∪ *R*) và **kết hợp**

VD: Tính giao hoán và kết hợp của phép hợp

*a. (E1 ∪ E2) ∪ E3 ≡ E1 ∪ (E2 ∪ E3)*

*b. (E1 ∩ E2) ∩ E3 ≡ E1 ∩ (E2 ∩ E3)*

#### b. Ánh xạ sang SQL

*R* ∪ *S* → **SELECT \* FROM** R **UNION SELECT \* FROM** S**;**.

**Lưu ý:**

**Yêu cầu Union Compatibility:** Cả hai truy vấn con (hai vế của UNION) phải tương thích tập hợp (cùng số lượng và loại thuộc tính).

### B. Phép trừ (− hay ∖)

#### a. Khái niệm

Phép Trừ (Set Difference, ký hiệu ĐSQH: *R*−*S* hoặc *R*∖*S*) tạo ra một quan hệ mới chứa các bộ có mặt trong *R* nhưng **không** có mặt trong *S*.

**Trong SQL, phép Trừ được ánh xạ sang từ khóa EXCEPT**.

* + Lưu ý: Một số hệ thống quản trị cơ sở dữ liệu (DBMS) khác như Oracle hoặc MS SQLServer sử dụng từ khóa ***MINUS*** thay cho EXCEPT.
  + **Phép trừ không có tính giao hoán (*R*−*S* ≠ *S*−*R*).**

**Các cách biểu diễn thay thế (khá phổ biến trong thực tế):**

Phép Trừ thường được mô phỏng bằng cách sử dụng các phép toán truy vấn con (subqueries) hoặc phép kết ngoài (Outer Join), đặc biệt là trong các hệ thống không hỗ trợ trực tiếp EXCEPT.

* **Sử dụng NOT IN:** Dùng mệnh đề WHERE và toán tử NOT IN để loại trừ các bộ trong *R* mà tồn tại trong *S*.
* **Sử dụng NOT EXISTS:** Dùng mệnh đề WHERE và toán tử NOT EXISTS với một truy vấn con trên *S* để tìm các bộ trong *R* không có bộ nào khớp trong *S*. Phương pháp này được sử dụng để mô phỏng sự khác biệt tập hợp.
* **Sử dụng Phép kết ngoài trái (Left Outer Join) loại trừ:** *R*−*S* có thể được ánh xạ thành phép kết ngoài trái *R* LEFT JOIN *S* nhưng **chỉ giữ lại** các bộ trong *R* mà không có đối sánh nào trong *S*. Điều này được thực hiện bằng cách thêm điều kiện WHERE để kiểm tra rằng một thuộc tính khóa trong *S* là NULL (ví dụ: WHERE B.a1 IS NULL). Trong tài liệu, đây được gọi là Left Join Excluding (LJ(E)).

#### b. Ánh xạ sang SQL

*R* – *S hay (R \ S)* → **SELECT \* FROM** R **EXCEPT SELECT \* FROM** S**;**.

**Lưu ý chi tiết về phương pháp trên :**

* **Multiset Difference:** Để giữ lại các bản sao (multiset), SQL sử dụng toán tử **EXCEPT ALL**. Số lượng bản sao của một bộ trong kết quả bằng số bản sao của bộ đó trong *R* trừ đi số bản sao của bộ đó trong *S*, miễn là hiệu số đó dương.
* **Hạn chế của DBMS:** MySQL không hỗ trợ toán tử EXCEPT.

Các cách ánh xạ khác:

Vì EXCEPT không được hỗ trợ phổ biến, phép trừ thường được mô phỏng bằng

cách sử dụng các phép toán Join hoặc truy vấn con.

* + **Sử dụng NOT IN (Truy vấn con):**

*R* − *S* → **SELECT \* FROM** R

**WHERE** (A1, A2, ...)

**NOT IN (SELECT** A1, A2, ... **FROM** S**);**

Áp dụng khi hai quan hệ chỉ có một hoặc một vài thuộc tính tương ứng cần kiểm tra.

* + **Sử dụng NOT EXISTS (Truy vấn con tương quan):**

*R* − *S* → **SELECT** R.\* **FROM** R

**WHERE NOT EXISTS (SELECT \* FROM** S

**WHERE** R.Key = S.Key

**AND** R.A1 = S.A1 **...);**.

Phương pháp này mô phỏng phép trừ bằng cách tìm các bộ trong *R* không có bộ nào khớp trong *S*.

