به نام خدا

پاسخنامه‌ی تمرین اول درس طراحی پایگاه داده

نیم‌سال اول ۱۴۰۱-۱۴۰۲

سوال اول

به صورت کلی مدل های بسیار متفاوتی از ذخیره سازی داده وجود دارد که هر کدام نقاط ضعف و قوت خود را دارند و به همین دلیل کاربرد های متفاوتی پیدا کرده اند. در اینجا به صورت مختصر مدل های داده ای مطرح شده در صورت سوال را توضیح می‌دهیم که با آن‌ها کمی اشناتر شوید.

توجه داشته باشید: توضیح تمام جزییات یا تمام نقاط قوت و ضعف و حتی تمام کاربرهای هر کدام از این مدل های داده ای از حوصله‌ی این متن بسیار خارج است و صرفا هدف این است که با آن‌ها مقداری بیشتر آشنا شوید و همچنین این توانایی را پیدا کنید که با توجه به کاربرد بخصوص در پروژه‌های خود؛ بین مدل‌های داده‌ای مختلف انتخاب کنید.

Relational Model

این مدل یکی از معروف ترین مدل‌های رابطه ای است. در این مدل اطلاعات را به صورت یک سری جدول یا relation میبینیم و ارتباطات بین این داده ها را با استفاده از رابطه ی بین جدول‌ها نشان میدهیم.
در این مدل در سطح مختلف data integrity تعریف می شود:

1. Field Integrity (A.K.A domain integrity)
2. Table Integrity(A.K.A entity integrity)
3. Relation Integrity(A.K.A referential integrity)

این سه سطح از integrity که در این نوع از مدل های داده ای پیاده می شود یکی از نقاط قوت بزرگ این مدل است و معمولا در DBMS هایی که از این مدل استفاده می کنند پشتیبانی از transaction ها نیزبه صورت کامل پیاده سازی می شود که خود تضمینی است که در هر لحظه از زمان data integrity را کاملا داریم.
در مقابل این ویژگی های مثبت می‌توان گفت که اگر داده ها به نوعی باشند که روابط آن‌ها بسیار زیاد باشد(مانند دیتابیسی که قرار است نقشه ی خیابان های یک شهر را در خود ذخیره کند) جوابگو نیستند و سرعت جوابگویی به پرس‌جو های آن‌ها بسیار پایین می آید.
بعضی از DBMS هایی که این مدل را پیاده سازی میکنند عبارتند از:

Mysql, Postgres, Oracle SQL, Microsoft SQL server

Graph-based Model

در این مدل داده ها به صورت یک گراف(تعدادی گره یا node و تعدادی یال یا edge) نظر گرفته می‌شوند. در graph-based model خود داده ها در گره ها ذخیره می شود و ارتباط بین آن‌ها به صورت یال ها ذخیره می شوند به همین دلیل کاربرد این مدل بیشتر داده هایی است که به حدی ارتباط بین داده‌ها زیاد است که دیگر مدل رابطه ای برای آن‌ها جوابگو نیست.

معمولا از این مدل در recommendation engine ها ؛ ذخیره سازی نقشه‌ی جاده ها و ... یا کاربردهایی مشابه که ارتباط بین داده ها بسیار زیاد است و نیاز داریم که پرس‌جوهایی که از ارتباط داده‌ها مطرح می شود با سرعت بالایی جواب داده شوندپ؛ استفاده می‌شود.

بعضی از DBMS هایی که از این مدل استفاده را پیاده سازی میکنند عبارتند از:

Neo4j, Apache Giraph, ArangoDB, Amazon Neptune, Titan

Object Oriented Model

در این مدل سعی می شود که مفاهیم Object Oriented Programming مانند:

- Polymorphism
- Inheritance
- Encapsulation

را با مفاهیم پایه ای تر ؛ که معمولا در مدل رابطه ای دیده می شود ؛ ترکیب کنند.

