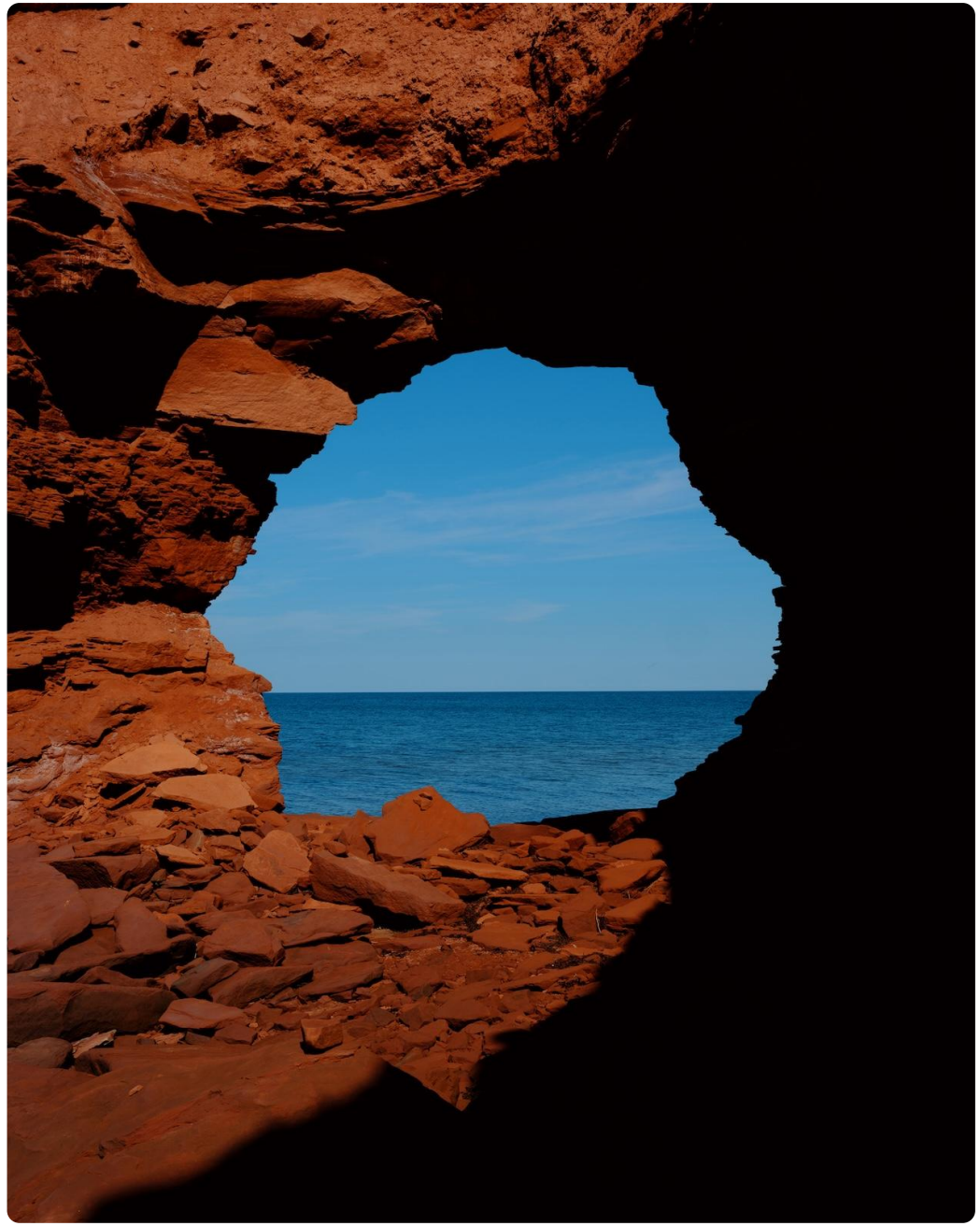
OpenAI

Harnessing LLMs for Scalable Data Minimization



Charles de Bourcy, PhD

Member of Technical Staff

What is Data Minimization?

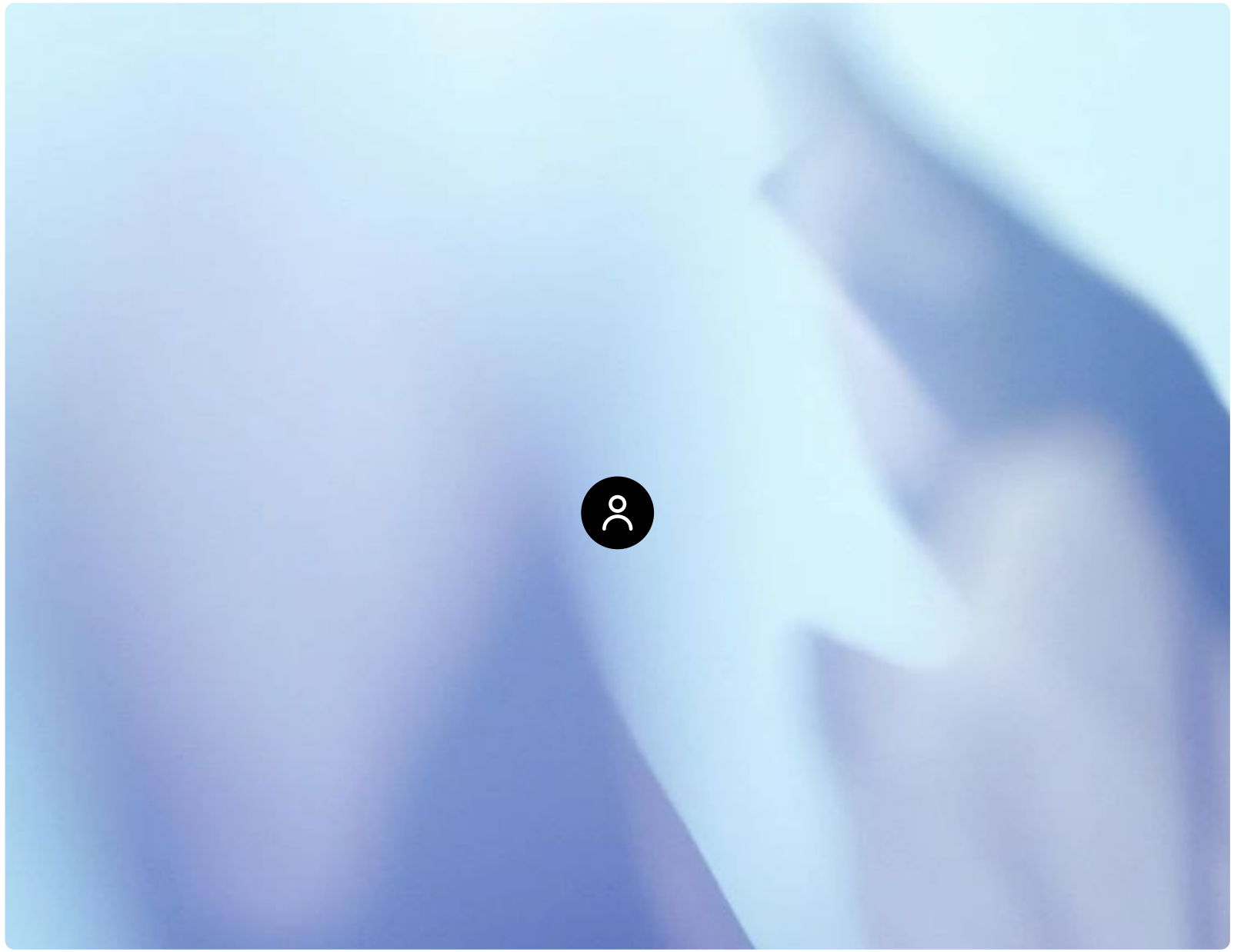
Only collect and retain PII as necessary for the specified purpose.

→ Don’t process irrelevant PII.

→ Expire PII when it is no longer needed.

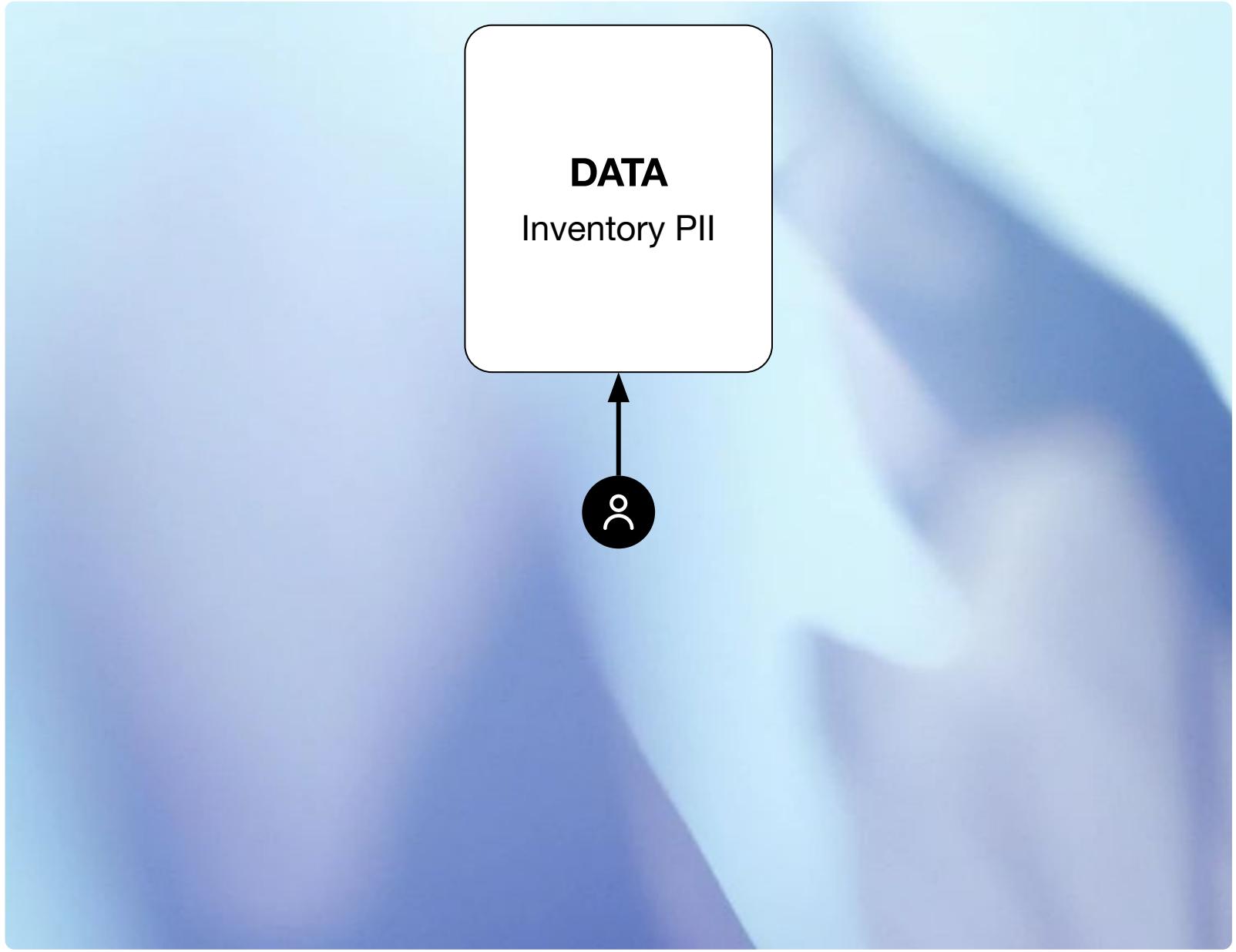
What does it take to scale Data Minimization?

Understand and influence **large numbers** of code modules, data stores and employees across an organization.



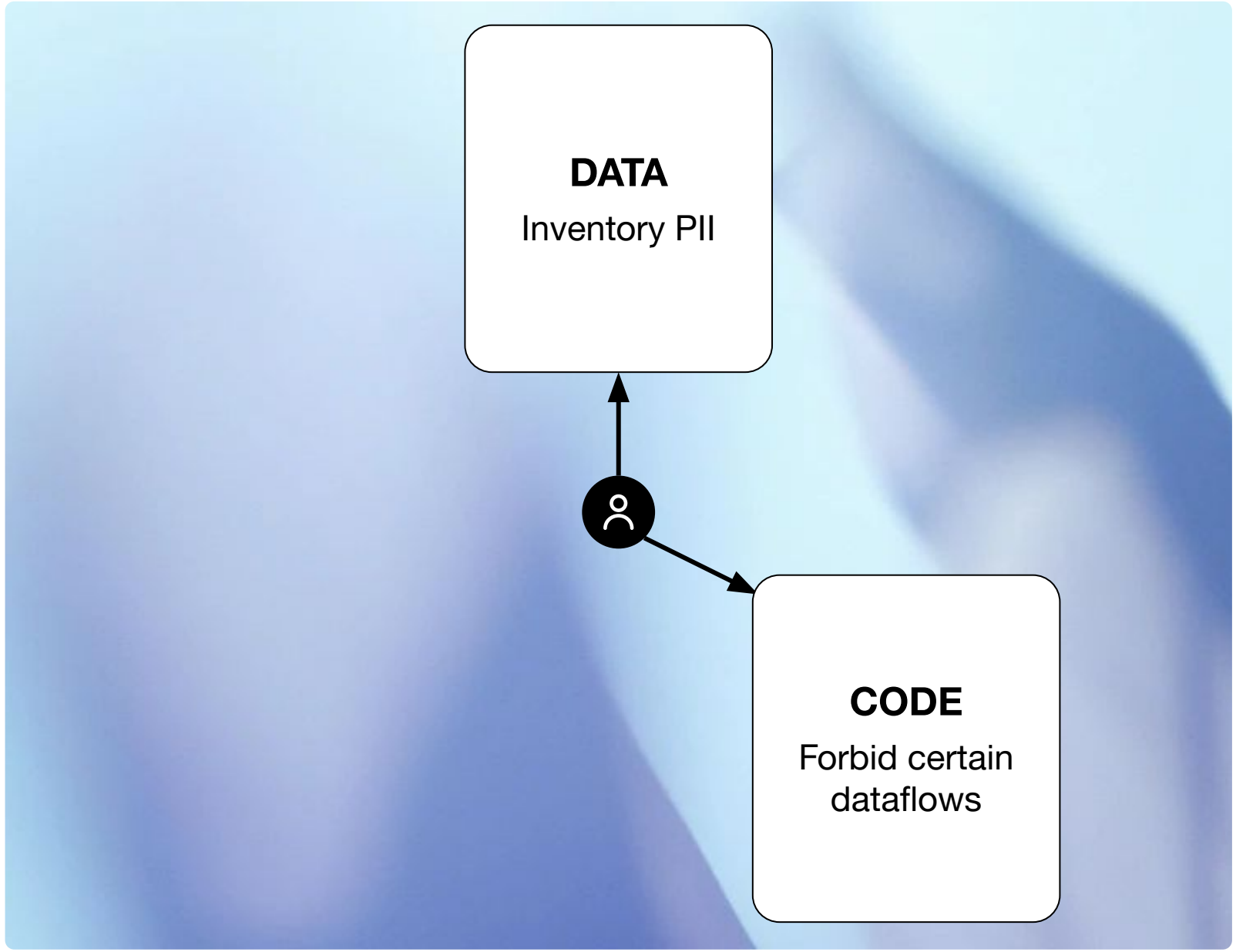
What does it take to scale Data Minimization?

