Chương 1: Các Thế Giới của Hệ Thống Cơ Sở Dữ Liệu

Mục tiêu

Hiểu các khái niệm:

- Thông tin (Information)
- Dữ liệu (Data)
- Cơ sở dữ liệu (Database)
- Hệ quản trị cơ sở dữ liệu (Database Management System - DBMS)
- Hệ thống cơ sở dữ liệu (Database System)

Nội dung

1. Sự phát triển của các hệ thống cơ sở dữ liệu
2. Tổng quan về hệ quản trị cơ sở dữ liệu

1.1 Sự phát triển của các hệ thống cơ sở dữ liệu

- **Dữ liệu (Data)**: Là các sự kiện thô, không có ý nghĩa ngữ cảnh (ví dụ: số, văn bản).
- **Thông tin (Information)**: ữ liệu đã được xử lý và tổ chức trong ngữ cảnh.D
- **Cơ sở dữ liệu (Database)**: Tập hợp thông tin tồn tại trong thời gian dài, được quản lý bởi DBMS. Thu thập dữ liệu hoặc thông tin có tổ chức có thể được truy cập, cập nhật và quản lý
- **Hệ quản trị cơ sở dữ liệu (Database Management System - DBMS)**: Phần mềm để tạo và duy trì cơ sở dữ liệu.
- **Hệ thống cơ sở dữ liệu (Database System)**: Gồm DBMS và dữ liệu.

1.1.1 Các mô hình dữ liệu ban đầu

- **Mô hình dữ liệu phân cấp (Hierarchical Data Model)**: Sử dụng trong các DBMS mainframe ban đầu như IMS của IBM.Nó là cấu trúc cây
- **Mô hình dữ liệu mạng (Network Data Model)**: Do Charles Bachman phát triển, cho phép mỗi bản ghi có nhiều bản ghi

cha và con.Nhược điểm là Không hỗ trợ ngôn ngữ truy vấn cấp cao

- **Hệ thống cơ sở dữ liệu quan hệ (Relational Database Systems)**: Được định nghĩa bởi Edgar F. Codd, Là tập hợp 2 hay nhiều bảng được liên kết bởi các mối quan hệ. SQL trở thành ngôn ngữ truy vấn quan trọng nhất, với sản phẩm thương mại đầu tiên là Oracle v.2.
- **Mô hình dữ liệu đối tượng (Object-Oriented Data Model)** là một mô hình kết hợp giữa các khái niệm của **lập trình hướng đối tượng (OOP)** và cơ sở dữ liệu, nơi dữ liệu được lưu trữ dưới dạng **đối tượng** thay vì các bảng hoặc bản ghi như trong các mô hình quan hệ truyền thống. Mô hình này cung cấp cách thức lưu trữ và truy vấn dữ liệu phức tạp, chẳng hạn như hình ảnh, âm thanh, hoặc các đối tượng 3D, bằng cách sử dụng các đối tượng.

1.1.2 Sự phát triển đến các hệ thống NoSQL và newSQL

- Các hệ thống DBMS ban đầu chạy trên các máy tính lớn và đắt tiền.
- Hiện nay, DBMS có thể chạy trên PC và thiết bị di động.
- Kích thước dữ liệu ngày càng lớn, nhiều cơ sở dữ liệu lưu trữ petabyte dữ liệu.

1.1.3 Tích hợp thông tin (Information Integration)

- **Kho dữ liệu (Data Warehouse)**: Sao chép thông tin từ nhiều cơ sở dữ liệu vào một cơ sở dữ liệu trung tâm.
- **Middleware**: Hỗ trợ mô hình tích hợp dữ liệu từ nhiều cơ sở dữ liệu khác nhau.

1.2 Tổng quan về hệ quản trị cơ sở dữ liệu (DBMS)

- **Thành phần của DBMS**: Bao gồm các cấu trúc dữ liệu và dòng dữ liệu.
- **Người dùng cơ sở dữ liệu**:

    - Quản trị viên (Database Administrator): Cấp quyền truy cập và giám sát sử dụng.
    - Nhà thiết kế (Database Designer): Xác định nội dung và cấu trúc cơ sở dữ liệu.
    - Người dùng cuối (End User): Sử dụng dữ liệu cho các truy vấn và báo cáo.

Xu hướng thiết kế cơ sở dữ liệu và DBMS

- Cơ sở dữ liệu không quan hệ (NoSQL): Ví dụ như MongoDB, Redis.
- Cơ sở dữ liệu đa mô hình (Multi-model Databases): Ví dụ như Oracle, ArangoDB.

Chương 2: Mô Hình Quan Hệ của Dữ Liệu

Mục tiêu

- Hiểu mô hình quan hệ (Relational Model) và thiết kế cơ sở dữ liệu dựa trên mô hình quan hệ.
- Khái niệm hóa dữ liệu sử dụng mô hình quan hệ.
- Hiểu các phép toán cơ bản của đại số quan hệ (Relational Algebra).
- Biểu diễn truy vấn sử dụng đại số quan hệ.

Nội dung

1. Tổng quan về các mô hình dữ liệu
2. Cơ bản về mô hình quan hệ
3. Ngôn ngữ truy vấn đại số

2.1 Tổng quan về các mô hình dữ liệu

- **Mô hình dữ liệu (Data Model)**: Tập hợp các khái niệm để mô tả dữ liệu gồm cấu trúc, các thao tác và các ràng buộc trên dữ liệu.
- **Mô hình dữ liệu bán cấu trúc (Semi-structured Data Model)**: Gồm XML và các tiêu chuẩn liên quan.

2.2 Cơ bản về mô hình quan hệ

- **Mô hình quan hệ (Relational Model)**: Gồm hai phần:
  - **Schema**: Tên quan hệ, tên thuộc tính và kiểu dữ liệu.
  - **Instance**: Một bảng với các hàng và cột.
- **Schema cơ sở dữ liệu (Database Schema)**: Tập hợp các schema của các quan hệ trong cơ sở dữ liệu.
- **Thuộc tính (Attribute)**: Có thể là khóa chính, khóa ngoại, thuộc tính nhiều giá trị, thuộc tính dẫn xuất.

Derived columns(thuộc tính dẫn xuất**):** là các cột được tính toán dựa trên các cột hoặc biểu thức khác, chẳng hạn như sử dụng các hàm hoặc phép toán số học.

2.3 Ngôn ngữ truy vấn đại số

- **Đại số quan hệ (Relational Algebra)**: Tập hợp các phép toán trên quan hệ, gồm:
  - **Phép toán tập hợp (Set Operations)**: Union, Intersection, Difference.
  - **Lựa chọn (Selection) và chiếu (Projection)**: Chọn các điều kiện và chiếu các thuộc tính.
  - **Phép nhân Cartesian (Cartesian Product) và phép nối (Joins)**: Tạo ra quan hệ mới từ các quan hệ ban đầu.
  - **Đổi tên (Rename)**: Đổi tên các thuộc tính và quan hệ.

Các ví dụ về phép toán đại số quan hệ

- **Lựa chọn**: σ<điều kiện>(R)
- **Chiếu**: π<danh sách thuộc tính>(R)
- **Phép nối tự nhiên (Natural Join)**: $R \bowtie S$
- **Biểu thức quan hệ (Relational Expression)**: Xây dựng bằng cách áp dụng các phép toán lên kết quả của các phép toán khác.

