

وظایف مهندسان داده

مهندسی داده بخشی از اکوسیستم کلان داده است و ارتباط نزدیکی با علم داده دارد. مهندسان داده معمولاً در پس زمینه کار می کنند و به اندازه دانشمندان داده مورد توجه قرار نمی گیرند، اما نقش کلیدی در فرآیند علم داده دارند.

مسئولیت های مهندس داده

مسئولیت های یک مهندس داده بسته به سطح بلوغ داده ای و میزان نیروی انسانی سازمان متفاوت است، اما برخی از داده، از وظایف اساسی این نقش هستند **(Transform)** و **تبدیل (Load)** **بارگذاری (Extract)**، وظایف، مانند استخراج

در ابتدایی ترین سطح، مهندسی داده شامل انتقال داده از یک سیستم یا فرمت به سیستم یا فرمت دیگر است. در قالب: مفاهیم رایج تر:

- **(Extract)** مهندسان داده داده ها را از یک منبع استخراج می کنند
- **(Transform)** سپس برخی تغییرات و پردازش ها را روی آن انجام می دهند
- در نهایت، داده ها را در محلی قرار می دهند که کاربران بتوانند به داده های باکیفیت و قابل اطمینان دسترسی **(Load)** داشته باشند

شناخته می شود و یکی از مفاهیم کلیدی در مهندسی داده **ETL (Extract, Transform, Load)** این فرآیند با نام محسوب می شود. حال با یک مثال به درک عمیق تر این حوزه می پردازیم.

مثال: فروشگاه آنلاین و نیاز به مهندسی داده

یک خرده فروش آنلاین را در نظر بگیرید که در وبسایت خود ویجت هایی با رنگ های مختلف می فروشد. این وبسایت از یک پایگاه داده رابطه ای پشتیبانی می شود که تمام تراکنش های انجام شده را ذخیره می کند.

اکنون می خواهیم به این سؤال پاسخ دهیم:

در سه ماهه گذشته، چند ویجت آبی رنگ به فروش رسیده است؟

روی پایگاه داده اجرا کرد تا پاسخ این پرسش را دریافت کرد. اما این مسئله نیازی به یک **SQL** در ابتدا می توان یک کوئری مهندس داده ندارد.

ورود به دنیای مهندسی داده

دیگر امکان پذیر نخواهد بود. علاوه بر **(Production Database)** با رشد فروشگاه، اجرای کوئری روی پایگاه داده تولیدی. این، ممکن است بیش از یک پایگاه داده برای ثبت تراکنش ها وجود داشته باشد: برای مثال، ممکن است پایگاه داده ای برای مناطق جغرافیایی مختلف تعریف شده باشد:

- یک پایگاه داده برای فروشندگان در آمریکای شمالی
- پایگاه های داده مجزا برای فروشندگان در آسیا، آفریقا و اروپا

در چنین شرایطی، مهندسی داده وارد عمل می شود.

مهندس داده اتصال هایی به تمام پایگاه های داده تراکنش های منطقه ای ایجاد می کند، داده ها را استخراج کرده و سپس **بارگذاری می کند (Data Warehouse)** در یک انبار داده.

در این مرحله، می توان تعداد ویجت های آبی فروخته شده را محاسبه کرد.

پرسش های پیچیده تر برای تحلیل داده

شرکت‌ها معمولاً به دنبال پاسخ سؤالات پیچیده‌تری هستند، از جمله:

- چگونه می‌توان متوجه شد که کدام مناطق بیشترین میزان فروش ویجت را دارند؟
- اوج زمان‌های فروش ویجت چه ساعاتی است؟
- چه تعداد کاربر ویجت‌ها را به سبد خرید خود اضافه کرده و سپس حذف کرده‌اند؟
- چه ترکیبی از ویجت‌ها بیشتر با هم فروخته می‌شوند؟

مدل‌سازی داده‌ها و بهینه‌سازی (Data Pipelines) اینجا جایی است که مهندسان داده با ایجاد پایپ‌لاین‌های داده‌ای پردازش داده، ارزش واقعی خود را نشان می‌دهند.

پردازش پیچیده‌تر داده‌ها در مهندسی داده

داده‌ها در یک سیستم واحد است. بین این دو **(Load)** و **بارگذاری (Extract)** پاسخ به این سؤالات فراتر از استخراج ضروری است **(Transform)** مرحله، تبدیل داده.

در مناطق مختلف جغرافیایی است **(Time Zones)** یکی از چالش‌های کلیدی، تفاوت مناطق زمانی برای مثال، ایالات متحده به‌تنهایی چهار منطقه زمانی دارد. بنابراین، برای تحلیل دقیق، باید تمام فیلدهای زمانی به یک استاندارد یکسان تبدیل شوند.

