Images and Videos are Special.

Competitive edge

Rich in information

Unlock insights with ML

Complex and Challenging

Size

Volume

Unique processing requirements

Large amount of metadata

Sneak Peak into the Life of a Data Scientist

Training / Classifying Today

Ask around and find the right DB and table within DB to use for metadata

Get permission to access said DB / Table

Write complex queries on metadata to fetch the right URLs.

Now query metadata, let's say from a Postgres table

In [ ]:

```
def get_metadata(self, tags, probs,
                         lat=-1, long=-1,
                         range_dist=0,
                         comptype='and',
                         return_responses=False):

    if lat not in [-1, 999.9999999]:
        location_qstr = '''(latitude >= {} AND latitude <= {}) AND (longitude >= {} AND longitude <= {})'''.form

        qstr = ['''id IN (select id from (test_metadata INNER JOIN test_autotags a on test_metadata.id=a.metadat

        query = '''SELECT line_number, download_url, id, latitude, longitude, license_name FROM test_metadata WH
    else:
        qstr = ['''id IN (select id from (test_metadata INNER JOIN test_autotags a on test_metadata.id=a.metadat

        query = '''SELECT line_number, download_url, id, latitude, longitude, license_name FROM test_metadata WH

    start_t = time.time()
    self.db_cursor.execute(query)
    response = self.db_cursor.fetchall()
    endtime = time.time() - start_t

    if return_responses:
        return response

    out_dict = {'response_len':len(response),'response_time':endtime}

    return out_dict
```

Gets Worse

Allocate a large enough VM to contain expected dataset

Get permission to download images from relevant buckets

Download the images from the URLs to prepare the dataset

Take a few hours or sometimes days…

Finally, train / classify....

Write any code to pre-process if needed for training

In [ ]:

```
def rotate_image(self, image, angle):
  image_center = tuple(np.array(image.shape[1::-1]) / 2)
  rot_mat = cv2.getRotationMatrix2D(image_center, angle, 1.0)
  result = cv2.warpAffine(image, rot_mat, image.shape[1::-1], flags=cv2.INTER_LINEAR)
  return result
```

Now we can create a local dataset with pre-processed images to train on

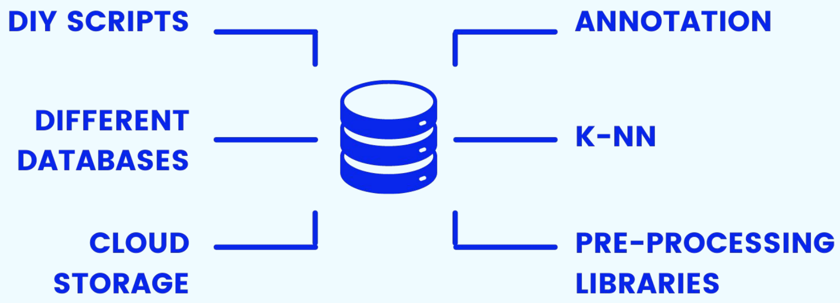
In [ ]:

```
def get_images(self, tags, probs, operations = [],
                    lat=-1, long=-1, range_dist=0, return_images=False, comptype='and'):
    metadata = self.get_metadata(tags, probs, lat, long, range_dist, comptype=comptype)

    img_array = []
    cols = ['line_number', 'download_url', 'id', 'latitude', 'longitude', 'license_name']

    for res in metadata:
        imgPath = "http://"+IMG_HOST+"/images/" + urlparse(res[cols.index('download_url')]).path
        try:
            imgdata = requests.get(imgPath)
            img = np.frombuffer(imgdata.content, dtype='uint8')

            # Warning -> cv2.imdecode returns None for some images
            # This seems to be fixed, but a possible source or error.
            decoded_img = cv2.imdecode(img, cv2.IMREAD_COLOR)

            # Check image is correct
            decoded_img = decoded_img if decoded_img is not None else img

            # Apply operations, if any
            for op in operations:
                if op["type"] == "resize":
                    height = op["height"]
                    width  = op["width"]
                    if height and width:
                        decoded_img = cv2.resize(decoded_img, dsize=(width, height))
                    else:
                        print("ERROR - Resize parameters not defined!")
                if op["type"] == "rotate":
                    angle = op["angle"]
                    if angle:
                        decoded_img = self.rotate_image(decoded_img, angle)
                    else:
                        print("ERROR - Resize parameters not defined!")
        except:
            print("Error processing image:", imgPath)
            decoded_img = None

        img_array.append(decoded_img)

    if return_images:
        out_dict["decoded_images"] = decoded_images

    return out_dict
```