Chương 3: Lý Thuyết Thiết Kế cho Cơ Sở Dữ Liệu Quan Hệ

Mục tiêu

Hiểu các khái niệm về:

- Phụ thuộc hàm (Functional Dependencies)
- Chuẩn hóa (Normalization)
- Phân rã (Decomposition)
- Phụ thuộc đa trị (Multi-valued Dependencies)

Nội dung

1. Phụ thuộc hàm
2. Các quy tắc về phụ thuộc hàm
3. Khóa và Siêu khóa
4. Các dạng chuẩn
5. Các lỗi dị thường
6. Các lỗi khác

3.1 Phụ thuộc hàm

- **Phụ thuộc hàm (Functional Dependency)**: Ràng buộc giữa hai tập thuộc tính trong một quan hệ.
- **Xác định phụ thuộc hàm**: Một tập thuộc tính X trong R quyết định một thuộc tính Y trong R (viết là $X \rightarrow Y$) nếu mỗi giá trị của X liên kết với chính xác một giá trị của Y. Lúc này Y sẽ phụ thuộc vào X.
- **Phụ thuộc hàm đúng (True Functional Dependency)**: Nếu hai bộ của R đồng ý về tất cả các thuộc tính trong X thì chúng cũng phải đồng ý về tất cả các thuộc tính trong Y.
- **Phụ thuộc hàm toàn phần (Full Functional Dependency)**: Trong một quan hệ, một thuộc tính Y được gọi là phụ thuộc toàn phần vào một tập thuộc tính X nếu Y phụ thuộc vào toàn bộ tập X và không phụ thuộc vào bất kỳ tập con đúng nào của X. Điều này có nghĩa là nếu bạn xóa bất kỳ thuộc tính nào từ X, phụ thuộc hàm sẽ không còn đúng.

- **Closure** của một tập hợp các thuộc tính (hay còn gọi là các thuộc tính ban đầu) là tập hợp tất cả các thuộc tính mà có thể được suy ra (hoặc xác định) từ tập hợp ban đầu đó thông qua các phụ thuộc hàm.

3.2 Các quy tắc về phụ thuộc hàm

- **Quy tắc của Armstrong (Armstrong's Axioms)**:
  - **Phản xạ (Reflexivity)**: Nếu X là tập con của Y thì Y → X.
  - **Bổ sung (Augmentation)**: Nếu X → Y thì XZ → YZ với mọi Z.
  - **Chuyển tiếp (Transitivity)**: Nếu X → Y và Y → Z thì X → Z.
  - **Kết hợp (Union)**: Nếu X → Y và X → Z thì X → YZ.
  - **Phân rã (Decomposition)**: Nếu X → YZ thì X → Y và X → Z.
  - **Giả chuyển tiếp (Pseudotransitivity)**: Nếu X → Y và WY → Z thì WX → Z.
  - **Phụ thuộc tầm thường (Trivial Functional Dependency)**: Nếu vế phải là tập con của vế trái.

3.3 Khóa và Siêu khóa

- **Khóa (Key)**: Một tập thuộc tính xác định duy nhất các bộ trong một quan hệ.
- **Siêu khóa (Super Key)**: Một tập thuộc tính chứa khóa và thỏa mãn điều kiện xác định duy nhất tất cả các thuộc tính khác của quan hệ.

3.4 Các dạng chuẩn hóa

Chuẩn hóa cơ sở dữ liệu(Database Normalization) là quá trình sắp xếp các thuộc tính và bảng của cơ sở dữ liệu để giảm thiểu sự trùng lặp và phụ thuộc.

- **1NF (First Normal Form)**: Một quan hệ **R** là ở dạng **1NF** nếu mọi giá trị trong các thuộc tính của quan hệ là **nguyên tử**, tức là không có tập hợp, không có mảng hay giá trị đa trị trong bất kỳ thuộc tính nào.
- **2NF (Second Normal Form)**: Quan hệ R là 2NF nếu nó là 1NF. Mọi thuộc tính không khóa phụ thuộc hoàn toàn vào khóa chính (không có phụ thuộc từng phần, tức là không có sự phụ thuộc chỉ vào một phần của khóa chính composite).
- **3NF (Third Normal Form)**: Quan hệ R là 3NF nếu nó là 2NF. Mọi thuộc tính không khóa không phụ thuộc vào nhau qua một thuộc tính trung gian (không có **phụ thuộc bắc cầu** hoặc **transitive dependency**).
- **BCNF (Boyce-Codd Normal Form)**: Quan hệ R là BCNF nếu mọi phụ thuộc hàm không tầm thường $X \rightarrow Y$ thì X là siêu khóa.Nếu một phụ thuộc hàm là BCNF thì nó cũng thuộc dạng chuẩn hóa 1NF,2NF và 3NF

3.5 Các Lỗi Dị Thường (Anomalies)

- **Insertion Anomaly (Lỗi chèn)**: Xảy ra khi không thể chèn dữ liệu mới vào cơ sở dữ liệu mà không cần thêm dữ liệu không liên quan(dữ liệu đơn lẻ)
- **Update Anomaly (Lỗi cập nhật)**: Xảy ra khi việc cập nhật một phần dữ liệu yêu cầu cập nhật nhiều hàng để giữ cho dữ liệu đồng nhất.
- **Deletion Anomaly (Lỗi xóa)**: Xảy ra khi việc xóa dữ liệu cũng dẫn đến việc mất dữ liệu có liên quan không mong muốn.

3.6 Các Lỗi Khác (Other Errors)

- **Redundancy (Dư thừa dữ liệu)**: Xảy ra khi cùng một dữ liệu được lưu trữ ở nhiều nơi trong cơ sở dữ liệu.
- **Integrity Violation (Vi phạm toàn vẹn dữ liệu)**: Xảy ra khi dữ liệu không tuân thủ các ràng buộc toàn vẹn.

- **Deadlock (Tắc nghẽn)**: Xảy ra khi hai hoặc nhiều giao dịch giữ các khóa trên các tài nguyên và chờ đợi lẫn nhau để giải phóng các khóa đó.
- **Concurrency Issues (Vấn đề đồng thời)**: Xảy ra khi nhiều người dùng hoặc nhiều giao dịch cố gắng truy cập và sửa đổi cùng một dữ liệu đồng thời.
- **Data Corruption (Hỏng dữ liệu)**: Xảy ra khi dữ liệu bị hỏng do các vấn đề phần cứng, phần mềm hoặc lỗi hệ thống.
- **Performance Issues (Vấn đề hiệu suất)**: Xảy ra khi cơ sở dữ liệu hoạt động không hiệu quả, dẫn đến thời gian phản hồi chậm hoặc tắc nghẽn hệ thống.

Chương 4: Mô Hình Cơ Sở Dữ Liệu Cấp Cao

4.1 Quy trình thiết kế cơ sở dữ liệu

- **Phân tích yêu cầu (Requirements Analysis)**: Xác định nhu cầu của người dùng và những gì cơ sở dữ liệu phải làm.
- **Thiết kế khái niệm (Conceptual Design)**: Mô tả cấp cao (sử dụng sơ đồ ER).Mô hình khái niệm(conceptual model) trong cơ sở dữ liệu độc lập với cả phần cứng và phần mềm
- **Thiết kế logic (Logical Design)**: Chuyển đổi ERD thành mô hình quan hệ của DBMS để làm giảm sự bất thường khi cập nhật. Khi chuyển thì không cần quan trọng thứ tự các cột trong thực thể ở ERD
- **Tinh chỉnh schema (Schema Refinement)**: Đảm bảo tính nhất quán và chuẩn hóa.
- **Thiết kế vật lý (Physical Design)**: Bố trí đĩa và chỉ mục.
- **Thiết kế bảo mật (Security Design)**: Xác định quyền truy cập dữ liệu.