* + **Sử dụng Left Join Loại trừ (LJ(E)):**

*R* − *S* → **SELECT** R.\* **FROM** R

**LEFT JOIN** S **ON** R.Key = S.Key

**WHERE** S.Key **IS NULL;**.

Phương pháp này được gọi là Left Join Excluding (LJ(E)). Truy vấn này giữ lại tất cả các bộ từ *R* không tìm thấy đối sánh nào trong *S*, bằng cách kiểm tra giá trị NULL trong một thuộc tính khóa của *S*. Phương pháp này cũng được sử dụng để ánh xạ phép toán Trừ D (A - B).

### C. Phép Giao (∩)

#### a. Khái niệm

Phép Giao (∩, ký hiệu ĐSQH: *R* ∩ *S*) tạo ra một quan hệ mới bao gồm các bộ có mặt trong **cả** *R* và *S*.

**Trong SQL, phép Giao được ánh xạ trực tiếp sang từ khóa INTERSECT**.

* + Phép INTERSECT cũng tự động loại bỏ các bản sao. Tương tự như UNION, phiên bản INTERSECT ALL tồn tại trong một số hệ thống để giữ lại các bản sao (multisets).
  + Phép Giao có thể được tính toán chỉ bằng các phép Trừ: *R* ∩ *S* = *R* − (*R* − *S*).

**Các cách biểu diễn thay thế (rất hiệu quả và phổ biến):**

Phép Giao thường được biểu diễn bằng các phép toán kết (Join) hoặc truy vấn con vì không phải tất cả các hệ thống đều hỗ trợ INTERSECT (ví dụ: MySQL không hỗ trợ INTERSECT và EXCEPT).

* **Sử dụng Phép kết trong (Inner Join):** Đây là phương pháp hiệu quả nhất để tìm giao của hai quan hệ dựa trên tất cả các thuộc tính. Cụ thể:

*R* ∩ *S* ≡ *R* **INNER JOIN** *S* trên tất cả các thuộc tính chung.

* **Sử dụng IN:** Dùng mệnh đề WHERE và toán tử IN để kiểm tra các bộ trong *R* có tồn tại trong *S*.
* **Sử dụng EXISTS:** Dùng mệnh đề WHERE EXISTS với một truy vấn con trên *S* để tìm các bộ trong *R* có đối sánh trong *S*

#### b. Ánh xạ sang SQL

*R* ∩ *S* → **SELECT \* FROM** R **INTERSECT SELECT \* FROM** S**;**

**Lưu ý chi tiết về cách trên:**

* **Giữ lại trùng lặp (Multiset semantics):** Để giữ lại các bản sao, SQL sử dụng toán tử **INTERSECT ALL**. Số lượng bản sao của một bộ trong kết quả bằng số lượng bản sao tối thiểu của bộ đó trong *R* và *S*.
* **Hạn chế của DBMS:** MySQL không triển khai toán tử INTERSECT.

Các phương pháp ánh xạ thay thế:

Phép Giao là phép toán không cơ bản và có thể được biểu diễn bằng các phép toán khác trong ĐSQH, ví dụ: *R* ∩ *S* = *R* − (*R* − *S*). Trong SQL, nó thường được mô phỏng bằng các phép kết hoặc truy vấn con vì tính hiệu quả cao hơn.

* **Sử dụng Inner Join (IJ) (Phương pháp hiệu quả):**

*R* ∩ *S* → **SELECT** R.\* **FROM** R

**INNER JOIN** S **ON** R.A = S.A **AND** R.B = S.B **...;**.

Phép Join trong (Inner Join) dựa trên tất cả các thuộc tính chung (khi các bảng tương thích tập hợp) tạo ra kết quả giống hệt với phép Giao. Trong thực tế, người ta hay sử dụng Inner Join để tìm Intersection vì nó **nhanh hơn** so với Outer Join có điều kiện.

* **Sử dụng EXISTS (Truy vấn con tương quan):**

*R* ∩ *S* → **SELECT** R.\* **FROM** R

**WHERE EXISTS (SELECT \* FROM** S

**WHERE** R.A = S.A **AND** R.B = S.B **...);**.

Phương pháp này kiểm tra sự tồn tại của một bộ khớp hoàn toàn trong *S* đối với mỗi bộ trong *R*.

* **Sử dụng IN (Truy vấn con):**

*R* ∩ *S* → **SELECT DISTINCT** A **FROM** R

**WHERE** A **IN (SELECT** A **FROM** S**);**.

Áp dụng khi chỉ kiểm tra một thuộc tính hoặc một tập hợp các thuộc tính cụ thể. Toán tử = SOME là tương đương với IN.

