باسمه تعالی

تکلیف اول پایگاه داده یک
اسفند ۱۴۰۲

استاد درس
حمید روایی

سپهر عبادی
۹۹۳۳۲۴۳

سؤال ۱ :
توضیح دهید که چرا سیستم‌های **No-SQL** در دهه ۲۰۰۰ ظهور کردند و ویژگی‌های آن‌ها را با سیستم‌های پایگاه داده سنتی مقایسه کنید.

سیستم‌های پایگاه داده سنتی file-processing system ساده بود که سیستم رکوردها را در فایل‌های مختلف ذخیره می‌کند و به برنامه‌های متفاوتی برای استخراج رکوردها و افزودن رکوردها به فایل‌های مناسب نیاز دارد. نگهداری اطلاعات سازمانی در یک سیستم پردازش فایل دارای تعدادی معایب عمده است:

: Data redundancy and inconsistency

اطلاعات ممکن است در چندین فایل تکرار شود.

: Difficulty in accessing data

نیاز به نوشتن یک برنامه جدید برای انجام هر کار جدید

: Data isolation

از آنجایی که داده ها در فایل های مختلف پراکنده شده اند، و فایل ها ممکن است در فرمت های مختلف باشند، نوشتن برنامه های کاربردی جدید برای بازیابی داده های مناسب دشوار است.

: Integrity problems

مقادیر داده های ذخیره شده در پایگاه داده باید انواع خاصی از محدودیت های سازگاری را برآورده کند.

: Atomicity problems

یک سیستم کامپیوتری، مانند هر دستگاه دیگری، در معرض خرابی است. در بسیاری از برنامه‌ها، بسیار مهم است که در صورت بروز خطا، داده‌ها به حالت ثابتی که قبل از خرابی وجود داشت بازیابی شوند.

: Concurrent access by multiple users

تعامل به‌روزرسانی‌های همزمان ممکن است منجر به داده‌های متناقض شود.

Security problems

سؤال ۲ :

انواع **DBMS** ها را از جوانب مختلف در یک جدول باهم مقایسه کنید.

	Relational DBMS	**Object relational DBMS**	**Object-oriented DBMS**	**No-SQL DBMS**
Major Strengths	Leader in the database market Can handle diverse data needs	Based on established, proven technology, e.g., SQL Able to handle complex data	Able to handle complex data Direct support for object orientation	Able to handle complex data
Major Weaknesses	Cannot handle complex data No support for object orientation Impedance mismatch between tables and objects	Limited support for object orientation Impedance mismatch between tables and objects	Technology is still maturing Skills are hard to find	Technology is still maturing Skills are hard to find
Data Types Supported	Simple	Simple and Complex	Simple and Complex	Simple and Complex
Types of Application Systems Supported	Transaction processing and decision making	Transaction processing and decision making	Transaction processing and decision making	Primarily decision making
Existing Storage Formats	Organization dependent	Organization dependent	Organization dependent	Organization dependent
Future Needs	Good future prospects	Good future prospects	Good future prospects	Good future prospects

سؤال ۳ :
DBMS چگونه به درخواست کاربر پاسخ می‌دهد؟ (توضیح با رعایت ترتیب عملیات)

۱. پرس و جو (Querying): کاربران درخواست‌های خود را برای بازیابی، ویرایش، حذف یا اضافه کردن داده‌ها به سیستم می‌دهند. این درخواست‌ها معمولاً با استفاده از زبان پرس و جو (مانند SQL برای DBMS رابطه‌ای) فراهم می‌شوند. DBMS درخواست را می‌پذیرد و با استفاده از موارد دیگری مانند شاخص‌ها و مکانیزم‌های بهینه‌سازی، به بهترین روش ممکن اجرا می‌کند.

۲. تفسیر و اجرا:

– تفسیر: DBMS ابتدا درخواست را تفسیر می‌کند، به معنای تبدیل آن به یک فرمت قابل فهم برای سیستم.

