عنوان: **NOSQL قسمت اول**
نویسنده: حمید سامانی
تاریخ: ۱۳۹۱/۱۱/۲۴ ۱۷:۴۰
آدرس: www.dotnettips.info
گروه‌ها: NoSQL, Database, پایگاه‌داده, نوسی‌کوال, نواس‌کیوال

در این پست نگاهی کلی به ویژگی‌های پایگاه‌های داده NOSql خواهیم داشت و با بررسی تاریخچه و دلیل پیدایش این سیستم‌ها آشنا خواهیم شد.
با فراگیر شدن اینترنت در سال‌های اخیر و افزایش کاربران ، سیستم‌های RDBMS جوابگوی نیازهای برنامه‌نویسان در حوزه‌ی وب نبودند زیرا نیاز به نگهداری داده‌ها با حجم بالا و سرعت خواندن و نوشتن بالا از جمله نقط ضعف سیستم‌های RDBMS میباشد ، چرا که با افزایش شدید کاربران داده‌ها اصولا به صورت منطقی ساختار یکدست خود را جهت نگه‌داری از دست می‌دهند و به این ترتیب عملیات نرمال سازی منجر به ساخت جداول زیادی می‌شود که نتیجه آن برای هر کوئری عملیات Joinهای متعدد می‌باشد که سرعت خواندن و نوشتن را به خصوص برای برنامه‌های با گستره‌ی وب پایین می‌آورد و مشکلات دیگری در سیستم‌های RDBMS که ویژگی‌های سیستم‌های NoSql مشخص کننده آن مشکلات است که در ادامه به آن می‌پردازیم.
طبق تعریف کلی پایگاه داده NOSql عبارت است از:
نسل بعدی پایگاه داده (نسل از بعد RDBMS) که اصولا دارای چند ویژگی زیر باشد:
۱- داده‌ها در این سیستم به صورت رابطه‌ای (جدولی) نمی‌باشند
۲-داده‌ها به صورت توزیع شده نگهداری می‌شوند.
۳-سیستم نرم‌افزاری متن باز می‌باشد.
۴-پایگاه داده مقیاس پذیر به صورت افقی می‌باشد(در مطالب بعدی توضیح داده خواهد شد.)

همان‌گونه که گفته شد این نوع پایگاه داده به منظور رفع نیازهای برنامه‌های با حجم ورود و خروج داده بسیار بالا (برنامه‌های مدرن وب فعلی) ایجاد شدند.
شروع کار پیاده‌سازی این سیستم‌ها در اوایل سال ۲۰۰۹ شکل گرفت و با سرعت زیادی رشد کرد و همچنین ویژگی‌های کلی دیگری نیز به این نوع سیستم اضافه شد.
که این ویژگی‌ها عبارتند از:
Schema-free : بدون شَما ! ، با توجه به برنامه‌های وبی فعلی ممکن است شمای نگه‌داری داده‌ها (ساختار کلی) مرتبا و یا گهگاهی تغییر کند. لذا در این سیستم‌ها اصولا داده‌ها بدون شمای اولیه طراحی و ذخیره می‌شوند. (به عنوان مثال می‌توان در یک سیستم که مشخصات کاربران وارد سیستم می‌شود برای یک کاربر یک سری اطلاعات اضافی و برای کاربری دیگر از ورود اطلاعات اضافی صرف‌نظر کرد ، و در مقایسه با RDBMS به این ترتیب از ورود مقادیر Null و یا پیوندهای بیمورد جلوگیری کرد.
کنترل اطلاعات الزامی توسط لایه سرویس برنامه انجام می‌شود. (در زبان جاوا توسط jsr-303 و یاBean Validation ها)
easy replication support : در این سیستم ، نحوه‌ی گرفتن نسخه‌های پشتیبان و sync بودن نسخه‌های مختلف بسیار ساده و سر راست می‌باشد و سرور پایگاه داده به محض عدم توانایی خواندن و یا نوشتن از روی دیسک سراغ نسخه‌ی پشتیبان می‌رود و آن نسخه را به عنوان نسخه‌ی اصلی در نظر می‌گیرد.
Simple API : به دلیل متن‌باز بودن و فعال بودن Community این سیستم‌ها APIهای ساده و بهینه‌ای برای اکثر زبان‌های برنامه‌نویس محبوب ایجاد شده است که در پست‌های بعدی با ارائه مثال آنها را بررسی خواهیم کرد.
eventually consistent : در سیستم‌های RDBMS که داده‌ها خاصیت ACID را (در قالب Transaction) پیاده می‌کنند ، در این سیستم‌های داده‌ها در وضعیت BASE قرار دارند که سرنام کلمات Eventual Consistency ، Soft State ، Basically Available می‌باشد.
huge amount of data : این سیستم‌ها به منظور کار با داده‌های با حجم بالا ایجاد شده‌اند ، یک تعریف کلی می‌گوید اگر مقدار داده‌های نگهداری شده در پایگاه‌های داده برنامه شما ظرفیتی کمتر از یک ترابایت داده دارد از پایگاه داده RDBMS استفاده کنید واگر ظرفیت آن از واحد ترابایت فراتر می‌رود از سیستم‌های NOSql استفاده کنید.
به طور کلی پایگاه داده‌ای که در چارچوب موارد ذکر شده قرار گیرد را می‌توان از نوع NoSql که سرنام کلمه (Not Only SQL) می‌باشد قرار داد. تاکنون پیاده‌سازی‌های زیادی از این سیستم‌ها ایجاد شده است که رفتار و نحوه‌ی نگه‌داری داده‌ها (پرس‌وجو ها) در این سیستم‌ها با یکدیگر متفاوت می‌باشد.

