در مورد همهٔ نوع های داده ای در زبان SQL توضیح داده شود.
با ذکر مثال و قطعه کد نوع های داده ای توضیح داده شوند و پیاده سازی آنها شرح داده شود.
هر یک از نوع ها چه ویژگی هایی دارند و در چه مواردی استفاده میشوند؟

انتخاب نوع داده مناسب برای یک ستون، متغیر یا پارامتر، اهمیت زیادی دارد:

- نوع داده باید بتواند محدوده داده‌هایی را که قرار است در آن ذخیره شوند، به طور دقیق مشخص کند.
- نوع داده باید فضای ذخیره‌سازی مناسبی را اشغال کند.
- نوع داده باید با نیازهای برنامه‌نویسی مطابقت داشته باشد.

انواع داده در SQL به دو دسته کلی تقسیم می‌شوند:

- داده‌های پایه (Primitive):

این نوع داده‌ها، داده‌های اصلی و ساده‌ای هستند که می‌توان آنها را به طور مستقیم در یک ستون، متغیر یا پارامتر ذخیره کرد.

- داده‌های مشتق (Derived):

این نوع داده‌ها، از ترکیب چند نوع داده پایه ایجاد می‌شوند.

از این نوع داده ها برای ذخیره مقادیر، انجام محاسبات و تجزیه و تحلیل داده ها استفاده می شود.

داده‌های پایه در SQL به شرح زیر هستند:

داده‌های متنی (Character): این نوع داده‌ها، برای ذخیره متن استفاده می‌شوند.

- char(n)

رشته متنی با طول ثابت n ، طول رشته هنگام تعریف ستون یا متغیر مشخص میشود. اگر طول داده وارد شده از متغیر بیشتر باشد کاراکترهای اضافه حذف میشوند.

مثال: در این مثال یک جدول با یک ستون به نام name از نوع char ساخته شده است.

```
CREATE TABLE customers (
  id INT,
  name CHAR(10)
);

INSERT INTO customers (id, name) VALUES (1, 'John Doe');
```

- varchar(n)

رشته متنی با طول متغیر n ، طول رشته، در هنگام وارد کردن مقدار برای رشته مشخص می‌شود. اگر مقدار وارد شده برای این ستون، کوتاه‌تر از ۲۰ کاراکتر باشد، هیچ کاراکتری به آن اضافه نمی‌شود. اگر مقدار وارد شده برای این ستون، بلندتر از ۲۰ کاراکتر باشد، رشته بدون تغییر باقی می‌ماند.

مثال: در این مثال یک جدول با یک ستون به نام name از نوع varchar ساخته شده است.

```
CREATE TABLE customers (
  id INT,
  name VARCHAR(20)
);

INSERT INTO customers (id, name) VALUES (1, 'John Doe');
```

- nchar(n)

رشته متنی با طول ثابت n که از کاراکترهای Unicode استفاده می‌کند. طول رشته، در هنگام تعریف ستون یا متغیر مشخص می‌شود. اگر مقدار وارد شده برای رشته، از طول مشخص شده بیشتر باشد، رشته کوتاه می‌شود و کاراکترهای اضافی حذف می‌شوند.

مثال: در این مثال یک جدول با یک ستون به نام name از نوع nchar ساخته شده است.

```
CREATE TABLE customers (
  id INT,
  name NCHAR(10)
);
```

```
INSERT INTO customers (id, name) VALUES (1, 'John Doe');
```

- nvarchar(n)

رشته متنی با طول متغیر n که از کاراکترهای Unicode استفاده می‌کند. طول رشته، در هنگام وارد کردن مقدار برای رشته مشخص می‌شود.

مثال: در این مثال یک جدول با یک ستون به نام name از نوع nvarchar ساخته شده است.

```
CREATE TABLE customers (
  id INT,
  name NVARCHAR(20)
);

INSERT INTO customers (id, name) VALUES (1, 'John Doe');
```

داده‌های عددی (Numeric): این نوع داده‌ها، برای ذخیره اعداد استفاده می‌شوند.