4.2 Mô hình thực thể - mối quan hệ (ER model)

- **Thực thể (Entity)**: Đối tượng thực tế phân biệt được với các đối tượng khác, được mô tả bằng tập các thuộc tính.

- **Tập thực thể (Entity Set)**: Tập hợp các thực thể tương tự.
- **Mối quan hệ (Relationship)**: Liên kết giữa hai hoặc nhiều thực thể, có thể có thuộc tính riêng.
- **Sơ đồ ER (ER Diagram)**: Sử dụng các ký hiệu để biểu diễn thực thể, thuộc tính và mối quan hệ của chúng.

4.3 Sơ đồ thực thể - mối quan hệ (ERD)

Các ký hiệu của ERD (ERD Notations)

1. **Thực thể (Entity)**
   - Biểu diễn các đối tượng thực tế trong hệ thống.
   - Ký hiệu: Hình chữ nhật.
   - Ví dụ: Sinh viên, Khóa học.

Thực thể yếu (Weak Entity) là loại thực thể không thể được nhận dạng duy nhất chỉ bằng các thuộc tính riêng của nó. Nó phụ thuộc vào một thực thể khác để có thể được nhận dạng duy nhất.

2. **Thuộc tính (Attribute)**
   - Biểu diễn các đặc tính hoặc thuộc tính của thực thể.
   - Ký hiệu: Hình elip.
   - Ví dụ: Tên, Tuổi, Địa chỉ.
3. **Mối quan hệ (Relationship)**
   - Biểu diễn sự liên kết giữa các thực thể.
   - Ký hiệu: Hình thoi.
   - Ví dụ: Đăng ký, Dạy.

3 thành phần trên là thành phần chính cấu tạo nên ERD . Ngoài ra trong ERD còn có thêm một số thành phần bên dưới:

4. **Mối quan hệ yếu (Weak Relationship)**
   - Mối quan hệ giữa thực thể yếu và thực thể mạnh.
   - Ký hiệu: Hình thoi đôi.
   - Ví dụ: Thuộc tính phụ thuộc vào thuộc tính của thực thể khác.
5. **Lớp con (Subclass)**

- Biểu diễn phân cấp kế thừa giữa các thực thể.
- Ký hiệu: Đường thẳng nối các thực thể và một đường thẳng dọc xuống để phân cấp.
- Ví dụ: Nhân viên có thể là Quản lý hoặc Nhân viên kỹ thuật.

4.4 Thuộc tính của thực thể (Attributes of Entity)

1. **Thuộc tính khóa (Key Attribute)**
   - Thuộc tính duy nhất để nhận diện một thực thể.
   - Ký hiệu: Hình elip với gạch dưới.
   - Ví dụ: Mã sinh viên, Số chứng minh nhân dân.
2. **Thuộc tính đa trị (Multivalued Attribute)**
   - Thuộc tính có thể có nhiều giá trị.
   - Ký hiệu: Hình elip đôi.
   - Ví dụ: Số điện thoại, Email.
3. **Thuộc tính dẫn xuất (Derived Attribute)**
   - Thuộc tính có giá trị được tính toán từ các thuộc tính khác.
   - Ký hiệu: Hình elip đứt đoạn.
   - Ví dụ: Tuổi (tính từ ngày sinh), Tổng số tín chỉ đã học.
4. **Thuộc tính tổng hợp (Composite Attribute)**
   - Thuộc tính bao gồm nhiều thành phần nhỏ hơn.
   - Ký hiệu: Hình elip lớn bao gồm các hình elip nhỏ.
   - Ví dụ: Địa chỉ (bao gồm Số nhà, Đường, Quận/Huyện, Thành phố).

4.5 Mối quan hệ đa trị (Multivalued Relationships)

1. **Định nghĩa**
   - Mối quan hệ trong đó một thực thể có thể liên kết với nhiều giá trị của một thực thể khác.
   - Ký hiệu: Hình thoi nối với một hình elip đôi.
2. **Ví dụ**
   - Một sinh viên (Thực thể Sinh viên) có thể đăng ký nhiều khóa học (Mối quan hệ Đăng ký) và mỗi khóa học có thể có nhiều sinh viên đăng ký.

4.5 Các bước trong thiết kế cơ sở dữ liệu

1. Thu thập dữ liệu cần mô hình hóa.
2. Xác định các dữ liệu có thể mô hình hóa thành các thực thể trong thế giới thực.
3. Xác định các thuộc tính cho từng thực thể.
4. Phân loại tập thực thể thành tập thực thể yếu hoặc mạnh.
5. Phân loại thuộc tính của thực thể thành thuộc tính khóa, thuộc tính đa trị, thuộc tính tổng hợp, thuộc tính dẫn xuất.
6. Xác định mối quan hệ giữa các thực thể khác nhau.
7. Sử dụng các ký hiệu phù hợp để vẽ các thực thể, thuộc tính và mối quan hệ của chúng.

4.6 Bảng (Table)

4.6.1 Khái niệm:

- Bảng là cấu trúc lưu trữ dữ liệu chính trong cơ sở dữ liệu quan hệ. Mỗi bảng chứa các hàng (rows) và cột (columns), và mỗi hàng đại diện cho một bản ghi dữ liệu.

4.6.2 Cấu trúc của bảng:

- Cột (Columns): Đại diện cho các thuộc tính của dữ liệu.
- Hàng (Rows): Đại diện cho các bản ghi dữ liệu.

4.6.3 Các thành phần của bảng:

1. Tên bảng (Table Name): Tên của bảng phải là duy nhất trong cơ sở dữ liệu.
2. Cột (Columns): Mỗi cột trong bảng có một tên và một kiểu dữ liệu.

3. Khóa chính (Primary Key): Một thuộc tính hoặc tập hợp các thuộc tính xác định duy nhất mỗi hàng trong bảng.
4. Khóa ngoại (Foreign Key): Một thuộc tính hoặc tập hợp các thuộc tính trong bảng này liên kết với khóa chính của bảng khác.Để tạm thời vô hiệu hóa ràng buộc khóa ngoại trong các thao tác INSERT và UPDATE trong SQL, bạn có thể sử dụng câu lệnh ALTER TABLE với tùy chọn NOCHECK CONSTRAINT

   VD: ALTER TABLE B NOCHECK CONSTRAINT fk_nam

5. Các ràng buộc (Constraints): Bao gồm NOT NULL, UNIQUE, CHECK, và DEFAULT.

Chương 5: Ngôn Ngữ Truy Vấn Đại Số (Algebraic Query Language)

Mục tiêu

Hiểu lý do sử dụng khái niệm Bag (multi-set). Biết các phép toán quan hệ trên bag. Biết các phép toán mở rộng trên bag.

Nội dung

1. Quan hệ dạng bag
2. Các phép toán quan hệ trên bag
3. Các phép toán mở rộng trên bag

5.1 Quan hệ dạng bag

- **Bag** (multi-set): Cho phép cùng một bộ (tuple) xuất hiện nhiều lần trong một quan hệ.
- **Lý do sử dụng bag**: Một số phép toán quan hệ hiệu quả hơn khi sử dụng mô hình bag như phép union và projection.

5.2 Các phép toán quan hệ trên bag

- **Union**: Trong {R ∪ S}, bộ t xuất hiện (n + m) lần.