افزودن فیلد موقعیت جغرافیایی

علاوه بر تبدیل زمان، نیاز به روشی برای تشخیص فروش در هر منطقه وجود دارد. این کار را می‌توان با افزودن یک فیلد به داده‌ها انجام داد **(Location Field)** موقعیت مکانی.

حال این سؤال مطرح می‌شود:
این فیلد باید به چه فرمتی باشد؟

- **(Well-Known Text - WKT)** یا متن استاندارد مکانی **(Coordinates)** شامل مختصات جغرافیایی **(Spatial Format)** فرمت مکانی
- **فرمت متنی ساده:** که بعداً در یک پایپ‌لاین مهندسی داده قابل تبدیل و پردازش باشد

انتخاب فرمت مناسب بستگی به نیازهای کسب‌وکار و نحوه تحلیل داده‌ها دارد.

ایجاد پایپ‌لاین داده برای پردازش اطلاعات

در این مرحله، مهندس داده باید مراحل زیر را انجام دهد:

1. استخراج داده‌ها از هر پایگاه داده.
2. افزودن فیلد موقعیت مکانی برای مشخص کردن مکان هر تراکنش.
3. برای یکسان‌سازی مناطق زمانی **ISO 8601** تبدیل زمان محلی به استاندارد.
4. **(Data Warehouse)** بارگذاری داده‌ها در انبار داده.

(Data Pipeline) پایپ‌لاین داده

ترکیب این مراحل با ایجاد یک پایپ‌لاین داده انجام می‌شود.

ویژگی‌های پایپ‌لاین داده:

- وارد پایپ‌لاین می‌شوند **(Raw Data)** داده‌های خام.
- داده‌ها ممکن است ناقص باشند یا اشتباهات تایپی داشته باشند.
- می‌شوند **(Transformation)** و اصلاح **(Cleaning)** در طول مسیر، داده‌ها تمیزسازی.

- خواهند بود (**Querying**) در نهایت، داده‌های پردازش شده در انبار داده ذخیره می‌شوند و آماده کوئری‌گیری

در ادامه، یک دیاگرام پایپ‌لاین داده نمایش داده خواهد شد که نحوه اجرای این فرآیند را نشان می‌دهد

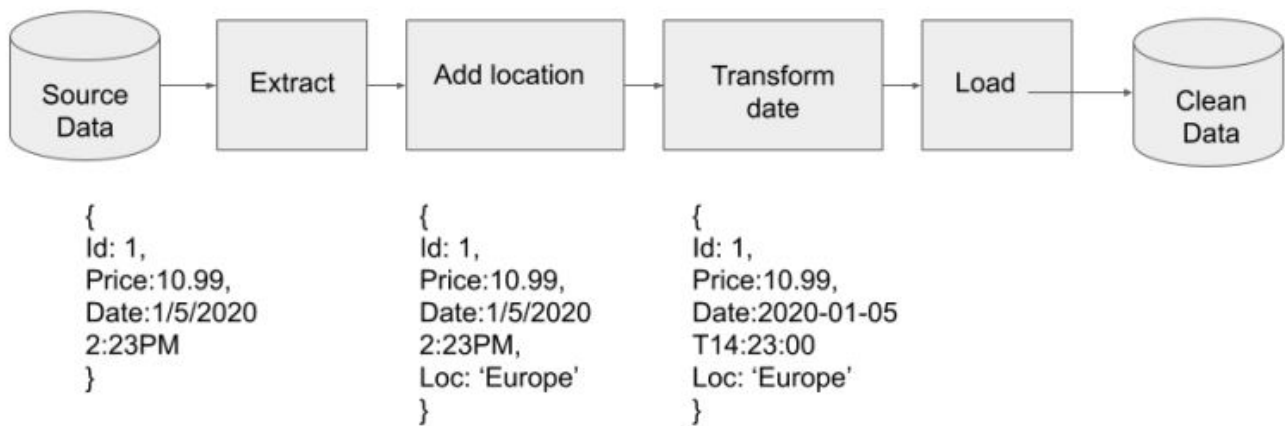


Figure 1.1 – A pipeline that adds a location and modifies the date

مهارت‌ها و دانش مورد نیاز برای مهندسی داده

در مثال قبلی، مشخص شد که **مهندسان داده** باید با فناوری‌های متنوعی آشنا باشند. این در حالی است که هنوز به نیازهای کسب‌وکار و فرآیندهای آن اشاره نکرده‌ایم.