Understand and influence **large numbers** of code modules, data stores and employees across an organization.



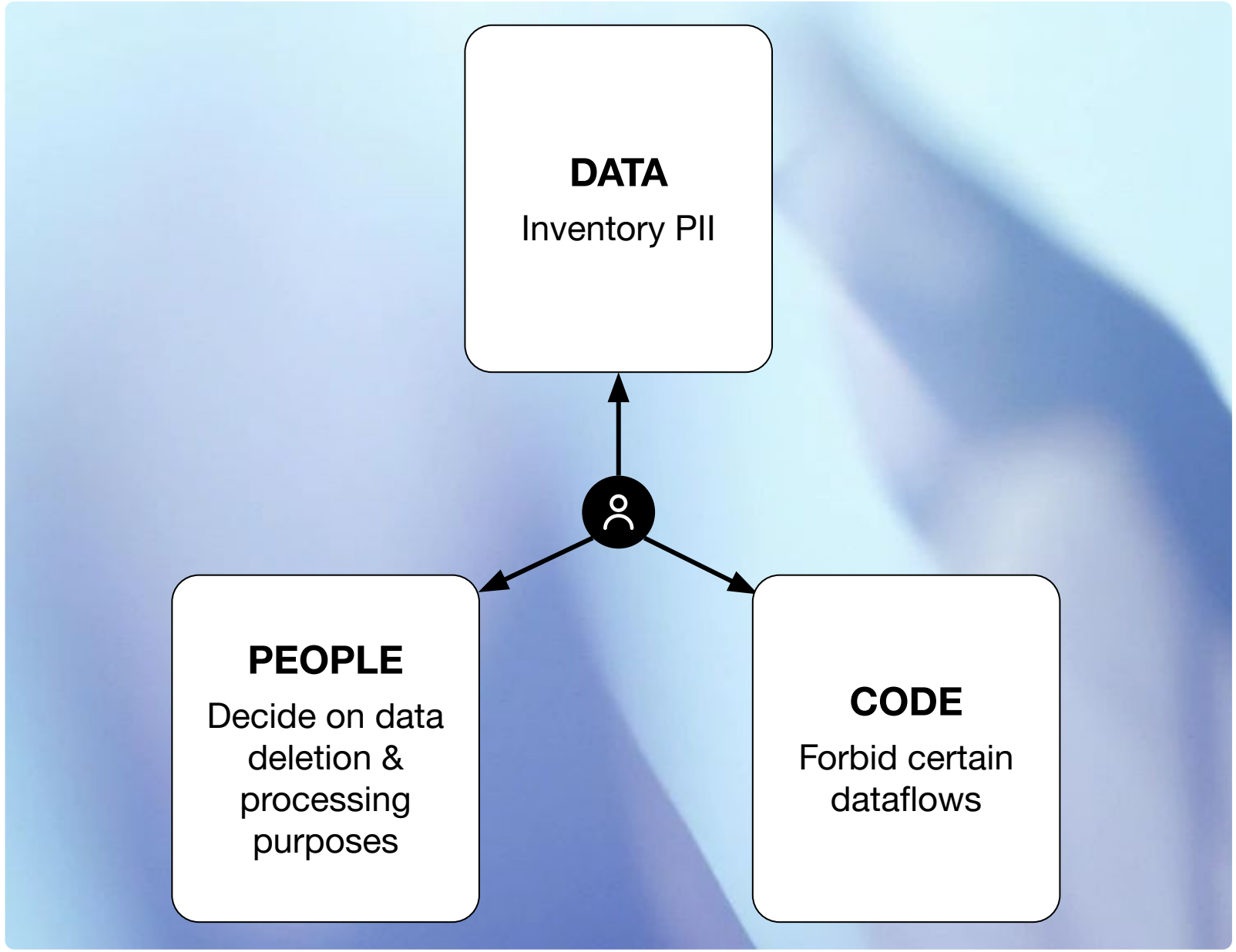
What does it take to scale Data Minimization?

Understand and influence **large numbers** of code modules, data stores and employees across an organization.

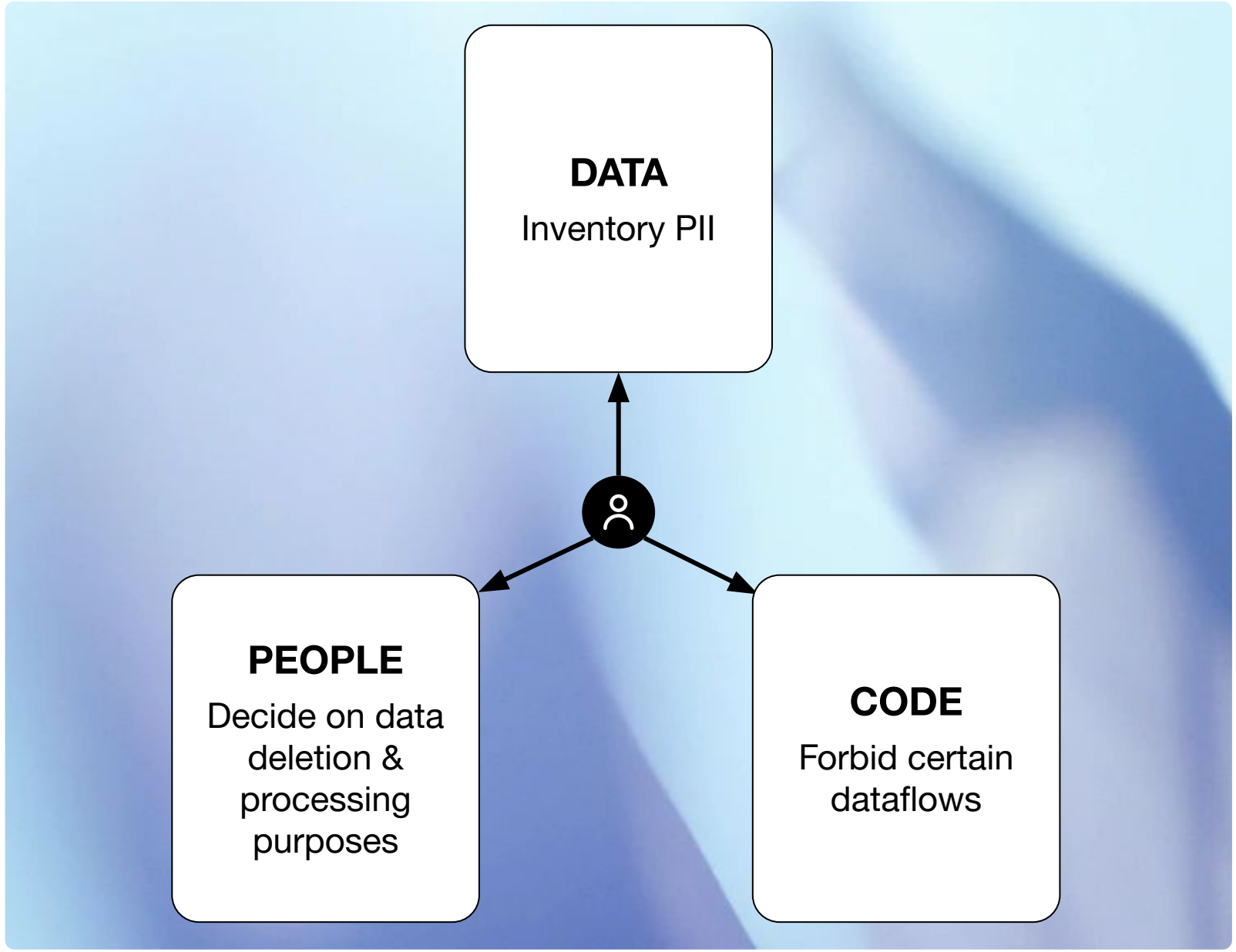


What does it take to scale Data Minimization?

Understand and influence **large numbers** of code modules, data stores and employees across an organization.



Can LLMs help improve accuracy & efficiency of Data Minimization processes?



Data

Task: Classify PII across diverse data stores

Task: Classify PII across diverse data stores

We can prompt, and later finetune, a foundation model to recognize PII categories in different forms and contexts.

Task: Classify PII across diverse data stores

We can prompt, and later finetune, a foundation model to recognize PII categories in different forms and contexts.

Table: contact_methods

account_type	phone
A	+1 234 567 8910
A	+55 1 55 1234 567 89
B	+1 800 123 4567

Task: Classify PII across diverse data stores

We can prompt, and later finetune, a foundation model to recognize PII categories in different forms and contexts.

Pick the correct label for this data object:

- *name*
- *phone_number*
- *email_address*
- *other*

Table: contact_methods

account_type	phone
A	+1 234 567 8910
A	+55 1 55 1234 567 89
B	+1 800 123 4567

Task: Classify PII across diverse data stores

We can prompt, and later finetune, a foundation model to recognize PII categories in different forms and contexts.

Pick the correct label for this data object:

- *name*
- ***phone_number***
- *email_address*
- *other*

Table: contact_methods

account_type	phone
A	+1 234 567 8910
A	+55 1 55 1234 567 89
B	+1 800 123 4567

Task: Classify PII across diverse data stores

Pick the correct label for this data object:

- *name*
- ***phone_number***
- *email_address*
- *other*

This approach can recognize a wider variety of PII with higher accuracy than traditional rules or regex-based approaches.

Table: contact_methods

account_type	phone
A	(234) 567-8910
A	+55 1 55 1234 567 89
B	001-800.123.4567

Task: Classify PII across diverse data stores

Pick the correct label for this data object:

- *name*
- ***phone_number***
- *email_address*
- *other*

This approach can recognize a wider variety of PII with higher accuracy than traditional rules or regex-based approaches.

Table: contact_methods

us_area_code	phone
408	1234567890
408	0987654321
800	1234567890

Task: Classify PII across diverse data stores

Pick the correct label for this data object:

- *name*
- *phone_number*
- *email_address*
- ***other***

In case of ambiguity, the model can take context and domain knowledge into account.

Table: accounts

account_type	user_id
A	1234567890
A	0987654321
B	1234567890

Task: Classify PII across diverse data stores

Pick the correct label for this data object:

- *name*
- ***phone_number***
- *email_address*
- *other*

In case of ambiguity, the model can take context and domain knowledge into account.

Table: accounts

account_type	user_id
A	+1 234 567 8910
A	+55 1 55 1234 567 89
B	+1 800 123 4567

Task: Classify PII across diverse data stores

Pick the correct label for this data object:

- *name*
- ***phone_number***
- *email_address*
- *other*

Since the foundation model has an understanding of the world, the categorization scheme can be adjusted flexibly.

Table: contact_methods

account_type	phone
A	+1 234 567 8910
A	+55 1 55 1234 567 89
B	+1 800 123 4567

Task: Classify PII across diverse data stores

Pick the correct label for this data object:

- ***contact_details***
- *not_contact_details*

Since the foundation model has an understanding of the world, the categorization scheme can be adjusted flexibly.

Table: contact_methods

account_type	phone
A	+1 234 567 8910
A	+55 1 55 1234 567 89
B	+1 800 123 4567

Task: Classify PII across diverse data stores

Pick the correct label for this data object:

- ***contact_details***
- *not_contact_details*

Since the foundation model has an understanding of the world, the categorization scheme can be adjusted flexibly.

Overhead is lower than creating new rules/regexes or targeted ML models for each new category.

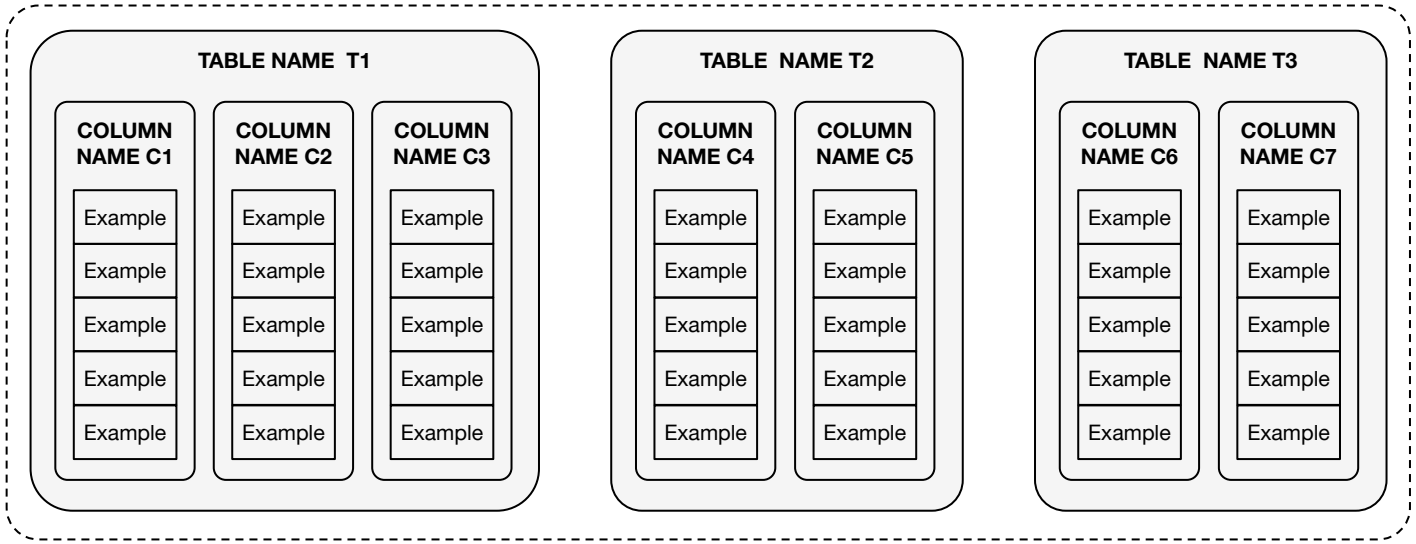
Table: contact_methods

account_type	email
A	me@example.com
A	test@example.com
B	jane@doe.info

Task: Classify PII across diverse data stores

This flexible, automatable approach becomes especially useful if a large organization has

