AWS re:Invent

NOV. 29 – DEC. 3, 2021 | LAS VEGAS, NV

DAT315

Quickly building for the future with Amazon RDS

Wanda He (she/her)
Sr. Database Specialist Solutions Architect
AWS

Yahav Biran (he/him)
Principal Solutions Architect
AWS

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

- Why open-source software database?
- Considerations for database engines and frameworks
- How to scale it quickly?
  - Connection management
  - Partitioning
  - Sharding/purpose-built database

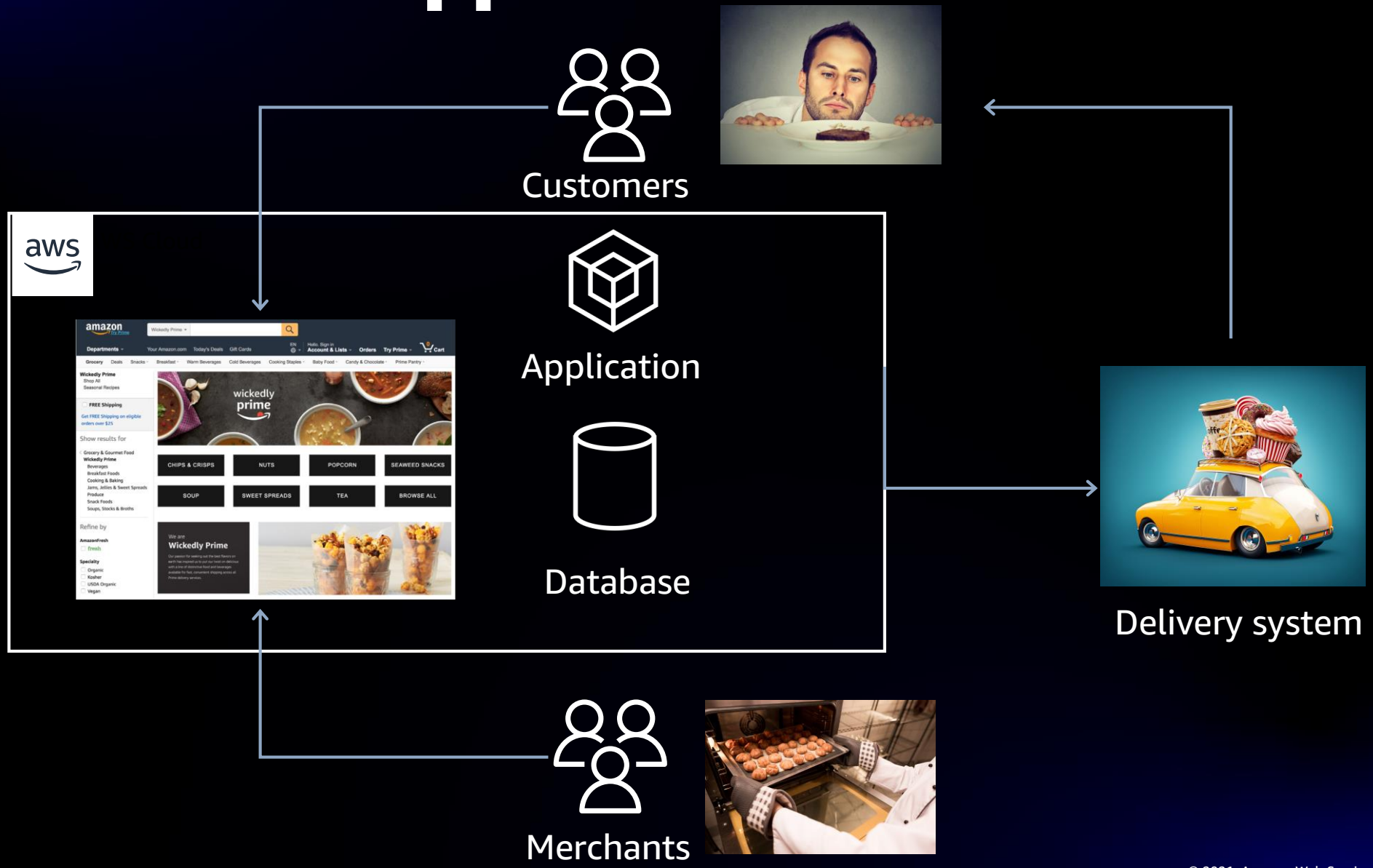
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why open-source software (OSS) database?

- We are developers
- We want to start simple
- We are familiar with OSS
- We will use a framework (e.g., Django, Rails) that is compatible with the database
- Let’s build an app!

