وظایف مهندسان داده

.مهندسی داده بخشی از اکوسیستم کلان‌داده است و ارتباط نزدیکی با علم داده دارد
مهندسان داده معمولاً در پس‌زمینه کار می‌کنند و به اندازه دانشمندان داده مورد توجه قرار نمی‌گیرند، اما نقش کلیدی .در فرآیند علم داده دارند

مسئولیت‌های مهندس داده

مسئولیت‌های یک مهندس داده بسته به سطح بلوغ داده‌ای 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 ابتدایی‌ترین سطح، مهندسی داده شامل انتقال داده از یک سیستم یا فرمت به سیستم یا فرمت دیگر است. در قالب :مفاهیم رایج‌تر

- مهندسان داده داده‌ها را از یک منبع استخراج می‌کنند (**Extract**)
- سپس برخی تغییرات و پردازش‌ها را روی آن انجام می‌دهند (**Transform**)
- در نهایت، داده‌ها را در محلی قرار می‌دهند که کاربران بتوانند به داده‌های باکیفیت و قابل اطمینان دسترسی داشته باشند (**Load**)

شناخته می‌شود و یکی از مفاهیم کلیدی در مهندسی داده **ETL (Extract, Transform, Load)** این فرآیند با نام .محسوب می‌شود. حال با یک مثال به درک عمیق‌تر این حوزه می‌پردازیم

مثال: فروشگاه آنلاین و نیاز به مهندسی داده

.یک خرده‌فروش آنلاین را در نظر بگیرید که در وب‌سایت خود ویجت‌هایی با رنگ‌های مختلف می‌فروشد
.این وب‌سایت از یک **پایگاه داده رابطه‌ای** پشتیبانی می‌شود که تمام تراکنش‌های انجام‌شده را ذخیره می‌کند

:اکنون می‌خواهیم به این سؤال پاسخ دهیم
در سه‌ماهه 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 می‌توان یک **کوئری** .مهندس داده ندارد

ورود به دنیای مهندسی داده

دیگر امکان‌پذیر 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 تراکنش‌ها وجود داشته باشد
:برای مثال، ممکن است پایگاه داده‌ای برای مناطق جغرافیایی مختلف تعریف شده باشد

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

.در چنین شرایطی، مهندسی داده وارد عمل می‌شود
مهندس داده اتصال‌هایی به تمام پایگاه‌های داده تراکنش‌های منطقه‌ای ایجاد می‌کند، داده‌ها را استخراج کرده و سپس .بارگذاری می‌کند (Data Warehouse) در یک انبار داده
.در این مرحله، می‌توان تعداد ویجت‌های آبی فروخته‌شده را محاسبه کرد

پرسش‌های پیچیده‌تر برای تحلیل داده

شرکت‌ها معمولاً به‌دنبال 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؟

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

پردازش پیچیده‌تر داده‌ها در مهندسی داده

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

یکی از چالش‌های کلیدی، تفاوت **مناطق زمانی (Time Zones)** در مناطق مختلف جغرافیایی است.
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 موقعیت مکانی (Location Field)** به داده‌ها انجام داد.

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

- **فرمت مکانی (Spatial Format):** شامل **مختصات جغرافیایی (Coordinates)** یا **متن استاندارد مکانی (Well-Known Text - WKT)**
- **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 **تبدیل زمان محلی به استاندارد ISO 8601**
4. **بارگذاری داده‌ها** در انبار داده (Data Warehouse)

پایپ‌لاین داده (Data Pipeline)

ترکیب این مراحل با ایجاد یک **پایپ‌لاین داده** انجام می‌شود.

ویژگی‌های پایپ‌لاین داده:

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

- خواهند بود **(Querying)** در نهایت، داده‌های پردازش‌شده 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. ◈