- Tinyint

عدد صحیح بین ۰ تا ۲۵۵

مثال: در این مثال یک جدول با ستون age از نوع TINYINT ساخته شده است، یعنی داده های این ستون میتواند مقادیری بین ۰ تا ۲۵۵ باشد.

```
CREATE TABLE customers (
  id INT,
  age TINYINT
);

INSERT INTO customers (id, age) VALUES (1, 20);
```

- Smallint

عدد صحیح بین ۳۲۷۶۸- تا ۳۲۷۶۷

مثال: در این مثال یک جدول با ستون age از نوع SMALLINT ساخته شده است، یعنی داده های این ستون میتواند مقادیری بین ۳۲۷۶۸- تا ۳۲۷۶۷ باشد.

```
CREATE TABLE customers (
  id INT,
  age SMALLINT
);

INSERT INTO customers (id, age) VALUES (1, 20000);
```

- Int

عدد صحیح بین ۲۱۴۷۴۸۳۶۴۸- تا ۲۱۴۷۴۸۳۶۴۷

مثال: در این مثال یک جدول با ستون age از نوع INT ساخته شده است، یعنی داده های این ستون میتواند مقادیری بین ۲۱۴۷۴۸۳۶۴۸- تا ۲۱۴۷۴۸۳۶۴۷ باشد.

```
CREATE TABLE customers (
  id INT,
  age INT
);

INSERT INTO customers (id, age) VALUES (1, 20000000);
```

- Bigint

عدد صحیح بین ۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷۵۸۸- تا ۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷۵۸۰۷

مثال: در این مثال یک جدول با ستون age از نوع BIGINT ساخته شده است، یعنی داده های این ستون میتواند مقادیری بین ۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷۵۸۸- تا ۹۲۲۳۳۷۲۰۳۶۸۵۴۷۷۵۸۰۷ باشد.

```
CREATE TABLE customers (
  id INT,
  age BIGINT
);

INSERT INTO customers (id, age) VALUES (1,
20000000000000000);
```

- Float

عدد اعشاری با دقت ۶ یا ۲۴ رقم

مثال: در این مثال یک جدول با ستون price از نوع FLOAT ساخته شده است.

```
CREATE TABLE customers (
  id INT,
  price FLOAT
);

INSERT INTO customers (id, price) VALUES (1, 123.456);
```

- Real

عدد اعشاری با دقت ۷ رقم

مثال: در این مثال یک جدول با ستون price از نوع REAL ساخته شده است.

```
CREATE TABLE customers (
  id INT,
  price REAL
);

INSERT INTO customers (id, price) VALUES (1,
123.456789);
```

- decimal(p,s)

عدد اعشاری با دقت p رقم و s رقم اعشار

مثال: در این مثال یک جدول با ستون price از نوع DECIMAL ساخته شده است. بنابراین، مقدار عدد اعشاری که می‌توان برای این ستون وارد کرد، باید بین ۰ تا ۹۹۹۹۹۹۹۹۹۹.۹۹ باشد. دقت عدد اعشاری، برابر با ۲ رقم است.

```
CREATE TABLE customers (
  id INT,
  price DECIMAL(10,2)
);

INSERT INTO customers (id, price) VALUES (1, 123.45);
```

داده‌های تاریخ و زمان (Datetime): این نوع داده‌ها، برای ذخیره تاریخ و زمان استفاده می‌شوند.

- Date

تاریخ، سال به صورت چهار رقمی، ماه به صورت دو رقمی و روز نیز به صورت دو رقمی ذخیره می شود. بنابراین، مقدار تاریخ بدون زمان که می‌توان برای این ستون وارد کرد، باید مطابق با فرمت YYYY-MM-DD باشد.

مثال: در مثال زیر یک جدول با ستون birth_date از نوع date تعریف شده است.

```
CREATE TABLE customers (
  id INT,
  birth_date DATE
);

INSERT INTO customers (id, birth_date) VALUES (1,
'2023-08-02');
```

- Time

زمان، ساعت به صورت دو رقمی، دقیقه به صورت دو رقمی، ثانیه به صورت دو رقمی ذخیره می شود. بنابراین، مقدار زمان بدون تاریخ که می‌توان برای این ستون وارد کرد، باید مطابق با فرمت HH:MM:SS باشد.

مثال: در این مثال یک جدول با ستون order_time از نوع TIME تعریف شده است.

```
CREATE TABLE customers (
  id INT,
  order_time TIME
);

INSERT INTO customers (id, order_time) VALUES (1,
'12:34:56');
```

- Datetime

تاریخ و زمان، این نوع داده، از ترکیب دو نوع داده date و time تشکیل شده است.