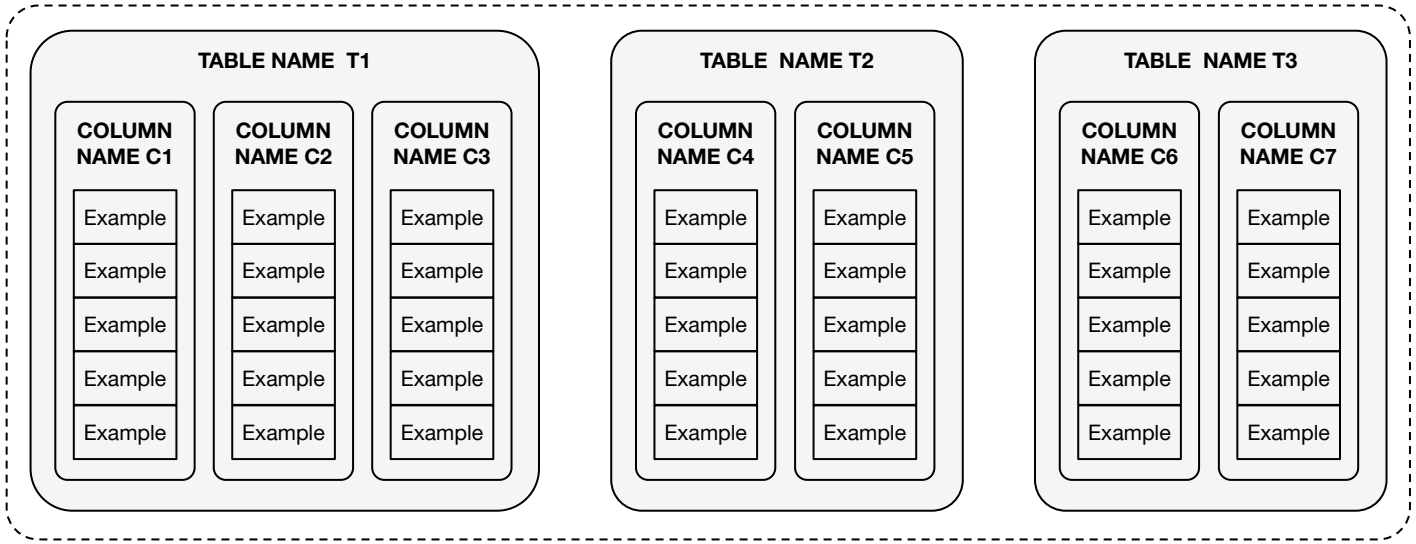
Task: Classify PII across diverse data stores



This flexible, automatable approach becomes especially useful if a large organization has

- many diverse data stores

Task: Classify PII across diverse data stores



This flexible, automatable approach becomes especially useful if a large organization has

- many diverse data stores
- a complex, potentially evolving PII taxonomy.

PII TAXONOMY

Category A
Category B
Category C
Category D
...

Task: Classify PII across diverse data stores

Then the approach shortens turnaround time for a draft classification with good accuracy.

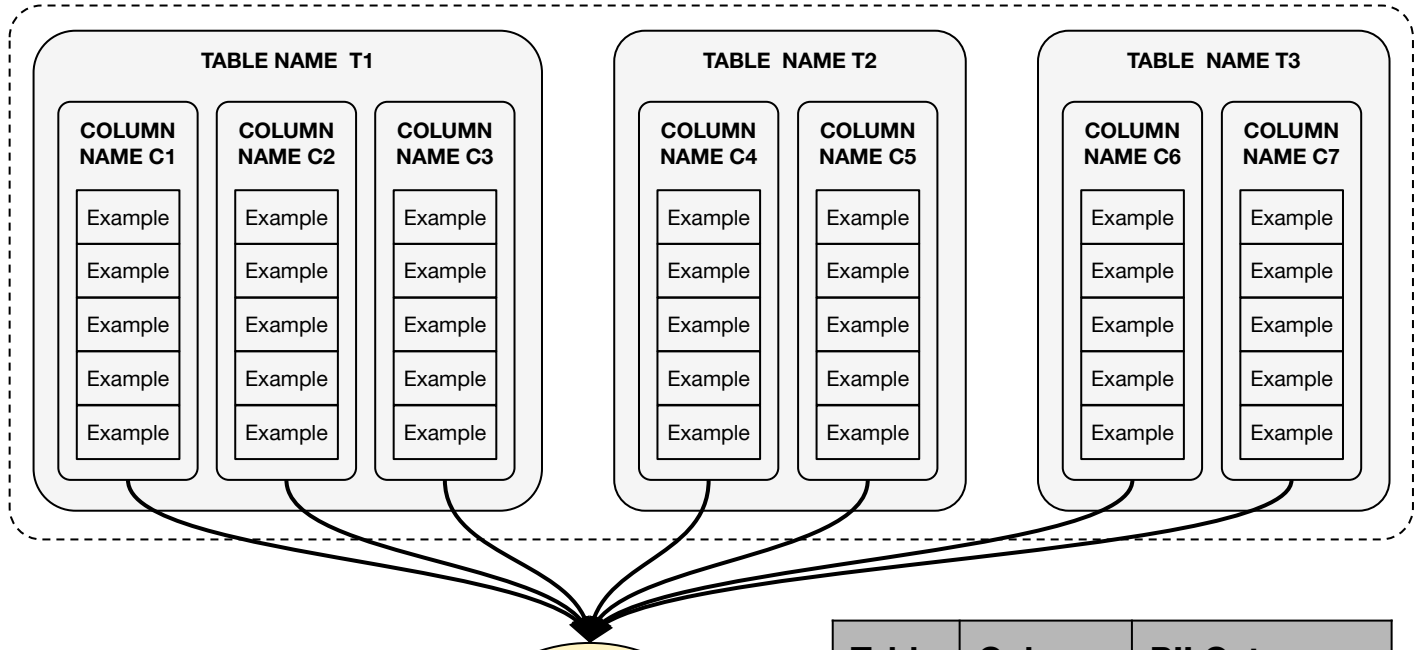


Table	Column	PII Category
T1	C1	A
T1	C2	B
T1	C3	None
T2	C4	None
T2	C5	C
T3	C6	None
T3	C7	A

Task: Classify PII across diverse data stores

Limitations include:

- Higher latency, more resource-intensive than lightweight rule engines or smaller models
- If not in-house, sending data to 3rd-party model provider could in itself conflict with Data Minimization (though mitigations like Zero Data Retention offerings exist).

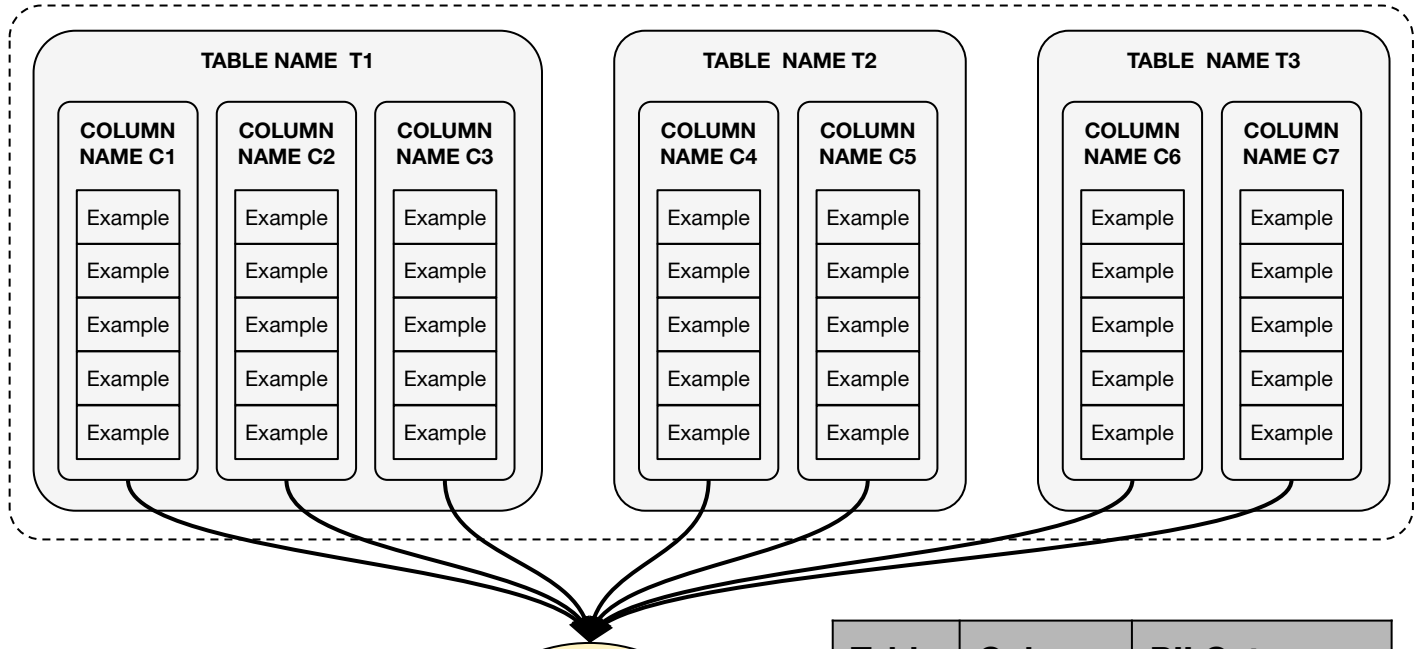


Table	Column	PII Category
T1	C1	A
T1	C2	B
T1	C3	None
T2	C4	None
T2	C5	C
T3	C6	None
T3	C7	A

Code

Task: Review and forbid certain data flows to minimize sprawl

Task: Review and forbid certain data flows to minimize sprawl

We can prompt an LLM-based coding agent to map out dataflows in a codebase.

Task: Review and forbid certain data flows to minimize sprawl

We can prompt an LLM-based coding agent to map out dataflows in a codebase.

“What external data does this app store, and where?”

app.py

```
app = App("Toy App")

@app.route("/submit_phone", methods=["POST"])
def submit():
    save_phone_number(**validate(request.json))
    return {"status": "ok"}

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

Task: Review and forbid certain data flows to minimize sprawl

We can prompt an LLM-based coding agent to map out dataflows in a codebase.

“What external data does this app store, and where?”

app.py

```
app = App("Toy App")

@app.route("/submit_phone", methods=["POST"])
def submit():
    save_phone_number(**validate(request.json))
    return {"status": "ok"}

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

App serves public internet traffic.

Task: Review and forbid certain data flows to minimize sprawl

We can prompt an LLM-based coding agent to map out dataflows in a codebase.

“What external data does this app store, and where?”

app.py

```
app = App("Toy App")

@app.route("/submit_phone", methods=["POST"])
def submit():
    save_phone_number(**validate(request.json))
    return {"status": "ok"}

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

App serves public internet traffic.

Takes phone number input.

Task: Review and forbid certain data flows to minimize sprawl

We can prompt an LLM-based coding agent to map out dataflows in a codebase.

“What external data does this app store, and where?”

app.py

```
app = App("Toy App")

@app.route("/submit_phone", methods=["POST"])
def submit():
    save_phone_number(**validate(request.json))
    return {"status": "ok"}

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

App serves public internet traffic.

Takes phone number input.

Let’s find the implementation…

Task: Review and forbid certain data flows to minimize sprawl

We can prompt an LLM-based coding agent to map out dataflows in a codebase.

“What external data does this app store, and where?”

app.py

```
app = App("Toy App")

@app.route("/submit_phone", methods=["POST"])
def submit():
    save_phone_number(**validate(request.json))
    return {"status": "ok"}

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

storage.py

```
cache = Cache.from_url(os.getenv("CACHE_URL"))

def save_phone_number(id, number):
  with db_session() as s:
    s.add(Phone(id, number)); s.commit()
    log("Saved phone", id)
    cache.set(f"user:{id}:phone", number)
```

App serves public internet traffic.

Takes phone number input.

Let’s find the implementation…

Task: Review and forbid certain data flows to minimize sprawl

We can prompt an LLM-based coding agent to map out dataflows in a codebase.

“What external data does this app store, and where?”

app.py

```
app = App("Toy App")

@app.route("/submit_phone", methods=["POST"])
def submit():
    save_phone_number(**validate(request.json))
    return {"status": "ok"}

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

storage.py

```
cache = Cache.from_url(os.getenv("CACHE_URL"))

def save_phone_number(id, number):
  with db_session() as s:
    s.add(Phone(id, number)); s.commit()
    log("Saved phone", id)
    cache.set(f"user:{id}:phone", number)
```

App serves public internet traffic.

Takes phone number input.

Let’s find the implementation…

Stores phone # with an ID in a database

Task: Review and forbid certain data flows to minimize sprawl

We can prompt an LLM-based coding agent to map out dataflows in a codebase.

“What external data does this app store, and where?”

app.py

```
app = App("Toy App")

@app.route("/submit_phone", methods=["POST"])
def submit():
    save_phone_number(**validate(request.json))
    return {"status": "ok"}

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

storage.py

```
cache = Cache.from_url(os.getenv("CACHE_URL"))

def save_phone_number(id, number):
  with db_session() as s:
    s.add(Phone(id, number)); s.commit()
    log("Saved phone", id)
    cache.set(f"user:{id}:phone", number)
```

App serves public internet traffic.

Takes phone number input.

Let’s find the implementation…

Stores phone # with an ID in a database & cache.

Task: Review and forbid certain data flows to minimize sprawl

We can prompt an LLM-based coding agent to map out dataflows in a codebase.

“What external data does this app store, and where?”

app.py

```
app = App("Toy App")

@app.route("/submit_phone", methods=["POST"])
def submit():
    save_phone_number(**validate(request.json))
    return {"status": "ok"}

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

storage.py

```
cache = Cache.from_url(os.getenv("CACHE_URL"))

def save_phone_number(id, number):
  with db_session() as s:
    s.add(Phone(id, number)); s.commit()
    log("Saved phone", id)
    cache.set(f"user:{id}:phone", number)
```

App serves public internet traffic.

Takes phone number input.

Let’s find the implementation…

Stores phone # with an ID in a database & cache.

Logs success event with an ID.

Task: Review and forbid certain data flows to minimize sprawl

We can prompt an LLM-based coding agent to map out dataflows in a codebase.