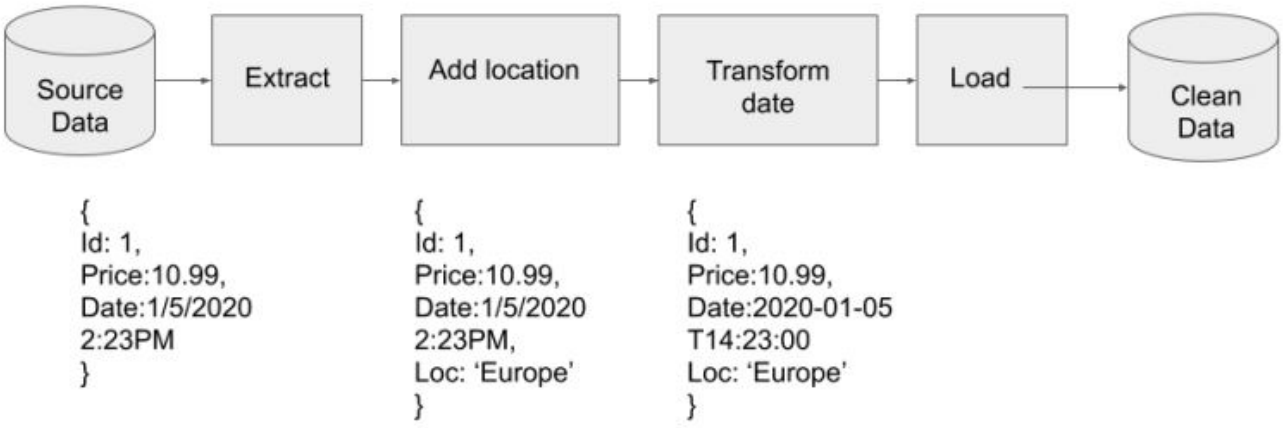

Figure 1.1 – A pipeline that adds a location and modifies the date

---

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**
- تسلط بر **SQL** برای استخراج داده‌ها
- تسلط بر **Python** برای پردازش داده‌ها و خودکارسازی عملیات

2. تبدیل داده (Transformation) و مدل‌سازی داده

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

3. بارگذاری داده در انبار داده (Data Warehouse)

- طراحی و ایجاد **Schema** برای انبار داده
- 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
- آشنایی با ابزارهای مدیریت داده مانند **Apache NiFi** و **Apache Airflow**
- کار با **سکوهای ابری (Cloud Platforms)** مانند **AWS, Google Cloud, Azure** برای ایجاد و مدیریت 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)**
- **به‌صورت ترکیبی (Hybrid) یا چند ابری (Multi-Cloud)**

.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 را به کار می‌برند، اما تمرکز آنها **Python** می‌دهد. هرچند که **مهندسان داده** و **دانشمندان داده** ابزارهای مشابهی مانند :متفاوت است

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

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 به‌شدت 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.
- مهندسان داده باید ابزارهایی را انتخاب کنند که بتوانند **داده‌های مختلف** را از **پایگاه‌های داده** **APIها و فایل‌ها**، 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** **در نزدیک به زمان واقعی (Near Real-Time Processing)** است.

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** اجتناب‌ناپذیر است SQL خاص، یادگیری.

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

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.

Widget	Region
Blue	1
Green	2
Red	3

RegionID	Name
1	N.America
2	Asia
3	Europe

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** پایگاه‌های داده **NoSQL** دیگر مانند

این پایگاه‌های داده برخلاف پایگاه‌های داده رابطه‌ای که داده‌ها را به‌صورت **رکورد** ذخیره می‌کنند، داده‌ها را به‌صورت **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.

- تمام این پایگاه‌های داده ستونی (مانند **Amazon Redshift**، **Google BigQuery**، و **Apache Cassandra**) **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

- **Elasticsearch** است که در واقع یک **موتور جستجو** مبتنی بر **NoSQL** یکی از پایگاه‌های داده **Apache Lucene** است.
- **Elasticsearch** مشابه **Apache Solr** 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** (زبان اختصاصی دامنه) برای پرس‌وجو استفاده می‌کند که **SQL** نیست است. بلکه یک **پرس‌وجوی** **JSON**
- ذخیره می‌کند و برخلاف پایگاه‌های داده رابطه‌ای، **(Documents)** داده‌ها را به‌صورت **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 به‌صورت **Batch** موتورهای پردازش داده به مهندسان داده این امکان را می‌دهند که داده‌ها را چه به‌صورت 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 کار **Python DataFrames** به‌خوبی با Apache Spark .بنویسند **Scala** و **Java**، **Python** تبدیلات خود را با استفاده از DataFrame، Spark دارای تبدیل کرده است. علاوه بر **Python** می‌کند، که آن را به ابزاری مناسب برای برنامه‌نویسان نیز هست **Resilient Distributed Datasets (RDDs)**.

RDDs

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

سایر موتورهای پردازش داده

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

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

Unbounded Stream چیست؟

جریانی از داده است که پایان مشخصی ندارد. برای مثال، **سنسور دما** که به‌طور مداوم دما **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

Data Pipelines (خطوط پردازش داده)

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 پردازش کنید تا تعداد فروش هر محصول را محاسبه کنید و در نهایت نتیجه را در **Apache Spark** از طریق **Data Warehouse** ذخیره کنید. این یک **Pipeline** محسوب می‌شود **پردازش داده**.