- **Intersection**: Trong {R ∩ S}, bộ t xuất hiện MIN(n, m) lần.
- **Difference**: Trong {R \ S}, bộ t xuất hiện MAX(0, n - m) lần.

Ví dụ về phép toán

- **Union**: Kết hợp các bộ từ hai quan hệ.
- **Intersection**: Lấy các bộ chung từ hai quan hệ.
- **Difference**: Lấy các bộ có trong quan hệ thứ nhất mà không có trong quan hệ thứ hai.

5.3 Các phép toán mở rộng trên bag

- **Duplicate Elimination (Loại bỏ trùng lặp)**: Sử dụng ký hiệu d(R).
- **Aggregation Operators (Phép toán tổng hợp)**: SUM, AVG, MIN, MAX, COUNT.
- **Grouping Operator (Phép toán nhóm)**: Sử dụng ký hiệu γL(R), nhóm các bộ theo các thuộc tính trong danh sách L.
- **Extended Projection (Phép chiếu mở rộng)**: Ký hiệu pL(R), có thể bao gồm các biểu thức và đổi tên thuộc tính.
- **Sorting Operator (Phép toán sắp xếp)**: Ký hiệu tL(R), sắp xếp các bộ theo các thuộc tính trong danh sách L.
- **Outer Join (Phép toán kết hợp ngoài)**: Các biến thể gồm Left Outer Join, Right Outer Join, Full Outer Join.

Chương 6: Ngôn Ngữ Cơ Sở Dữ Liệu SQL

Mục tiêu

- Sinh viên có thể viết script SQL.
- Sinh viên có thể soạn các truy vấn SQL sử dụng các toán tử tập hợp và bag, truy vấn con có liên quan, truy vấn tổng hợp.
- Sinh viên có thể thao tác thành thạo trên các truy vấn phức tạp.

Nội dung

1. Ràng buộc toàn vẹn
2. Ngôn ngữ truy vấn cấu trúc (SQL)
3. Truy vấn con (Subquery)

6.1 Ràng buộc toàn vẹn (Integrity Constraints)

- **Mục đích**: Ngăn chặn các bất đồng ngữ nghĩa trong dữ liệu.

1. Ràng buộc khóa (Key Constraints)

- **Khóa chính (Primary Key)**
    - Định nghĩa: Một thuộc tính hoặc tập hợp các thuộc tính xác định duy nhất mỗi bản ghi trong bảng.
    - Đặc điểm: Giá trị của khóa chính phải là duy nhất và không được NULL.
    - Ví dụ: PRIMARY KEY (MaSinhVien)

- **Khóa ứng viên (Candidate Key)**
    - Định nghĩa: Là một hoặc nhiều thuộc tính trong bảng mà mỗi thuộc tính đều có thể được sử dụng như khóa chính.
    - Đặc điểm: Giá trị của khóa ứng viên cũng phải là duy nhất và không được NULL.
    - Ví dụ: UNIQUE (SoCMND)

2. Ràng buộc thuộc tính (Attribute Constraints)

- **NULL/NOT NULL**
    - Định nghĩa: Xác định xem một thuộc tính có thể chấp nhận giá trị NULL hay không.
    - NULL: Cho phép thuộc tính nhận giá trị NULL.
    - NOT NULL: Không cho phép thuộc tính nhận giá trị NULL.
    - Ví dụ: TenSinhVien VARCHAR(50) NOT NULL
- **CHECK**

    - Định nghĩa: Đảm bảo rằng các giá trị của một thuộc tính thỏa mãn một điều kiện nhất định.
    - Đặc điểm: Có thể áp dụng trên một hoặc nhiều thuộc tính.
    - Ví dụ: CHECK (Tuoi >= 18)
- **DEFAULT**
    - Định nghĩa: Đặt giá trị mặc định cho một thuộc tính nếu không có giá trị nào được cung cấp khi chèn dữ liệu.
    - Đặc điểm: Giá trị mặc định được gán tự động nếu không có giá trị nào khác được chỉ định.
    - Ví dụ: Luong DECIMAL(10, 2) DEFAULT 5000.00
- **UNIQUE KEY**
    - **Định nghĩa**: Là ràng buộc đảm bảo rằng giá trị trong một hoặc nhiều thuộc tính của bảng là duy nhất, không được trùng lặp với bất kỳ giá trị nào trong cùng một bảng. Một bảng có thể có nhiều khóa ứng viên, nhưng chỉ có một khóa chính.
    - **Đặc điểm**: Giá trị của khóa ứng viên phải là duy nhất và có thể được NULL.

3. Ràng buộc toàn vẹn tham chiếu (Referential Integrity Constraints)

- **Khóa ngoại (Foreign Key)**
    - Định nghĩa: Một thuộc tính hoặc tập hợp các thuộc tính trong bảng này liên kết với khóa chính của bảng khác.
    - Đặc điểm: Đảm bảo rằng mỗi giá trị trong khóa ngoại phải tồn tại trong bảng mà nó tham chiếu.
    - Ví dụ: FOREIGN KEY (MaLop) REFERENCES Lop(MaLop)

4. Ràng buộc toàn cục (Global Constraints)

- **CHECK hoặc CREATE ASSERTION**

- Định nghĩa: Ràng buộc toàn cục là các ràng buộc áp dụng trên toàn bộ cơ sở dữ liệu, không chỉ trên một bảng duy nhất.
- **CHECK**: Tương tự như ràng buộc thuộc tính, nhưng có thể áp dụng trên nhiều bảng.
- **CREATE ASSERTION**: Được sử dụng để định nghĩa các ràng buộc phức tạp hơn.
- Ví dụ về CHECK toàn cục: CHECK (Luong > 0 AND Luong < 1000000)
- Ví dụ về CREATE ASSERTION:

```
CREATE ASSERTION valid_salary
CHECK (NOT EXISTS (
  SELECT *
  FROM NhanVien
  WHERE Luong < 0 OR Luong > 1000000
));
```

6.2 Ngôn ngữ Truy vấn Cấu trúc (SQL)

SQL (Structured Query Language) là ngôn ngữ tiêu chuẩn để tương tác với cơ sở dữ liệu quan hệ. SQL bao gồm các nhóm lệnh chính, mỗi nhóm phục vụ một chức năng cụ thể trong quản lý và thao tác dữ liệu

1. Ngôn ngữ định nghĩa dữ liệu (DDL)

- **Lệnh CREATE**: Tạo cơ sở dữ liệu, bảng.
- **Lệnh ALTER**: Thay đổi cấu trúc bảng.
- **Lệnh DROP**: Xóa bảng hoặc cơ sở dữ liệu (toàn bộ cấu trúc và dữ liệu đều bị xóa) , trigger, function, procedure.
- **TRUNCATE**: Xóa tất cả dữ liệu trong bảng nhưng giữ lại cấu trúc.

2. Ngôn ngữ Truy vấn Dữ liệu (DQL) cái này có thể gộp với DML

- **Lệnh SELECT**: Truy vấn dữ liệu từ bảng.Có thể hoạt động trên một hoặc nhiều bảng.

3. Ngôn ngữ thao tác dữ liệu (DML)

- **Lệnh INSERT**: Thêm dữ liệu vào bảng.Hoạt động trên một bảng đơn.
- **Lệnh UPDATE**: Cập nhật dữ liệu trong bảng.Hoạt động trên một bảng đơn, nhưng có thể sử dụng thông tin từ các bảng khác.
- **Lệnh DELETE**: Xóa dữ liệu từ bảng.Hoạt động trên một bảng đơn, nhưng có thể sử dụng thông tin từ các bảng khác.