“What external data does this app store, and where?”

app.py

```
app = App("Toy App")

@app.route("/submit_phone", methods=["POST"])
def submit():
    save_phone_number(**validate(request.json))
    return {"status": "ok"}

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

storage.py

```
cache = Cache.from_url(os.getenv("CACHE_URL"))

def save_phone_number(id, number):
  with db_session() as s:
    s.add(Phone(id, number)); s.commit()
    log("Saved phone", id)
    cache.set(f"user:{id}:phone", number)
```

App serves public internet traffic.

Takes phone number input.

Let’s find the implementation…

Stores phone # with an ID in a database & cache.

Logs success event with an ID.

Let’s find the storage infra…

Task: Review and forbid certain data flows to minimize sprawl

We can prompt an LLM-based coding agent to map out dataflows in a codebase.

“What external data does this app store, and where?”

app.py

```
app = App("Toy App")

@app.route("/submit_phone", methods=["POST"])
def submit():
    save_phone_number(**validate(request.json))
    return {"status": "ok"}

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

storage.py

```
cache = Cache.from_url(os.getenv("CACHE_URL"))

def save_phone_number(id, number):
  with db_session() as s:
    s.add(Phone(id, number)); s.commit()
    log("Saved phone", id)
    cache.set(f"user:{id}:phone", number)
```

infra/main.tf

```
resource "app_deployment" "toy_app" {
  env = {
    DB_URL    = module.db.connection_url
    CACHE_URL = module.cache.connection_url
    LOG_SINK  = module.logging.sink_url
  }
}
```

App serves public internet traffic.

Takes phone number input.

Let’s find the implementation…

Stores phone # with an ID in a database & cache.

Logs success event with an ID.

Let’s find the storage infra…

Found app’s infra code and relevant storage references.

Task: Review and forbid certain data flows to minimize sprawl

We can prompt an LLM-based coding agent to map out dataflows in a codebase.

“What external data does this app store, and where?”

app.py

```
app = App("Toy App")

@app.route("/submit_phone", methods=["POST"])
def submit():
    save_phone_number(**validate(request.json))
    return {"status": "ok"}

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

storage.py

```
cache = Cache.from_url(os.getenv("CACHE_URL"))

def save_phone_number(id, number):
  with db_session() as s:
    s.add(Phone(id, number)); s.commit()
    log("Saved phone", id)
    cache.set(f"user:{id}:phone", number)
```

infra/main.tf

```
resource "app_deployment" "toy_app" {
  env = {
    DB_URL    = module.db.connection_url
    CACHE_URL = module.cache.connection_url
    LOG_SINK  = module.logging.sink_url
  }
}
```

App serves public internet traffic.

Takes phone number input.

Let’s find the implementation…

Stores phone # with an ID in a database & cache.

Logs success event with an ID.

Let’s find the storage infra…

Found app’s infra code and relevant storage references.

Let’s put findings in the requested format…

Task: Review and forbid certain data flows to minimize sprawl

We can prompt an LLM-based coding agent to map out dataflows in a codebase.

“What external data does this app store, and where?”

app.py

```
app = App("Toy App")

@app.route("/submit_phone", methods=["POST"])
def submit():
    save_phone_number(**validate(request.json))
    return {"status": "ok"}

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

storage.py

```
cache = Cache.from_url(os.getenv("CACHE_URL"))

def save_phone_number(id, number):
  with db_session() as s:
    s.add(Phone(id, number)); s.commit()
    log("Saved phone", id)
    cache.set(f"user:{id}:phone", number)
```

infra/main.tf

```
resource "app_deployment" "toy_app" {
  env = {
    DB_URL    = module.db.connection_url
    CACHE_URL = module.cache.connection_url
    LOG_SINK  = module.logging.sink_url
  }
}
```

App serves public internet traffic.

Takes phone number input.

Let’s find the implementation…

Stores phone # with an ID in a database & cache.

Logs success event with an ID.

Let’s find the storage infra…

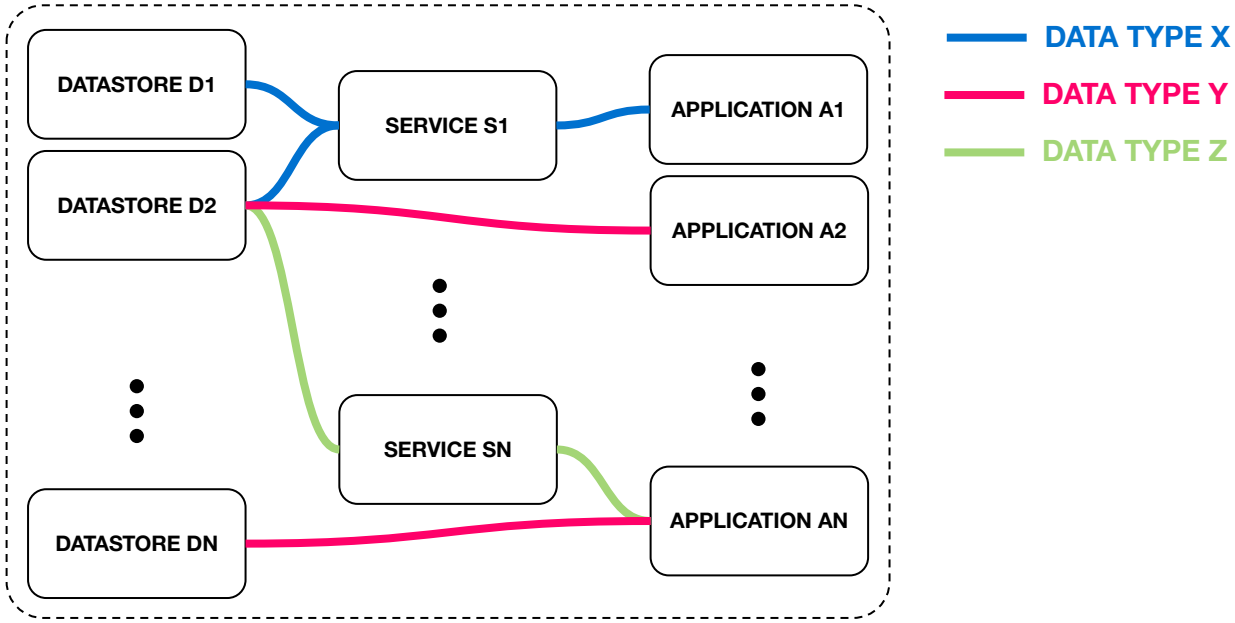
Found app’s infra code and relevant storage references.

Let’s put findings in the requested format…

Application	Datastore	Data type	Infra Reference
Toy App	SQL Database	user_id, phone_number	DB_URL
Toy App	Cache	user_id, phone_number	CACHE_URL
Toy App	Application logs	user_id	LOG_SINK

Task: Review and forbid certain data flows to minimize sprawl

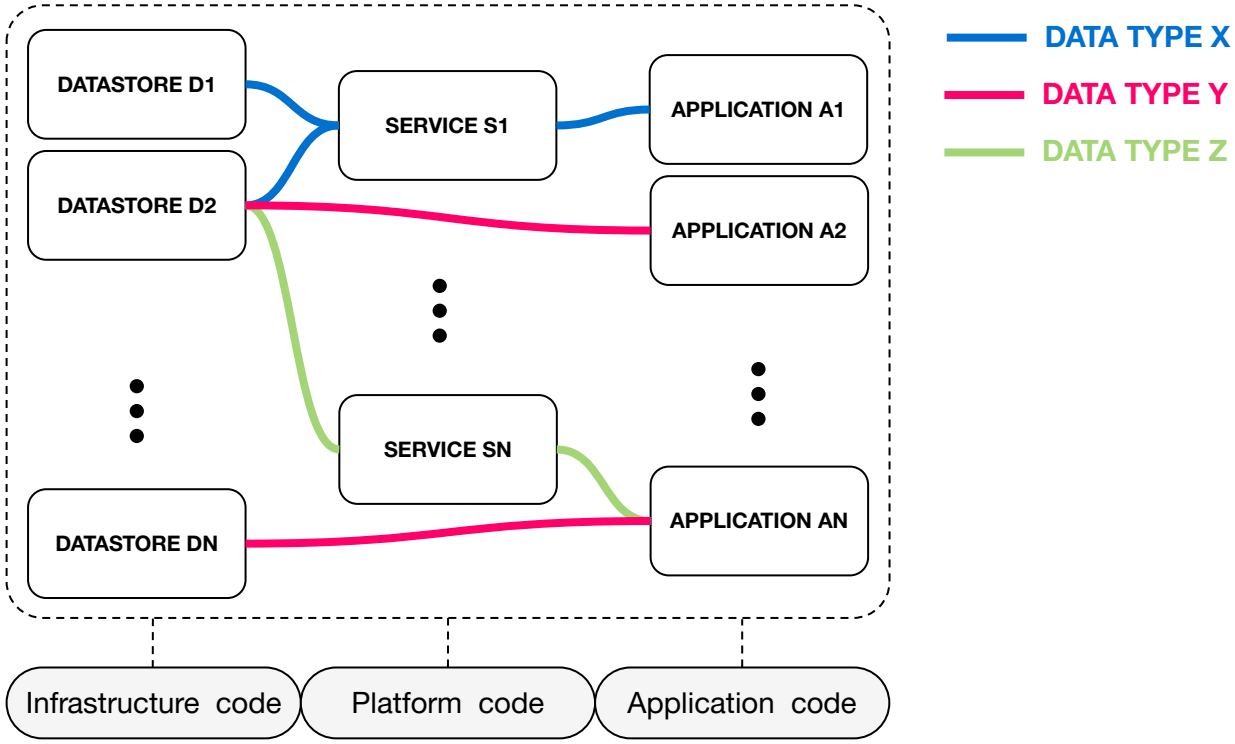
Particularly useful in large complex systems that are continually evolving.



Task: Review and forbid certain data flows to minimize sprawl

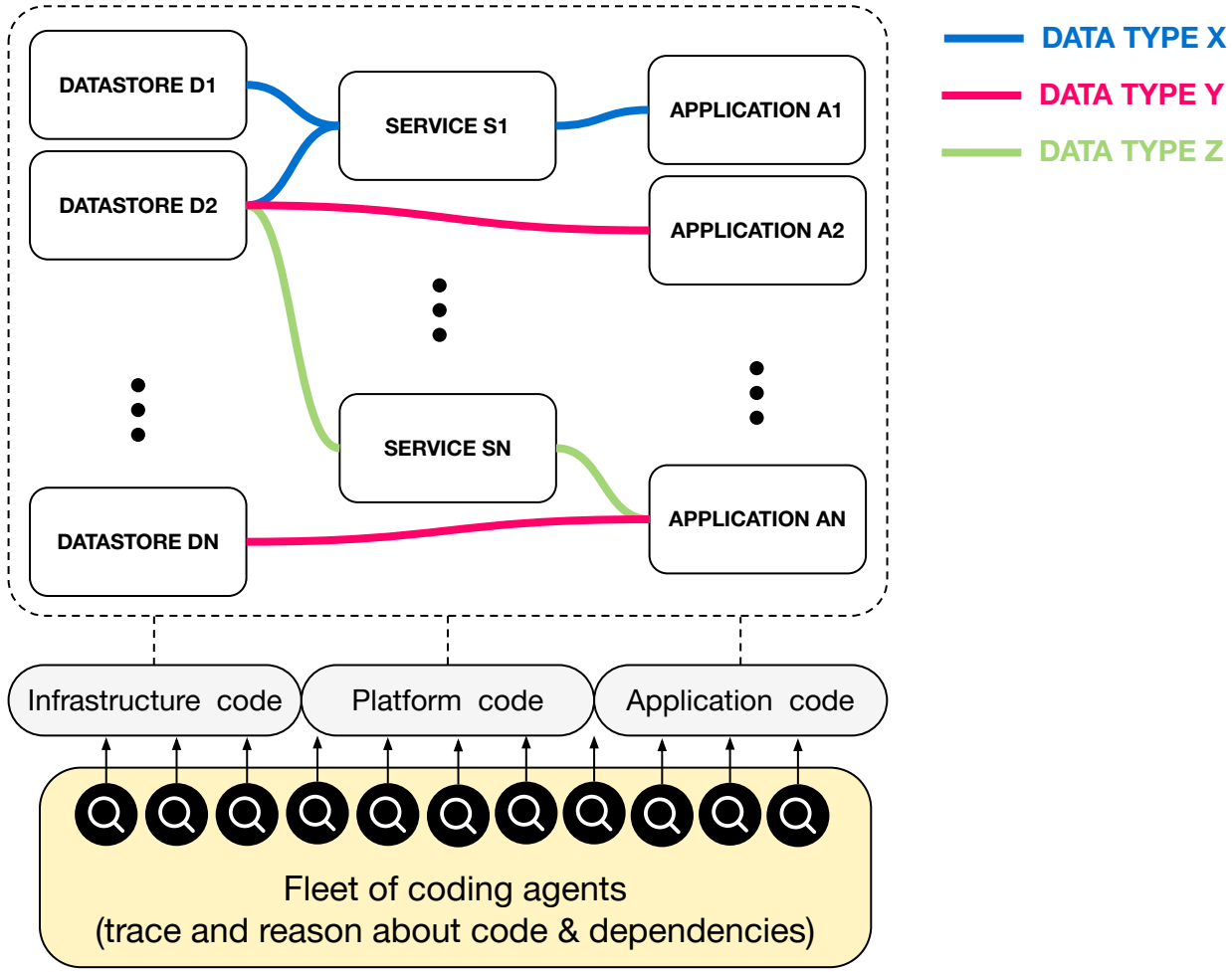
Particularly useful in large complex systems that are continually evolving.

There, improved tools for understanding code can reduce manual effort of reviewing newly built systems.