1. استخراج داده از منابع مختلف

- آشنایی با فرمت‌های متنوع فایل‌ها و پایگاه‌های داده مختلف
- برای استخراج داده‌ها SQL تسلط بر
- برای پردازش داده‌ها و خودکارسازی عملیات Python تسلط بر

2. و مدل‌سازی داده (Transformation) تبدیل داده

- و ساختارهای داده (**Data Modeling**) دانش مدل‌سازی داده
- درک فرآیندهای کسب‌وکار و نیازهای تحلیلی
- طراحی مدل‌های داده‌ای متناسب با نیازهای سازمان

3. بارگذاری داده در انبار داده (Data Warehouse)

- برای انبار داده **Schema** طراحی و ایجاد
- دانش پایه‌ای درباره طراحی انبار داده
- آشنایی با پایگاه‌های داده تحلیلی و ابزارهای مرتبط

4. مدیریت زیرساخت‌های پردازش داده

- مدیریت سرورهای لینوکسی و پیکربندی سرویس‌ها
- **Apache NiFi** و **Apache Airflow** آشنایی با ابزارهای مدیریت داده مانند
- برای ایجاد و مدیریت **AWS, Google Cloud, Azure** مانند (**Cloud Platforms**) کار با سکوها ابری زیرساخت‌ها

تعریف جامع‌تری از مهندسی داده

اکنون که مثالی از وظایف مهندس داده بررسی شد، می‌توان یک تعریف گسترده‌تر از مهندسی داده ارائه داد.

تعریف مهندسی داده

مهندسی داده شامل توسعه، اجرا و نگهداری زیرساخت‌های داده است که می‌تواند:

- در محل (On-Premises)
- در فضای ابری (Cloud)
- یا چند ابری (Multi-Cloud) یا چند ابری (Hybrid) به صورت ترکیبی

باشد.

و (Transform) تبدیل، (Extract) این زیرساخت شامل پایگاه‌های داده و پایپ‌لاین‌های داده است که برای استخراج داده‌ها طراحی و پیاده‌سازی می‌شوند (Load) بارگذاری.

تفاوت مهندسی داده و علم داده

مهندسی داده زیربنای اصلی برای اجرای علم داده است. بسته به میزان بلوغ داده‌ای یک سازمان، ممکن است از دانشمندان داده انتظار برود که داده‌های مورد نیاز خود را پاک‌سازی و انتقال دهند، اما این کار بهره‌وری آنها را کاهش می‌دهد. هرچند که مهندسان داده و دانشمندان داده ابزارهای مشابهی مانند Python می‌دهد.

علم داده	مهندسی داده
استفاده از داده‌ها برای ساخت مدل‌های آماری و محاسبات ریاضی	طراحی و مدیریت فرمت‌ها، مدل‌ها و ساختارهای داده برای انتقال کارآمد داده‌ها
استخراج داده‌ها از انبار داده برای تحلیل و مدل‌های یادگیری ماشین	(Data Warehouse) ایجاد و نگهداری انبار داده
تحلیل داده و ارائه بینش کسب‌وکار	پیاده‌سازی پایپ‌لاین‌های داده برای پردازش خودکار داده‌ها
توسعه الگوریتم‌ها برای پیش‌بینی و تحلیل داده	بهینه‌سازی زیرساخت برای مقیاس‌پذیری و پردازش سریع داده‌ها

همکاری بین مهندسان داده و دانشمندان داده

- دانشمندان داده از انبار داده ایجادشده توسط مهندسان داده استفاده می‌کنند.
- مدل‌های یادگیری ماشین ممکن است در پایپ‌لاین داده یکپارچه شوند.
- درک نیازهای دانشمندان داده، به بهینه‌سازی فرآیندهای مهندسی داده کمک می‌کند.

ابزارهای مهندسی داده

برای ساخت پایپ‌لاین‌های داده، مهندسان داده باید ابزارهای مناسبی را انتخاب کنند. مهندسی داده بخشی از اکوسیستم است و باید سه ویژگی اصلی کلان‌داده را در نظر بگیرد (Big Data) کلان‌داده:

۱. حجم (Volume)

- حجم داده‌ها به شدت افزایش یافته است.
- جابجایی هزار رکورد در یک پایگاه داده با جابجایی میلیون‌ها ردیف یا پردازش میلیون‌ها تراکنش در دقیقه کاملاً متفاوت است.

۲. تنوع (Variety)

- داده‌ها در فرمت‌های متنوع ذخیره می‌شوند.
- ها و فایل‌ها API، مهندسان داده باید ابزارهایی را انتخاب کنند که بتوانند داده‌های مختلف را از پایگاه‌های داده پردازش کنند.

