

# مقدمه‌ای بر الستیک سرچ

## درس ارائه مطالب علمی و فنی

پارسا محمدیان

دانشگاه صنعتی شریف

۲۱ خرداد ۱۴۰۰



## ۱ Elasticsearch چیست؟

- ◀ آشنایی با ویژگی‌های معماری
- ◀ کاربردهای الستیک سرچ در صنعت

## ۲ مفاهیم اولیه‌ی الستیک سرچ

- ◀ آشنایی با اجزا
- ◀ مقایسه با دیتابیس‌های رابطه‌ای

## ۳ کارایی در عملیات مختلف

- ◀ جزئیات تست انجام شده
- ◀ عملیات ساختن مخزن
- ◀ عملیات بارگذاری داده
- ◀ عملیات جستجو زیررشته
- ◀ عملیات جستجو بازه
- ◀ عملیات تجمعی (Aggregation)

## ۴ الستیک سرچ کمی عمیق‌تر

## ۵ منابع



- ◀ موتور جستجو متن‌باز بر پایه‌ی لوسین (Lucene)
  - ◀ لوسین موتور جستجو متن‌باز نوشته شده به زبان جاوا است
- ◀ توزیع شده (Distributed)
- ◀ مقیاس‌پذیر (Scalability)
- ◀ امکان جستجو Full-text
- ◀ رابط HTTP و RESTful
- ◀ اسناد غیروابسته به قالب (Scheme-free) و بر پایه‌ی JSON



# Elasticsearch چیست؟

کاربردهای الاستیک سرچ در صنعت



شکل: نمونه‌های کاربردهای مختلف و مشتری‌ها<sup>۱</sup>

<sup>۱</sup> تصویر از [۱]

- ◀ خوشه یا Cluster  $\Leftarrow$  تعدادی از سرورهای الاستیک سرچ که به هم متصل هستند
- ◀ گره یا Node  $\Leftarrow$  هر یک از سرورهای الاستیک سرچ در Cluster
- ◀ سند یا Document  $\Leftarrow$  هر یک از اسناد متنی که بارگذاری می‌شود
- ◀ مخزن یا Index  $\Leftarrow$  مخازنی که دارای تعدادی سند با قالب یکسان هستند
- ◀ نگاشت یا Mapping  $\Leftarrow$  قالب هر Index
- ◀ تکه داده یا Shard  $\Leftarrow$  هر Index تعدادی Shard دارد
- ◀ بخش یا Segment  $\Leftarrow$  هر Shard تعداد Segment دارد



الستیک سرچ	دیتابیس‌های رابطه‌ای
Index	Database
Type <sup>a</sup>	Table
Document	Row
Field	Column
Mapping	Schema

<sup>a</sup>Removed since version 6.00

## کمی نکات منفی در مورد الستیک!

◀ مفهوم Transaction ندارد

◀ پشتیبانی محدودتر از Join

## نکته مهم

این محدودیت‌ها برای دیتابیس‌های NoSQL و Distributed قابل انتظار است.



- ▶ داده بارگذاری شده اطلاعات املاک تهران موجود در دیوار است. [۴]
- ▶ این داده شامل ۳۷۲۳۳۲ رکورد و حجم خام 143 MB است.
- ▶ تست‌ها بر روی یک کامپیوتر شخصی با رم 8 GB و پردازنده Core i7-8550 انجام شده است.

## ابزارهای مورد استفاده

برای کوئری زدن به الاستیک از ابزار Kibana که یک وب اپلیکیشن است استفاده شده است. همچنین برای بارگذاری داده در الاستیک با استفاده از ابزار Curl درخواست HTTP فرستاده شده است. برای ارتباط با SQLServer از Azure Data Studio استفاده شده است.