ApertureData

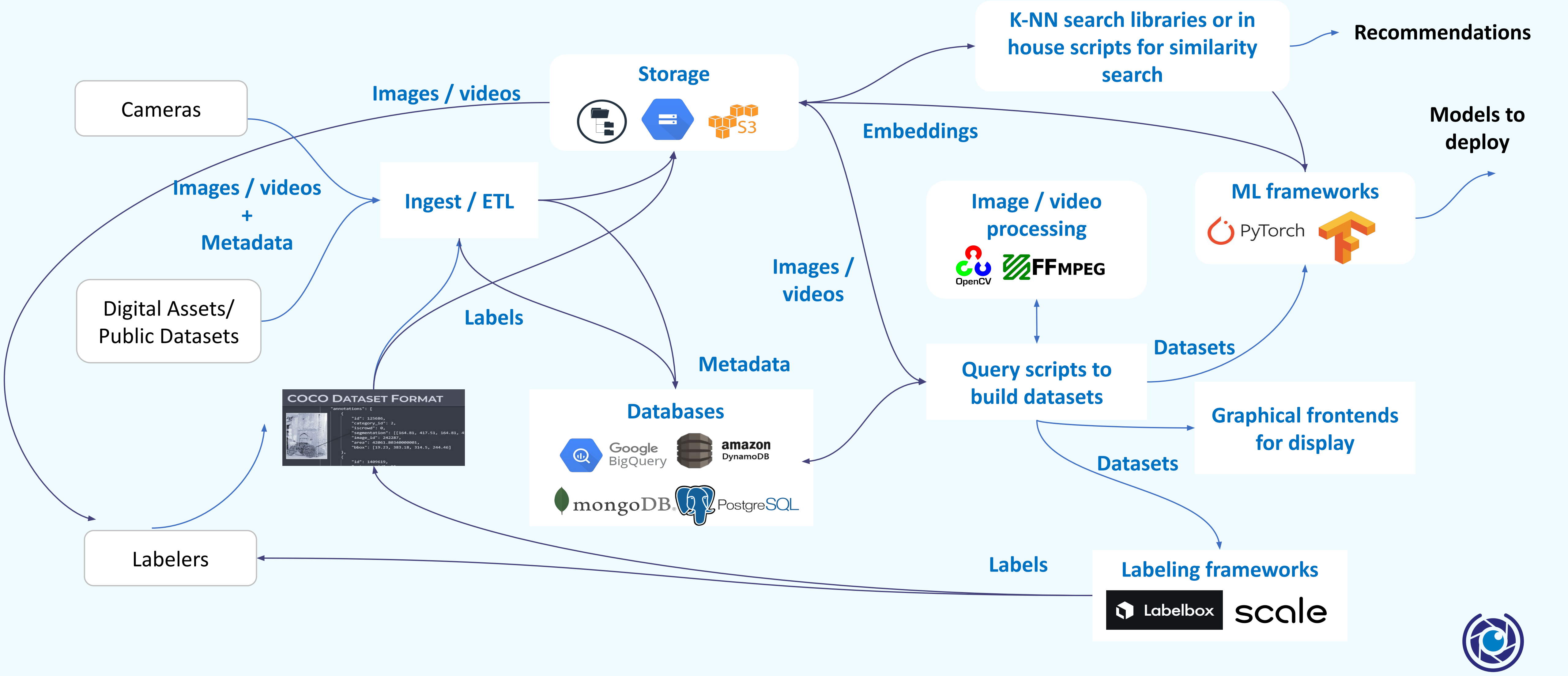
Just Because Your Data Is Unstructured Doesn’t Mean it Should be Onerous

Data management platforms are not designed for images & video-based ML/Analytics



- Multiple technologies to solve one problem add to cost & complexity
- Challenging data lifecycle management
- Long data engineering delays when tuning ML
- Reluctance in refreshing training datasets or updating schema
- Lack of reuse