۳. سرعت (Velocity)

- سرعت تولید داده‌ها در حال افزایش است.
- ردیابی فعالیت میلیون‌ها کاربر در یک شبکه اجتماعی یا ثبت خرید کاربران در سطح جهانی نیازمند پردازش داده است. (Near Real-Time Processing) در نزدیک به زمان واقعی.

بخش بعدی: معرفی ابزارهای رایج در مهندسی داده

در ادامه، با ابزارهای پرکاربرد مهندسی داده که برای پردازش و مدیریت حجم بالای داده استفاده می‌شوند، آشنا خواهیم شد.

زبان‌های برنامه‌نویسی در مهندسی داده

استفاده کنید و چه یک زبان برنامه‌نویسی **low-code** زبان اصلی مهندسی داده محسوب می‌شود. چه از ابزارهای **SQL** اجتناب ناپذیر است **SQL** خاص، یادگیری

- به مهندسان داده کمک می‌کند تا کوئری‌ها را بهینه‌سازی کرده و تحول‌های داده‌ای را مدیریت کنند **SQL** تسلط بر
- دارند **SQL** نیز ابزارهایی برای اجرای **NoSQL** ها و پایگاه‌های داده **Data Lake** حتی

زبان‌های رایج در مهندسی داده

✓ Python

- کراس‌پلتفرم و مستندات قوی
- پشتیبانی از کتابخانه‌های گسترده‌ای مثل:
 - **pandas, matplotlib, numpy, scipy** (برای پردازش و تحلیل داده)
 - **scikit-learn, tensorflow, pytorch, NLTK** (برای یادگیری ماشین و پردازش زبان طبیعی)

پایگاه‌های داده

در بیشتر سیستم‌های تولید، داده‌ها در پایگاه‌های داده رابطه‌ای ذخیره می‌شوند.

- استفاده می‌کنند **Microsoft SQL Server** یا **Oracle** راه‌حل‌های اختصاصی معمولاً از
- استفاده می‌کنند **PostgreSQL** یا **MySQL** راه‌حل‌های متن‌باز معمولاً از

این پایگاه‌های داده داده‌ها را در رکوردها ذخیره کرده و برای ثبت تراکنش‌ها مناسب هستند.

برای پیوستن داده‌ها از یک جدول به **(Primary Keys)** همچنین، روابطی بین جداول وجود دارد که از کلیدهای اصلی جدول دیگر استفاده می‌شود، که این ویژگی آنها را به پایگاه‌های داده رابطه‌ای تبدیل می‌کند.

مدل ساده داده و روابط بین جداول

در تصویر زیر، یک مدل ساده داده و روابط میان جداول نمایش داده شده است.

