**Describe database objects**

Completed100 XP

* 9 minutes

In addition to tables, a relational database can contain other structures that help to optimize data organization, encapsulate programmatic actions, and improve the speed of access. In this unit, you'll learn about three of these structures in more detail: *views*, *stored procedures*, and *indexes*.

**What is a view?**

A view is a virtual table based on the results of a **SELECT** query. You can think of a view as a window on specified rows in one or more underlying tables. For example, you could create a view on the **Order** and **Customer** tables that retrieves order and customer data to provide a single object that makes it easy to determine delivery addresses for orders:

SQLCopy

CREATE VIEW Deliveries

AS

SELECT o.OrderNo, o.OrderDate,

c.FirstName, c.LastName, c.Address, c.City

FROM Order AS o JOIN Customer AS c

ON o.Customer = c.ID;

You can query the view and filter the data in much the same way as a table. The following query finds details of orders for customers who live in Seattle:

SQLCopy

SELECT OrderNo, OrderDate, LastName, Address

FROM Deliveries

WHERE City = 'Seattle';

**What is a stored procedure?**

A stored procedure defines SQL statements that can be run on command. Stored procedures are used to encapsulate programmatic logic in a database for actions that applications need to perform when working with data.

You can define a stored procedure with parameters to create a flexible solution for common actions that might need to be applied to data based on a specific key or criteria. For example, the following stored procedure could be defined to change the name of a product based on the specified product ID.

SQLCopy

CREATE PROCEDURE RenameProduct

@ProductID INT,

@NewName VARCHAR(20)

AS

UPDATE Product

SET Name = @NewName

WHERE ID = @ProductID;

When a product must be renamed, you can execute the stored procedure, passing the ID of the product and the new name to be assigned:

SQLCopy

EXEC RenameProduct 201, 'Spanner';

**What is an index?**

An index helps you search for data in a table. Think of an index over a table like an index at the back of a book. A book index contains a sorted set of references, with the pages on which each reference occurs. When you want to find a reference to an item in the book, you look it up through the index. You can use the page numbers in the index to go directly to the correct pages in the book. Without an index, you might have to read through the entire book to find the references you're looking for.

When you create an index in a database, you specify a column from the table, and the index contains a copy of this data in a sorted order, with pointers to the corresponding rows in the table. When the user runs a query that specifies this column in the **WHERE** clause, the database management system can use this index to fetch the data more quickly than if it had to scan through the entire table row by row.

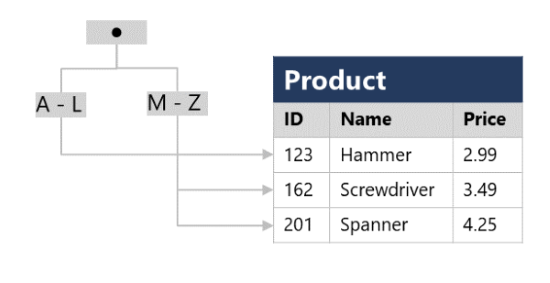
For example, you could use the following code to create an index on the **Name** column of the **Product** table:

SQLCopy

CREATE INDEX idx\_ProductName

ON Product(Name);

The index creates a tree-based structure that the database system's query optimizer can use to quickly find rows in the **Product** table based on a specified **Name**.



For a table containing few rows, using the index is probably not any more efficient than simply reading the entire table and finding the rows requested by the query (in which case the query optimizer will ignore the index). However, when a table has many rows, indexes can dramatically improve the performance of queries.

You can create many indexes on a table. So, if you also wanted to find products based on price, creating another index on the **Price** column in the **Product** table might be useful. However, indexes aren't free. An index consumes storage space, and each time you insert, update, or delete data in a table, the indexes for that table must be maintained. This additional work can slow down insert, update, and delete operations. You must strike a balance between having indexes that speed up your queries versus the cost of performing other operations.

**Các đối tượng trong cơ sở dữ liệu**

Ngoài các bảng, một cơ sở dữ liệu quan hệ có thể chứa các cấu trúc khác giúp tối ưu hóa tổ chức dữ liệu, đóng gói các hành động theo chương trình và cải thiện tốc độ truy cập. Trong bài học này, bạn sẽ tìm hiểu chi tiết hơn về ba cấu trúc này: **ki检视图 (kǎn jiàn wèi tú)** (view - dạng xem), **存储过程 (cúntrù chéng xuè)** (stored procedures - thủ tục lưu trữ) và **索 引 (suǒyǐn)** (index - chỉ mục).

**Dạng xem (View) là gì?**

Dạng xem là một bảng ảo dựa trên kết quả của truy vấn SELECT. Bạn có thể coi dạng xem như một cửa sổ trên các hàng được chỉ định trong một hoặc nhiều bảng nền. Ví dụ, bạn có thể tạo một dạng xem trên các bảng Đơn hàng và Khách hàng để truy xuất dữ liệu đơn hàng và khách hàng nhằm cung cấp một đối tượng duy nhất giúp dễ dàng xác định địa chỉ giao hàng cho các đơn hàng:

SQL

CREATE VIEW Deliveries

AS

SELECT o.OrderNo, o.OrderDate,

      c.FirstName, c.LastName, c.Address, c.City

FROM Order AS o JOIN Customer AS c

ON o.Customer = c.ID;

Use code [with caution.](https://gemini.google.com/faq#coding)

content\_copy

Bạn có thể truy vấn dạng xem và lọc dữ liệu theo cách tương tự như một bảng. Truy vấn sau tìm chi tiết các đơn hàng cho khách hàng sống ở Seattle:

SQL

SELECT OrderNo, OrderDate, LastName, Address

FROM Deliveries

WHERE City = 'Seattle';

Use code [with caution.](https://gemini.google.com/faq#coding)

content\_copy

**Thủ tục lưu trữ (Stored Procedure) là gì?**

Thủ tục lưu trữ định nghĩa các câu lệnh SQL có thể được chạy theo lệnh. Thủ tục lưu trữ được sử dụng để đóng gói logic theo chương trình trong cơ sở dữ liệu cho các hành động mà ứng dụng cần thực hiện khi làm việc với dữ liệu.