* **Sử dụng Left/Right Outer Join Bao gồm (LJ(I)/RJ(I)):**

*R* ∩ *S* → **SELECT** R.\* **FROM** R

**LEFT JOIN** S **ON** R.Key = S.Key

**WHERE** S.Key **IS NOT NULL;**.

Phương pháp này còn được gọi là Left Join Including (LJ(I)). Cách này ánh xạ phép toán Giao (*I*) bằng cách sử dụng Left Join và chỉ giữ lại các bộ trong *R* mà tìm thấy đối sánh trong *S*, kiểm tra bằng điều kiện một thuộc tính khóa của *S* **IS NOT NULL**. Tuy nhiên, các nguồn tin nhấn mạnh rằng Inner Join (IJ) là phương pháp hiệu quả hơn để tìm Intersection

### D. Ví dụ chuyển đổi truy vấn tập hợp

Các phép toán tập hợp (∪,∩,∖) đòi hỏi các quan hệ tham gia phải **tương thích tập hợp** (*Union Compatibility*). Sự tương thích này yêu cầu chúng phải có cùng số lượng thuộc tính (arity) và các thuộc tính tương ứng phải có cùng miền giá trị (domain compatibility).

#### a. Ánh xạ Phép Hợp (∪) → UNION

Phép toán Hợp (*C*∪*N*) tìm ra tất cả các bản ghi tồn tại trong *C* hoặc *N*. Kết quả sẽ loại bỏ các bản ghi trùng lặp (trừ khi dùng ALL).

**Ví dụ D.1: Hợp hai bảng (Loại bỏ trùng lặp)**

Mục tiêu: Tìm danh sách tất cả các sản phẩm hiện có và sản phẩm mới (Discontinued + New + Common).

ĐSQH: *C* ∪ *N*

**SELECT \* FROM** C **UNION SELECT \* FROM** N;

*Lưu ý:* Nếu sử dụng UNION ALL, các bản sao sẽ được giữ lại.

ĐSQH (Multiset): *C* ∪all ​*N*

**SELECT \* FROM** C **UNION ALL SELECT \* FROM** N;

#### b. Ánh xạ Phép Giao (∩) → INTERSECT (và các phương pháp thay thế)

Phép toán Giao (*C* ∩ *N*) tìm các bản ghi sản phẩm/nhà cung cấp/chi phí **chung** (common records) có mặt trong cả *C* và *N*.

**Ví dụ D.2: Sử dụng INTERSECT (Chuẩn SQL)**

ĐSQH: *C* ∩ *N*

**SELECT \* FROM** C **INTERSECT SELECT \* FROM** N;

*Lưu ý:* Toán tử INTERSECT **tự động loại bỏ trùng lặp**. Tuy nhiên, **MySQL không hỗ trợ** toán tử này.

**Ví dụ D.3: Sử dụng INNER JOIN (IJ) (Phương pháp hiệu quả)**

Phép Giao có thể được ánh xạ bằng phép **Kết Trong (Inner Join)** trên *tất cả các thuộc tính* chung. Phương pháp này **thường hiệu quả hơn** so với việc sử dụng Outer Join có điều kiện.

ĐSQH: *C* ∩ *N* ≡ *πC*.∗​(*C*⋈all attributes​*N*)

**SELECT** C.\* **FROM** C

**INNER JOIN** N **ON** (C.PID = N.PID)

**AND** (C.SID = N.SID)

**AND** (C.Cost = N.Cost);

**Ví dụ D.4: Sử dụng EXISTS (Truy vấn con tương quan)**

Phương pháp này kiểm tra liệu có tồn tại bộ khớp hoàn toàn trong *N* cho mỗi bộ trong *C* hay không.

ĐSQH: *C* ∩ *N* ≡ *σ*∃*n*∈*N*(*C*=*n*)​(*C*)

**SELECT** \* **FROM C**

**WHERE EXISTS** (**SELECT** \* **FROM** N

**WHERE** (C.PID = N.PID)

**AND** (C.SID = N.SID)

**AND** (C.Cost = N.Cost););

Phương pháp này được coi là một truy vấn chuyên biệt (specialized query) và **không hiệu quả** bằng INNER JOIN (Q3).