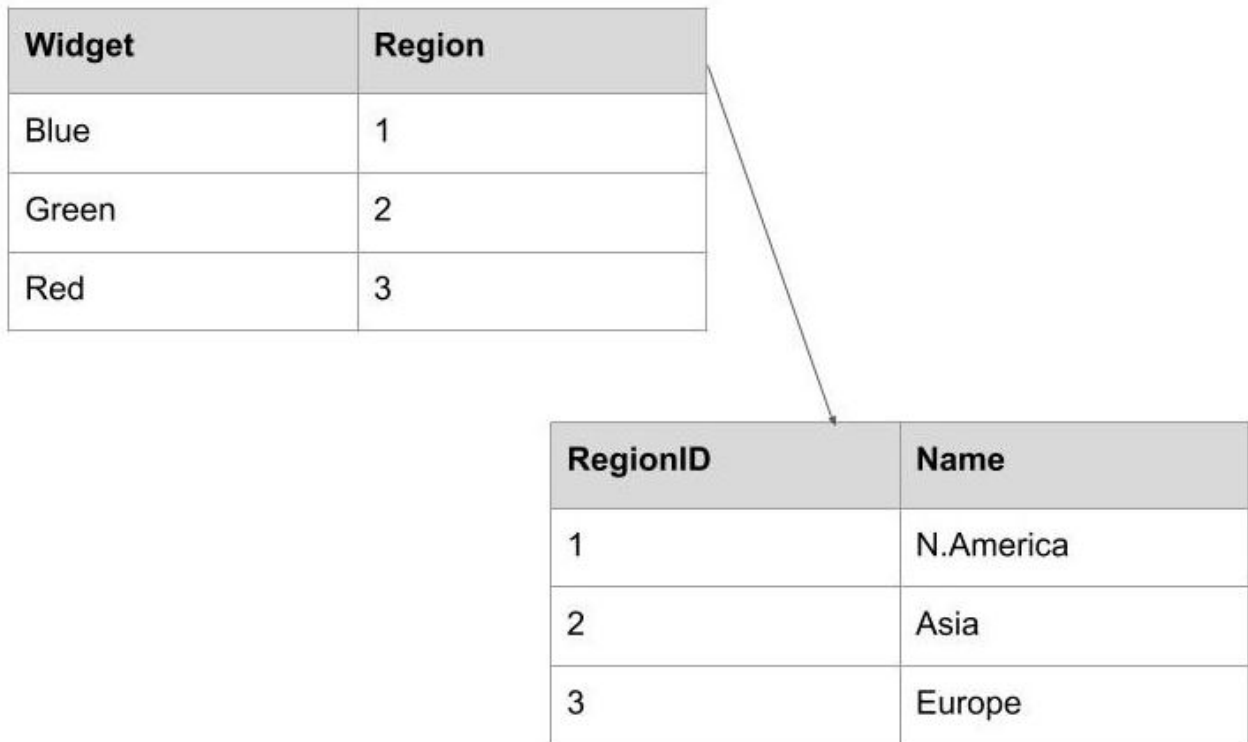


Figure 1.2 – Relational tables joined on Region = RegionID.

پایگاه‌های داده در انبار داده (Data Warehousing)

رایج‌ترین پایگاه‌های داده‌ای که در انبار داده‌ها استفاده می‌شوند عبارتند از:

- **Amazon Redshift**
- **Google BigQuery**
- **Apache Cassandra**
- **Elasticsearch** دیگر مانند **NoSQL** پایگاه‌های داده

این پایگاه‌های داده برخلاف پایگاه‌های داده رابطه‌ای که داده‌ها را به صورت **رکورد** ذخیره می‌کنند، داده‌ها را به صورت **ستونی (Columnar Format)** ذخیره می‌کنند، همانطور که در تصویر زیر نشان داده شده است.

Widget	Region
Blue	1
Green	2
Red	3

Widget	Blue
	Green
	Red
Region	1
	2
	3

Figure 1.3 – Rows stored in a columnar format

NoSQL پایگاه‌های داده ستونی و

برای پرس‌وجوهای سریع‌تر مناسب‌تر هستند و به همین دلیل برای (Columnar Databases) پایگاه‌های داده ستونی انبار داده‌ها بسیار کارآمد هستند.

- **Amazon Redshift، Google BigQuery، و Apache Cassandra** تمام این پایگاه‌های داده ستونی (مانند **Cassandra Query Language (CQL)** از **Cassandra** کوئری شوند، اگرچه **SQL** می‌توانند با استفاده از **SQL** استفاده می‌کند که مشابه است.

و مستندات NoSQL پایگاه‌های داده

- **Apache Lucene** است که در واقع یک موتور جستجو مبتنی بر **NoSQL** یکی از پایگاه‌های داده **Elasticsearch** است.
- **Apache Solr** مشابه **Elasticsearch** است اما کاربرپسندتر می‌باشد.
- برای یادگیری ماشین، **X-Pack plugins** این پایگاه داده متن‌باز است، اما اجزای اختصاصی نیز دارد، از جمله گراف‌ها، امنیت، و هشدارها/نظارت‌ها.
- **Elasticsearch** از **Elastic Query DSL** (که برای پرس‌وجو استفاده می‌کند که) **SQL** نیست زبان اختصاصی دامنه) برای پرس‌وجو استفاده می‌کند که **JSON** بلکه یک پرس‌وجوی ذخیره می‌کند و برخلاف پایگاه‌های داده رابطه‌ای، **(Documents)** داده‌ها را به صورت مدارک **Elasticsearch** غیررابطه‌ای است.

پردازش داده‌ها

زمانی که یک مهندس داده داده‌ها را از پایگاه داده استخراج می‌کند، نیاز به تحول یا پردازش آنها دارد. در دنیای داده‌های بزرگ، استفاده از یک موتور پردازش داده به شدت کمک‌کننده است.

موتورهای پردازش داده

و چه به صورت **Batch** موتورهای پردازش داده به مهندسان داده این امکان را می‌دهند که داده‌ها را چه به صورت پردازش و تبدیل کنند. این موتورها اجرای موازی وظایف مربوط به تبدیل داده را فراهم می‌کنند **Stream**.

Apache Spark

است. این موتور به مهندسان داده اجازه می‌دهد که **Apache Spark** یکی از محبوب‌ترین موتورهای پردازش داده بنویسند **Scala** و **Java** و **Python** تبدیلات خود را با استفاده از.