Bạn có thể định nghĩa một thủ tục lưu trữ với các tham số để tạo giải pháp linh hoạt cho các hành động thông thường có thể cần được áp dụng cho dữ liệu dựa trên một khóa hoặc tiêu chí cụ thể. Ví dụ, thủ tục lưu trữ sau có thể được định nghĩa để thay đổi tên của sản phẩm dựa trên ID sản phẩm được chỉ định.

SQL

CREATE PROCEDURE RenameProduct

        @ProductID INT,

        @NewName VARCHAR(20)

AS

UPDATE Product

SET Name = @NewName

WHERE ID = @ProductID;

Use code [with caution.](https://gemini.google.com/faq#coding)

content\_copy

Khi cần đổi tên sản phẩm, bạn có thể thực thi thủ tục lưu trữ, truyền ID của sản phẩm và tên mới cần gán:

SQL

EXEC RenameProduct 201, 'Spanner';

Use code [with caution.](https://gemini.google.com/faq#coding)

content\_copy

**Chỉ mục (Index) là gì?**

Chỉ mục giúp bạn tìm kiếm dữ liệu trong bảng. Hãy nghĩ về chỉ mục trên một bảng giống như chỉ mục ở cuối sách. Chỉ mục sách chứa một tập hợp các tham chiếu được sắp xếp, với các trang mà mỗi tham chiếu xuất hiện. Khi bạn muốn tìm một tham chiếu đến một mục trong sách, bạn sẽ tra cứu nó thông qua chỉ mục. Bạn có thể sử dụng số trang trong chỉ mục để đi trực tiếp đến các trang chính xác trong sách. Nếu không có chỉ mục, bạn có thể phải đọc toàn bộ sách để tìm các tham chiếu mình đang tìm kiếm.

Khi bạn tạo chỉ mục trong cơ sở dữ liệu, bạn chỉ định một cột từ bảng và chỉ mục chứa một bản sao của dữ liệu này theo thứ tự được sắp xếp, với các con trỏ đến các hàng tương ứng trong bảng. Khi người dùng chạy truy vấn chỉ định cột này trong mệnh đề WHERE, hệ thống quản lý cơ sở dữ liệu có thể sử dụng chỉ mục này để lấy dữ liệu nhanh hơn so với việc phải quét toàn bộ bảng từng hàng một.

Ví dụ, bạn có thể sử dụng đoạn mã sau để tạo chỉ mục cho cột Name của bảng Product:

SQL

CREATE INDEX idx\_ProductName

ON Product(Name);

Use code [with caution.](https://gemini.google.com/faq#coding)

content\_copy

Chỉ mục tạo ra một cấu trúc dựa trên cây mà bộ tối ưu hóa truy vấn của hệ thống cơ sở dữ liệu có thể sử dụng để nhanh chóng tìm các hàng trong bảng Product dựa trên một Name được chỉ định.

Đối với một bảng chứa vài hàng, việc sử dụng chỉ mục có thể không hiệu quả hơn việc chỉ đơn giản đọc toàn bộ bảng và tìm các hàng được yêu cầu bởi truy vấn (trong trường hợp này, bộ tối ưu hóa truy vấn sẽ bỏ qua chỉ mục). Tuy nhiên, khi một bảng có nhiều hàng, chỉ mục có thể cải thiện đáng kể hiệu suất của các truy vấn.

**Hiệu suất và chi phí của chỉ mục:**

* **Hiệu suất:**
  + Chỉ mục có thể giúp truy vấn dữ liệu nhanh hơn bằng cách cho phép hệ thống quản lý cơ sở dữ liệu truy cập trực tiếp vào các hàng cụ thể mà không cần quét toàn bộ bảng.
  + Hiệu quả của chỉ mục tăng lên khi số lượng hàng trong bảng tăng lên.
* **Chi phí:**
  + Chỉ mục sử dụng dung lượng lưu trữ.
  + Mỗi khi dữ liệu trong bảng được chèn, cập nhật hoặc xóa, các chỉ mục cho bảng đó cũng phải được cập nhật. Việc cập nhật này có thể làm chậm các thao tác chèn, cập nhật và xóa.

**Lưu ý:**

* Bạn cần cân bằng giữa việc sử dụng chỉ mục để tăng tốc truy vấn và chi phí lưu trữ và cập nhật chỉ mục.
* Có nhiều loại chỉ mục khác nhau, mỗi loại có ưu điểm và nhược điểm riêng. Bạn nên chọn loại chỉ mục phù hợp nhất với nhu cầu của mình.

**Ví dụ về các loại chỉ mục:**

* **Chỉ mục cụm (Clustered index):**
  + Là loại chỉ mục duy nhất có thể có trên một bảng.
  + Sắp xếp dữ liệu trong bảng theo thứ tự cụ thể.
  + Cấu trúc dữ liệu được sử dụng để lưu trữ dữ liệu thực tế của bảng.
* **Chỉ mục không cụm (Non-clustered index):**
  + Có thể có nhiều chỉ mục không cụm trên một bảng.
  + Sắp xếp dữ liệu trong bảng theo một hoặc nhiều cột.
  + Không ảnh hưởng đến cấu trúc dữ liệu được sử dụng để lưu trữ dữ liệu thực tế của bảng.