4. Ngôn ngữ thao tác dữ liệu (DML)

- **GRANT**: Cấp quyền truy cập cho người dùng hoặc vai trò.
- **REVOKE**: Thu hồi quyền truy cập từ người dùng hoặc vai trò.

5. Ngôn ngữ Kiểm soát Giao dịch (TCL)

- **COMMIT**: Lưu vĩnh viễn các thay đổi trong một giao dịch.
- **ROLLBACK**: Hủy bỏ các thay đổi chưa lưu trong một giao dịch.
- **SAVEPOINT**: Đặt điểm đánh dấu trong một giao dịch để có thể quay lại khi cần.

6.3 Truy vấn con (Subquery)

- **Truy vấn con**: Một truy vấn được sử dụng để hỗ trợ trong việc đánh giá một truy vấn khác.
- **Các loại truy vấn con**:
    - Truy vấn con trả về một giá trị.
    - Truy vấn con trả về nhiều giá trị.
    - Truy vấn con tương quan (correlated subquery): Được đánh giá nhiều lần.
    - Truy vấn con trong mệnh đề FROM: Sử dụng làm biến bộ.

6.5 Các phép toán trên toàn bộ quan hệ (Full-Relation Operations)

- **Loại bỏ trùng lặp (Eliminating Duplicates)**: Sử dụng từ khóa DISTINCT.
- **Phép toán tập hợp (Set Operations)**: UNION, INTERSECT, EXCEPT.
- **Phép toán nhóm và tổng hợp (Grouping and Aggregation)**: Sử dụng GROUP BY và các toán tử tổng hợp như SUM, AVG, MIN, MAX, COUNT.
- **Mệnh đề HAVING**: Được sử dụng để áp dụng các điều kiện trên các nhóm sau khi nhóm bằng GROUP BY

WHERE là điều kiện của SELECT CÒN HAVING là điều kiện của GROUP BY

Chương 7: Các Vấn Đề Thực Tế của Ứng Dụng Cơ Sở Dữ Liệu

Mục tiêu

Hiểu các khái niệm về:

- Giao dịch và các tính chất của chúng (ACID)
- Áp dụng giao dịch trong lập trình ứng dụng cơ sở dữ liệu
- Vai trò của các kỹ thuật đánh chỉ mục (indexing techniques)
- Triển khai các chỉ mục cho tối ưu hóa truy vấn
- Hiểu khái niệm và cách sử dụng views
- Hiểu kế hoạch thực thi truy vấn để phân tích tối ưu hóa truy vấn

Nội dung

1. Giao dịch trong SQL
2. Chỉ mục trong SQL và tối ưu hóa truy vấn
3. Views

7.1 Giao dịch trong SQL

- **Giao dịch (Transaction)**: Một nhóm các thao tác cần được thực hiện cùng nhau. Có thể tồn tại được trong Store Procedure nhưng không thể tồn tại trong Functions

- Trong ngữ cảnh của các **giao dịch trong cơ sở dữ liệu** (database transactions), khi sử dụng đối tượng **Connection** (ví dụ như trong ADO.NET hoặc các giao diện cơ sở dữ liệu khác), có một số phương thức được sử dụng để quản lý vòng đời của giao dịch. Các phương thức này bao gồm:
    - **Begin Transaction (Bắt đầu giao dịch)**: Phương thức này bắt đầu một giao dịch mới. Mọi thay đổi được thực hiện sau khi bắt đầu giao dịch sẽ không được áp dụng vĩnh viễn vào cơ sở dữ liệu cho đến khi giao dịch được cam kết.
    - **Commit Transaction (Cam kết giao dịch)**: Phương thức này cam kết giao dịch, có nghĩa là tất cả các thay đổi đã thực hiện trong giao dịch sẽ được lưu vào cơ sở dữ liệu vĩnh viễn.
    - **Rollback Transaction (Hủy bỏ giao dịch)**: Phương thức này hủy bỏ giao dịch, đồng thời hoàn tác tất cả các thay đổi đã thực hiện trong giao dịch.
- **ACID**: Các tính chất của giao dịch:
    - **Atomicity (Tính nguyên tử)**: Giao dịch phải được thực hiện toàn bộ hoặc không thực hiện gì cả.
    - **Consistency (Tính nhất quán)**: Giao dịch phải duy trì tính nhất quán của cơ sở dữ liệu.
    - **Isolation (Tính cô lập)**: Giao dịch phải được thực hiện như thể nó là giao dịch duy nhất trên hệ thống.
    - **Durability (Tính bền vững)**: Các thay đổi được thực hiện bởi giao dịch phải được duy trì ngay cả khi có sự cố.

Ví dụ về giao dịch

- **Chuyển tiền giữa hai tài khoản**:
    - Trừ 500$ từ tài khoản A.
    - Cộng 500$ vào tài khoản B.
    - Nếu có lỗi xảy ra sau bước 1 nhưng trước bước 2, cần phải hoàn nguyên lại bước 1.

7.2 Chỉ mục trong SQL và tối ưu hóa truy vấn

❖ **Chỉ mục (Index)**: Cấu trúc dữ liệu giúp tìm kiếm các bộ có giá trị cố định cho một thuộc tính A hiệu quả.

Đặc điểm và Lợi ích của Chỉ Mục:

❖ **Chỉ mục là một cấu trúc dữ liệu (Index is a data structure)**
  ➢ Chỉ mục được sử dụng để tăng tốc độ truy vấn dữ liệu bằng cách cung cấp một cách nhanh chóng để tìm các hàng dựa trên các giá trị cột.

❖ **Chỉ mục có thể được tạo trên bất kỳ thuộc tính nào (Index can be created on any attribute)**
  ➢ Không nhất thiết phải là khóa chính của bảng. Nó có thể là bất kỳ thuộc tính nào hoặc tập hợp các thuộc tính nào.

❖ **Chỉ mục thường được triển khai như cây tìm kiếm nhị phân (Index as a binary search tree)**
  ➢ Một trong những cách triển khai phổ biến của chỉ mục là cây tìm kiếm nhị phân (B-tree), trong đó mỗi cặp khóa và vị trí (key, location) liên kết với một tập các vị trí của các bộ.

Nhược điểm

Tốn kém không gian lưu trữ: Nhiều chỉ mục sẽ tiêu tốn nhiều không gian lưu trữ.

Giảm hiệu suất ghi: Chèn, cập nhật và xóa dữ liệu sẽ chậm hơn do phải cập nhật nhiều chỉ mục.

Tốn kém hiệu suất duy trì: Bảo trì các chỉ mục đòi hỏi nhiều tài nguyên và thời gian.

Tăng độ phức tạp của quản lý: Việc quản lý nhiều chỉ mục trở nên phức tạp hơn.

Tác động tiêu cực đến tối ưu hóa truy vấn: Bộ tối ưu hóa truy vấn có thể gặp khó khăn trong việc chọn chỉ mục tốt nhất.