Python کار می‌کند، که آن را به ابزاری مناسب برای برنامه‌نویسان **Python DataFrames** به‌خوبی با **Apache Spark** نیز هست (**Resilient Distributed Datasets (RDDs)** دارای **DataFrame** تبدیل کرده است. علاوه بر

از اشیا هستند که عمدتاً از طریق بارگذاری داده از یک (**Immutable**) مجموعه‌ای توزیع‌شده و غیرقابل تغییر **RDDs** **RDD** ها پردازش سریع و توزیع‌شده را ممکن می‌سازند و وظایف تعریف‌شده در یک **RDD**. منبع خارجی ایجاد می‌شوند ها سعی نمی‌کنند به صورت **RDD**، **DataFrame** اجرا می‌شوند. برخلاف (**Cluster**) روی گره‌های مختلف یک کلاستر داده را تشخیص دهند (**Schema**) خودکار ساختار.

سایر موتورهای پردازش داده

:موتورهای پردازشی دیگری نیز وجود دارند، از جمله **Spark**، علاوه بر

- **Apache Storm**: برای اتصال این **Bolt** برای خواندن داده و **Spout** این موتور از **Apache Storm**. پردازشی ایجاد کرد **Pipeline** اجزا به یکدیگر، می‌توان یک
- **Apache Flink** و **Apache Samza**: این دو فریمورک مدرن برای پردازش **Stream** و **Batch** امکان **Unbounded Stream** پردازش را فراهم می‌کنند.

چیست؟ **Unbounded Stream**

جریانی از داده است که پایان مشخصی ندارد. برای مثال، **سنسور دما** که به‌طور مداوم **Unbounded Stream** یک محسوب می‌شود **Unbounded Stream** را گزارش می‌دهد، یک

گزینه‌های مناسبی برای **Samza** و **Flink**، برای استریم داده‌ها از یک سیستم استفاده می‌کنید **Apache Kafka** اگر از /خواهید آموخت **Apache Kafka** پردازش داده‌های دریافتی هستند. در ادامه این کتاب، اطلاعات بیشتری درباره

(خطوط پردازش داده) **Data Pipelines**

یک زبان برنامه‌نویسی، یک موتور، (**Transactional Database**) ترکیبی از یک پایگاه داده تراکنشی **Data Pipeline** یک است (**Data Warehouse**) پردازش داده و یک انبار داده.

Data Pipeline مثال ساده از یک

تصور کنید که می‌خواهید تمام رکوردهای مربوط به فروش یک محصول را از پایگاه داده استخراج کنید، سپس داده‌ها را **Data** پردازش کنید تا تعداد فروش هر محصول را محاسبه کنید و در نهایت نتیجه را در **Apache Spark** از طریق **Warehouse** پردازش داده محسوب می‌شود **Pipeline** ذخیره کنید. این یک

چندان مفید نخواهد بود. برای **Pipeline** اما اگر بخواهید این فرآیند را هر بار به صورت دستی اجرا کنید، این دارید که بتواند پردازش‌ها را در بازه‌های زمانی مشخص اجرا کند **Scheduler** اتوماتیک‌سازی اجرای آن، نیاز به یک

Pipeline با **Crontab** برنامه‌ریزی

برای **cron job** در لینوکس است. شما می‌توانید یک **crontab** استفاده از Pipeline ساده‌ترین راه برای زمان‌بندی اجرای خود تعریف کنید تا در **فواصل زمانی مشخص** اجرا شود **Python** اجرای فایل

را هر ۶ ساعت اجرا می‌کند **pipeline.py** مثال تعریف یک کران جاب که اسکریپت

```
0 */6 * * * /usr/bin/python3 /path/to/pipeline.py
```

به صورت **دوره‌ای** اجرا خواهد شد Pipeline، با این کار، بدون نیاز به اجرای دستی

Crontab در Pipeline مشکلات مدیریت

دشوار می‌شود. برخی از چالش‌ها عبارت‌اند از **crontab** ها، مدیریت آن‌ها در **Pipeline** با افزایش تعداد

- چگونه متوجه شوید که کدام پردازش **موفقیت‌آمیز** بوده و کدام یک **ناموفق** اجرا **Pipeline مانیتورینگ** اجرای شده است؟
- هایی اجرا شده‌اند و کدام یک **اجرا نشده‌اند**؟ Pipeline **های اجرا شده**: چگونه بدانید که چه **Pipeline رهگیری**
- سریع‌تر از وظیفه بعدی اجرا شود، چگونه می‌توان داده‌ها (**Tasks**) اگر یکی از وظایف **Backpressure** مدیریت را کنترل کرد تا وظایف بعدی با حجم بالای داده دچار مشکل نشوند؟