DIY Solution



Data Science / ML Teams Want

TECH

Unified single technology

Holistic and purpose built

OUTCOME

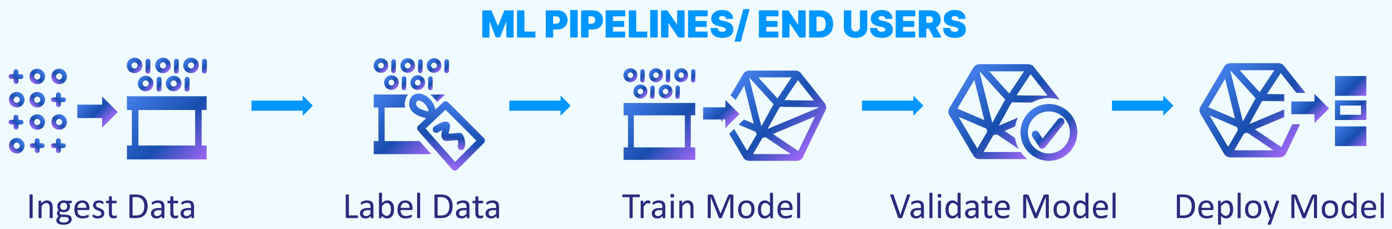
Enhanced Productivity

Simplified data engineering

Faster ML iteration

System that evolves as rapidly as ML and scales rapidly with data growth

The Missing Piece: Purpose-built Database for Visual Analytics