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

Crontab با Pipeline 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
```

به‌صورت **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 کدام‌یک **اجرا نشده‌اند**؟Pipeline **های اجراشده**: چگونه بدانید که چه**Pipeline رهگیری**
- سریع‌تر از وظیفه بعدی اجرا شود، چگونه می‌توان داده‌ها **(Tasks) وظایف** اگر یکی از :**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**، **Pipelineها** با پیچیده‌تر شدن ها خواهید داشت، مانندPipeline:

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

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

Apache Airflow

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

Airflow اجزای اصلی

از چندین مؤلفه کلیدی تشکیل شده است Airflow:

- **Web Server**: رابط کاربری برای مانیتورینگ و مدیریت DAGها
- **Scheduler**: زمان‌بندی و اجرای وظایف (Tasks)

- **Metastore**: پایگاه داده‌ای برای ذخیره اطلاعات مربوط به DAGها، اجرای Taskها، و وضعیت آن‌ها
- **Queueing System**: مدیریت صف اجرای وظایف
- **Executors**: اجرای وظایف روی یک یا چندین Worker

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

DAG در Airflow چیست؟

Airflow از Directed Acyclic Graphs (DAGs) برای تعریف Pipelineهای داده استفاده می‌کند.

DAG یک کد Python وظایف (Tasks) است که شامل تعریف و وابستگی‌های آن‌ها است.

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

مثال یک DAG در Airflow

در سناریوی قبلی که یک **Pipeline** داشتیم **پردازش فروش محصولات**، DAG به‌صورت زیر خواهد بود:

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

در نمودار DAG، این فرآیند به‌صورت یک **جریان متوالی از Task**ها نمایش داده می‌شود.

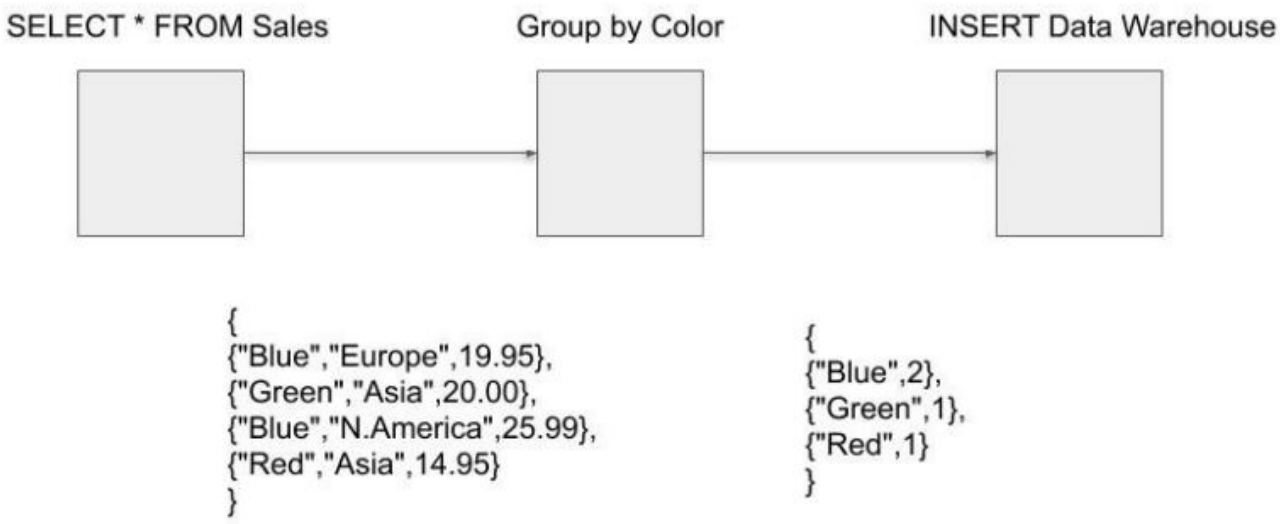

Figure 1.4 – A DAG showing the flow of data between nodes. The task follows the arrows (is directed) from left to right

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.

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

Apache NiFi

Apache NiFi ساخت یکی دیگر از فریمورک‌های محبوب برای **Data Pipeline** است که مانند **Airflow** از **DAG 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.

Apache NiFi مزایای

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

NiFi در مقایسه با سایر ابزارها

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**

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

Python در Data Pipeline جایگزین دیگری برای – Luigi

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

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

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**.