Data Pipeline راهکار جایگزین برای مدیریت پیشرفته‌ی

دیگر کافی نخواهد بود و شما نیاز به یک **فریمورک قدرتمندتر** برای مدیریت **Crontab**، **Pipeline** ها با پیچیده‌تر شدن **Pipeline** ها خواهید داشت، مانند:

- **Apache Airflow**
- **Luigi**
- **Prefect**

و **مدیریت (Task Dependencies)** این ابزارها امکاناتی مانند **مانیتورینگ**، **لاگ‌گیری**، **مدیریت وابستگی بین وظایف** را در اختیار شما قرار می‌دهند (**Backpressure Handling**) **حجم داده**

Apache Airflow

Airbnb است. این پلتفرم توسط **Python** در **Data Pipeline** محبوب‌ترین فریمورک برای ساخت **Apache Airflow** محسوب می‌شود (**Workflow Management Platform**) توسعه داده شده و یک **مدیر گردش کار**

Airflow اجزای اصلی

از چندین مؤلفه کلیدی تشکیل شده است Airflow:

- **Web Server**: مدیریت و DAGها رابط کاربری برای مانیتورینگ و
- **Scheduler**: (Tasks) زمان‌بندی و اجرای وظایف
- **Metastore**: ها، اجرای DAG پایگاه داده‌ای برای ذخیره اطلاعات مربوط به
- **Queueing System**: مدیریت صف اجرای وظایف
- **Executors**: Worker اجرای وظایف روی یک یا چندین

معمولاً به صورت **کلاستر** با چندین **Production** را روی یک **سرور واحد اجرا کرد**، اما در محیط **Airflow** می‌توان **Executor Node** راه‌اندازی می‌شود.

چیست؟ DAG در Airflow

های داده استفاده می‌کند Pipeline برای تعریف **Directed Acyclic Graphs (DAGs)** از Airflow.

- و وابستگی‌های آن‌ها است (Tasks) است که شامل تعریف وظایف Python یک کد DAG
- وابسته به خود Task بعد از Task به این معنی است که وظایف دارای ترتیب مشخصی هستند و هر Directed اجرا می‌شود.
- وجود ندارد و جریان داده فقط در یک جهت حرکت می‌کند (Loop) به این معنی است که حلقه Acyclic.

Airflow در DAG مثال یک

به صورت زیر خواهد بود DAG، پردازش فروش محصولات داشتیم Pipeline در سناریوی قبلی که یک

1. برای دریافت اطلاعات فروش SQL Query اجرای
2. برای محاسبه تعداد فروش محصولات Spark اجرای پردازش در
3. Data Warehouse ذخیره خروجی پردازش در.