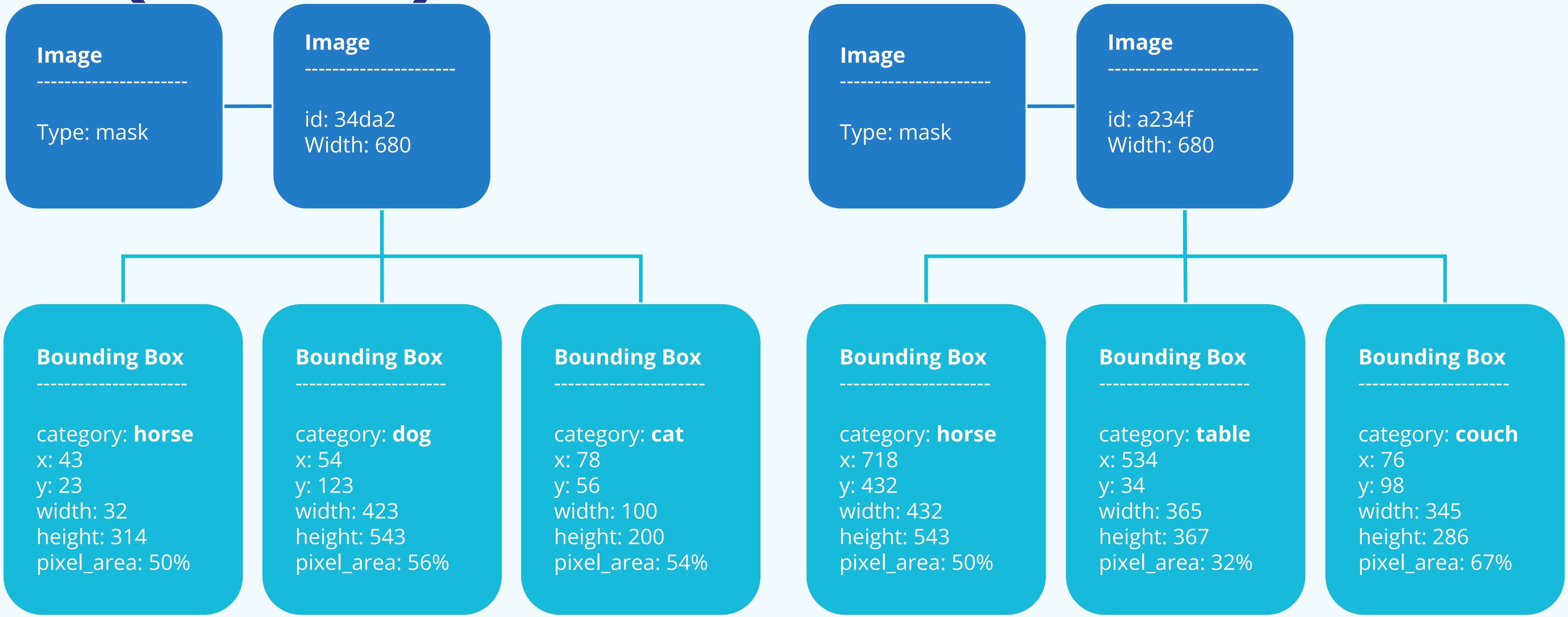


ApertureDB

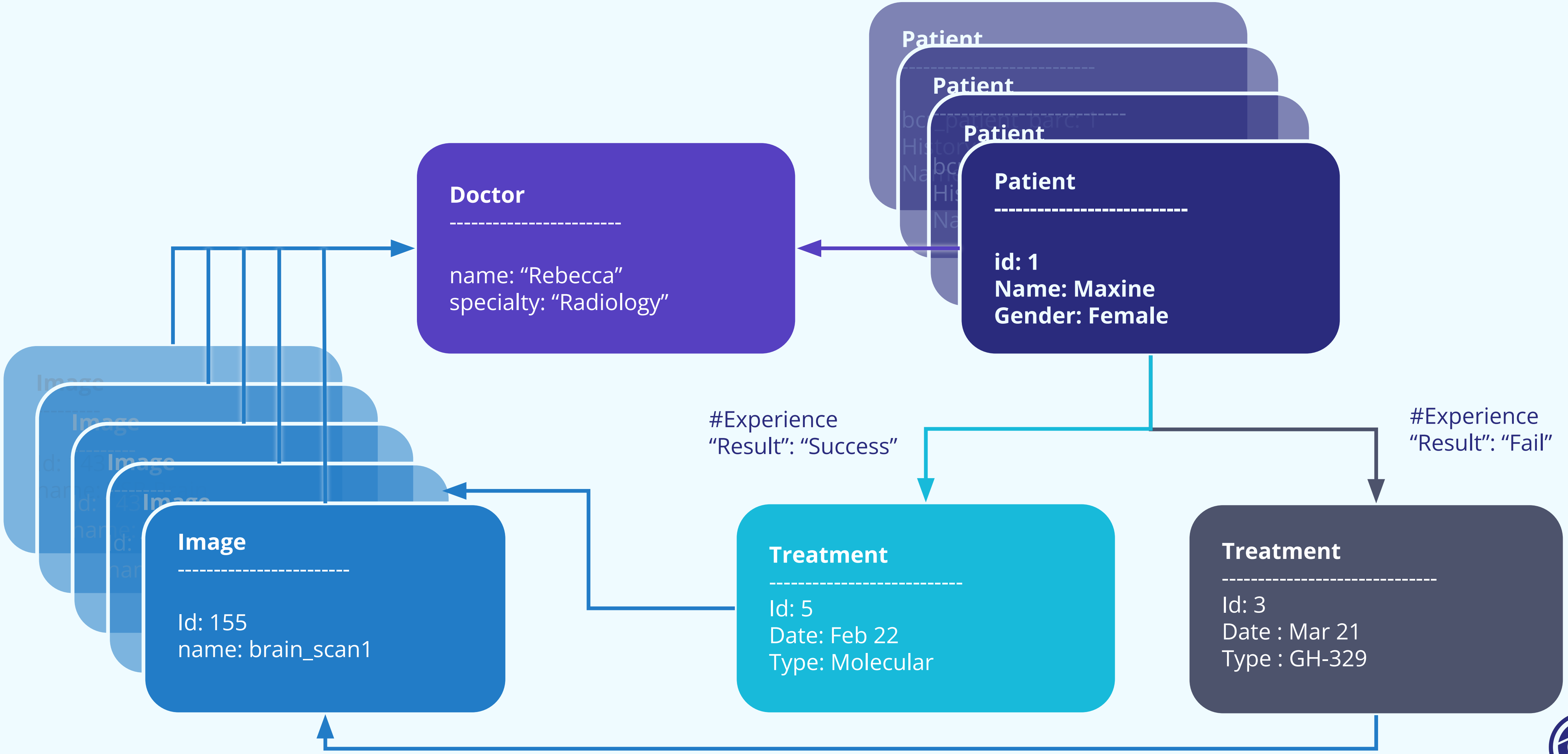
VISUAL DATA

Native support for images, videos, and pre-processing operations

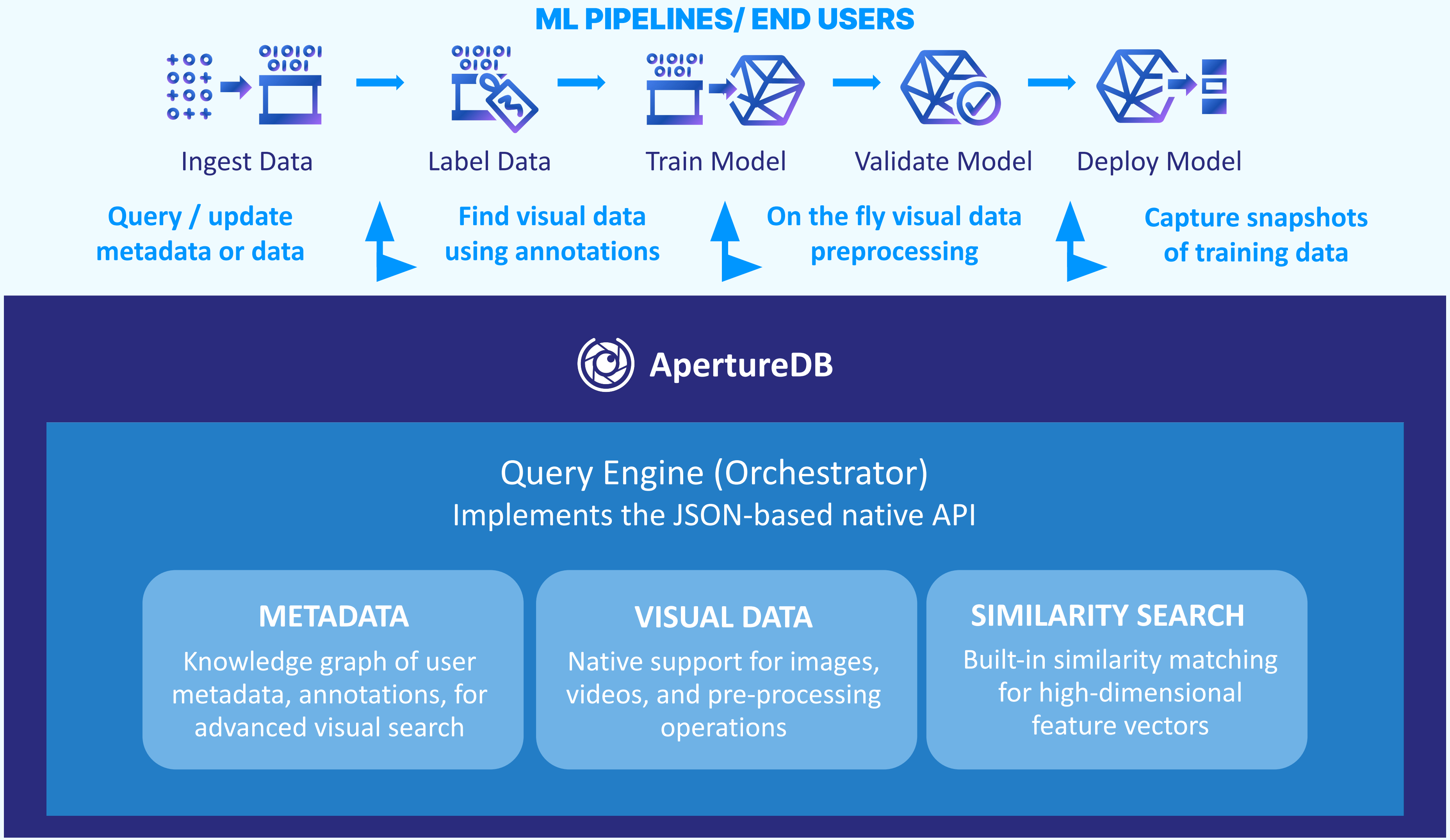
Example Metadata Schema (COCO)



Another Example Schema



Purpose-built Database for Visual Analytics



High Performance: Design Choices Matter

vs.