# کارایی در عملیات مختلف

## عملیات ساختن مخزن

The screenshot displays two application windows side-by-side. The left window is SQL Server Enterprise Manager, showing a query editor with the command `CREATE DATABASE Comparison;` and a Messages pane indicating successful execution at 12:47:51 AM. The right window is Kibana, showing a PUT request to `/_mapping` for the `comparison` index, with a response showing `"index": "comparison"`. A Windows PowerShell window is also visible in the background.

SQL Server Enterprise Manager - SQLQuery\_1 - localhost:master (Integrated) - Azure...

Windows PowerShell - PS C:\Users\Parisa>

Elastic - 127.0.0.1:5601/app/kibana#/dev\_tools/console

Console Search Profiler Grok Debugger Painless Lab BETA

History Settings Help

1 PUT /\_mapping

2 {

3 "index": "comparison"

4 }

5 }

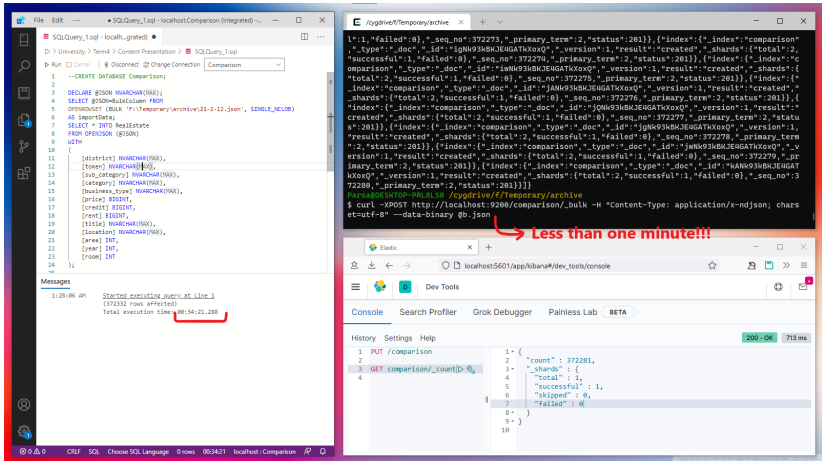
6

200 - OK 1207 ms





# کارایی در عملیات مختلف



# کارایی در عملیات مختلف

## عملیات جستجو زیررشته

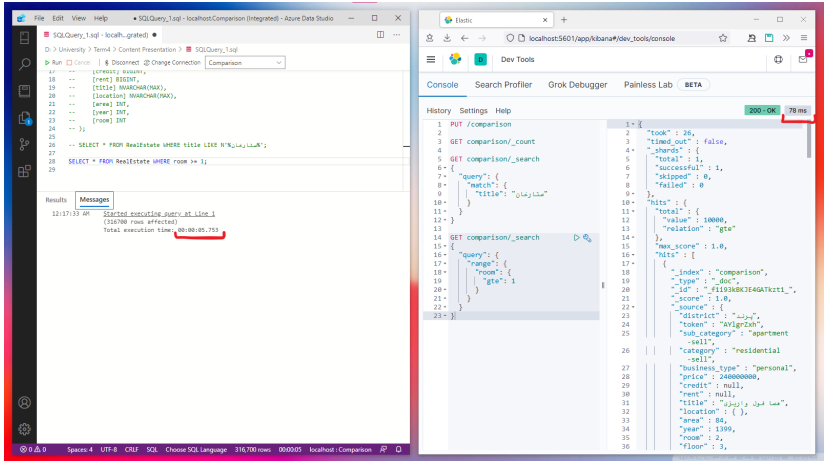
The image shows two side-by-side windows. The left window is SQL Studio, displaying a table schema for 'SQLQuery\_1.sql' and a query: `SELECT * FROM RealEstate WHERE title LIKE N'%مشاركان%'`. The 'Results' tab shows the execution started at 12:13:09 AM, affecting 1857 rows in 00:00:01.258. The right window is the Elasticsearch console, showing a REST client request: `PUT /comparison` and a response with a 200 status and 103ms duration. The response body is a JSON document representing a real estate listing.

```
1 PUT /comparison
2
3 GET comparison/_count
4
5 GET comparison/_search
6 {
7   "query": {
8     "match": {
9       "title": "مشاركان"
10    }
11  }
12 }
13
```

```
1 {
2   "took": 27,
3   "timed_out": false,
4   "_shards": {
5     "total": 1,
6     "successful": 1,
7     "skipped": 0,
8     "failed": 0
9   },
10  "hits": {
11    "total": {
12      "value": 1767,
13      "relation": "eq"
14    },
15    "max_score": 8.026751,
16    "hits": [
17      {
18        "_index": "comparison",
19        "_type": "doc",
20        "_id": "rQJk93k8KJf4GAtk98Lq",
21        "_score": 8.026751,
22        "district": "مشاركان",
23        "token": "AVL4887T",
24        "sub_category": "shop-rent",
25        "category": "commercial-rent",
26        "business_type": "personal",
27        "price": null,
28        "credit": 300000000,
29        "rent": 15000000,
30        "title": "مشاركان",
31        "location": { },
32        "area": 78,
33        "year": 1373,
34        "room": 0
35      }
36    ]
37  },
38}
```