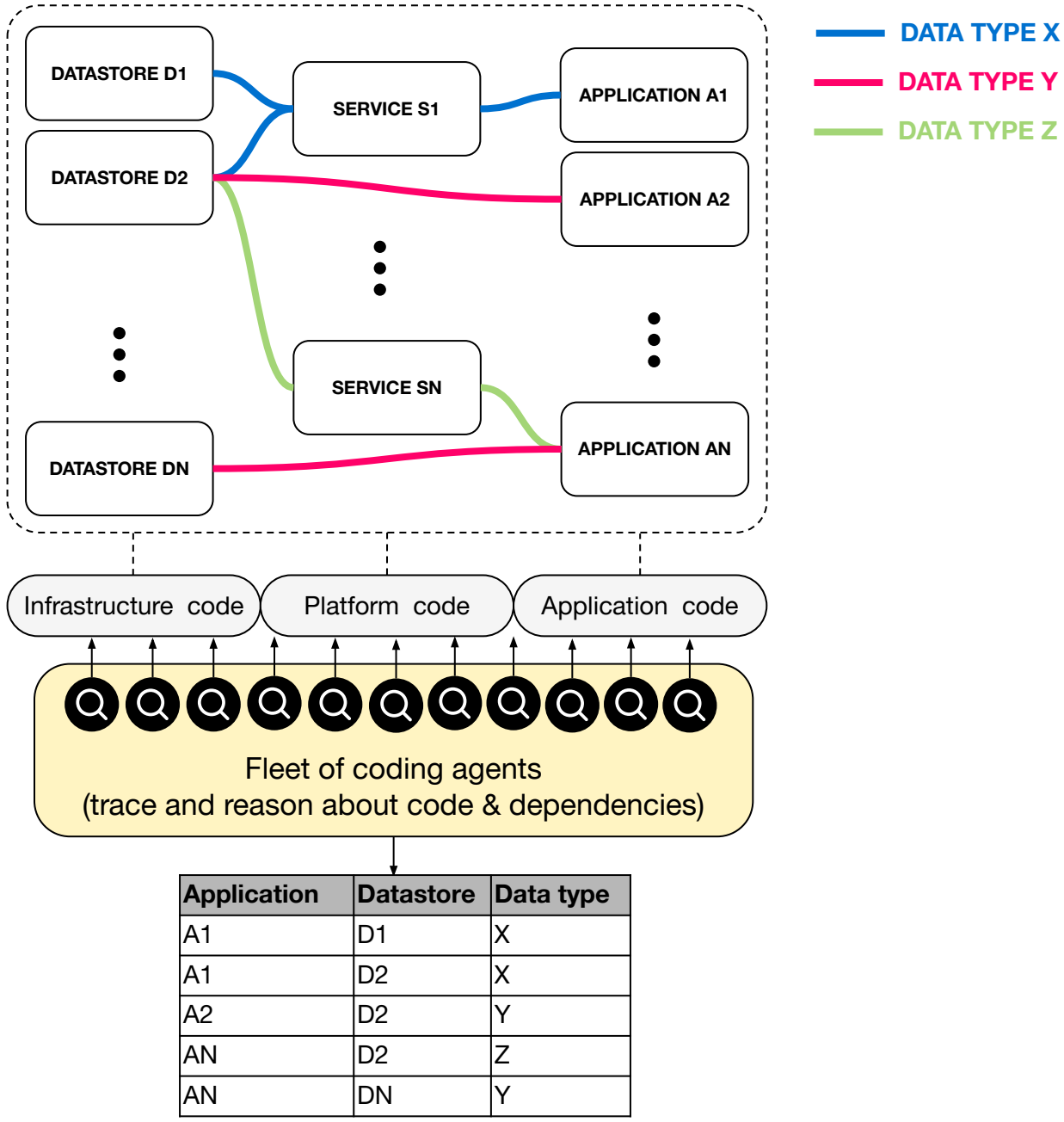
Task: Review and forbid certain data flows to minimize sprawl

Parallelized LLM-based coding agents can regularly extract up-to-date insights from thousands of ever-changing modules for review.



Task: Review and forbid certain data flows to minimize sprawl

Parallelized LLM-based coding agents can regularly extract up-to-date insights from thousands of ever-changing modules for review.

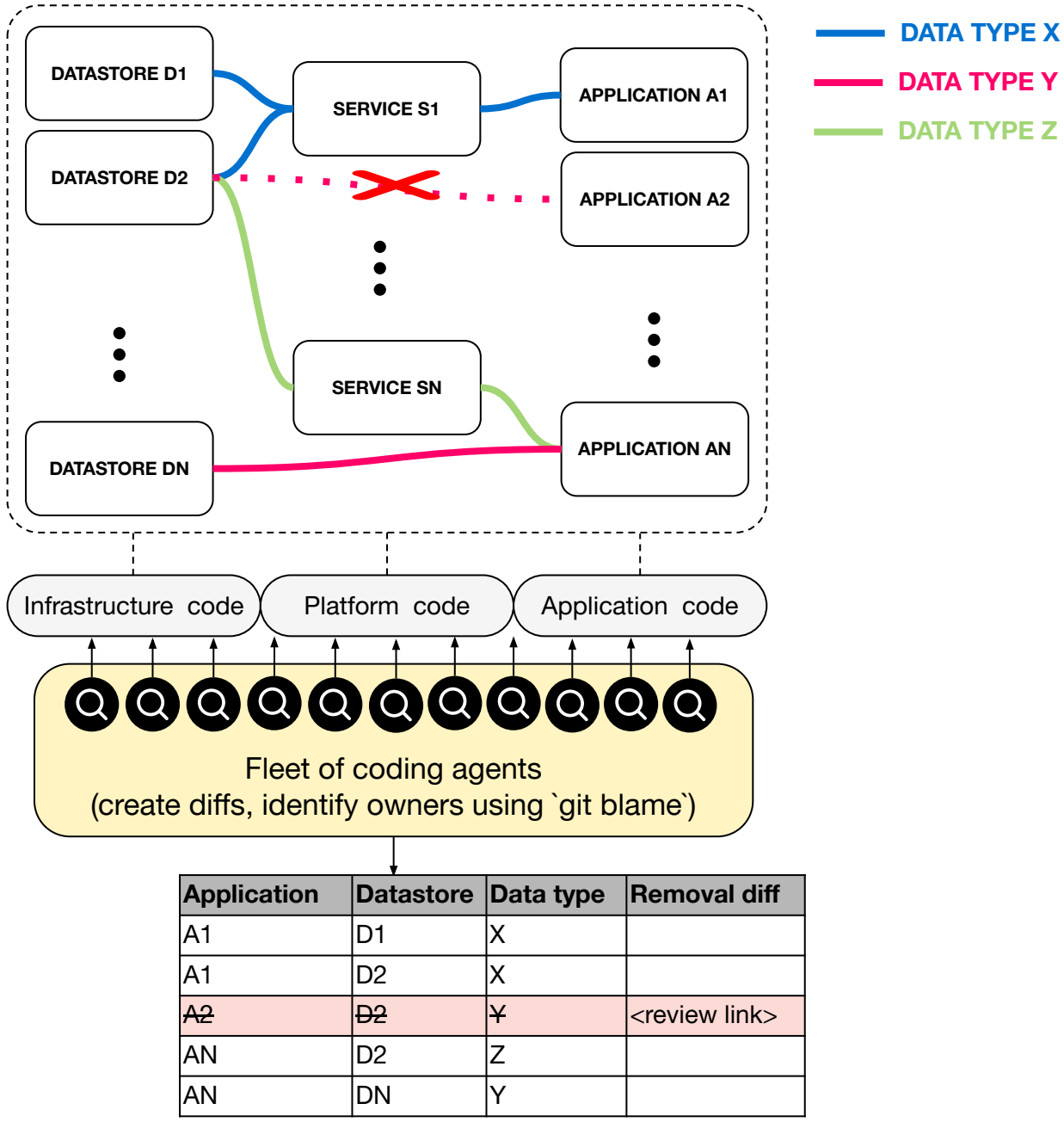


Application	Datastore	Data type
A1	D1	X
A1	D2	X
A2	D2	Y
AN	D2	Z
AN	DN	Y

Task: Review and forbid certain data flows to minimize sprawl

They can also accelerate removal of undesired data flows by drafting the required diffs.

Example: exclude certain data types from logs.

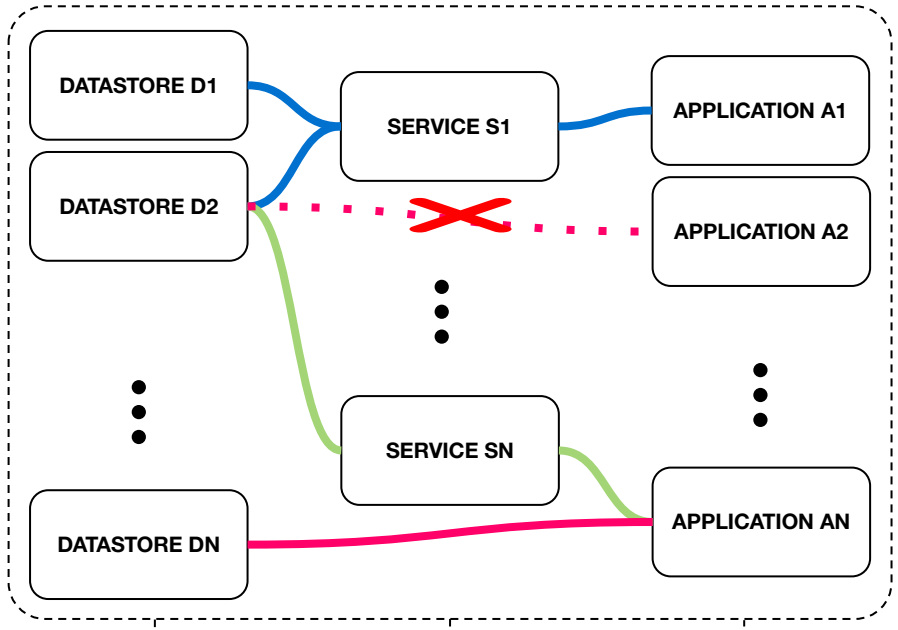


Application	Datastore	Data type	Removal diff
A1	D1	X	
A1	D2	X	
~~A2~~	~~D2~~	~~Y~~	<review link>
AN	D2	Z	
AN	DN	Y	

Task: Review and forbid certain data flows to minimize sprawl

Advantages compared to traditional static analysis include:

- tracing across multiple layers of the stack without custom parsers
- reduced maintenance overhead, rapid iteration
- semantic understanding of data and its flows, which can improve precision for privacy use cases.

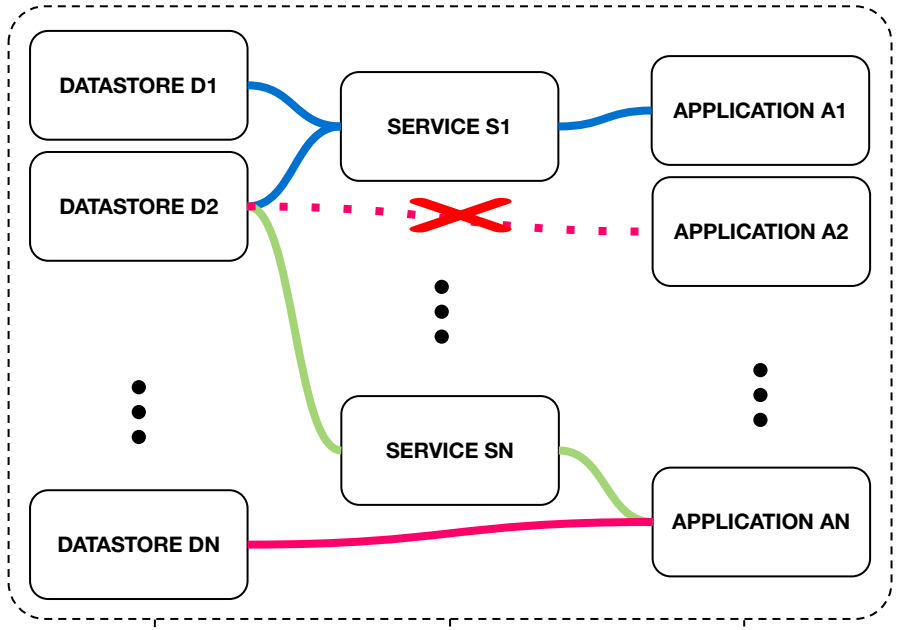


Application	Datastore	Data type	Removal diff
A1	D1	X	
A1	D2	X	
~~A2~~	~~D2~~	~~Y~~	<review link>
AN	D2	Z	
AN	DN	Y	

Task: Review and forbid certain data flows to minimize sprawl

Limitations include:

- Context length caps the amount of code that can be considered in 1 analysis, so judicious division into separate agent tasks is essential.
- False positives can occur. Specifying detailed analysis recipes and cross-referencing results against production logs helps.



Fleet of coding agents
(create diffs, identify owners using `git blame`)

Application	Datastore	Data type	Removal diff
A1	D1	X	
A1	D2	X	
~~A2~~	~~D2~~	~~Y~~	<review link>
AN	D2	Z	
AN	DN	Y	

People

Task: Facilitate organizational decisions around data deletion

Task: Facilitate organizational decisions around data deletion

Is table X still needed?
Who owns it?

Task: Facilitate organizational decisions around data deletion

How can we figure out the answers to ownership questions without taking too much of anyone's time?

Is table X still needed?
Who owns it?

Task: Facilitate organizational decisions around data deletion

How can we figure out the answers to ownership questions without taking too much of anyone's time?

[MM/DD/YYYY] "Team A is excited to announce pipeline P which will help accomplish Z"

Is table X still needed?
Who owns it?

Task: Facilitate organizational decisions around data deletion

How can we figure out the answers to ownership questions without taking too much of anyone's time?

[MM/DD/YYYY] "Team A is excited to announce pipeline P which will help accomplish Z"

[MM/DD/YYYY] "To accommodate event E, pipeline P has been migrated to produce table Y and table X will be deprecated"

Is table X still needed?
Who owns it?

Task: Facilitate organizational decisions around data deletion

How can we figure out the answers to ownership questions without taking too much of anyone's time?