جهت پیاده سازی پایگاه داده با این سیستم‌ها تا حدودی نگرش کلی به داده‌ها و نحوه‌ی چیدمان آنها تغییر می‌کند ، به صورت کلی

مباحث مربوط به normalization و de-normalization و تصور داده‌ها به صورت جدولی کنار می‌رود.
سیستم NoSql به جهت دسته‌بندی نحوه‌ی ذخیره‌سازی داده‌ها و ارتباط بین آنها به ۴ دسته کلی تقسیم می‌شود که معرفی کلی آن دسته‌بندی‌ها موضوع مطلب بعدی می‌باشد.

نظرات خوانندگان

نویسنده: هاشمی
تاریخ: ۱۳۹۴/۰۲/۰۲ ۱۲:۴۰

سلام
من یه سوال دارم
خاصیت Base که شامل Eventual Consistency ، Soft State ، Basically Available به چه معنی هست؟ میشه توضیح بدید؟

نویسنده: محسن خان
تاریخ: ۱۳۹۴/۰۲/۰۲ ۱۳:۳۵

به مطلب مروری بر مفاهیم مقدماتی NoSQL مراجعه کنید.

عنوان: **NOSQL قسمت دوم**
نویسنده: حمید سامانی
تاریخ: ۱۳۹۱/۱۱/۲۶ ۹:۲۵
آدرس: www.dotnettips.info
گروه‌ها: NoSQL, Database, پایگاه‌داده, نوسی‌کوال, نواس‌کیوال, key-value, کلید-مقدار

در مطلب قبلی با تعاریف سیستم‌های NoSQL آشنا شدیم و به طور کلی ویژگی‌های یک سیستم NoSQL را بررسی کردیم.

در این مطلب دسته‌بندی کلی و نوع ساختار داده‌ای این سیستم‌ها و بررسی ساده‌ترین آنها را مرور می‌کنیم.