# کارایی در عملیات مختلف

## عملیات جستجو بازه



# کارایی در عملیات مختلف

## عملیات تجمعی (Aggregation)

The image displays two side-by-side windows. The left window is the SQLQuery\_1.sql editor in Azure Data Studio, showing a series of SQL queries for testing aggregation functions. The right window is the Elastic console, showing the execution of these queries and the resulting JSON data.

**SQLQuery\_1.sql (Left Window):**

```
18 -- [rent] BIGINT,  
19 -- [title] NVARCHAR(MAX),  
20 -- [location] NVARCHAR(MAX),  
21 -- [area] INT,  
22 -- [year] INT,  
23 -- [room] INT  
24 -- ];  
25  
26 -- SELECT * FROM RealEstate WHERE title LIKE N'%مشارکان%';  
27  
28 -- SELECT * FROM RealEstate WHERE room >= 1;  
29  
30 SELECT AVG(room) AS 'roomAverage'  
31 FROM RealEstate;
```

**Elastic Console (Right Window):**

The console shows the execution of the following queries:

- PUT /comparison**: Returns a document with fields like `_type`, `_id`, `_score`, `_source`, `district`, `token`, `sub_category`, `sell`, `category`, `business_type`, `price`, `credit`, `rent`, `title`, `location`, `area`, `year`, `room`, `agency`, `floor`, `elevator`, `parking`, and `storage`.
- GET comparison/\_count**: Returns the count of documents.
- GET comparison/\_search**: Returns a search result with a range query for `room` (gte: 1) and an aggregation for `room_avg`.
- POST comparison/\_search**: Returns a search result with a range query for `room` (gte: 1) and an aggregation for `room_avg`.

The console also shows the execution of the `SELECT AVG(room) AS 'roomAverage' FROM RealEstate;` query, which returns a single row with the average room value.



سند ۱ :

سند ۲ :



- [1] Howard Chen. *Elastic Q1 FY2021 Analysis*. URL: <https://blog.publiccomps.com/elasticsearchq1fy2021/> (visited on 06/06/2021).
- [2] Zachary Tong Clinton Gormley. *Elasticsearch: The Definitive Guide: A Distributed Real-Time Search and Analytics Engine*. O'Reilly Media, 2015. ISBN: 1449358543.
- [3] *Elastic Stack and Product Documentation*. URL: <https://www.elastic.co/guide/index.html> (visited on 04/19/2021).
- [4] Amirali Taheri. *tehran-real-estate-prices*. URL: <https://www.kaggle.com/amiralitaheri/tehranrealestateprices> (visited on 06/11/2021).

اگر جایی منبع ذکر نشده از [۳] و [۲] استفاده شده است.