[MM/DD/YYYY] "Team A is excited to announce pipeline P which will help accomplish Z"

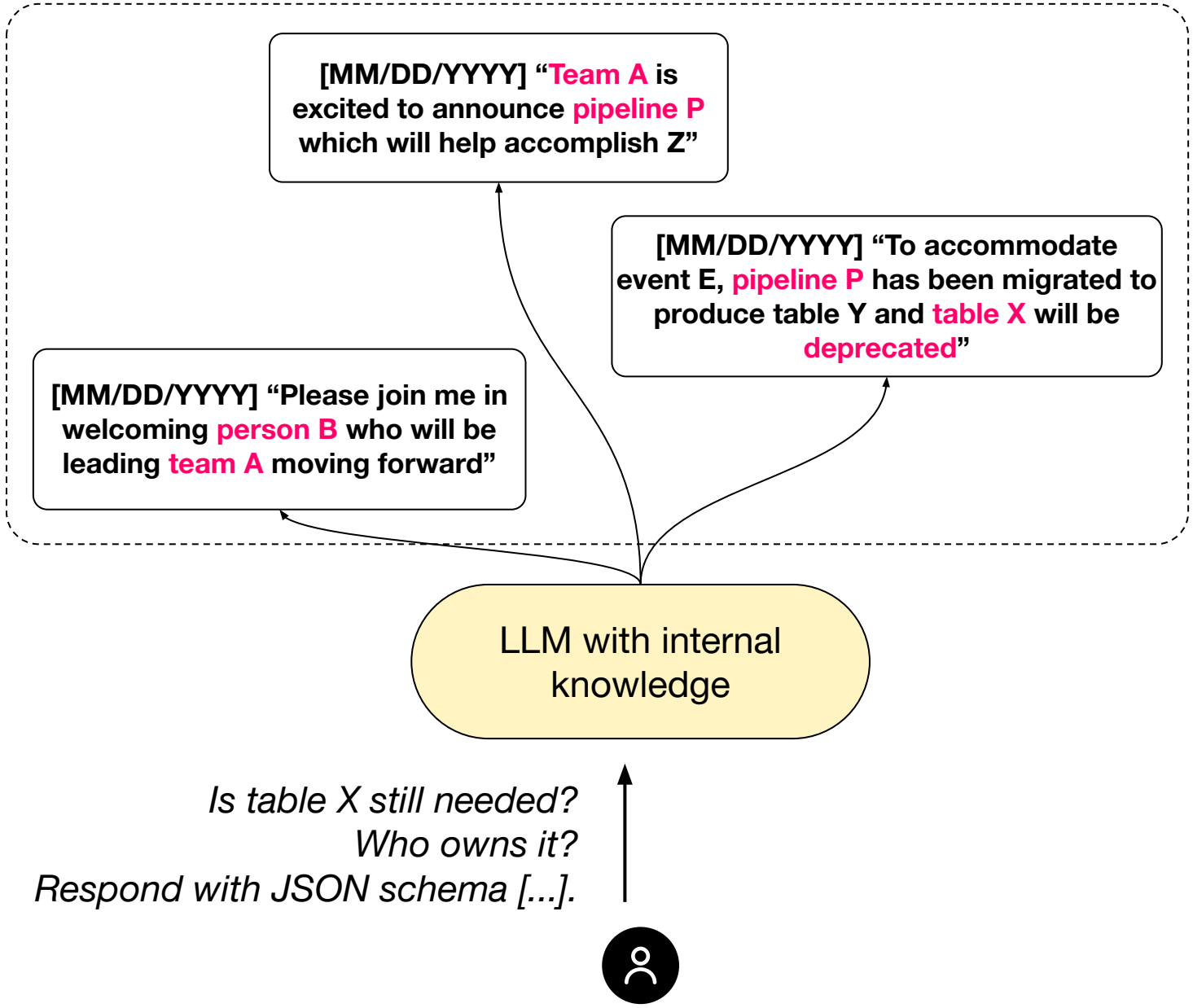
[MM/DD/YYYY] "To accommodate event E, pipeline P has been migrated to produce table Y and table X will be deprecated"

[MM/DD/YYYY] "Please join me in welcoming person B who will be leading team A moving forward"

Is table X still needed?
Who owns it?

Task: Facilitate organizational decisions around data deletion

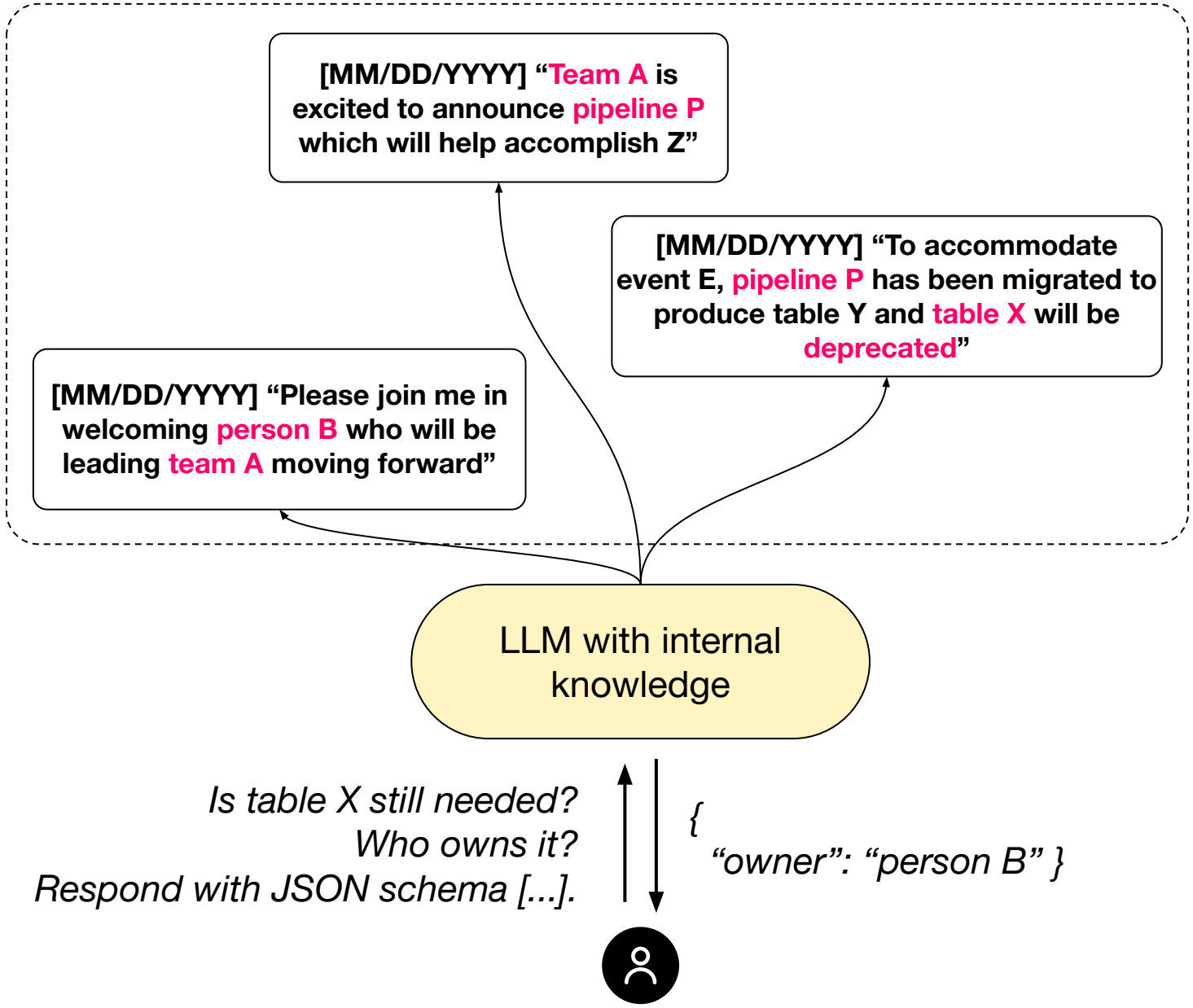
Reasoning over internal documents can help



Task: Facilitate organizational decisions around data deletion

Reasoning over internal documents can help

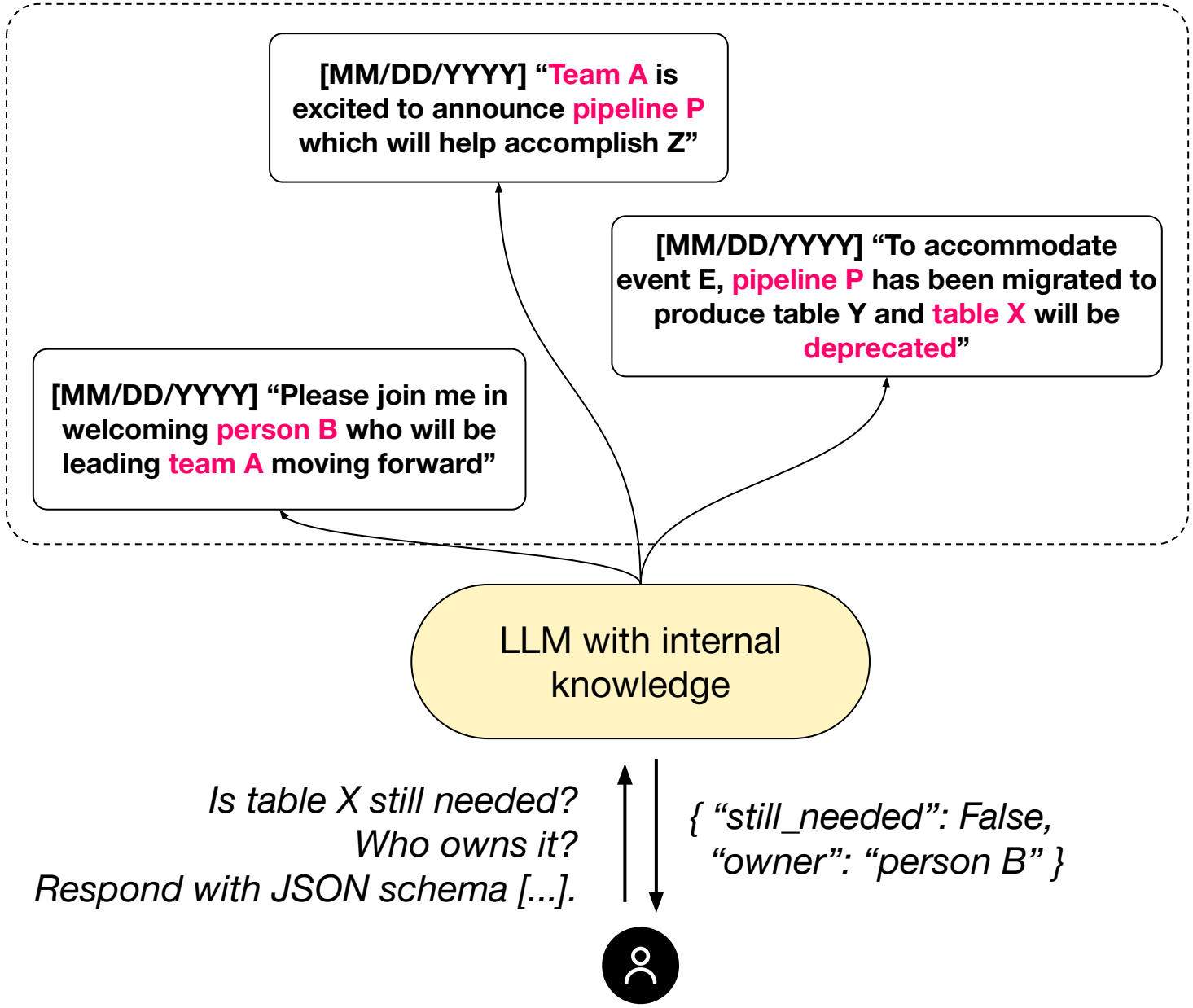
- automatically close gaps in ownership frameworks (e.g. departing employees)



Task: Facilitate organizational decisions around data deletion

Reasoning over internal documents can help

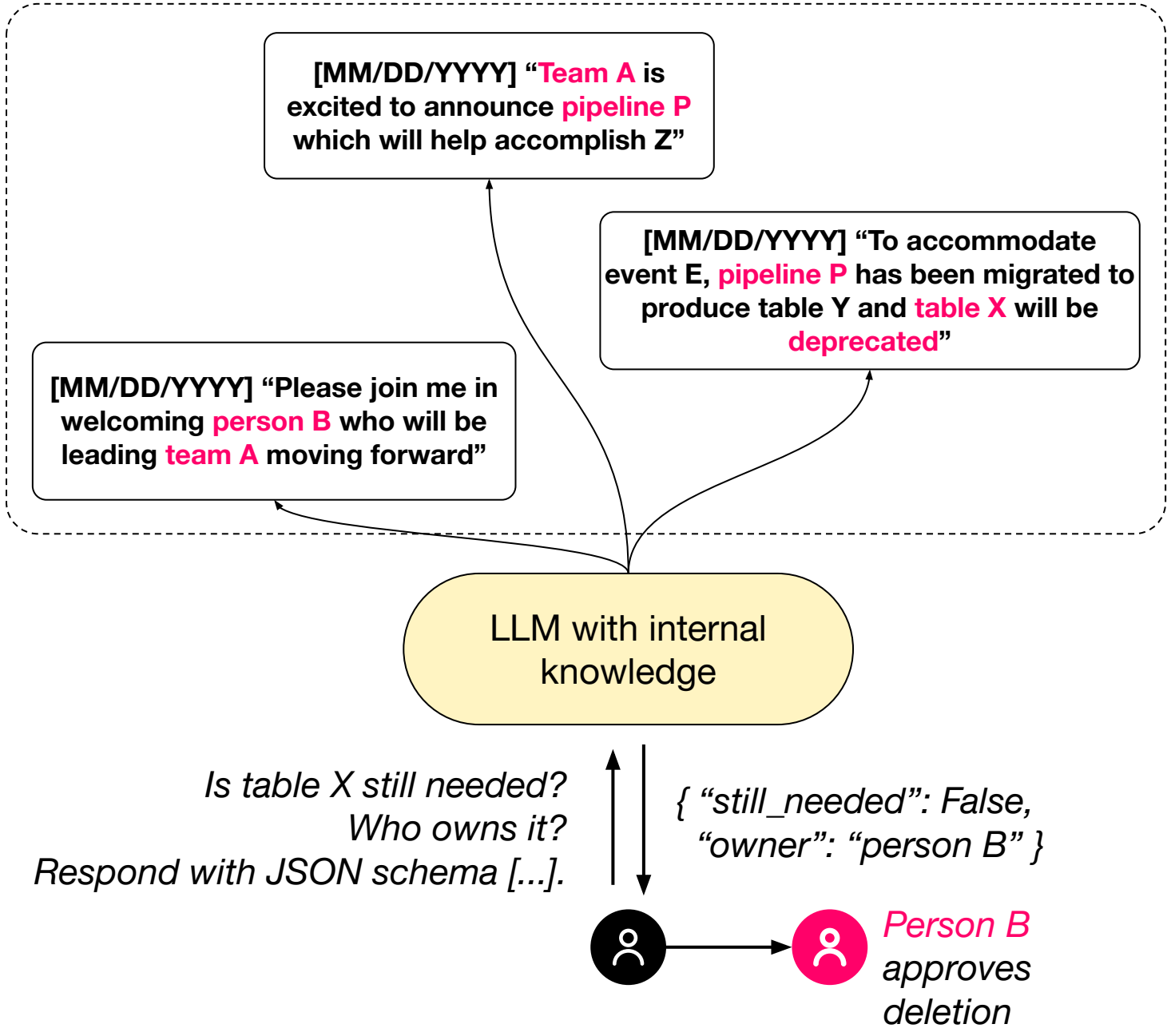
- automatically close gaps in ownership frameworks (e.g. departing employees)
- synthesize usage information to facilitate deletion decisions