– بررسی مجوزها: سیستم بررسی می‌کند که آیا کاربر مجاز به انجام عملیات موردنظر است یا خیر.

– اجرا: سپس، عملیات موردنظر انجام می‌شود. این شامل جستجو در داده‌ها، ویرایش آن‌ها، حذف یا اضافه کردن داده‌ها به پایگاه داده است.

۳. پردازش نتایج:

– پس از اجرای درخواست، نتایج به کاربر بازگردانده می‌شوند. این نتایج ممکن است شامل داده‌های بازیابی شده، پیام‌های خطا، یا دیگر اطلاعات مورد نیاز باشد.

– در صورتی که نتایجی باشد، آنها به صورت معمول به کاربر ارسال می‌شوند تا بتواند با آنها برای مقاصد موردنظر خود کار کند.

۴. بهینه‌سازی:

– DBMS ممکن است از بهینه‌سازی‌های مختلفی استفاده کند تا عملیات را سریع‌تر و کارآمدتر اجرا کند. این شامل استفاده از شاخص‌ها، نمایه‌ها، و بهینه‌سازی‌های دیگر در سطح سیستم است.

۵. تراکنش‌ها:

– در صورتی که درخواست مربوط به تراکنش باشد، DBMS مسئول اجرای تراکنش و تضمین اینکه عملیات‌ها به صورت ACID (Atomicity، Consistency، Isolation، Durability) انجام شوند.

در نهایت، DBMS به واسطه این فرآیندها و عملکردها، به درخواست‌های کاربران پاسخ می‌دهد و امکان مدیریت و دسترسی به داده‌ها را فراهم می‌کند.

سؤال ۴ :
ORM چیست و چه دغدغه هایی را برطرف می کند.چند مثال از ORM های موجود بزنید.

ORM یا Object-Relational Mapping یک تکنیک برنامه‌نویسی است که به برنامه‌نویسان امکان می‌دهد تا ارتباط بین داده‌های مدل شیءگرا و داده‌های موجود در پایگاه داده رابطه‌ای (RDBMS) را مدیریت کنند. به طور ساده، ORM این امکان را فراهم می‌کند که بتوانید با داده‌های پایگاه داده به شکل شیءها و کلاس‌ها در برنامه‌نویسی شیءگرا برخورد کنید.

با استفاده از ORM، برنامه‌نویسان نیازی به نوشتن کدهای پیچیده برای اتصال، استخراج، و تبدیل داده‌های موجود در پایگاه داده به داده‌های قابل استفاده در برنامه‌شان ندارند. به جای آن، آنها می‌توانند از مدل‌های شیءگرایی که با زبان‌های برنامه‌نویسی محبوبی مانند Java یا Python تعریف شده‌اند، استفاده کنند و ORM برای مدیریت ارتباط بین این مدل‌ها و پایگاه داده مربوطه مسئول باشد.

در زیر چند مثال از ORMهای معروف ذکر شده است:

۱. Hibernate: Hibernate یکی از معروف‌ترین ORMها برای زبان Java است. این ابزار به برنامه‌نویسان امکان می‌دهد تا با استفاده از کلاس‌ها و شیءها به جای جداول پایگاه داده کار کنند و پیچیدگی‌های مربوط به استفاده از SQL را پنهان کند.

۲. Entity Framework: Entity Framework یک ORM برای زبان برنامه‌نویسی C# است. این ابزار توسعه دهندگان را قادر می‌سازد تا با استفاده از مدل‌های شیءگرا که در کدهای C# تعریف شده‌اند، به صورت شفاف با پایگاه داده‌های Microsoft SQL Server و دیگر پایگاه‌های داده ارتباط برقرار کنند.

۳. Django ORM: Django یک چارچوب توسعه وب برای زبان برنامه‌نویسی Python است و ORM آن به برنامه‌نویسان اجازه می‌دهد تا با استفاده از کلاس‌ها و مدل‌های شیءگرا به جای جداول پایگاه داده کار کنند.