مثال: در این مثال یک جدول با ستون last_login از نوع DATETIME تعریف شده است.

```
CREATE TABLE customers (
  id INT,
  last_login DATETIME
);

INSERT INTO customers (id, last_login) VALUES (1,
'2023-08-20 12:34:56');
```

- datetimeoffset

تاریخ و زمان با دقت میلی‌ثانیه و اختلاف ساعت با گرینویچ. نابراین، مقدار تاریخ و زمانی که می‌توان برای این ستون وارد کرد، باید بین ۰۰۰۱-۰۱-۰۱ ۰۰:۰۰:۰۰ تا ۹۹۹۹-۱۲-۳۱ ۲۳:۵۹:۵۹ باشد. دقت تاریخ و زمان، برابر با ۷ رقم است. اختلاف زمانی، در قالب +HH:mm یا -HH:mm وارد می‌شود.

مثال: در این مثال یک جدول با ستون order_date از نوع DATETIMEOFFSET تعریف شده است.

```
CREATE TABLE customers (
  id INT,
  order_date DATETIMEOFFSET
);

INSERT INTO customers (id, order_date) VALUES (1,
'2023-07-20 12:00:00 +04:00');
```

داده‌های منطقی (Boolean): این نوع داده‌ها، برای ذخیره مقادیر منطقی true یا false استفاده می‌شوند. برای true مقدار ۱ و برای false مقدار ۰ در نظر گرفته می شود.

- Bit

یک بیت

مثال: در مثال زیر یک جدول با ستون active با نوع داده BIT تعریف شده است.

```
CREATE TABLE customers (
  id INT,
  active BIT
);

INSERT INTO customers (id, active) VALUES (1, 1);
```

- Boolean

یک عدد صحیح ۰ یا ۱

مثال: در مثال زیر یک جدول با ستون is_active با نوع داده BOOLEAN تعریف شده است.

```
CREATE TABLE customers (
  id INT,
  is_active BOOLEAN
);

INSERT INTO customers (id, is_active) VALUES (1,
TRUE);
```

داده‌های مشتق در SQL به شرح زیر هستند:

- داده‌های آرایه (Array)

این نوع داده‌ها، مجموعه‌ای از داده‌های مشابه هستند که می‌توانند در یک ستون، متغیر یا پارامتر ذخیره شوند.

مثال: در مثال زیر یک جدول با ستون names با نوع داده ARRAY تعریف شده است. بنابراین، مقداری که می‌توان برای این ستون وارد کرد، باید یک آرایه از رشته‌ها باشد.

```
CREATE TABLE customers (
  id INT,
  names ARRAY(VARCHAR(255))
);

INSERT INTO customers (id, names) VALUES (1,
ARRAY['John Doe', 'Jane Doe']);
```

- داده‌های (XML)

این نوع داده‌ها، برای ذخیره داده‌های XML استفاده می‌شوند.

مثال: در مثال زیر یک جدول با ستون profile با نوع داده XML تعریف شده است. بنابراین، مقداری که می‌توان برای این ستون وارد کرد، باید یک سند XML باشد.

```
CREATE TABLE customers (
  id INT,
  profile XML
);

INSERT INTO customers (id, profile) VALUES (1,
'<profile><name>John
Doe</name><age>30</age></profile>');
```

- داده‌های USER-DEFINED TYPE

نوع داده‌ای که توسط کاربر تعریف شده است.

مثال: در مثال زیر نوع داده my_type تعریف شده است. این نوع داده، از دو فیلد id و name تشکیل شده است. سپس، ستون name با نوع داده my_type تعریف شده است. بنابراین، مقداری که می‌توان برای این ستون وارد کرد، باید یک مقدار از نوع داده my_type باشد.

```
CREATE TYPE my_type AS (
  id INT,
  name VARCHAR(255)
);

CREATE TABLE customers (
  id INT,
  name my_type
);

INSERT INTO customers (id, name) VALUES (1, my_type(1,
'John Doe'));
```

هر یک از این نوع های داده‌ای چگونه در حافظه تخصیص و چگونه پیاده سازی شده اند؟

- داده‌های متنی (Character)

داده‌های متنی، در حافظه به صورت آرایه‌ای از کاراکترها ذخیره می‌شوند. طول آرایه، برابر با طول رشته متنی است. هر کاراکتر، با یک کد عددی مشخص می‌شود. این کد عددی، معمولاً در یک جدول کد (Code Page) تعریف می‌شود.

برای مثال، اگر طول رشته متنی برابر با ۱۰ باشد، آرایه متنی، ۱۰ کاراکتر را در حافظه اشغال خواهد کرد.