در حالت کلی پایگاهای داده NoSQL به ۴ دسته تقسیم می‌شوند که به ترتیب پیچیدگی ذخیره‌سازی داده‌ها عبارتند از:
Key/Value Store Databases

Document Databases
Graph Databases
Column Family Databases
در حالت کلی در پایگاه‌های‌داده NoSQL داده‌ها در قالب KEY/VALUE (کلید/مقدار) نگه‌داری می‌شوند ، به این صورت که مقادیر توسط کلید یکتایی نگاشت شده و ذخیره می‌شوند، هر مقدار صرفا توسط همان کلید نگاشت شده قابل بازگردانی می‌باشد و راهی جهت دریافت مقدار بدون دانستن کلید وجود ندارد . در این ساختارداده منظور از مقادیر، داده‌های اصلی برنامه هستند که نیاز به نگه‌داری دارند و کلیدها نیز رشته‌هایی هستند که توسط برنامه‌نویس ایجاد می‌شوند.
به دلیل موجود بودن این نوع ساختار داده‌ای در اکثر کتابخانه‌های زبان‌های برنامه‌نویسی (به عنوان مثال پیاده‌سازی‌های مختلف اینترفیس Map شامل HashMap ، HashTable و موارد دیگر در کتابخانه‌های JDK) این نوع ساختار برای اکثر برنامه‌نویسان آشنا بوده و فراگیری آن نیز ساده می‌باشد.

بدیهی است که اعمال فرهنگ داده‌ای (درج ، حذف ، جستجو) در این سیستم به دلیل اینکه داده‌ها به صورت کلید/مقدار ذخیره می‌شوند دارای پیچیدگی زمانی O(1) می‌باشد که بهینه‌ترین حالت ممکن به لحاظ طراحی می‌باشد. همان‌گونه که مستحضرید در الگوریتم‌هایی که دارای پیچیدگی زمانی با مقدار ثابت دارند کم یا زیاد بودن داده‌ها تاثیری در کارایی الگوریتم نداشته و همواره با هر حجم داده‌ای زمان ثابتی جهت پردازش نیاز می‌باشد.

:Key/Value Store Databases

این سیستم ساده‌ترین حالت از دسته‌بندی‌های NoSQL می‌باشد ، به طور کلی جهت استفاده در سیستم‌هایی است که داده‌ها متمایز از یکدیگر هستند و اصولا Availability و یا در دسترس بودن داده‌ها نسبت به سایر موارد نظیر پایائی اهمیت بالاتری دارد.

از موارد استفاده این گونه سیستم‌ها به موارد زیر می‌توان اشاره کرد:
در پلتفرم‌های اشتراک گذاری داده‌ها . هدف کلی صرفا هندل کردن آپلود محتوی (باینری) و به صورت همزمان بروز کردن در سمت دیگر می‌باشد.(اپلیکیشنی مانند اینستاگرام را تصور کنید) در اینگونه نرم‌افزارها با تعداد بسیار زیاد کاربر و تقاضا، استفاده از این نوع پایگاه داده به مراتب کارایی و سرعت را بالاتر می‌برد. و با توجه به عدم پیش‌بینی حجم داده‌ها یکی از ویژگی‌های این نوع پایگاه داده تحت عنوان Horizontal Scaling مطرح می‌شود که در صورت Overflow شدن سرور، داده‌ها را به سمت سرور دیگری می‌توان هدایت کرد وبدون مشکل پردازش را ادامه داد ، این ویژگی یک وجه تمایز کارایی این سیستم با سیستم‌های RDBMS می‌باشد که جهت مقابله با چنین وضعیتی تنها راه پیش‌رو بالا بردن امکانات سرور می‌باشد و به طور کلی داده‌ها را در یک سرور می‌توان نگه‌داری کرد (البته راه‌حل‌هایی نظیر پارتیشن کردن و غیره وجود دارد که به مراتب پیچیدگی و کارایی کمتری نسبت به Horizontal Scaling در پایگاه‌های داده NoSQL دارد.)

برای Cache کردن صفحات بسیار کارا می‌باشد ، به عنوان مثال می‌توان آدرس درخواست را به عنوان Key در نظر گرفت و مقدار آن را نیز معادل JSON نتیجه که توسط کلاینت پردازش خواهد شد قرار داد.