ها نمایش داده می‌شود Task این فرآیند به صورت یک جریان متوالی از DAG، در نمودار

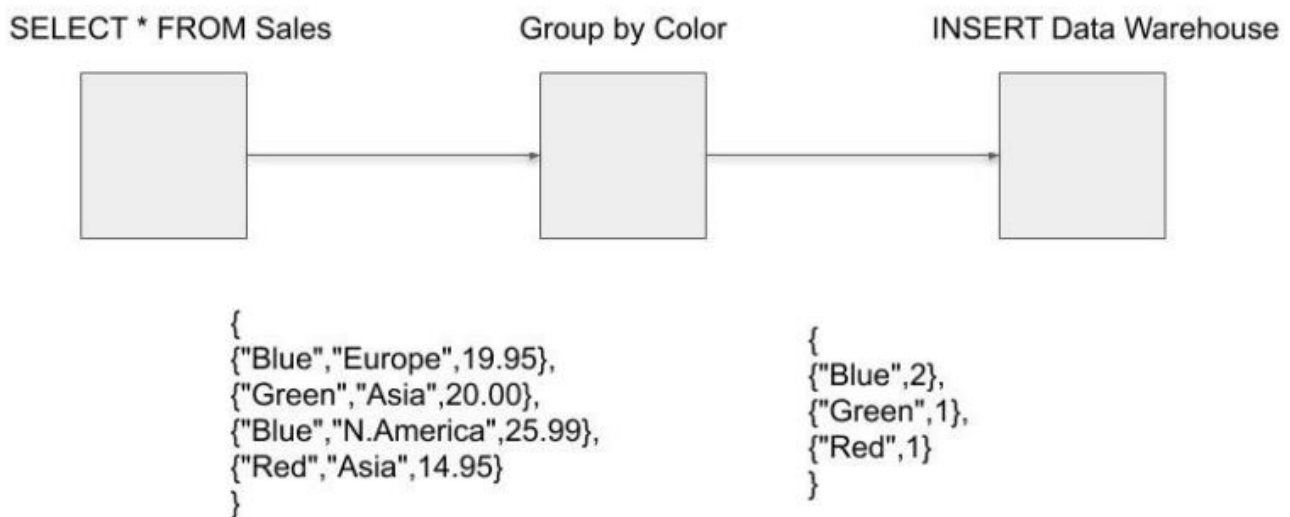


Figure 1.4 – A DAG showing the flow of data between nodes. The task follows the arrows (is directed) from left to right

Airflow در مقایسه با Apache NiFi

برای نمایش مفاهیم مهندسی داده Apache NiFi را پوشش می‌دهد، اما بیشتر از Airflow هرچند این کتاب مبانی استفاده خواهد کرد.

Data است و بیشتر برای (Real-Time) یک ابزار گرافیکی برای انتقال و پردازش داده‌ها در لحظه Apache NiFi: نکته بیشتر برای زمان‌بندی و هماهنگی پردازش‌های داده‌ای Airflow استفاده می‌شود، در حالی که ETL و Ingestion مناسب است.

Apache NiFi

DAG از Airflow است که مانند Data Pipeline یکی دیگر از فریمورک‌های محبوب برای ساخت Apache NiFi آژانس امنیت ملی آمریکا) توسعه داده شد و اکنون در بسیاری از NSA استفاده می‌کند. این فریمورک ابتدا توسط

نهادهای دولتی و سازمانی مورد استفاده قرار می‌گیرد.

مزایای Apache NiFi

- ✓ نصب و راه‌اندازی ساده‌تر دارد و برای مهندسان داده تازه‌کار مناسب است، **Airflow** راه‌اندازی آسان: نسبت به
- ✓ نیازی به کدنویسی زیاد نیست و بسیاری از پردازش‌ها را می‌توان از طریق: **(GUI)** رابط کاربری گرافیکی قدرتمند
- ✓ پیگیری ساده انجام داد
- ✓ **Jython, Clojure, Scala, Groovy** پشتیبانی از زبان‌های مختلف: امکان نوشتن پردازش‌ها در
- ✓ **Pipeline** سرور و اجرای توزیع‌شده: امکان اجرای **Clustering** قابلیت
- ✓ و مانیتورینگ: برای کنترل حجم داده‌های در حال پردازش **Backpressure** مدیریت
- ✓ **NiFi Registry** کنترل نسخه: از طریق
- ✓ و منابع توزیع‌شده جمع‌آوری کرد IoT می‌توان داده‌ها را از دستگاه‌های **Minifi** از طریق **Edge** داده‌برداری در

در مقایسه با سایر ابزارها NiFi

ابزار	زبان اصلی	رابط گرافیکی	مناسب برای	قابلیت Clustering	استفاده برای
Apache NiFi	Java	دارد	پردازش داده در لحظه (Real-Time)	بله	Data Ingestion, ETL, IoT
Apache Airflow	Python	دارد	زمان‌بندی و هماهنگی پردازش‌ها	بله	Batch Processing, Orchestration
Luigi (Spotify)	Python	دارد	مدیریت وابستگی‌های پردازش داده	خیر	Task Automation, ETL

مناسب است، **Batch Processing** در **(Task Orchestration)** برای هماهنگی وظایف **Airflow**: تفاوت اصلی **(Real-Time ETL)** در لحظه **ETL** و **(Streaming Data)** بیشتر برای انتقال و پردازش داده‌های جریانی **NiFi** درحالی‌که استفاده می‌شود.

Luigi در Python جایگزین دیگری برای – Data Pipeline

برای **Python-based** توسعه یافته، یکی دیگر از گزینه‌های **Spotify** که توسط **Luigi** ابزار **NiFi** و **Airflow** علاوه بر است **Data Pipeline** مدیریت.

- ها استفاده می‌کند Task برای اتصال **Graph Structure** از
- دارد **Airflow** رابط گرافیکی مشابه
- مناسب است **(Batch Processing)** برای زمان‌بندی وظایف و مدیریت وابستگی‌ها در پردازش دسته‌ای

پوشش داده نخواهد شد، اما این ابزار همچنان گزینه‌ای قدرتمند و سبک **Luigi** تمرکز دارد و **NiFi** نکته: این کتاب بر روی محسوب می‌شود **Python** برای مهندسان داده‌ی