- داده‌های عددی (Numeric)

داده‌های عددی، در حافظه به صورت آرایه‌ای از اعداد ذخیره می‌شوند. طول آرایه، برابر با تعداد ارقام عدد است. هر عدد، با یک کد عددی مشخص می‌شود. این کد عددی، معمولاً در یک قالب عددی (Numeric Format) تعریف می‌شود.

برای مثال، اگر عدد برابر با ۱۲۳۴۵۶ باشد، آرایه عددی، ۶ عدد را در حافظه اشغال خواهد کرد.

- داده‌های تاریخ و زمان (Datetime)

داده‌های تاریخ و زمان، در حافظه به صورت آرایه‌ای از اعداد ذخیره می‌شوند. طول آرایه، برابر با تعداد ارقام تاریخ و زمان است. هر عدد، با یک کد عددی مشخص می‌شود. این کد عددی، معمولاً در یک قالب تاریخ و زمان (Datetime Format) تعریف می‌شود.

برای مثال، اگر تاریخ و زمان برابر با ۲۰-۰۸-۲۰۲۳ ۱۲:۴۵:۰۰ باشد، آرایه تاریخ و زمان، ۱۴ عدد را در حافظه اشغال خواهد کرد. شش عدد اول آرایه، برای ذخیره تاریخ استفاده می‌شوند. هشت عدد بعدی آرایه، برای ذخیره زمان استفاده می‌شوند.

- داده‌های منطقی (Boolean)

داده‌های منطقی، در حافظه به صورت یک بیت ذخیره می‌شوند. مقدار بیت، برابر با مقدار منطقی true یا false است.

برای مثال، مقدار منطقی true با مقدار بیت ۱ و مقدار منطقی false با مقدار بیت ۰ ذخیره می‌شود.

نوع داده	اندازه در حافظه
char(n)	n بایت
varchar(n)	n بایت
nchar(n)	n بایت
nvarchar(n)	n بایت
tinyint	1 بایت
smallint	2 بایت
int	4 بایت
bigint	8 بایت
float	4 یا 8 بایت
real	4 بایت
decimal(p,s)	p + s + 2 بایت

چه عملگرهایی برای این نوع ها تعریف شده اند؟

برای انواع داده‌های پایه در SQL، عملگرهای زیر تعریف شده‌اند:

- عملگرهای مقایسه (Comparison Operators)

این عملگرها برای مقایسه دو مقدار از یک نوع داده استفاده می‌شوند. نتیجه مقایسه، یک مقدار منطقی true یا false است.

  - = مساوی
  - <> نابرابر
  - > کوچکتر از
  - => کوچکتر یا مساوی
  - < بزرگتر از
  - =< بزرگتر یا مساوی

- عملگرهای منطقی (Logical Operators)

این عملگرها برای ترکیب دو یا چند عبارت منطقی استفاده می‌شوند. نتیجه ترکیب، یک مقدار منطقی true یا false است.

  - AND و
  - OR یا
  - NOT نه

- عملگرهای تخصیص(Assignment Operators)

این عملگرها برای تخصیص یک مقدار به یک متغیر یا ستون استفاده می‌شوند.

  - = تخصیص
  - =+ جمع و تخصیص
  - =- تفریق و تخصیص
  - =* ضرب و تخصیص
  - =/ تقسیم و تخصیص

- عملگرهای دودویی (Binary Operators)

این عملگرها برای انجام عملیات ریاضی دوتایی بر روی دو مقدار استفاده می‌شوند.

  - + جمع
  - تفریق
  - ضرب
  - / تقسیم
  - % باقی‌مانده
  - ** توان

- عملگرهای یک‌تایی (Unary Operators)

این عملگرها برای انجام عملیات ریاضی یک‌تایی بر روی یک مقدار استفاده می‌شوند.

  - + مثبت
  - - منفی
  - ! معکوس منطقی

برای انواع داده‌های مشتق، عملگرهای زیر تعریف شده است:

- برای انواع داده‌های آرایه عملگرهای زیر تعریف شده است:
  - []: دسترسی به عنصر یک آرایه
  - @: طول یک آرایه

- برای انواع داده‌های XML، عملگرهای زیر تعریف شده‌اند:
  - . : دسترسی به عنصر یک سندXML
  - // : دسترسی به تمام عناصر یک سند XML که با یک الگوی خاص مطابقت دارند.

اگر لیست ها و رشته ها و آرایه های انجمنی در زبان SQL تعریف شده اند، پیاده سازی آنها چگونه است؟

در زبان SQL، لیست ها، رشته ها و آرایه های انجمنی به عنوان نوع داده های تعریف نشده وجود ندارند. با این حال، می توان آنها را با استفاده از توابع و عبارات SQL پیاده سازی کرد.