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Let’s build an app



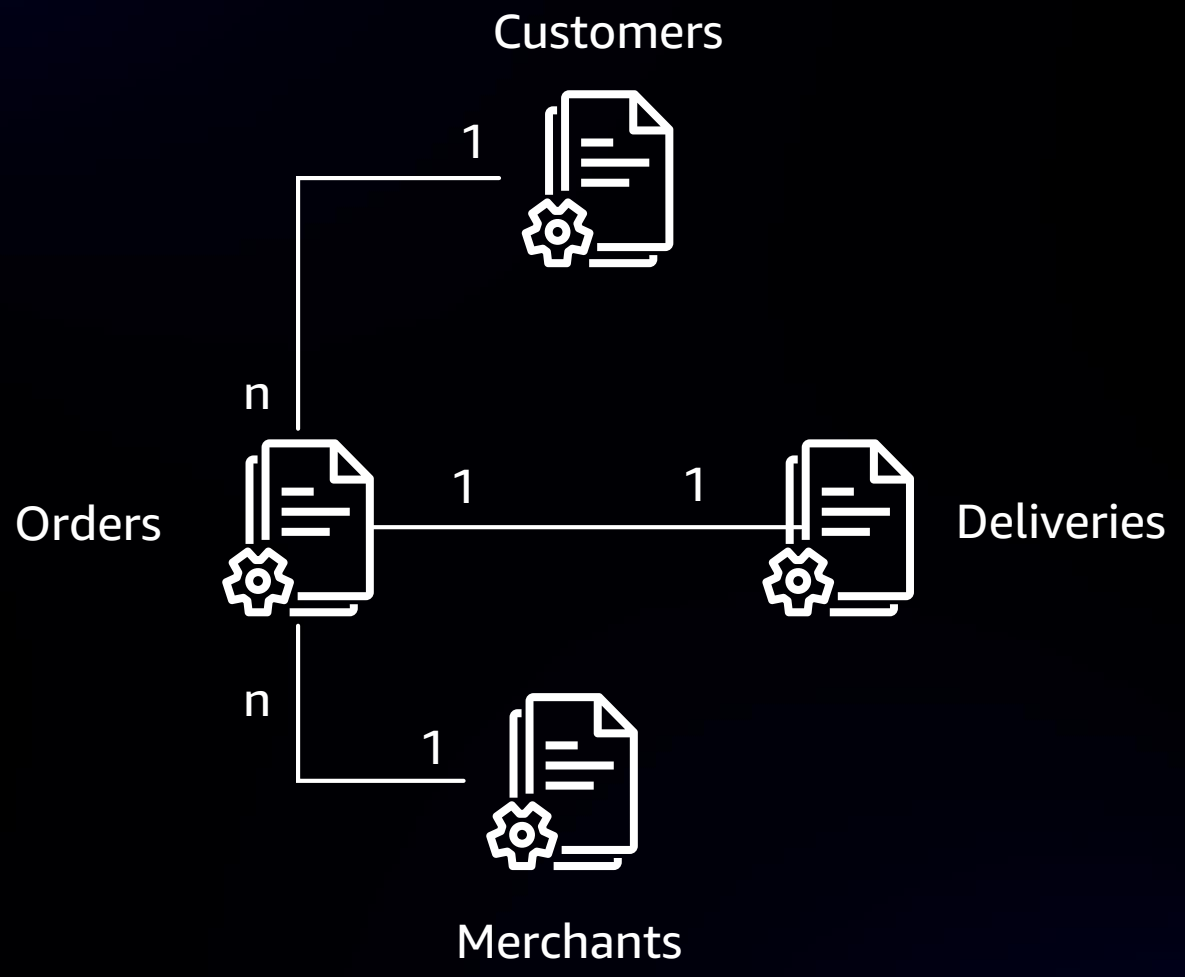
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Example app



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Example app



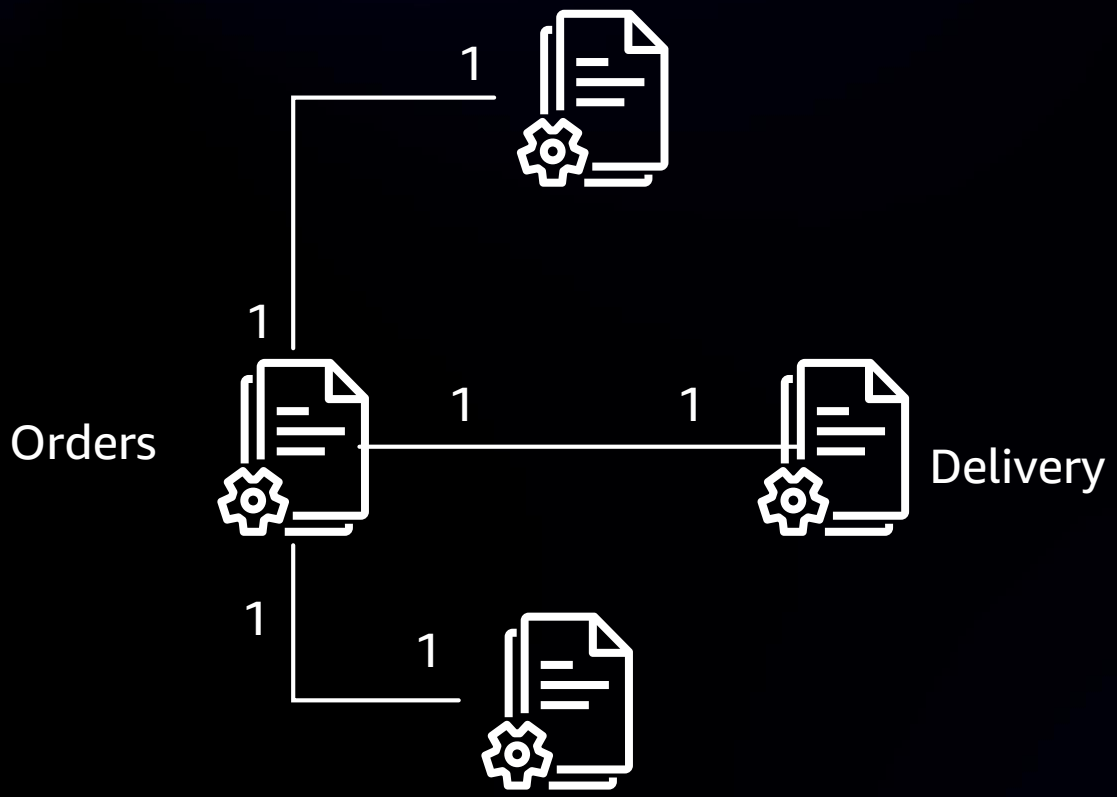
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why use an object-relational mapping (ORM)?