یک نسخه کپی شده از توئیتر که کاملا توسط این نوع پایگاه داده پیاده شده است نیز از این آدرس قابل مشاهده است. این برنامه به زبان‌های ruby , php و java نوشته شده است و سورس نیز در مخارن github می‌جود می‌باشد. (یک نمونه پیاده سازی ایده‌آل جهت آشنایی با نحوه‌ی مدیریت داده‌ها در این نوع پایگاه داده)

از پیاده‌سازی‌های این نوع پایگاه داده به موارد زیر می‌توان اشاره کرد:

Amazon SimpleDB

Memcached

Oracle Key/value Pair

Redis

هر یک از پیاده‌سازی‌ها دارای ویژگی‌های مربوط به خود هستند به عنوان مثال Memcached داده‌ها را صرفا در DRAM ذخیره می‌کند که نتیجه‌ی آن Volatile بودن داده‌ها می‌باشد و به هیچ وجه از این سیستم جهت نگهداری دائمی داده‌ها نباید استفاده شود. از طرف دیگر Redis داده‌ها را علاوه بر حافظه اصلی در حافظه جانبی نیز ذخیره می‌کند که نتیجه‌ی آن سرعت بالا در کنار پایائی می‌باشد.

همان‌گونه که در تعریف کلی عنوان شد یکی از ویژگی‌های این سیستم‌ها متن‌باز بودن انها می‌باشد که نتیجه‌ی آن وجود پیاده‌سازی‌های متنوع از هر کدام می‌باشد ، لازم است قبل از انتخاب هر سیستم به خوبی با ویژگی‌های اکثر سیستم‌های محبوب و پراستفاده آشنا شویم و با توجه به نیاز سیستم را انتخاب کنیم.

در مطلب بعدی با نوع دوم یعنی Document Databases آشنا خواهیم شد.

نظرات خوانندگان

نویسنده: مجید هزاری
تاریخ: ۱۳۹۱/۱۱/۲۸ ۱۵:۴۷

عالی است.
متشکرم.

نویسنده: احمد ولی پور
تاریخ: ۱۳۹۱/۱۱/۲۸ ۱۷:۵۵

یه سوال برام پیش اومده:
با رایج شدن nosql پایگاه داده هایی مثل Oracle یا Sql Server چی میشن؟

نویسنده: مجید هزاری
تاریخ: ۱۳۹۱/۱۱/۲۸ ۱۹:۵۰

اینها تداخلی با یکدیگر ندارند.
NOSQL تنها برای رفع نیاز هایی ظهور کرده است که RelDB در آنها ضعیف بوده. همانطور که NoSQL در زمینه هایی که RelDB قوی است ضعیف عمل خواهد کرد.
(البته من کاملا مختصر گفتم)

نویسنده: حمید سامانی
تاریخ: ۱۳۹۱/۱۱/۲۸ ۲۰:۱۷

در حالت کلی هرکدام از پایگاه‌داده‌ها بسته به نیاز استفاده می‌شن ، توی برنامه‌های اینترپرایز وبی مفهوم Polyglot Persistence مطرحه (که می‌شه اونو نگه‌داری یا ذخیره سازی چند زبانی ترجمه کرد) که می‌گه توی یک سیستم از چندین نوع پایگاه داده ~~می‌شه~~ (باید) استفاده کرد. به عنوان مثال برای نگه‌داری داده‌هایی جهت گزارش گیری و یا ایجاد Transactionها بهترین گزینه همان سیستم‌های RDBMS هستند ، در مطالب آتی به این موضوع اشاره بیشتری خواهم کرد ، مارتین فویلر در این مطلب مفهوم Polyglot Persistence را به خوبی توضیح داده‌اند.

