سوال اول

الف) Graph Database Model
در مدل گراف همانطور که از اسم آن پیداست داده‌ها به صورت گراف یعنی مجموعه‌ای از راس‌ها و همچنین یال‌ها که نشان دهنده روابط بین راس‌هاست ذخیره شده و نشان داده می‌شوند. درحالیکه در مدل رابطه‌ای داده‌ها به صورت جداول ذخیره می‌شوند. یکی از تفاوت‌های اساسی مدل گراف و مدل رابطه‌ای در میزان ساختارمندی آن‌هاست. در مدل گراف ما هیچ ساختار بسیار سختگیرانه یا به اصطلاح restiricted schema نداریم و به همین خاطر انعطاف‌پذیری این مدل بسیار بیشتر از مدل رابطه‌ای است. تمرکز اصلی مدل گراف در نحوه پیاده سازی روابط بین موجودیت‌هاست تا خود موجودیت‌ها برخلاف مدل رابطه‌ای که در آن تعریف روابط به خاطر پیش آمدن عمل join که معمولا عملی پرهزینه است در مدل گراف روابط به صورت صریح ذخیره می‌شوند به همین خاطر انجام query هایی که مربوط به روابط بین موجودیت‌ها باشند در مدل گراف بسیار بهینه است.
یکی از معایب اصلی مدل گراف نبود یک زبان استاندارد و مشترک است بر خلاف SQL که زبان استاندارد پایگاه‌های داده مبتنی بر مدل رابطه‌ای است. در مدل گراف بسته به ابزار مورد استفاده زبان‌های متفاوتی وجود دارند. یکی از پراستفاده ترین پایگاه‌های داده مبتنی بر مدل گراف Neo4j می‌باشد این پایگاه داده با زبان Java و Scala توسعه داده شده. در این پایگاه داده‌ها از زبان Cypher برای پرس و جو استفاده می‌شود این زبان به طور ویژه برای استفاده در مدل گراف بهینه شده‌است.

ب) Key-Value Model
در این مدل داده‌ها به صورت جفت‌هایی از key ها و value ها ذخیره می‌شوند. با استفاده از key می‌توان به داده‌ها دسترسی پیدا کرد و عملیات‌های مختلف همچون delete, insert, edit را اجرا کرد. key می‌تواند از هر نوعی باشد اما باید یکتا باشد چرا که تمیز دادن رکوردهای مختلف داده توسط این key امکان پذیر است. value می‌تواند انواع ساده‌ای مانند عدد یا استرینگ یا انواع پیچیده و ترکیبی باشد. این مدل معمولا در عموم زبان‌های برنامه نویسی به صورت داده ساختارهایی مانند دیکشنری موجود است. اما زمانی که از Key-Value Database صحبت می‌کنیم منظور این است که قصد داریم داده‌ها را به صورت persistant ذخیره کنیم و این داده‌ها قرار است توسط یک DBMS کنترل و مدیریت شوند. تفاوت عمده‌ای که این مدل با مدل رابطه‌ای دارد نبود یک restiricted schema است در واقع برخلاف مدل رابطه‌ای که در آن هر جدول دارای ساختار مشخص است در مدل Key-Value هر رکورد می‌تواند schema ی متفاوت و مختص به خود را داشته باشد و نیازی نیست که تمام رکوردها ازین ساختار پیروی کنند. یکی از نقاط قوت این مدل توانایی horizontally scale کردن بسیار آسان و کم هزینه است. علت این امر قابلیت تقسیم بسیار بالای آن است چرا که رکورد ها وابستگی بسیار کمی به هم دارند. یکی از زمینه های پر استفاده این مدل هنگامی است که نیاز به یک مکانیزم caching داریم که قرار با تعداد و فرکانس بسیار بالا به آن دسترسی

داشته باشیم. یکی از محبوب‌ترین پایگاه داده‌های این مدل پایگاه داده Redis می‌باشد. Redis یک پایگاه داده in-memory است به همین خاطر سرعت و پرفورمنس بسیار بالایی دارد.