APACHE HTTP SERVER

PostgreSQL

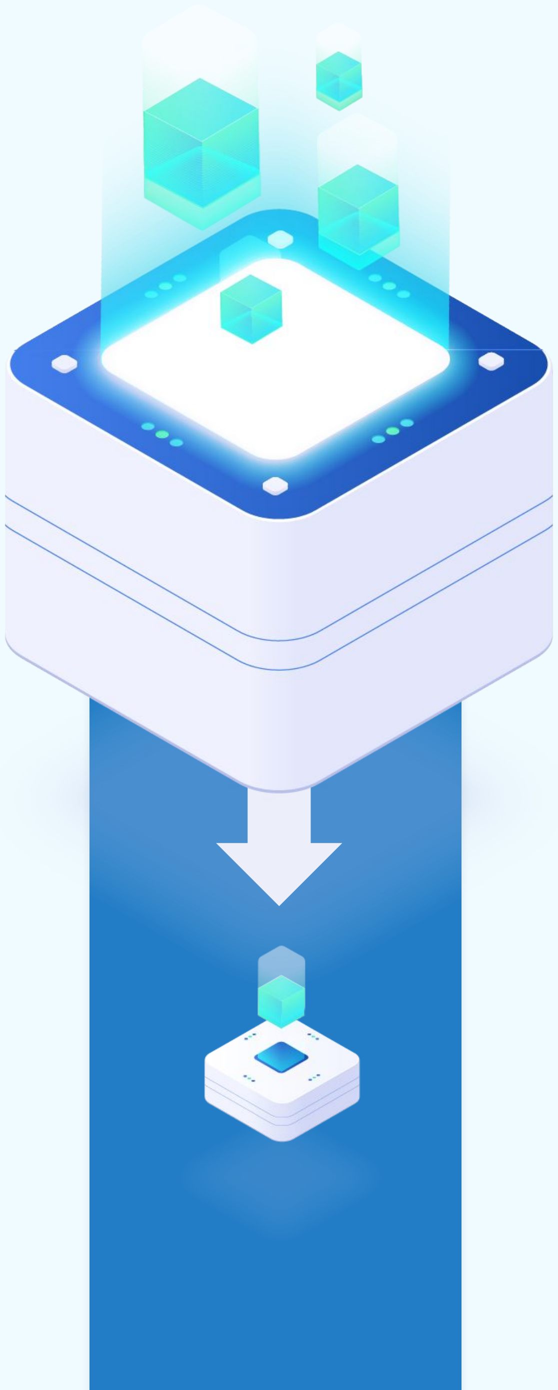
OpenCV

- **METADATA-BASED** visual search queries to find right set of images
- **YFCC100M (~100 MILLION IMAGES) DATASET**
- Up to **35X FASTER** and **15X ON AVERAGE** [paper in VLDB 2021]

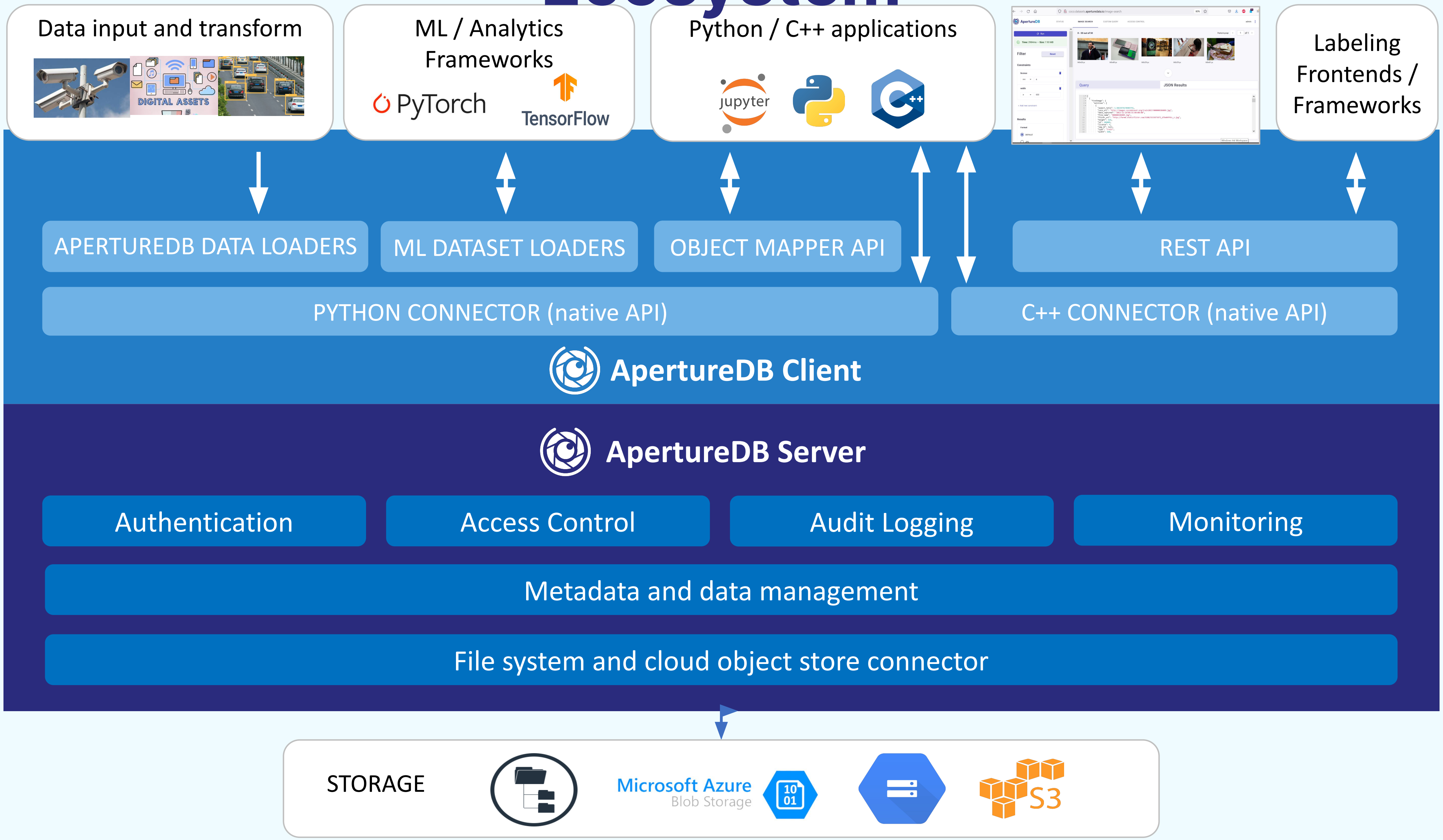
Resource Efficient: Preprocessing Near Data