نویسنده: saremi
تاریخ: ۱۳۹۲/۱۱/۲۹ ۱۷:۲۲

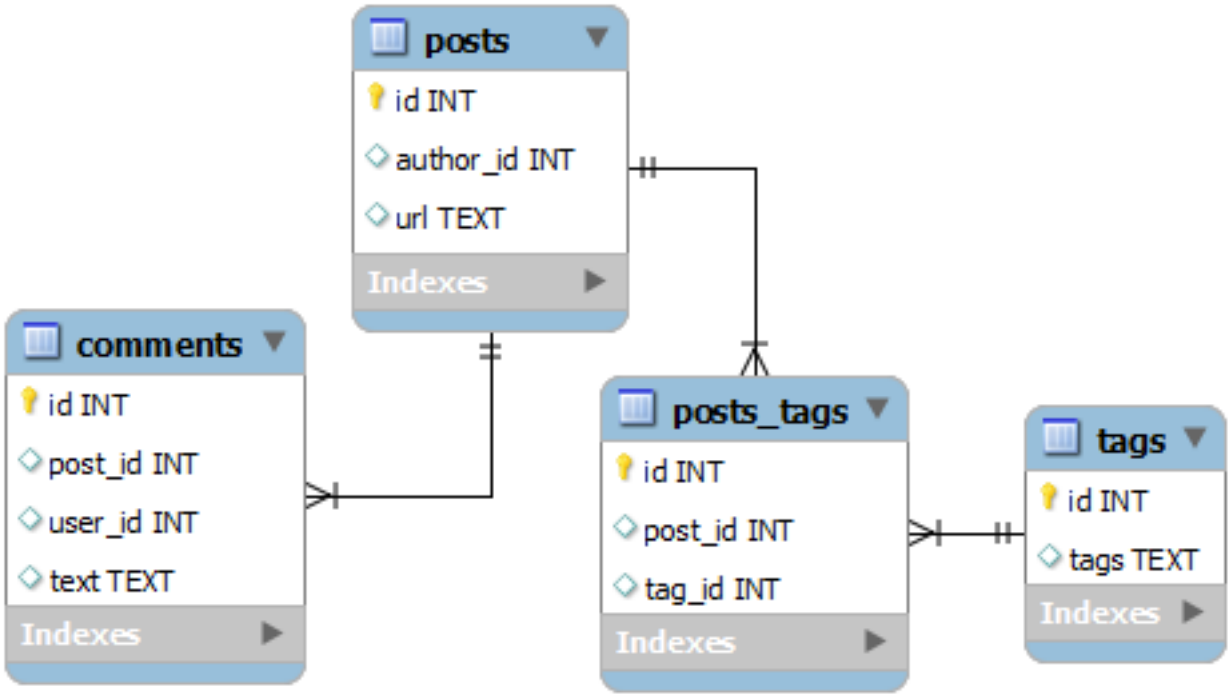
سلام
می خواستم بپرسم bucket در key value store دقیقا چیه؟
بعد خیلی از جاها راجع به hash table و hash code هم مطالبی گفته اند. آیا منظور فقط hash کردن کلید است یا فرآیند پیچیده‌تر از این حرفاست؟

عنوان: **NOSQL قسمت سوم**
نویسنده: حمید سامانی
تاریخ: ۱۳۹۱/۱۱/۲۹ ۱۹:۱۰
آدرس: www.dotnettips.info
گروه‌ها: NoSQL, Database, نوسی‌کوال, نواس‌کیوال, mongodb, پایگاه

در مطلب قبلی با نوع اول پایگاه‌های‌داده NoSQL یعنی Key/Value Store آشنا شدیم و در این مطلب به معرفی دسته دوم یعنی Document Database خواهیم پرداخت.

در این نوع پایگاه داده ، داده‌ها مانند نوع اول در قالب کلید/مقدار ذخیره می‌شوند و بازگردانی مقادیر نیز دقیقا مشابه نوع اول یعنی Key/Value Store بر اساس کلید م ی‌باشد. اما تفاوت این سیستم با نوع اول در دسته‌بندی داده‌های مرتبط با یکدیگر در قالب یک Document می‌باشد. سعی کردم در این مطلب با ذکر مثال مطالب را شفاف‌تر بیان کنم:

به عنوان مثال اگر بخواهیم جداول مربوط به پست‌های یک سیستم CMS را بصورت رابطه‌ای پیاده کنیم ، یکی از ساده‌ترین حالات پایه برای پست‌های این سیستم در حالت نرمال به صورت زیر می‌باشد.