به صورت کلی فلسفه ی این مدل این است که در لایه ی دیتابیس دقیقا همانند لایه ی نرم افزار Object و Class ها را داشته باشیم و به همین دلیل در این مدل اجازه ی این موضوع داده می شود که هر Object همزمان هم state فعلی آن داخل دیتابیس ذخیره شود و هم behaviour آن که از طریق method های آن است نیز ذخیره شود.

از فلسفه های بزرگ این مدل می‌توان به این اشاره کرد که با توجه به اینکه ساختار پایگاه داده بسیار شبیه به ساختار object ها در نرم افزار است؛ کدی که برای ذخیره سازی object ها در لایه ی نرم افزار خواهیم داشت بسیار ساده تر خواهد بود.

از معایب این مدل می‌توان به این اشاره کرد که ذخیره کردن هر object همراه با behaviour های آن می‌تواند overhead زیادی را به DBMS اضافه کند و آن را به شدت کند می‌کند و امروزه تمام زبان‌هایی که برای تولید نرم افزار استفاده می‌شوند لزوما Object Oriented نیستند.

مثال هایی از تکنولوژی هایی که از این مدل استفاده میکنند عبارتند از:

WakandaDB, ObjectDB, ZODB

که اکثر این تکنولوژی ها برای اینکه overhead خود را کمی کم تر کنند برای object های یک زبان بخصوص پشتیبانی دارند و همه‌ی زبان ها را پشتیبانی نمی کنند.

Key Value Model

در این مدل ؛ داده ها به صورت key-value pair دیده می شوند و روابط بین داده‌ها تقریبا به صورت کامل نادیده گرفته می شود. معمولا در پیاده سازی این مدل از یک HashMap و Hash Function بسیار قدرتمند استفاده می شود که در واقع به ما اجازه می دهند که با اطمینان بالایی هر دیتایی را در O(1) به دست آوریم.

در مقابل این سرعت بالا؛ HashMap ها نیاز به فضای بسیار زیادی برای ذخیره سازی دارند و در صورتی که حجم داده ای که میخواهیم ذخیره سازی کنیم زیاد باشد یا روابط زیادی بین آن‌ها برقرار باشد میزان حافظه ی مورد نیاز بسیار زیاد می شود و عملا کارکردن با این حجم از داده غیر ممکن می شود. به همین دلیل از این مدل معمولا در کنار مدل های دیگر به عنوان یک حافظه ی cache استفاده می شود و معمولا داده‌هایی که با استفاده از این مدل ذخیره سازی شده‌اند در RAM که دسترسی به ان بسیار سریع‌تر نسبت به hard disk است؛ ذخیره سازی می شوند.

دو بازیگر بزرگ در این زمینه دو تکنولوژی زیر هستند:

Redis, Memcache

که هر دو بیشتر به عنوان یک cache عمل می کنند.

Time series Model

در بعضی از سناریوها نرم افزار در حال دریافت یا تولید اطلاعات زیادی در طول زمان است و برای ما گرفتن داده ی مربوط به یک زمان مشخص اهمیت کمتری نسبت به دیدن روند تغییرات در طول بازه ای زمانی دارد؛ در این سناریو استفاده از مدل time series بسیار مناسب است.

در این نوع از دیتابیس ها هر داده ای همراه با یک time tag ذخیره سازی می شود و معمولا بعدا این امکان وجود دارد که در مورد داده ی ذخیره شده در طول زمان پرس‌جو کرد. (به طور مثال تغییرات

میانگین کاربران انلاین در یک ساعت اخیر؛ تعداد پیام های رد و بدل شده در یک نرم افزار پیام رسانی در ۱۰ روز اخیر؛ تعداد request هایی که توسط یک server در هر لحظه در طول ده دیقه ی اخیر handle شده اند و ...)

در این مدل ارتباط بین داده ها در نظر گرفته نمی شود و بیشتر این ویژگی در نظر گرفته می شود که یک ویژگی بخصوص در هر لحظه از زمان چه مقداری داشته است.

به همین دلیل مزیت این مدل در شرایطی است که داده هایی که با آن‌ها سروکار داریم در طول زمان بسیار تغییر می‌کنند و مدل‌های قبلی کارآمدی لازم برای ذخیره سازی این حجم از تغییرات در طول زمان را ندارند.