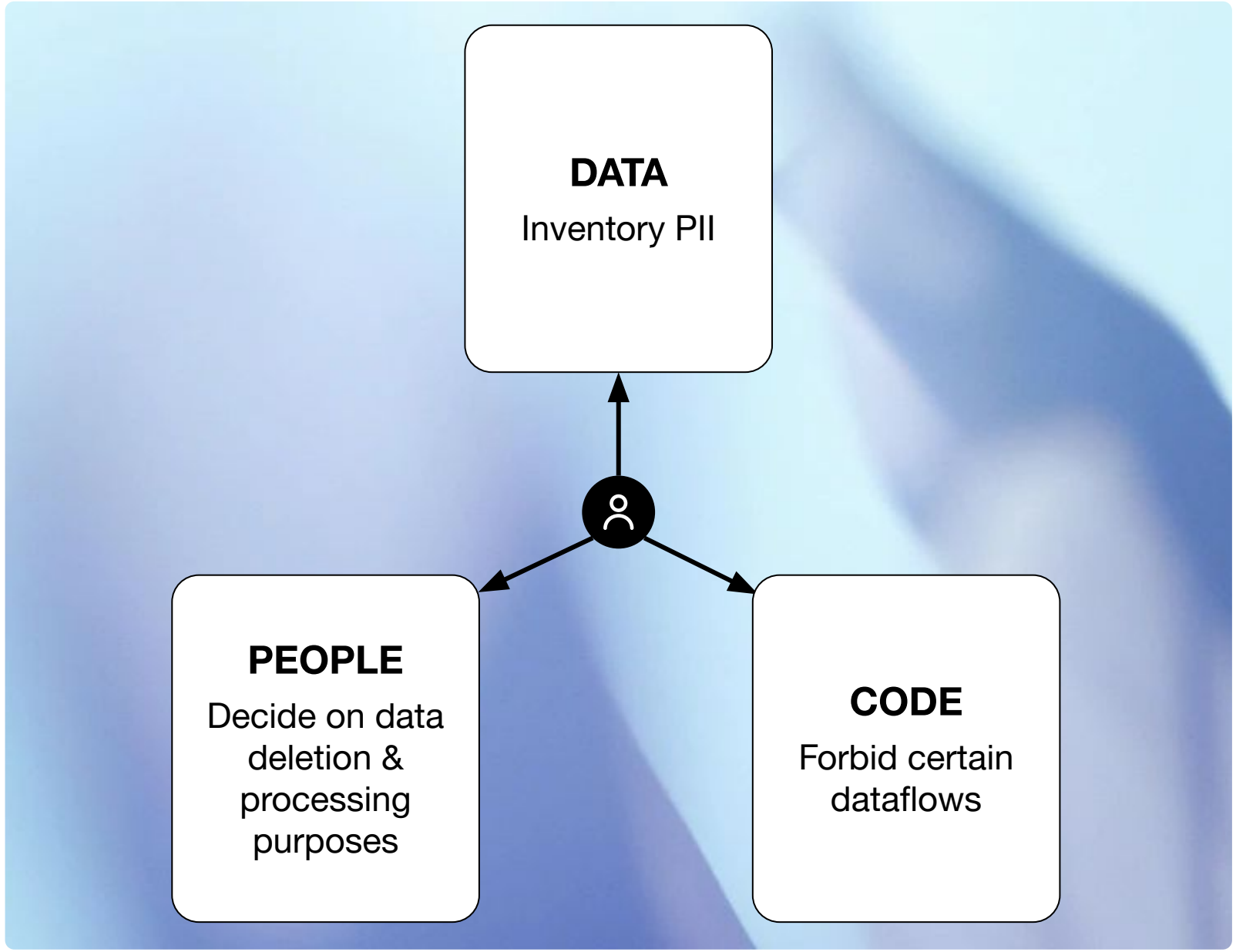
Task: Facilitate organizational decisions around data deletion

Reasoning over internal documents can help

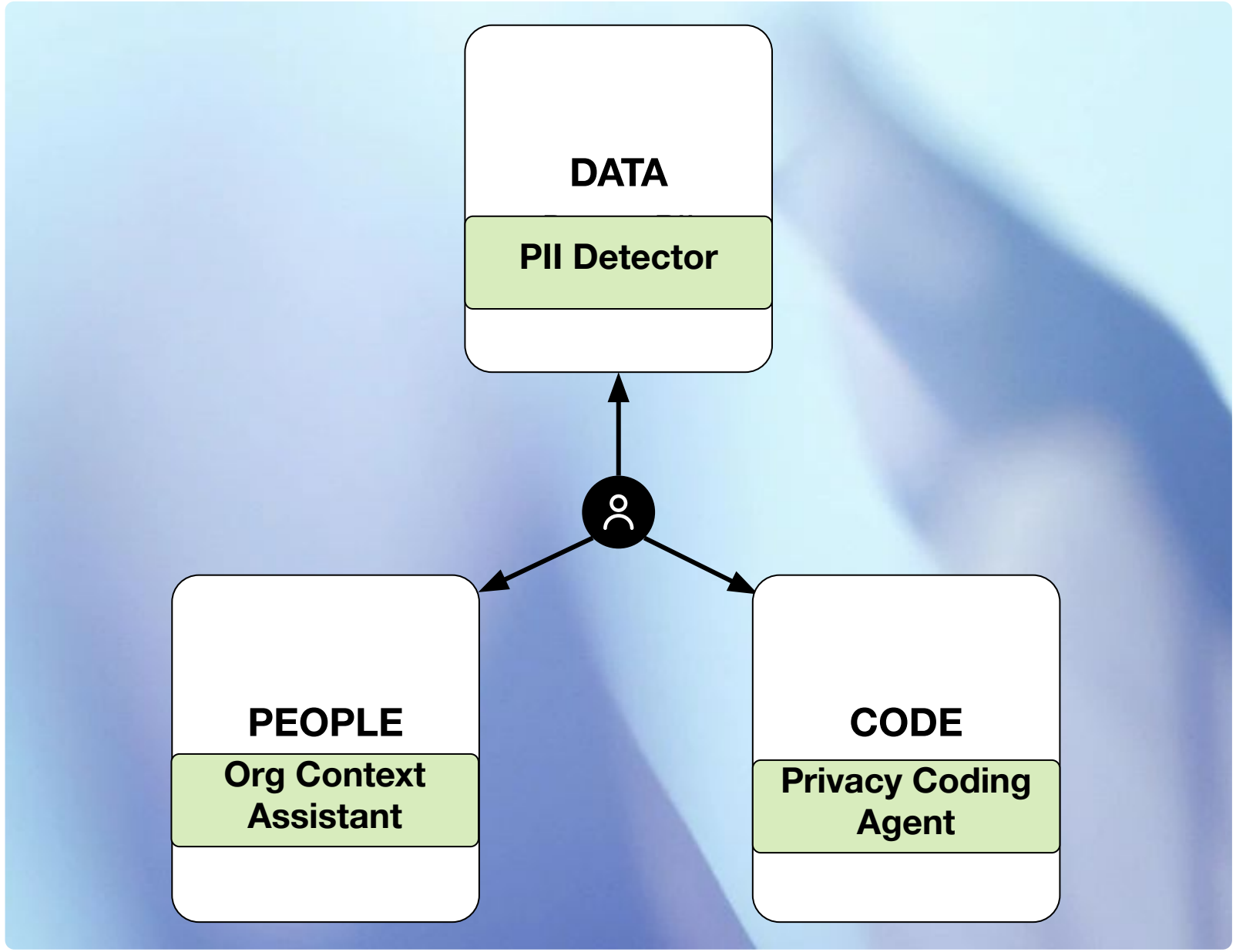
- automatically close gaps in ownership frameworks (e.g. departing employees)
- synthesize usage information to facilitate deletion decisions



Can LLMs help improve accuracy & efficiency of Data Minimization processes?



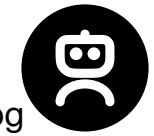
Yes, LLMs provide new architectural components for proactive Data Minimization



Yes, LLMs provide new architectural components for proactive Data Minimization

Putting it all together…

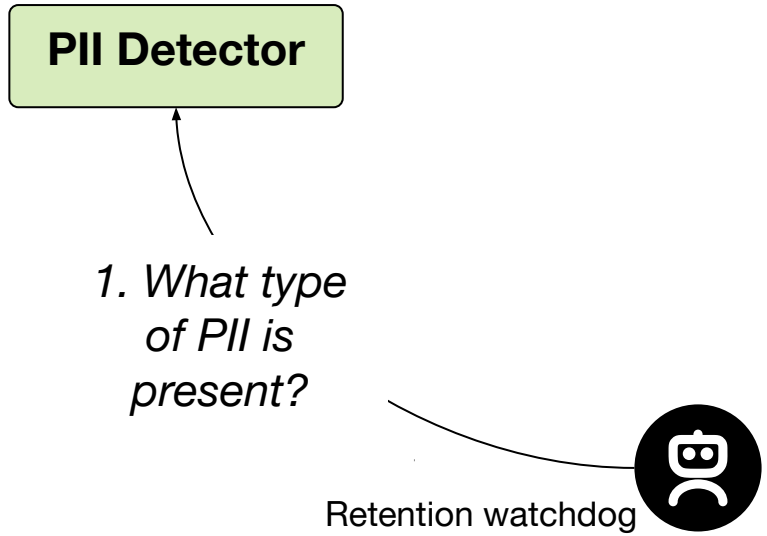
Automated systems can e.g. nudge an organization towards retention adjustments.

Retention watchdog 

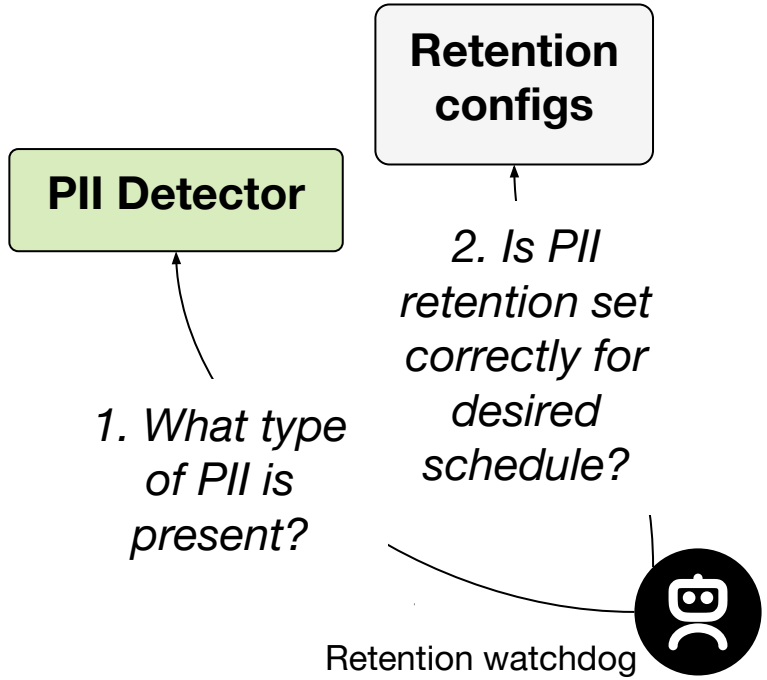
Yes, LLMs provide new architectural components for proactive Data Minimization

Putting it all together…

Automated systems can e.g. nudge an organization towards retention adjustments.



Yes, LLMs provide new architectural components for proactive Data Minimization



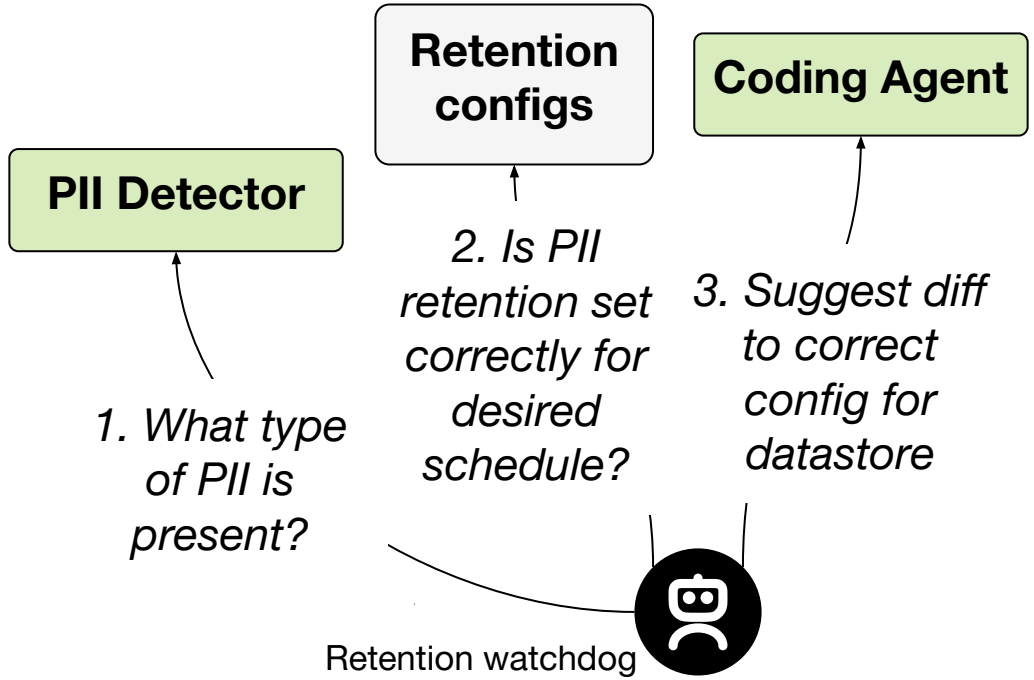
Putting it all together…

Automated systems can e.g. nudge an organization towards retention adjustments.