جداول واضح بوده و نیازی به توضیح ندارد ، حال نحوه‌ی ذخیره‌سازی داده‌ها در سیستم Document Database برای چنین مثالی را بررسی می‌کنیم:

```
{
_id: ObjectID(‘4bf9e8e17cef4644108761bb’),
Title: ‘NoSQL Part3’,
url: ‘http://dotnettips.info/yyy/xxxx’,
author: ‘hamid samani’,
tags: [‘databases’,’mongoDB’],
comments:[
{user: ‘unknown user’,
 text:’unknown test’
},
{user:unknown user2’,
 text:’unknown text2
```

```
}
]
}
```

همانگونه که مشاهد می‌کنید نحوه‌ی ذخیره‌سازی داده‌ها بسیار با سیستم رابطه‌ای متفاوت می‌باشد ، با جمع‌بندی تفاوت نحوه‌ی نگه‌داری داده‌ها در این سیستم و RDBMS و بررسی این سیستم نکات اصلی به شرح زیر می‌باشند:

-۱فرمت ذخیره سازی داده‌ها مشابه فرمت JSON می‌باشد.

-۲به مجموعه داده‌های مرتبط به یکدیگر Document گفته می‌شود.

-۳در این سیستم JOIN ها وجود ندارند و داده‌های مرتبط کنار یکدیگر قرار می‌گیرند ، و یا به تعریف دقیق‌تر داده‌ها در یک داکیومنت اصلی Embed می‌شوند .

به عنوان مثال در اینجا مقدار comment ها برابر با آرایه‌ای از Document ها می‌باشد.

-۴مقادیر می‌توانند بصورت آرایه نیز در نظر گرفته شوند.

-۵در سیستم‌های RDBMS در صورتی که بخواهیم از وجود JOIN ها صرفنظر کنیم. به عدم توانایی در نرمال‌سازی برخواهیم خورد که یکی از معایب عدم نرمال‌سازی وجود مقادیر Null در جداول می‌باشد؛ اما در این سیستم به دلیل Schema free بودن می‌توان ساختارهای متفاوت برای Document ها در نظر گرفت.

به عنوان مثال برای یک پست می‌توان مقدار n کامنت تعریف کرد و برای پست دیگر هیچ کامنتی تعریف نکرد.

-۶در این سیستم اصولا نیازی به تعریف ساختار از قبل موجود نمی‌باشد و به محض اعلان دستور قرار دادن داده‌ها در پایگاه‌داده ساختار متناسب ایجاد می‌شود.

با مقایسه دستورات CRUD در هر دو نوع پایگاه داده با نحوه‌ی کوئری گرفتن از Document Database آشنا می‌شویم:

در SQL برای ایجاد جدول خواهیم داشت:

```
CREATE TABLE posts (
    id INT NOT NULL
        AUTO_INCREMENT,
    author_id INT NOT NULL,
    url VARCHAR(50),
    PRIMARY KEY (id)
)
```

دستور فوق در Document Database معادل است با:

```
db.posts.insert({id: “256” , author_id:”546”,url:"http://example.com/xxx"}) // با قرار دادن مقدار نوع ساختار مشخص می‌شود
```

در SQL جهت خواندن خواهیم داشت:

```
SELECT * from posts
WHERE author_id > 100
```

و معادل آن برابر است با:

```
db.posts.find({author_id:{$gt:”1000”}})
```

در SQL جهت بروزرسانی داریم:

```
UPDATE posts
SET author_id= "123"
```

که معادل است با:

```
db.posts.update({ $set: { author_id: "123" }})
```

در SQL جهت حذف خواهیم داشت:

```
DELETE FROM posts
WHERE author_id= "654"
```

که معادل است با:

```
db.posts.remove( { author_id: "654" } )
```

همانگونه که مشاهده می‌فرمایید نوشتن کوئری برای این پایگاه داده ساده بوده و زبان آن نیز بر پایه جاوا اسکریپت می‌باشد که برای اکثر برنامه‌نویسان قابل درک است.