63% reduction

in data transferred over the network using pre-processing within our API



Seamlessly Integrate Across Data Science Ecosystem



Beyond Performance

TIME TO SETUP INFRASTRUCTURE

E.g. **6-9 months faster**

At least **3-4 fewer modules** in ML infrastructure

SCALE

1.3+ billion metadata entities with as many relationships

Over **300+ million** images

Go Beyond Helping Data Scientists

Data Engineers

Simpler data lifecycle management

Data scientists / ML Engineers

Easy data(set) creation, search and access for visualizing, training, model iteration

Data Science Managers

Team collaboration, faster results

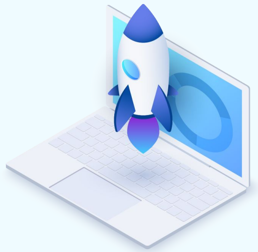
Infrastructure teams

Reduced maintenance & complexity

ApertureDB

FASTER

Model tuning and deployment
Keyword / label searches
Similarity searches



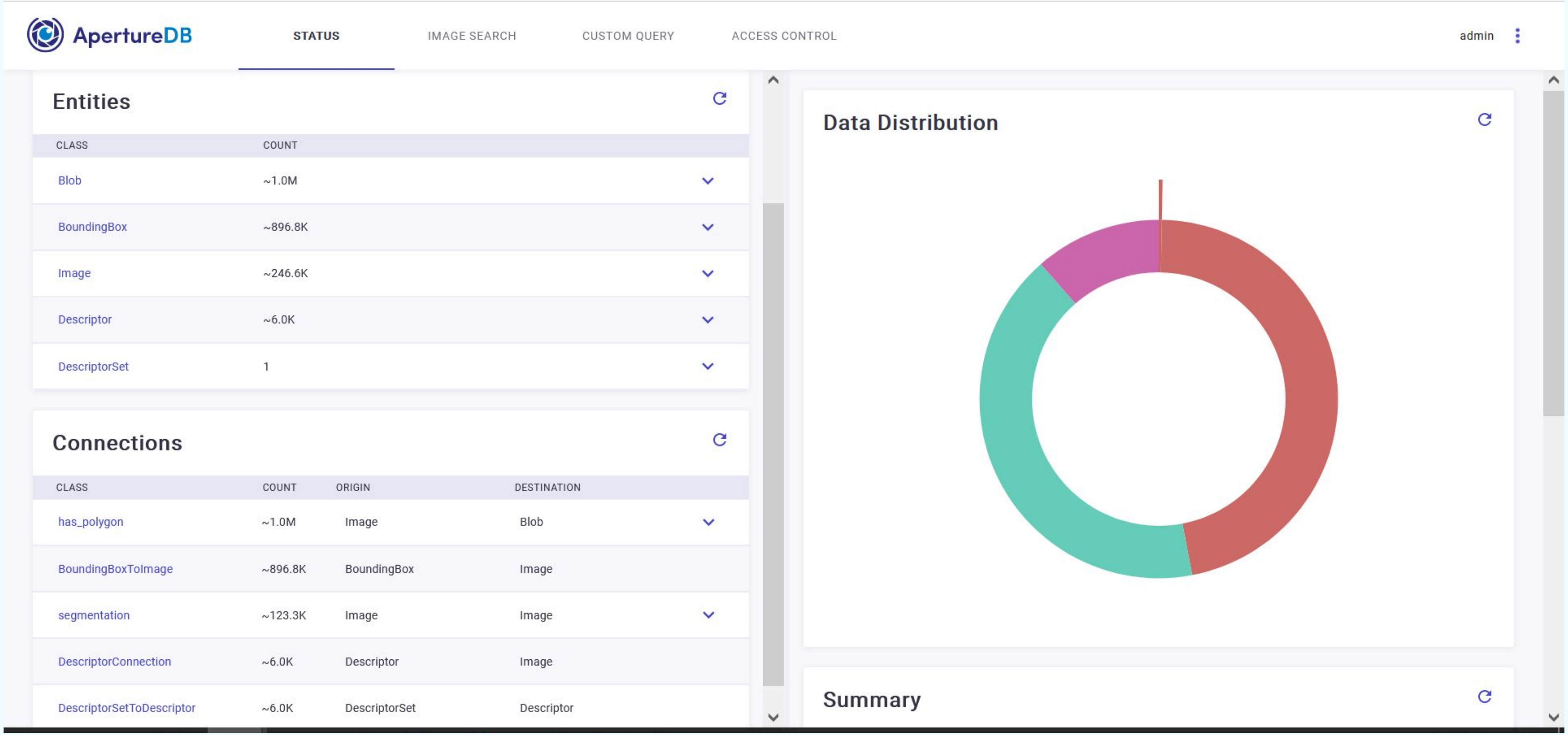
SRE Teams

Security, privacy, monitoring, reliability, availability

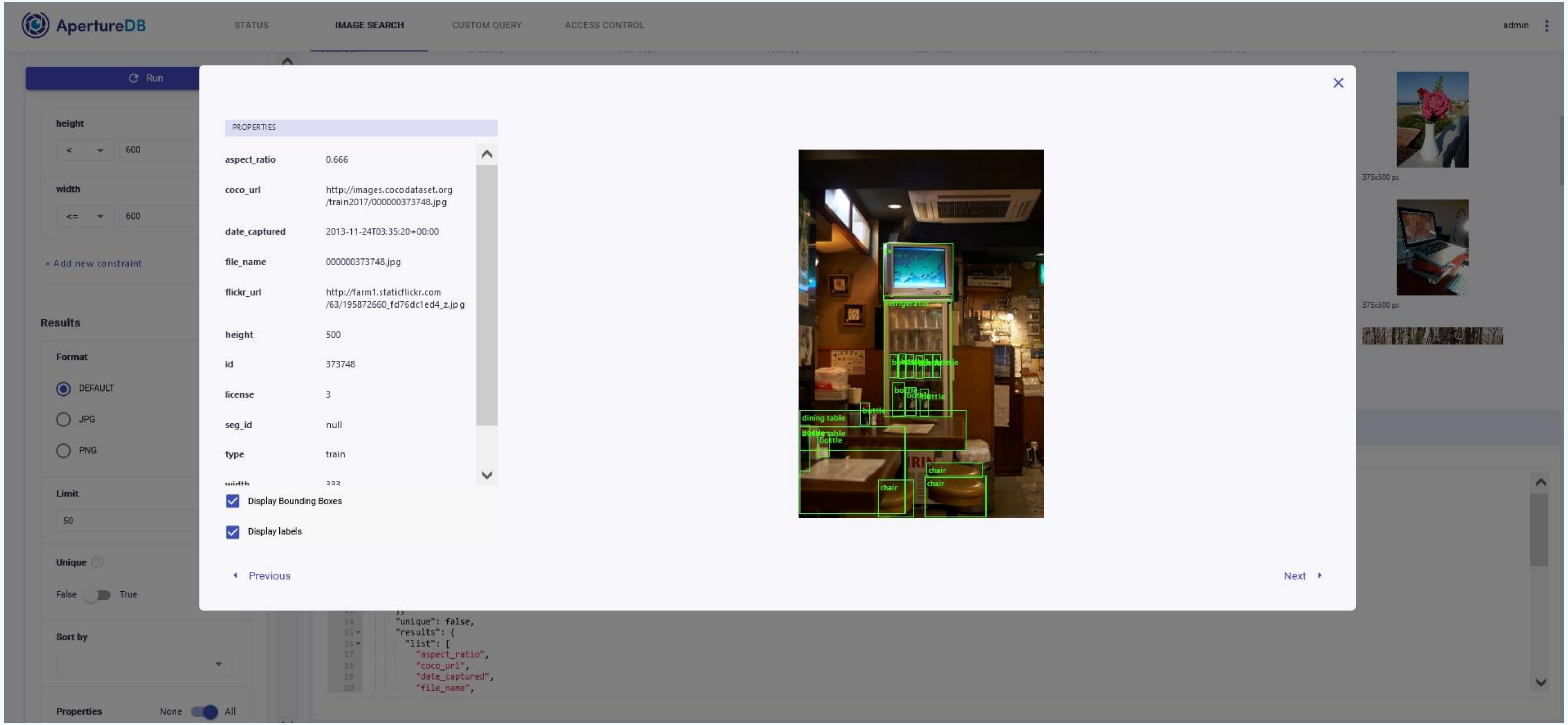
Classification or object detection	Visual inspection or activity recognition	Similarity based visual recommendations

Simplify various data steps in the life of a Data Scientist

00 – Know the metadata



01 – What's in your dataset?