Yes, LLMs provide new architectural components for proactive Data Minimization

Putting it all together…

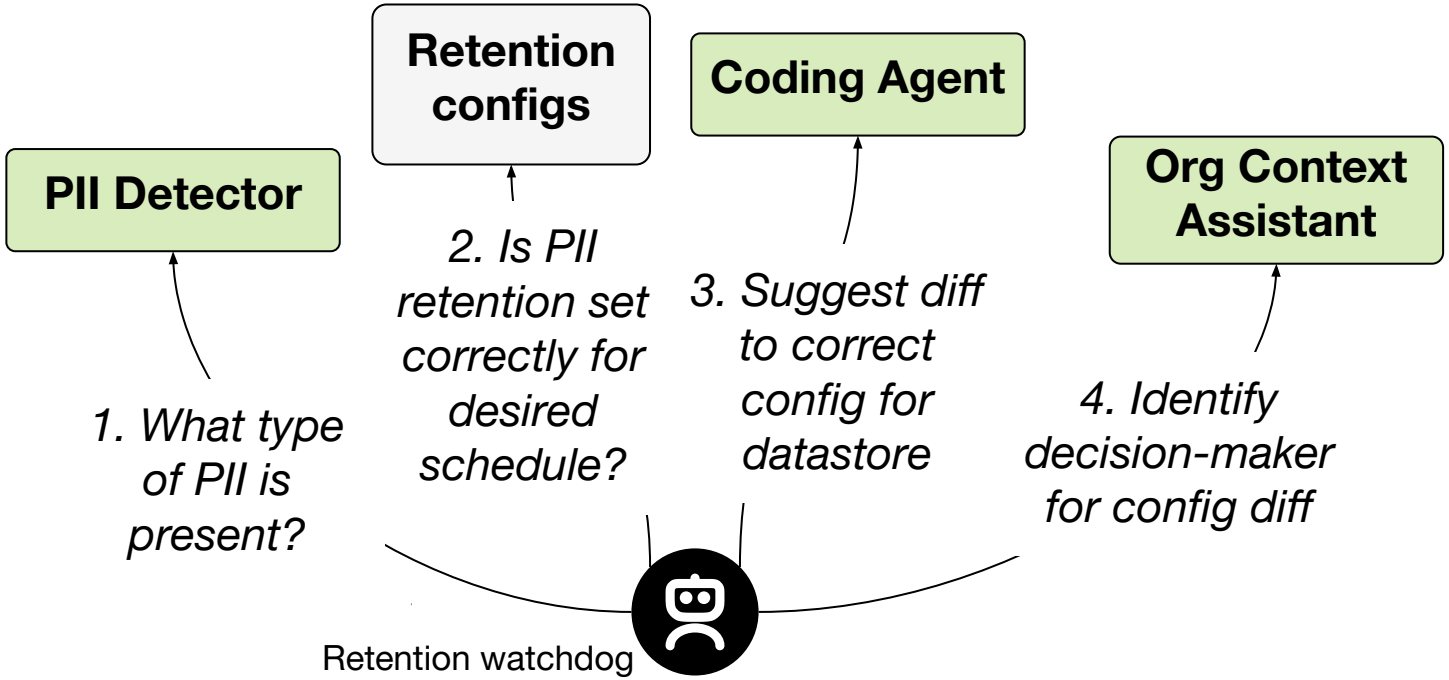
Automated systems can e.g. nudge an organization towards retention adjustments.



Yes, LLMs provide new architectural components for proactive Data Minimization

Putting it all together…

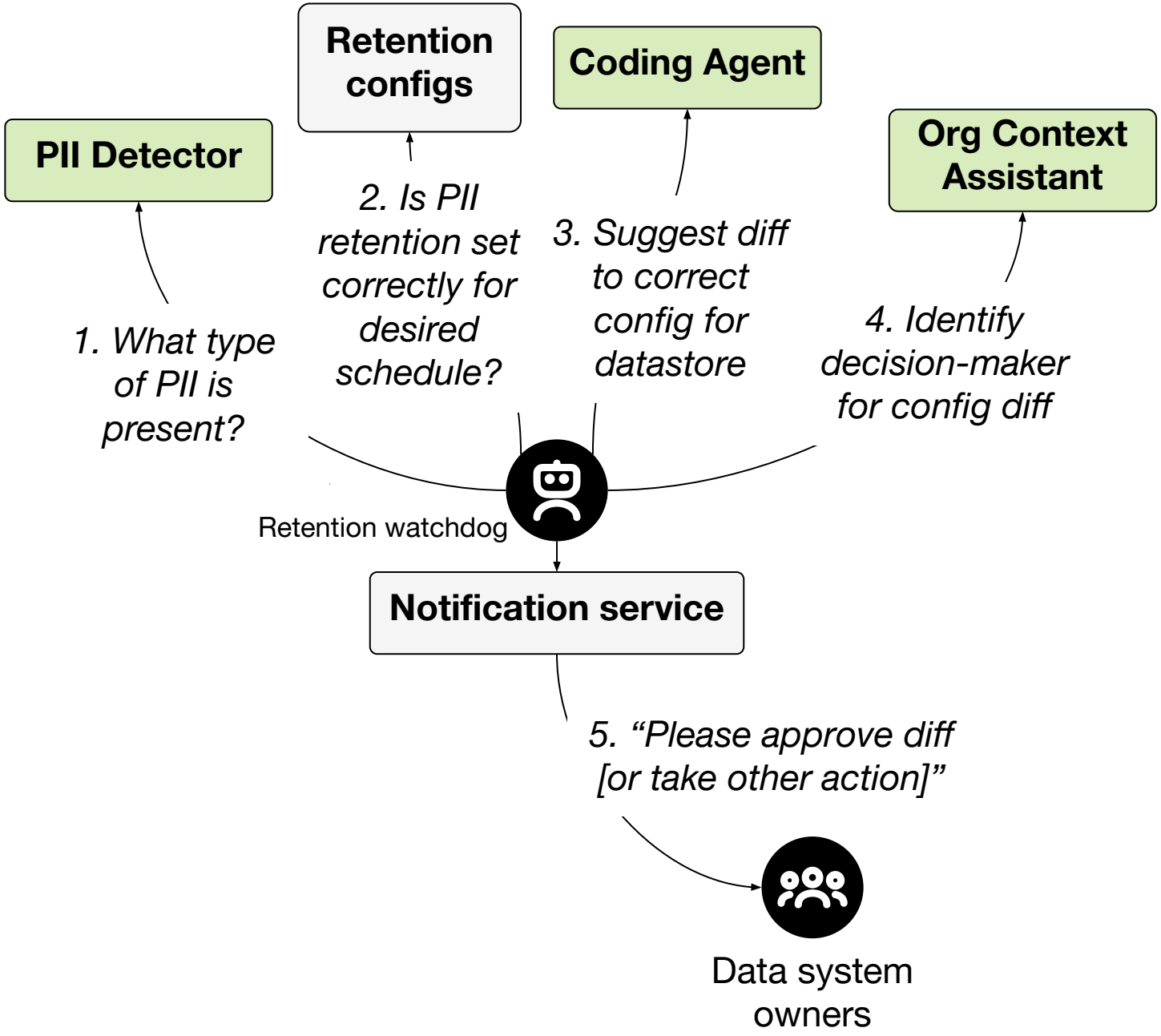
Automated systems can e.g. nudge an organization towards retention adjustments.



Yes, LLMs provide new architectural components for proactive Data Minimization

Putting it all together…

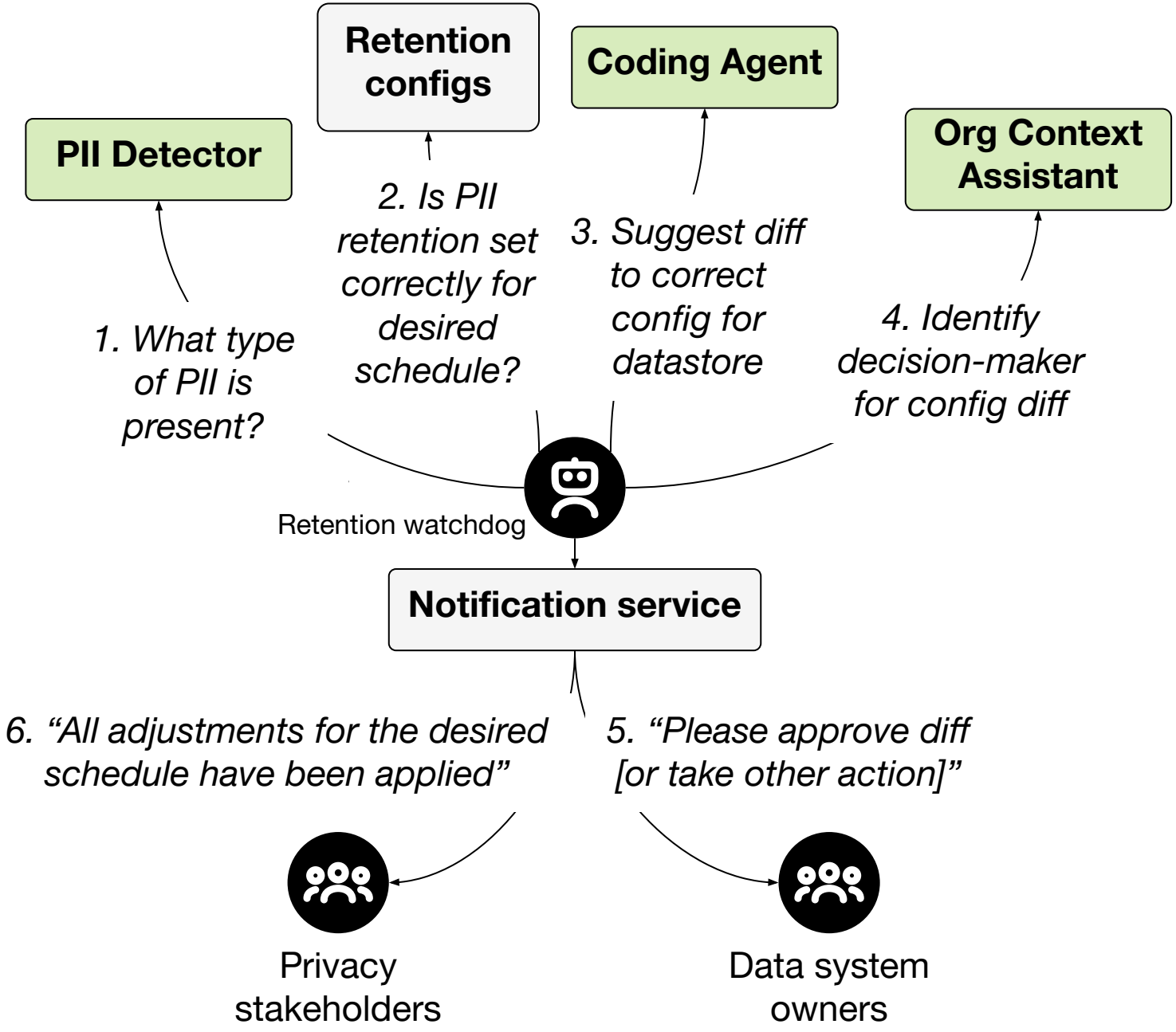
Automated systems can e.g. nudge an organization towards retention adjustments.



Yes, LLMs provide new architectural components for proactive Data Minimization

Putting it all together…

Automated systems can e.g. nudge an organization towards retention adjustments.

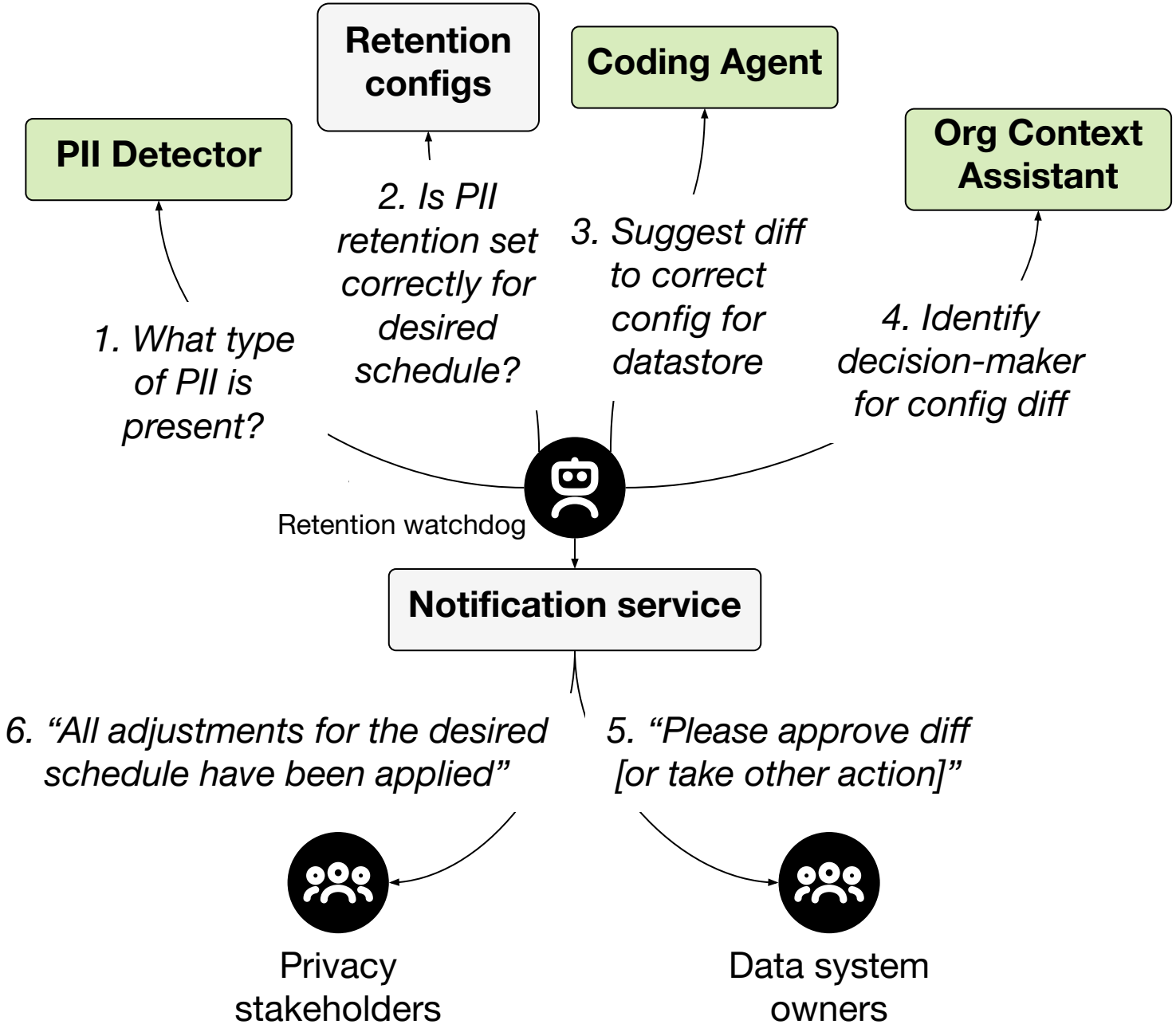


Yes, LLMs provide new architectural components for proactive Data Minimization

Putting it all together…

Automated systems can e.g. nudge an organization towards retention adjustments.

This can optimize the signal from nuanced human judgments while reducing effort and errors.



LLM-based tools can improve accuracy of Data Minimization and give focus time back to product developers.

In experiments, tools were good at: PII detection in context, converting code & docs into structured insights for governance.

We’re interested in collaborating with and learning from this community to advance privacy *for* and *with* AI.

Thank you