تاکنون توسط شرکت‌های مختلف پیاده‌سازی‌های مختلفی از این سیستم انجام شده است که از مهم‌ترین و پر استفاده‌ترین آنها می‌توان به موارد زیر اشاره کرد:

MongoDB

CouchDB

RavenDB

نظرات خوانندگان

نویسنده: سعید یزدانی
تاریخ: ۱۳۹۱/۱۱/۲۹ ۱۹:۵۷

با تشکر از مطلب زیباتون
یک سوال داشتم ایا این روش اونقدر به بلوغ رسیده که بشه در پروژه‌ها روش حساب کرد . یا اینکه فعلا از همون روش قبلی استفاده کنیم
سوال دیگر من هم این هست که به نظر شما در nosql اینده ایی دیده میشه ؟
با تشکر

نویسنده: سعید یزدانی
تاریخ: ۱۳۹۱/۱۱/۲۹ ۱۹:۵۹

اگر هم امکان داره refrence ی در این زمینه هست link بدید

نویسنده: حمید سامانی
تاریخ: ۱۳۹۱/۱۱/۲۹ ۲۱:۳

در رابطه با سوال اولتون عارضم که در حال حاضر همه‌ی شرکت‌های بزرگ و فعال در این صنعت مثل گوگل ، فیس ب و ک توئیتر و از این شیوه استفاده می‌کنند ، در حالت کلی این مبحث یک تکنولوژی خاص نیست که مصرفی باشه و بعد از مدتی تاریخش بگذره ، یک Movement و یا یک نگرش کلی در تعریف عامه از مجموعه‌ای از راه حل‌ها به منظور رفع مشکلاتRDBMS در پردازش داده‌های بزرگ (BigData) ، داده‌ها در حوزه‌ی وب هم که رشدی نمایی دارند.
در رابطه با سوال دوم هم بستگی به خود فرد و یا شرکت مربوطه داره ، در حوزه‌ی نرم‌افزارهای داخلی به دلیل پایین‌تر بودن حجم داده‌ها الزامی در استفاده از این روش‌ها نیست. (استفاده و یا عدم استفاده مستقیما به نوع نرم‌افزار و ساختار آن بستگی دارد)

نویسنده: حمید سامانی
تاریخ: ۱۳۹۱/۱۱/۲۹ ۲۱:۵

از اینجا که شما شروع کنید به همه جا لینک می‌شوید :)

نویسنده: سعید یزدانی
تاریخ: ۱۳۹۱/۱۱/۲۹ ۲۱:۲۴

ممنون بابت جواب کاملتون

نویسنده: توحید عزیزی
تاریخ: ۱۳۹۱/۱۱/۳۰ ۳:۵۲

سلام

سپاسگزارم از موضوع جالبی که انتخاب کرده اید و مطالب خوبی که می‌نویسید.
آیا امکان دارد که در مورد هر کدام از انواع دیتابیس نوسیکوئل، مثالهای بیشتری بزنید.
از یک سرویس رایگان برای نوشتن مثال‌ها می‌تواند استفاده کرد که برای همه در دسترس باشد، مثل: cloudant.com
با تشکر

نویسنده: Meysam Navaei
تاریخ: ۱۳۹۱/۱۱/۳۰ ۹:۱۶

سلام

توحید این سایت که معرفی کردی خیلی جالب بود.می خاستم بینم محدودیت حجمی در استفاده ازش وجود داره یا نه نامحدود. ریسک محسوب نمیشه‌ی پروژه بزرگ داشته باشی و بخای دیتابیس رو از سرویس این سایت استفاده کنی؟منظورم اینکه از این سایت‌ها نباشه که یهو محدودیت ایجاد بکنه و یا پولی بشه و...