❖ **Các loại chỉ mục**:
  ➢ Chỉ mục có cụm (Clustered Index)

➢ Chỉ mục không có cụm (Non-clustered Index)

```
CREATE CLUSTERED INDEX index_name ON
dbo.Tablename(ColumnName1, ColumnName2...);
CREATE NONCLUSTERED INDEX index_name ON
dbo.Tablename(ColumnName1, ColumnName2...);
DROP INDEX index_name;
```

7.3 Views

- **Views**: Một mối quan hệ ảo được định nghĩa bởi một biểu thức như truy vấn SQL.
- **Các loại views**:
    - Views đơn giản (Simple views): Có thể INSERT, UPDATE, DELETE thông qua view.
    - Views phức tạp (Complex views): Không thể cập nhật trực tiếp, chỉ có thể đọc.

```
CREATE VIEW view_name AS SELECT * FROM tblEmployee
WHERE depNum=1;
```

Update View

- **View phải dựa trên một bảng duy nhất**: Không thể cập nhật view nếu nó được tạo từ nhiều bảng.
- **View phải bao gồm khóa chính của bảng gốc**: Đảm bảo mỗi hàng trong view có thể được xác định duy nhất để cập nhật.
- **Không sử dụng các hàm tổng hợp**: View không được có các hàm như SUM(), AVG(), v.v. vì chúng làm mất đi sự liên kết giữa các hàng và bảng gốc.

- **Không sử dụng DISTINCT**: DISTINCT có thể loại bỏ các bản sao, gây khó khăn trong việc xác định hàng cần cập nhật.
- **Không sử dụng GROUP BY hoặc HAVING**: Các câu lệnh này làm việc với các nhóm dữ liệu, không thể xác định hàng cần cập nhật.
- **Không có các subquery trong định nghĩa của view**: Subqueries gây phức tạp và không xác định rõ cách cập nhật dữ liệu.
- **Nếu view dựa trên view khác, view gốc phải có thể cập nhật được**: View phụ thuộc vào view khác chỉ có thể cập nhật nếu view gốc cũng có thể cập nhật.
- **Không sử dụng hằng số, chuỗi hoặc biểu thức trong các trường đầu ra của view**: Các trường trong view phải là các cột trực tiếp từ bảng gốc để dễ dàng cập nhật.

Chương 8: Lập Trình Cơ Sở Dữ Liệu trên SQL Server

Mục tiêu

- Hiểu khái niệm và cách sử dụng triggers, stored-procedure, cursors, và functions
- Hiểu sự khác biệt giữa lập trình T-SQL và các ngôn ngữ lập trình khác
- Hiểu lợi ích của trigger, stored-procedure so với các câu lệnh SQL

Nội dung

1. Lập trình T-SQL
2. Stored-procedure
3. Functions
4. Triggers
5. Cursors

8.1 Lập trình T-SQL

- **Biến (Variables)**:

```
DECLARE @local_variable AS data_type = initial_value;
SET @variable_name = value;
PRINT @variable_name;
```

Câu lệnh điều khiển luồng:

```
IF condition
  BEGIN
    -- statements
  END
ELSE
  BEGIN
    -- statements
  END
```

```

```

WHILE:

```
WHILE condition
  BEGIN
    -- statements
  END
```

8.2 Stored-procedure

- **Stored-procedure**: là một tập hợp các câu lệnh Transact-SQL (T-SQL) được lưu trữ trong cơ sở dữ liệu. Các câu lệnh này được biên dịch thành một kế hoạch thực thi duy nhất khi **Stored-procedure** được thực thi. Stored Procedures có thể thực hiện các tác vụ mà SQL thông thường không thể thực hiện một cách dễ dàng, chẳng hạn như điều khiển luồng (flow control), xử lý lỗi, và thao tác dữ liệu phức tạp.
- **Ví dụ tạo stored-procedure**:

```
CREATE PROCEDURE procedure_name
AS
  sql_statement;
EXEC procedure_name;
```

8.3 Functions

- **Các loại functions**:
    - Scalar functions: là các hàm trả về một giá trị duy nhất, có thể là một giá trị nguyên, số thập phân, chuỗi ký tự, hoặc bất kỳ kiểu dữ liệu đơn giản nào khác.
    - Inline table-valued functions: là các hàm trả về một bảng dữ liệu (table) và được định nghĩa bằng một câu lệnh đơn SELECT.
    - Multi-statement table-valued functions: là các hàm trả về một bảng dữ liệu (table) và có thể bao gồm nhiều câu lệnh SQL bên trong thân hàm.

  Các hàm do người dùng định nghĩa (UDF) trong SQL thường được gọi bằng cách sử dụng câu lệnh SELECT khi chúng trả về một giá trị.

- **Ví dụ tạo function**:

```
CREATE FUNCTION function_name (@param1 data_type, @param2 data_type)
RETURNS return_data_type
AS
BEGIN
  -- function body
  RETURN value;
END;
```

8.4 Triggers

- **Triggers**: Kích hoạt khi một sự kiện (INSERT, UPDATE, DELETE) xảy ra. Có 3 loại trigger tương ứng với 3 sự kiện trên
- Khi làm việc với trigger trong SQL Server, có hai bảng đặc biệt mà bạn cần lưu ý: **inserted** và **deleted**. Những bảng này chỉ tồn tại trong

phạm vi của một trigger và được sử dụng để theo dõi các thay đổi dữ liệu khi một sự kiện (INSERT, UPDATE, DELETE) xảy ra.

- ❖ **Bảng inserted**

- **Bảng inserted** chứa các hàng mới được chèn vào hoặc các hàng mới sau khi được cập nhật.
- **Khi nào sử dụng**:
  - **INSERT Trigger**: Bảng inserted chứa các hàng mới được chèn vào.
  - **UPDATE Trigger**: Bảng inserted chứa các hàng mới sau khi cập nhật.

- ❖ **Bảng deleted**

- **Bảng deleted** chứa các hàng cũ bị xóa hoặc các hàng cũ trước khi được cập nhật.
- **Khi nào sử dụng**:
  - **DELETE Trigger**: Bảng deleted chứa các hàng bị xóa.
  - **UPDATE Trigger**: Bảng deleted chứa các hàng trước khi cập nhật.
- **Ví dụ tạo trigger**:

```
CREATE TRIGGER trigger_name ON table_name
AFTER INSERT, UPDATE, DELETE
AS
BEGIN
  -- trigger body
END;
```

8.5 Cursors

- **Cursors**: Cho phép duyệt qua tập kết quả của từng truy vấn và xử lí từng bộ riêng lẻ.Tuy nhiên, sử dụng cursors thường không được coi là phương pháp tối ưu nhất cho các hoạt động như chèn, cập nhật hoặc xóa dữ liệu vì chúng có thể làm chậm hiệu suất của hệ thống.
- **Ví dụ sử dụng cursor**:

```
DECLARE cursor_name CURSOR FOR SELECT statement;
OPEN cursor_name;
FETCH NEXT FROM cursor_name INTO @var1, @var2;
CLOSE cursor_name;
DEALLOCATE cursor_name;
```

Các Mối Quan Hệ trong Cơ Sở Dữ Liệu và Những Điều Cần Lưu Ý Khi Code

1. Mối quan hệ 1-1 (One-to-One)

Định nghĩa:

- Mỗi thực thể trong tập thực thể này chỉ liên kết với một thực thể trong tập thực thể kia và ngược lại.

Điều cần lưu ý khi code:

- Đảm bảo rằng mỗi thực thể chỉ có thể liên kết với một thực thể khác duy nhất.
- Thường sử dụng khóa ngoại (foreign key) duy nhất hoặc các ràng buộc toàn vẹn (unique constraints) để đảm bảo tính toàn vẹn của mối quan hệ.
- Cân nhắc việc gộp các bảng lại thành một bảng nếu có ít thuộc tính để tránh truy vấn phức tạp không cần thiết.