- لیست ها

لیست ها می توانند با استفاده از تابع ARRAY() پیاده سازی شوند .این تابع یک آرایه از مقادیر را ایجاد می کند .به عنوان مثال، کد زیر یک لیست از اعداد صحیح ایجاد می کند:

```
SELECT ARRAY(1, 2, 3, 4, 5);
```

- رشته ها

رشته ها می توانند با استفاده از تابع CONCAT() پیاده سازی شوند .این تابع رشته های داده را به یکدیگر متصل می کند .به عنوان مثال، کد زیر دو رشته را به یکدیگر متصل می کند:

```
SELECT CONCAT('Hello', 'World');
```

- آرایه های انجمنی

آرایه های انجمنی می توانند با استفاده از تابعMAP() پیاده سازی شوند .این تابع یک آرایه از جفت های کلید-مقدار را ایجاد می کند .به عنوان مثال، کد زیر یک آرایه انجمنی از نام و سن افراد ایجاد می کند:

```
SELECT MAP('name', 'John Doe', 'age', 30);
```

اگر اشاره گرها و متغیرهای مرجع در زبان SQL تعریف شده اند، پیاده سازی آنها چگونه است؟

در زبان SQL ، اشاره گرها و متغیرهای مرجع به عنوان نوع داده های تعریف نشده وجود ندارند. با این حال، می توان آنها را با استفاده از توابع و عبارات SQL پیاده سازی کرد.

- اشاره گرها

اشاره گرها می توانند با استفاده از تابع ROW() پیاده سازی شوند. این تابع یک ردیف از یک جدول را به عنوان یک اشاره گر بازمی گرداند. به عنوان مثال، کد زیر یک اشاره گر به ردیف اول جدول PERSON ایجاد می کند:

```
SELECT ROW(1, 'John Doe', 30) FROM PERSON;
```

- متغیرهای مرجع

متغیرهای مرجع می توانند با استفاده از تابع REF() پیاده سازی شوند. این تابع یک اشاره گر به یک متغیر را ایجاد می کند. به عنوان مثال، کد زیر یک متغیر مرجع به متغیر x ایجاد می کند:

```
DECLARE x INT;
SELECT REF(x);
```

در زبان SQL چه سازوکارهایی برای رفع مشکلات نشتی حافظه و اشاره گر معلق وجود دارد؟

در زبانSQL ، دو سازوکار اصلی برای رفع مشکلات نشتی حافظه و اشاره گر معلق وجود دارد:

- حذف متغیرها و اشاره گرهای استفاده شده

این سازوکار ساده ترین و موثرترین روش برای جلوگیری از مشکلات نشتی حافظه و اشاره گر معلق است. به طور کلی، هر متغیر یا اشاره گر که دیگر استفاده نمی شود باید بلافاصله حذف شود. این کار می تواند با استفاده از دستورDROP انجام شود.

به عنوان مثال، کد زیر یک متغیر x ایجاد می کند و سپس آن را حذف می کند:

```
DECLARE x INT;
x = 1;
DROP x;
```

- استفاده از توابع FREE() و CLOSE()

این توابع در سیستم های مدیریت پایگاه داده (DBMS) مختلف برای آزاد کردن حافظه تخصیص یافته به متغیرها و اشاره گرها استفاده می شوند.

به عنوان مثال، کد زیر یک اشاره گر p به یک ردیف از جدول PERSON ایجاد می کند و سپس از تابع FREE() برای آزاد کردن حافظه تخصیص یافته به آن استفاده می کند:

```
DECLARE p ROW;
SELECT ROW(1, 'John Doe', 30) INTO p FROM PERSON;
FREE p;
```

سایر روش ها:

- استفاده از متغیرهای محلی

متغیرهای محلی تنها در محدوده بلوک کدی که در آن تعریف شده اند قابل دسترسی هستند. بنابراین، استفاده از متغیرهای محلی به جلوگیری از نشتی حافظه کمک می کند.

- عدم استفاده از اشاره گرهای بی‌نیاز

اشاره گرها به طور خودکار آزاد نمی شوند. بنابراین، باید از اشاره گرهایی که دیگر استفاده نمی شوند استفاده نکنید.

- استفاده از ابزارهای عیب‌یابی

ابزارهای عیب‌یابی می توانند به شما کمک کنند تا مشکلات نشتی حافظه و اشاره گر معلق را شناسایی کنید.