نویسنده: حمید سامانی
تاریخ: ۱۳۹۱/۱۱/۳۰ ۱۶:۱۵

سلام
سعی می‌کنم مثال‌های بیشتری را در مطالب آتی بگنجانم.
با سپاس :)

نویسنده: توحید عزیزی
تاریخ: ۱۳۹۱/۱۲/۰۳ ۹:۳

سلام.
نسخه رایگانش محدودیت داره: فکر کنم 2000 کوئری در روز.
اگر می‌خواهید پروژه‌ی بزرگ روش ببرید، باید از نسخه‌های تجاریش استفاده کنید. البته من خودم تستش نکرده ام هنوز.
https://cloudant.com/#home-pricing

نویسنده: masi
تاریخ: ۱۳۹۲/۰۲/۱۹ ۱:۵۶

سلام ، واقعا مطالب خوبی بود هر جا رو گشتم کاملتر و جامع‌تر از همه بودید ، موضوع پروژه‌ی من روی این موضوعه ، ای کاش میشد در مورد موضوع زیر صحبت کنید. key value store

نویسنده: جواد زبیدی
تاریخ: ۱۳۹۲/۰۵/۱۶ ۳:۳۳

سلام تشکر از مطلب بسیار مفیدتون .
می خواستم بدونم که کدام یک از روش‌ها بیشتر امتحان خودش رو توی داده‌های زیاد پس داده .و بشه راحتر باهاش کار کرد .
روش
Document store
Key value
روش هایی دیگری رو هم توی سایت دیدم اگر امکان داره مزایا و معایب هر کدوم رو توضیح دهید ممنون.

نویسنده: دادخواه
تاریخ: ۱۳۹۲/۰۶/۰۷ ۱۲:۱۰

سلام
تشکر از مطالب خوبتون
اما چند تا سوال دارم.
1- از این سه تا پایگاه داده که در اخر نوشتید فکر کنم فقط MongoDB مجانی باشه. درسته؟
2- آیا دستورات در همه این پایگاه داده‌ها به همین صورت است؟
3- آیا همه سرورها و هاست‌ها از این پایگاه داده‌ها مانند MS SQL پشتیبانی می‌کنند و یا سرورهای خاص را باید پیدا کرد؟
تشکر

نویسنده: محسن خان

تاریخ: ۱۳۹۲/۰۶/۰۷ ۱۲:۲۵

اگر مطالب مقدماتی‌تر رو مطالعه می‌کردید، می‌دید که اصلا هدف از بانک اطلاعاتی NoSQL این نیست که باهاش سایت معمولی درست کنند اون هم روی سرور اجاره‌ای با 100 مگ فضا. هدفش توزیع شده بودن در سرورهایی متعدد و یا با پراکندگی جغرافیایی بالا است.

نتیجه گیری؟ ابزار زده نباشید. اول مفاهیم رو مطالعه کنید. اول تئوری کار مهمه.

نویسنده: saremi
تاریخ: ۱۳۹۲/۱۱/۲۴ ۱۶:۴۴

با سلام؛ میخواستم در مورد ،HQL، CQL ،UNQL، map reduce و... بپرسم. توی همون سایتی که لینکش رو دادین این‌ها جزو انواع query methodها هستند. من دقیق نمی‌فهمم الان ما دستورات مشابه sql رو معادلش رو با java script نوشتیم. در مورد تفاوت اینها و استفاده شون اگر میشه کمی توضیح بدین لطفا. دقیقا توی انواع مختلف پایگاه داده با چه زبانی کوئری نویسی می‌شه؟ با تشکر

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۱/۲۴ ۱۶:۵۲

در مورد تفاوت این‌ها در مطلب مروری بر مفاهیم مقدماتی NoSQL بیشتر توضیح داده شده. برچسب NoSQL را بهتر است دنبال کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۱/۲۴ ۱۷:۴۹

در مورد MongoDB یک کتابچه‌ی فارسی 90 صفحه‌ای موجود است .

نویسنده: salam
تاریخ: ۱۳۹۳/۰۵/۰۹ ۱۱:۱۶

سلام
در قسمت دوم این مطلب اومده که "در حالت کلی پایگاهای داده NoSQL به ۴ دسته تقسیم می‌شوند " دسته 3 و 4 را توضیح نمی‌دین؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۰۹ ۱۱:۲۶

برای دنبال کردن مطالب هم خانواده در این سایت، در ذیل هر مطلب یک سری گروه یا برچسب تعریف شده‌اند. برای مثال اگر برچسب NoSQL را دنبال کنید، در مطالب دیگری پاسخ خود را خواهید یافت.