2. Mối quan hệ 1-N (One-to-Many)

Định nghĩa:

- Một thực thể trong tập thực thể này liên kết với nhiều thực thể trong tập thực thể kia, nhưng mỗi thực thể trong tập thực thể kia chỉ liên kết với một thực thể trong tập thực thể này.

Điều cần lưu ý khi code:

- Sử dụng khóa ngoại để liên kết các bảng với nhau.
- Đảm bảo rằng khóa ngoại trong bảng "nhiều" (many) tham chiếu đến khóa chính trong bảng "một" (one).
- Khi xóa một thực thể trong bảng "một", cần cân nhắc việc xóa hoặc cập nhật các thực thể liên quan trong bảng "nhiều".

3. Mối quan hệ N-N (Many-to-Many)

Định nghĩa:

- Mỗi thực thể trong một tập thực thể có thể liên kết với nhiều thực thể trong tập thực thể kia và ngược lại.

Điều cần lưu ý khi code:

- Tạo bảng liên kết (junction table) để lưu trữ các mối quan hệ giữa hai bảng chính.
- Bảng liên kết chứa ít nhất hai khóa ngoại, mỗi khóa ngoại tham chiếu đến khóa chính của một trong hai bảng chính.
- Đảm bảo tính toàn vẹn tham chiếu bằng cách sử dụng ràng buộc toàn vẹn trên các khóa ngoại trong bảng liên kết.

4. Mối quan hệ cha-con (Hierarchical)

Định nghĩa:

- Một thực thể cha liên kết với nhiều thực thể con. Mối quan hệ này thường được sử dụng trong mô hình phân cấp.

Điều cần lưu ý khi code:

- Sử dụng khóa ngoại để liên kết thực thể con với thực thể cha.

- Đảm bảo rằng mỗi thực thể con có một khóa ngoại tham chiếu đến khóa chính của thực thể cha.
- Xem xét việc sử dụng cấu trúc dữ liệu như cây (tree) hoặc đồ thị (graph) để biểu diễn mối quan hệ phân cấp.

5. Mối quan hệ mạnh-yếu (Strong-Weak)

Định nghĩa:

- Mối quan hệ giữa một thực thể mạnh (có khóa chính riêng) và một thực thể yếu (phụ thuộc vào thực thể mạnh để xác định duy nhất).

Điều cần lưu ý khi code:

- Thực thể yếu phải có một khóa ngoại tham chiếu đến khóa chính của thực thể mạnh.
- Khóa chính của thực thể yếu thường bao gồm khóa ngoại tham chiếu và một hoặc nhiều thuộc tính riêng của thực thể yếu.
- Đảm bảo rằng các thực thể yếu không thể tồn tại nếu không có thực thể mạnh tương ứng, thường bằng cách sử dụng các ràng buộc toàn vẹn tham chiếu (referential integrity constraints).

Lý Thuyết Về Hệ Quản Trị Cơ Sở Dữ Liệu Quan Hệ (RDBMS)

Hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) là một hệ thống quản lý cơ sở dữ liệu dựa trên mô hình quan hệ. Mô hình này tổ chức dữ liệu thành các bảng (tables) bao gồm các hàng (rows) và cột (columns).

Thành Phần Chính Của RDBMS

1. **Bộ sưu tập các bảng (Collection of Tables)**:
    - **Bảng (Table)**: Là cấu trúc cơ bản trong RDBMS, dùng để lưu trữ dữ liệu. Mỗi bảng chứa các hàng và cột, nơi mỗi hàng đại diện cho một bản ghi dữ liệu và mỗi cột đại diện cho một thuộc tính của dữ liệu đó.
    - **Bộ sưu tập các bảng** là câu trả lời đúng vì RDBMS quản lý và lưu trữ dữ liệu chủ yếu thông qua các bảng.
2. **Bộ sưu tập các bản ghi (Collection of Records)**:

- **Bản ghi (Record)**: Mỗi hàng trong bảng là một bản ghi. Tuy nhiên, nói RDBMS là bộ sưu tập các bản ghi không chính xác, vì bản ghi chỉ là thành phần của bảng.

3. **Bộ sưu tập các khóa (Collection of Keys)**:
   - **Khóa (Key)**: Khóa là các thuộc tính hoặc tập hợp các thuộc tính được sử dụng để xác định duy nhất các bản ghi trong bảng. Mặc dù khóa rất quan trọng, chúng không phải là yếu tố cơ bản duy nhất cấu thành RDBMS.
4. **Bộ sưu tập các trường (Collection of Fields)**:
   - **Trường (Field)**: Trường là các cột trong bảng, đại diện cho thuộc tính của dữ liệu. Giống như bản ghi, trường cũng chỉ là thành phần của bảng.

Sự khác nhau của DBMS với RDBMS

So sánh tổng quát

Đặc điểm	DBMS	RDBMS
Mô hình dữ liệu	Phân cấp, mạng, hướng đối tượng, tệp tin	Quan hệ (relational)
Lưu trữ dữ liệu	Tệp tin	Bảng (tables)
Tính toàn vẹn dữ liệu	Hạn chế hoặc không có	Hỗ trợ ràng buộc toàn vẹn dữ liệu
Giao dịch (Transactions)	Hạn chế hoặc không có	Hỗ trợ ACID
Ngôn ngữ truy vấn	Không bắt buộc phải dùng SQL	Sử dụng SQL
Quản lý quyền truy cập	Hạn chế hoặc không có	Bảo mật và quản lý quyền truy cập
Ví dụ	File System, XML Database	MySQL, PostgreSQL, Oracle, SQL Server

Các Loại JOIN trong SQL

JOIN là một câu lệnh trong SQL được sử dụng để kết hợp các hàng từ hai hoặc nhiều bảng dựa trên một điều kiện liên quan giữa chúng. Dưới đây là các loại JOIN phổ biến trong SQL:

1. INNER JOIN

- **Khái niệm**: INNER JOIN trả về các hàng khi có sự phù hợp trong cả hai bảng. Chỉ những hàng có giá trị khớp nhau trong cả hai bảng mới được chọn.

- **Khi không chỉ định loại join nào ở mệnh đề from thì mặc định sẽ là INNER JOIN. Equi Joins cũng same thằng này nhưng chỉ có thể dùng toán tử = để nối 2 bảng, còn inner join có thể dùng > , <,..**

2. LEFT JOIN (LEFT OUTER JOIN)

- **Khái niệm**: LEFT JOIN trả về tất cả các hàng từ bảng bên trái, và các hàng phù hợp từ bảng bên phải. Nếu không có sự phù hợp, kết quả sẽ chứa NULL từ bảng bên phải.

3. RIGHT JOIN (RIGHT OUTER JOIN)

- **Khái niệm**: RIGHT JOIN trả về tất cả các hàng từ bảng bên phải, và các hàng phù hợp từ bảng bên trái. Nếu không có sự phù hợp, kết quả sẽ chứa NULL từ bảng bên trái.

4. FULL JOIN (FULL OUTER JOIN)

- **Khái niệm**: FULL JOIN trả về tất cả các hàng khi có sự phù hợp trong một trong các bảng. Nếu không có sự phù hợp, kết quả sẽ chứa NULL từ bảng này hoặc bảng kia.

5. CROSS JOIN

- **Khái niệm**: CROSS JOIN trả về tích Đề-các của các hàng từ hai bảng. Kết quả sẽ là số lượng hàng của bảng này nhân với số lượng hàng của bảng kia.