ج) Time Series Model
در این مدل همانطور که از نامش برمیاد سری های زمانی داده‌های مورد استفاده ما هستند. سری زمانی در واقع داده ای است که دارای یک کلید زمان یا timestamp می‌باشد که زمان وقوع اطلاعات داده را نشان می‌دهد و همینطور value که می‌تواند هرچیزی باشد. در دنیای امروزه در موارد زیادی داده‌های ما به صورت سری زمانی هستند. به عنوان مثال در بحث IOT ما نیاز داریم داده‌های سنسورها و device های مختلف موجود در شبکه را که در بازه‌های زمانی متوالی ایجاد و ارسال می‌شوند ذخیره کرده و برای پردازش‌های بعدی آماده کنیم. در زمینه بورس ومسائل financial هم سری‌های زمانی نقش پررنگی دارند به عنوان مثال قیمت یک سهام خاص در بازه‌های مختلف زمانی.
اما تفاوت اساسی سری‌های زمانی با داده های عادی چیست. در سری های زمانی کلید اصلی همواره زمان یا timestamp می‌باشد. در این نوع از داده ما همواره داده‌های جدیدی را ذخیره می‌کنیم و معمولا داده‌های قدیمی را ویرایش یا edit نمی‌کنیم چراکه یک رویداد در یک زمان خاص رخ داده و امکان تغییر آن وجود ندارد. در سری های زمانی کلید اصلی که همان timestamp می‌باشد معمولا به صورت صعودی می‌باشد چراکه در روند ایجاد داده‌ها زمان همواره رو به جلو حرکت می‌کند. مساله بعدی حجم بسیار بالای اطلاعات ورودی در زمان کم است یک سامانه‌ی IOT ممکن است شامل هزاران سنسور باشد که هر کدام قرار است در هر ثانیه اطلاعات ثبت شده خود را به سمت سرور ارسال کنند در نتیجه scalibility اهمیت بسیار بالایی در پایگاه داده‌های سری زمانی دارد.
یکی از پر استفاده ترین پایگاه داده‌های سری زمانی پایگاه داده Prometheus می‌باشد. این پایگاه داده به طور ویژه برای پایش و نظارت (Monitoring) سامانه‌های نرم افزاری ایجاد شده و با زبان Golang توسعه داده شده است. این پایگاه داده اطلاعات مربوط به سلامت و کارایی هر سرویس نرم افزاری (Metrics) را در بازه‌های زمانی مشخص از طریق پروتکل HTTP جمع‌آوری (Scrape) می‌کند. در نتیجه می‌توان در آینده روی این داده‌ها تحلیل انجام داده یا آن‌ها مصورسازی (Visualization) کرد. یکی از کاربردهای این داده‌ها ایجاد یک اخطار (Alert) برای سرویس نرم‌افزاری است. به عنوان مثال اگر در بازه‌ی زمانی مشخص مدت زمان پاسخ (Response Time) یک سرویس از حد مورد انتظار ما بالاتر بود یک اخطار ایجاد شده و مارا از وجود مشکل در سلامتی سیستم آگاه می‌کند.

سوال دوم

در واقع Data Dictionary یک فایل یا مجموعه‌ای از فایل‌هاست که شامل فراداده‌ها یا metadata های مربوط به یک پایگاه داده است. به عنوان مثال اینکه چه جداولی در پایگاه داده موجود هستند، چه کاربرانی به چه داده‌هایی دسترسی دارند، محدودیت‌های مربوط به هر داده چیست و غیره. در Data Dictionary همچنین معنی و هدف داده‌های موجود در پایگاه داده هم وجود دارد. در Data

Dictionary اطلاعاتی درباره نحوه ذخیره سازی فیزیکی داده‌ها نیز وجود دارد. در اکثر اوقات کاربران با سطح دسترسی معمولی به Data Dictionary دسترسی ندارند و وظیفه مدیریت و کنترل این ماژول به عهده DBA یا Database Administrator ها است.

به طور دقیقتر در Data Dictionary اطلاعاتی نظیر

نام تمام جداول پایگاه داده و Schema ی مربوط به آن‌ها

جزئیاتی مربوط به هر جدول از جمله صاحبان آن‌ها، محدودیت‌های امنیتی و زمان ایجاد آن

جزئیاتی مربوط به نحوه ذخیره سازی فیزیکی جداول

محدودیت‌های هر جدول و اطلاعات کلیدهای اصلی و کلیدهای خارجی

وجود Data Dictionary با ایجاد آگاهی از Schema جداول و همچنین محدودیت‌های هر داده و سطوح دسترسی باعث ایجاد Data Integrity در داده‌ها می‌شود چراکه بدون این اطلاعات مدیریت و کنترل و همچین Validation داده‌ها در پایگاه داده امکان‌پذیر نیست.