02 – Filter, train / classify

Create a PyTorch Dataset

```
In [1]:
import time
from aperturedb import Connector, Status
from aperturedb import PyTorchDataset, Images
from aperturedb import ProgressBar
import AlexNetClassifier as alexnet

db = Connector.Connector("aperturedb.local", user="admin", password="admin")

const = Images.Constraints()
const.greaterequal("license", 0)

dataset = PyTorchDataset.ApertureDBDatasetConstraints(db, constraints=const)

total = len(dataset)
print("Total images in the dataset:", total)
```

```
Total images in the dataset: 123287
```

```
In [2]:
img, label = dataset[70]

from aperturedb import NotebookHelpers as nh
from PIL import Image
from IPython.display import display as ds

ds(Image.fromarray(img))
```

Classify Image using AlexNet

```
In [3]:
classifier  = alexnet.AlexNetClassifier()

label, conf = classifier.classify(img)

print(label, conf)
```

```
ballplayer, baseball player 84.28474426269531
```

03 – Debug your dataset

```
In [2]:  from aperturedb import Connector, Images

         db = Connector.Connector("aperturedb.local", user="admin", password

         imgs  = Images.Images(db)
         const = Images.Constraints()
         const.greater("width",  600)
         const.greater("height", 600)

         imgs.search(constraints=const)
         print("Total results:", imgs.total_results())

         imgs.display(limit=3, show_bboxes=True)

         Total results: 5112
```

Display Segmentation

```
In [12]: from aperturedb import Connector, Images

         db = Connector.Connector("aperturedb.local", user="admin", password=

         imgs  = Images.Images(db)
         const = Images.Constraints()

         const.equal("license", 2)

         imgs.search(constraints=const)
         print("Total results:", imgs.total_results())

         Total results: 17027

In [13]: imgs.display(show_segmentation=True, limit=10)
```

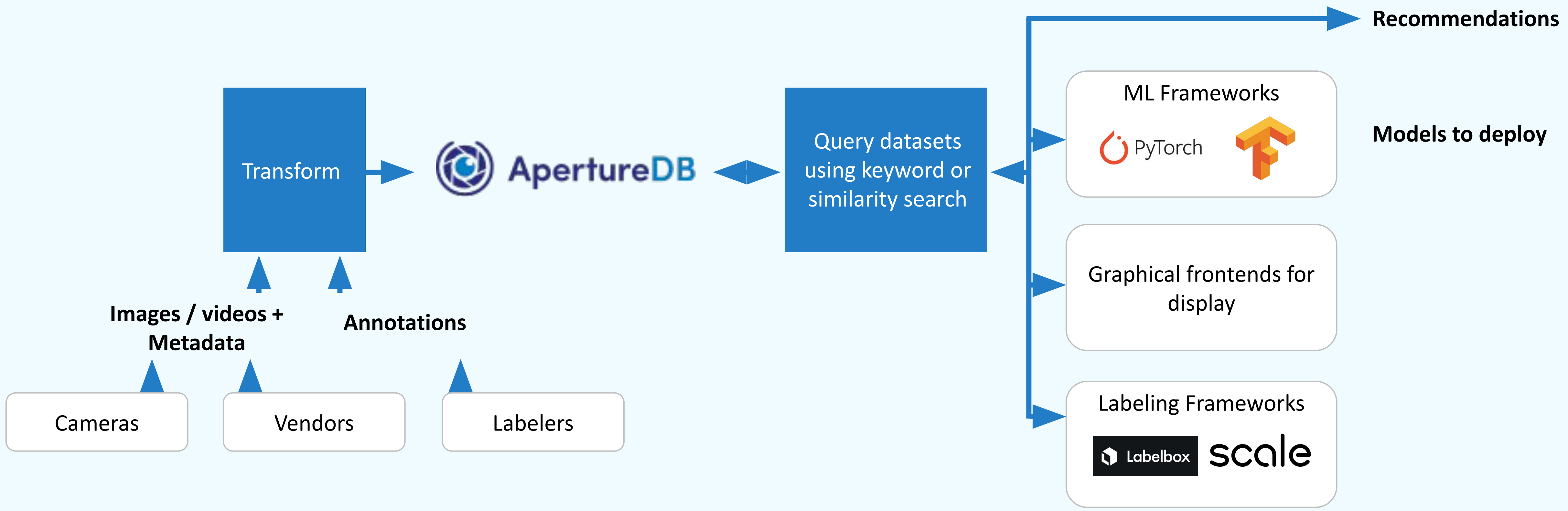
04 – Find Without Keywords

Search for similar images

In [1]:
```
from aperturedb import Connector, Images

db = Connector.Connector("aperturedb.local", user="admin", password="admin")

imgs  = Images.Images(db)
const = Images.Constraints()

const.equal("yfcc_id", 5552231605) # Filter 1 out of 120K images

imgs.search(constraints=const)
imgs.display()
```

In [2]:
```
similar = imgs.get_similar_images("coco_descriptors", 15)
similar.display(limit=40)
```

Simpler Data Pipeline Shifts Focus to ML / Data Science



Write to us if you want to develop, deploy, or have cool ideas for ApertureDB!

team@aperturedata.io