در صنعت معمولا اطلاعات دریافتی از سنسورها را یا اطلاعاتی که یک نرم افزار از شیوه ی کار خود تولید می کند(به طور مثال یک تابع بخصوص چند بار توسط کاربران مختلف در ده روز اخیر فراخوانی شده است و ...)را در این نوع از database ها ذخیره سازی می‌کنند.

بعضی تکنولوژی هایی که این مدل را پیاده سازی میکنند:

Prometheus, TimescaleDB, Graphite

Document Oriented Model

در این مدل داده ای داده ها به صورت مجموعه ای از document ها ذخیره سازی می شوند که معمولا این داده‌ها به صورت فایل هایی از جنس های زیر ذخیره سازی می شوند:

- JSON (javascript object notation)
- XML (Extensible Markup Language)

در این مدل روابط بین document ها معمولا وجود ندارد و سعی می شود که روابط بین داده ها در همان document ذخیره سازی شود.

این روش این امکان را به DBMS می دهد که پرس‌جو هایی که نیاز به دانستن روابط مختلف بین داده‌های را دارد را بسیار سریع تر و با overhead بسیار کمتری نسبت به مدل رابطه ای جواب بدهد.

معمولا DBMS هایی که از این مدل استفاده می‌کنند بسیار ساده و سریع و در عین پرقدرت هستند و به صورت کلی setup اولیه ی آن‌ها بسیار ساده است به همین دلیل این مدل در حال تبدیل شدن به یکی از معروف ترین مدل‌های رابطه ای در بین عموم برنامه نویسان است.

به دلیل ساختار و معماری این مدل امکان پشتیبانی از transaction ها یا اصلا وجود ندارد یا بسیار ضعیف تر از آنچه که DBMS های رابطه ای ارایه می‌دهند؛ ارایه داده شده اند به همین دلیل معمولا در حالت هایی که high consistency از ویژگی های حیاتی مورد نیاز ما باشد از این نوع database استفاده نمی‌شود.

از معروف ترین تکنولوژی هایی که از این مدل استفاده میکنند؛ عبارتند از:

MongoDB, Amazon DynamoDB, CouchDB, Aerospike

سوال دوم

سوال اول

به صورت کلی DBMS خود یک نرم افزار است که نیاز به ذخیره سازی داده های مختلفی در راستای کارکرد خودش دارد. معمولا DBMS ها از data dictionary برای ذخیره سازی meta data ی خود در مورد جدول ها؛ روابط ؛ ستون‌ها و ... استفاده می کنند.

سوال دوم

اینکه دقیقا هر DBMS چه چیزی را در data dictionary خود ذخیره می‌کند به جزییات پیاده سازی آن DBMS و حتی فلسفه‌ی ساخت آن برمیگردد اما به صورت کلی می‌توان گفت که اطلاعات زیر معمولا در data dictionary ها ذخیره می شوند:

- Field Name
- Data Format
- Column Domain
- Field Size (Actual number of bytes used for storing this data)
- Column Description
- Relation name
- Type of Relationship between tables
- etc

سوال سوم

با توجه به اینکه Data Dictionary حاوی اطلاعات مهمی از قبیل Relation Name و Column Domain و ... است حفظ کردن data integrity برای DBMS بدون داشتن این نوع data غیرممکن خواهد بود.

به صورت کلی هنگامی که DBMS قصد دارد یک سطر جدید در یک realation ایجاد کند ابتدا باید داده‌های جدید را با توجه به سه سطح data integrity مطرح شده در سوال قبل سنجش کند و اگر data

dictionary وجود نداشته باشد دیگر DBMS این توانایی را از دست می‌دهد که بفهمد ایا اطلاعات سطر جدید سه سطح integrity را بهم میزنند یا نه.

به همین دلیل Data dictionary با استفاده از اطلاعاتی که در خورد دارد به DBMS کمک می‌کند که Data integrityدر تمام سطوح حفظ کند.