**Ví dụ D.5: Sử dụng LEFT JOIN INCLUDING (LJ(I))**

Phép Giao (*I*) có thể được ánh xạ bằng Left Join Including (LJ(I)). Điều này đòi hỏi phải sử dụng điều kiện WHERE ... IS NOT NULL (Including Predicate) trên một thuộc tính khóa của bảng bên phải (*N*) để giữ lại các bản ghi khớp.

ĐSQH: *I* =LJ (*I*)

**SELECT** C.\* **FROM** C

**LEFT JOIN** N **ON** (C.PID = N.PID)

**AND** (C.SID = N.SID) **AND** (C.Cost = N.Cost)

**WHERE** (N.PID) **IS NOT NULL**;

#### c. Ánh xạ Phép Hiệu (∖) → EXCEPT (và các phương pháp thay thế)

Phép toán Hiệu (*C* ∖ *N*) tìm các bản ghi **chỉ có trong** *C* mà **không có** trong *N*. Trong bối cảnh sản phẩm, đây là các bản ghi đã ngừng sản xuất (Discontinued Records).

**Ví dụ D.6: Sử dụng EXCEPT/MINUS (Chuẩn SQL)**

ĐSQH: *C* ∖ *N*

**SELECT** \* **FROM** C **EXCEPT SELECT** \* **FROM** N;

*Lưu ý:* Oracle sử dụng từ khóa **MINUS** thay cho EXCEPT. Cả hai toán tử này đều yêu cầu **tính tương thích tập hợp**.

**Ví dụ D.7: Sử dụng LEFT JOIN EXCLUDING (LJ(E)) (Phương pháp thay thế phổ biến)**

Phép Hiệu *D*(*C* − *N*) được ánh xạ trực tiếp bằng cách sử dụng **Left Join Excluding (LJ(E))**. Phương pháp này giữ lại tất cả các bộ từ *C* không tìm thấy đối sánh trong *N*.

ĐSQH: *D*(*C* − *N*) = LJ(*E*)

**SELECT** C.\* **FROM** C

**LEFT JOIN** N **ON** (C.PID = N.PID)

**AND** (C.SID = N.SID) **AND** (C.Cost = N.Cost)

**WHERE** (N.PID) **IS NULL**;

*Lưu ý quan trọng:* Điều kiện WHERE (N.PID) Is Null là predicate "Excluding" (E). Ta **chỉ cần kiểm tra một thuộc tính khóa** (ví dụ: PID) của bảng bên phải (*N*) là đủ, vì nếu thuộc tính khóa này là NULL, thì toàn bộ bộ đó không có đối sánh.

**Ví dụ D.8: Tìm các sản phẩm mới (New Records) D(N-C)**

Phép toán Hiệu *D*(*N* − *C*) (các sản phẩm có trong *N* nhưng không có trong *C*, tức là Newly Available) là đối xứng của *D*(*C* − *N*) và được ánh xạ bằng **Right Join Excluding (RJ(E))**.

ĐSQH: *D*(*N* − *C*)=RJ(*E*)

**SELECT** N.\* **FROM** C

**RIGHT JOIN** N **ON** (C.PID = N.PID)

**AND** (C.SID = N.SID) **AND** (C.Cost = N.Cost)

**WHERE** (C.PID) **IS NULL**;

## 4.1.3. Ánh xạ tích đề-các

## 4.2. Ánh xạ các phép kết nối

## 4.2.1. Ánh xạ Natural Join

## 4.2.2. Ánh xạ Outer Join

## 4.3. Ánh xạ các toán tử nâng cao khác

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

# TÀI LIỆU THAM KHẢO

sách **Fundamentals of Database Systems** của **Elmasri & Navathe (6th edition) Chương 6**

**Tài liệu bạn cung cấp trên GeeksforGeeks có tiêu đề "Introduction of Relational Algebra in DBMS"**

**"Mapping relation algebra operators into SQL queries: A database case study"  
  
Unit 4 Relational Algebra (Using SQL DML Syntax): Data Manipulation Language For Relations  
  
Database System Concepts (Silberschatz, Korth, Sudarshan, 7th Edition (2020)), 1. Đại số Quan hệ Cơ bản (Basic Relational Algebra) nằm ở chapter 2, Đại số Quan hệ Nâng cao và Ngôn ngữ Truy vấn Khác Vị trí: Chương 27 (Online Chapter: Pure Query Languages)**

**A Practical Approach to Design, Implementation, and Management, elational Algebra cơ bản → Chapter 5, Ứng dụng nâng cao → Chapter 23 (Query Processing) & Chapter 24 (Distributed DBMSs)**