۴. SQLAlchemy: SQLAlchemy یک ابزار ORM برای زبان Python است که به برنامه‌نویسان اجازه می‌دهد تا با استفاده از مدل‌های شیءگرا به جای SQL بیانیه‌ها کار کنند و با پایگاه داده‌های مختلف ارتباط برقرار کنند.

سؤال ۵ :
مراحل دهگانه طراحی یک پایگاه داده را نام برده و هر کدام را مختصرا توضیح دهید.

1. **Identify Entities:**
   Identify the roles, events, locations tangible things or concepts about which the end-users want to store data.
2. **Find Relationships:**
   Find the natural associations between pairs of entities using a relationship matrix.
3. **Draw Rough ERD:**
   Put entities in rectangles and relationships on line segments connecting the entities.
4. **Fill in Cardinality:**
   Determine the number of occurrences of one entity for a single occurrence of the related entity.
5. **Define Primary Keys:**
   Identify the data attribute(s) that uniquely identify one and only one occurrence of each entity.
6. **Draw Key-Based ERD:**
   Eliminate Many-to-Many relationships and include primary and foreign keys in each entity.
7. **Identify Attributes:**
   Name the information details (fields) which are essential to the system under development.
8. **Map Attributes:**
   For each attribute, match it with exactly one entity that it describes.
9. **Draw fully attributed ERD:**
   Adjust the ERD from step 6 to account for entities or relationships discovered in step 8.
10. **Check Results:**
    Does the final Entity Relationship Diagram accurately depict the system data?

سؤال ۶ :

تفاوت بین **weak entity set** و **strong entity set** توضیح داده و با توجه به اینکه میتوان هر **weak entity set** را به **strong entity set** تبدیل کرد، این کار چه مشکلی ایجاد میکند؟

۱. Strong Entity Set :

یک موجودیت است که به تنهایی می‌تواند توسط یک شناسه یکتا (Primary Key) تعیین شود و به صورت مستقل از دیگر موجودیت‌ها در پایگاه داده ذخیره می‌شود.
Strong entity set معمولاً ویژگی‌هایی دارد که مستقل از موجودیت‌های دیگر است و به تنهایی می‌تواند از آنها اطلاعات را نگهداری کند.

۲. Weak Entity Set :

یک موجودیت است که برای تعیین هویت خود به یک موجودیت دیگر به نام "موجودیت صاحب" وابسته است. به عبارت دیگر، هویت یک موجودیت ضعیف فقط به دنبال ارتباط با یک موجودیت قوی (یا چندین موجودیت قوی) است.
در weak entity set، شناسه آن به عنوان کلید اصلی به تنهایی کافی نیست، بلکه به همراه شناسه موجودیت صاحب باید برای تعیین هویت استفاده شود.

حالا در مورد تبدیل weak entity set به strong entity set، اگر یک weak entity set را به strong entity set تبدیل کنیم، باید معماری داده‌ای را تغییر دهیم و مجموعه‌ای از ویژگی‌های دیگر را برای این موجودیت اضافه کنیم تا بتواند به صورت مستقل از موجودیت‌های دیگر در پایگاه داده ذخیره شود. این ممکن است باعث ایجاد پیچیدگی‌های معماری شود و ممکن است منجر به تغییرات گسترده در سیستم شود.

مشکلاتی که ممکن است ایجاد شود عبارتند از:
- افزایش پیچیدگی معماری و طراحی سیستم.
- نیاز به تغییرات گسترده در کدهای برنامه‌نویسی و در معماری پایگاه داده.

- افزایش احتمال خطاها و مشکلات اجرایی به دلیل تغییرات گسترده.

بنابراین، تبدیل weak entity set به strong entity set می‌تواند منجر به مشکلاتی مانند افزایش پیچیدگی و هزینه توسعه شود، بنابراین باید با دقت و با توجه به نیازها و الزامات سیستم این تغییرات را اعمال کرد.