6. SELF JOIN

- **Khái niệm**: SELF JOIN là một join trong đó bảng được join với chính nó. Thường được sử dụng để tìm các mối quan hệ trong cùng một bảng, chẳng hạn như quản lý và nhân viên.

Lý Thuyết Về Xử Lý Lỗi Trong Giao Dịch (Transaction) Trong SQL

Trong SQL, khi một lỗi xảy ra trong quá trình thực hiện giao dịch (transaction) và không được xử lý hoặc không được rollback (hoàn tác) một cách rõ ràng, hệ quản trị cơ sở dữ liệu sẽ tự động quyết định trạng thái của giao dịch đó. Các hệ quản trị cơ sở dữ liệu thường xử lý các lỗi trong giao dịch bằng cách rollback giao dịch để đảm bảo tính toàn vẹn của dữ liệu.

Một số lệnh trong sql server để xử lí chuỗi:

1. LEFT

- **Chức năng**: Trả về một số ký tự nhất định từ đầu chuỗi.
- **Cú pháp**: LEFT(string, number_of_characters)

Ví dụ:
SELECT LEFT('FPTUDN', 3) AS 'LeftString';

- **Kết quả**: 'FPT'

2. RIGHT

- **Chức năng**: Trả về một số ký tự nhất định từ cuối chuỗi.
- **Cú pháp**: RIGHT(string, number_of_characters)

Ví dụ:
SELECT RIGHT('FPTUDN', 3) AS 'RightString';

- **Kết quả**: 'UDN'

3. LEN

- **Chức năng**: Trả về độ dài của chuỗi.
- **Cú pháp**: LEN(string)

Ví dụ:
SELECT LEN('FPTUDN') AS 'Length';

- **Kết quả**: 6

4. CHARINDEX

- **Chức năng**: Trả về vị trí bắt đầu của một chuỗi con trong một chuỗi.
- **Cú pháp**: CHARINDEX(substring, string)

Ví dụ:

```
SELECT CHARINDEX('PT', 'FPTUDN') AS 'Index';
```

- **Kết quả**: 2

5. PATINDEX

- **Chức năng**: Trả về vị trí của một mẫu trong một chuỗi.
- **Cú pháp**: PATINDEX('%pattern%', string)

Ví dụ:

```
SELECT PATINDEX('%PT%', 'FPTUDN') AS 'PatternIndex';
```

- **Kết quả**: 2

6. REPLACE

- **Chức năng**: Thay thế tất cả các lần xuất hiện của một chuỗi con bằng một chuỗi con khác.
- **Cú pháp**: REPLACE(string, old_substring, new_substring)

Ví dụ:

```
SELECT REPLACE('FPTUDN', 'UDN', 'University') AS 'ReplacedString';
```

- **Kết quả**: 'FPTUniversity'

7. RTRIM và LTRIM

- **Chức năng**: Loại bỏ khoảng trắng ở cuối (RTRIM) hoặc đầu (LTRIM) của chuỗi.
- **Cú pháp**:
    - RTRIM(string)

- LTRIM(string)

Ví dụ:
SELECT RTRIM(' FPTUDN ') AS 'RTRIMString';
SELECT LTRIM(' FPTUDN ') AS 'LTRIMString';

- **Kết quả**:
  - ' FPTUDN' (cho RTRIM)
  - 'FPTUDN ' (cho LTRIM)

8. CONCAT

- **Chức năng**: Kết hợp nhiều chuỗi thành một chuỗi duy nhất.
- **Cú pháp**: CONCAT(string1, string2, ...)

Ví dụ:
SELECT CONCAT('FPT', 'UDN') AS 'ConcatenatedString';

- **Kết quả**: 'FPTUDN'

9. SUBSTRING.
Lệnh SUBSTRING được sử dụng để trích xuất một phần của chuỗi từ một chuỗi ban đầu.

Cú pháp của SUBSTRING
SUBSTRING(string, start_position, length)

- **string**: Chuỗi ban đầu từ đó bạn muốn trích xuất.
- **start_position**: Vị trí bắt đầu trích xuất trong chuỗi (bắt đầu từ 1).
- **length**: Số lượng ký tự cần trích xuất.

Ví dụ:
SELECT SUBSTRING('HelloWorld', 6, 5);

Kết quả:
World

Kiểu dữ liệu trong SQL

- Kiểu số (Number type): Bao gồm các kiểu dữ liệu như INT, FLOAT, DECIMAL, etc.
- Chuỗi ký tự (String type): Bao gồm các kiểu dữ liệu như VARCHAR, CHAR, TEXT, etc.
- Kiểu ngày và giờ (Date and time type): Bao gồm các kiểu dữ liệu như DATE, TIME, DATETIME, etc.
- Kiểu bảng (Table type): Một hàm có thể trả về một bảng, đặc biệt là các hàm kiểu Table-Valued Function.

Toán Tử Nhị Phân và Đơn Nhân trong Đại Số Quan Hệ

Trong đại số quan hệ (Relational Algebra), các toán tử có thể được phân thành hai loại: toán tử nhị phân và toán tử đơn nhân.

- Toán tử nhị phân (Binary Operation): Toán tử yêu cầu hai toán hạng (hai bảng) để thực hiện phép toán.
- Toán tử đơn nhân (Unary Operation): Toán tử chỉ yêu cầu một toán hạng (một bảng) để thực hiện phép toán.

Query Compiler:

Query Parsing → Query Preprocessing → Query Optimization → Query Execution

1. **Query Parsing (Phân tích truy vấn)**:

- **Mục đích**: Phân tích cú pháp của truy vấn SQL để đảm bảo rằng truy vấn hợp lệ và không có lỗi cú pháp.
- **Công việc**: Trình biên dịch sẽ chuyển đổi truy vấn SQL thành một cấu trúc dữ liệu như **cây phân tích (parse tree)** hoặc **cây cú pháp trừu tượng (abstract syntax tree - AST)**.
- **Kết quả**: Xác nhận truy vấn hợp lệ về mặt cú pháp.

2. **Query Preprocessing (Tiền xử lý truy vấn)**:
   - **Mục đích**: Xử lý các tác vụ như kiểm tra quyền truy cập, kiểm tra các lỗi ngữ nghĩa và các phép biến đổi đơn giản.
   - **Công việc**: Bao gồm các bước như kiểm tra bảng, kiểm tra cột, xem xét các alias, v.v.
   - **Kết quả**: Truy vấn đã được chuẩn bị để chuyển sang giai đoạn tối ưu hóa.
3. **Query Optimization (Tối ưu hóa truy vấn)**:
   - **Mục đích**: Tối ưu hóa kế hoạch truy vấn để chọn cách thức thực thi hiệu quả nhất.
   - **Công việc**: Tối ưu hóa truy vấn bằng cách thay đổi cấu trúc của kế hoạch thực thi. Ví dụ, thay đổi thứ tự các phép toán, sử dụng chỉ mục (indexes) hoặc chuyển đổi các phép toán sao cho hiệu quả hơn.
   - **Kết quả**: Kế hoạch truy vấn tối ưu nhất để thực thi.
4. **Query Execution (Thực thi truy vấn)**:
   - **Mục đích**: Thực hiện kế hoạch truy vấn đã tối ưu hóa.
   - **Công việc**: Trình biên dịch truyền tải kế hoạch thực thi đến phần xử lý của cơ sở dữ liệu, nơi các thao tác thực sự được thực hiện trên dữ liệu.
   - **Kết quả**: Dữ liệu được truy xuất, cập nhật, hoặc xóa theo yêu cầu của truy vấn.