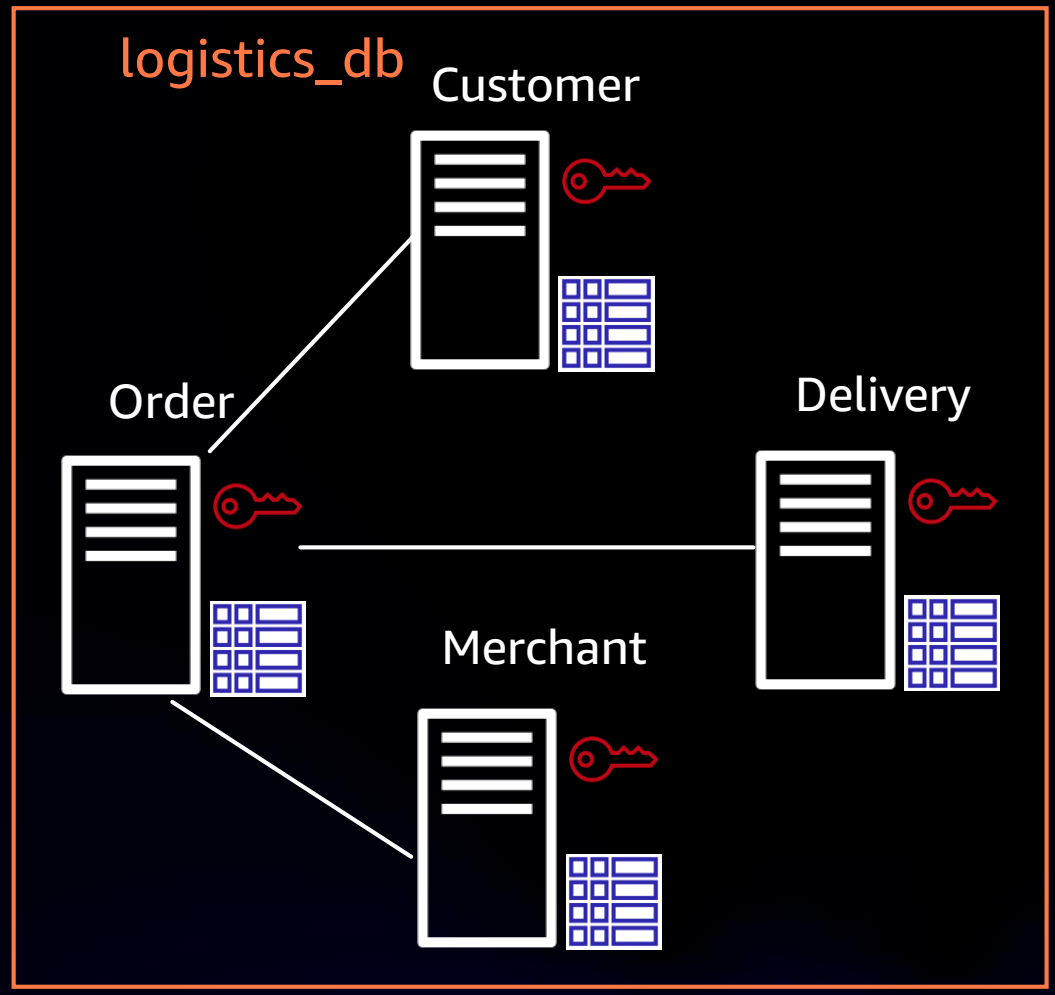
- We don’t know much about the architecture and the app schema
- We need the object model to align with the DB schema
- We want the database objects to just exist based on our model
- Therefore, we want to use an ORM

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Example app



ORM



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

“My app is great but starts to get slow. What can I do differently?”

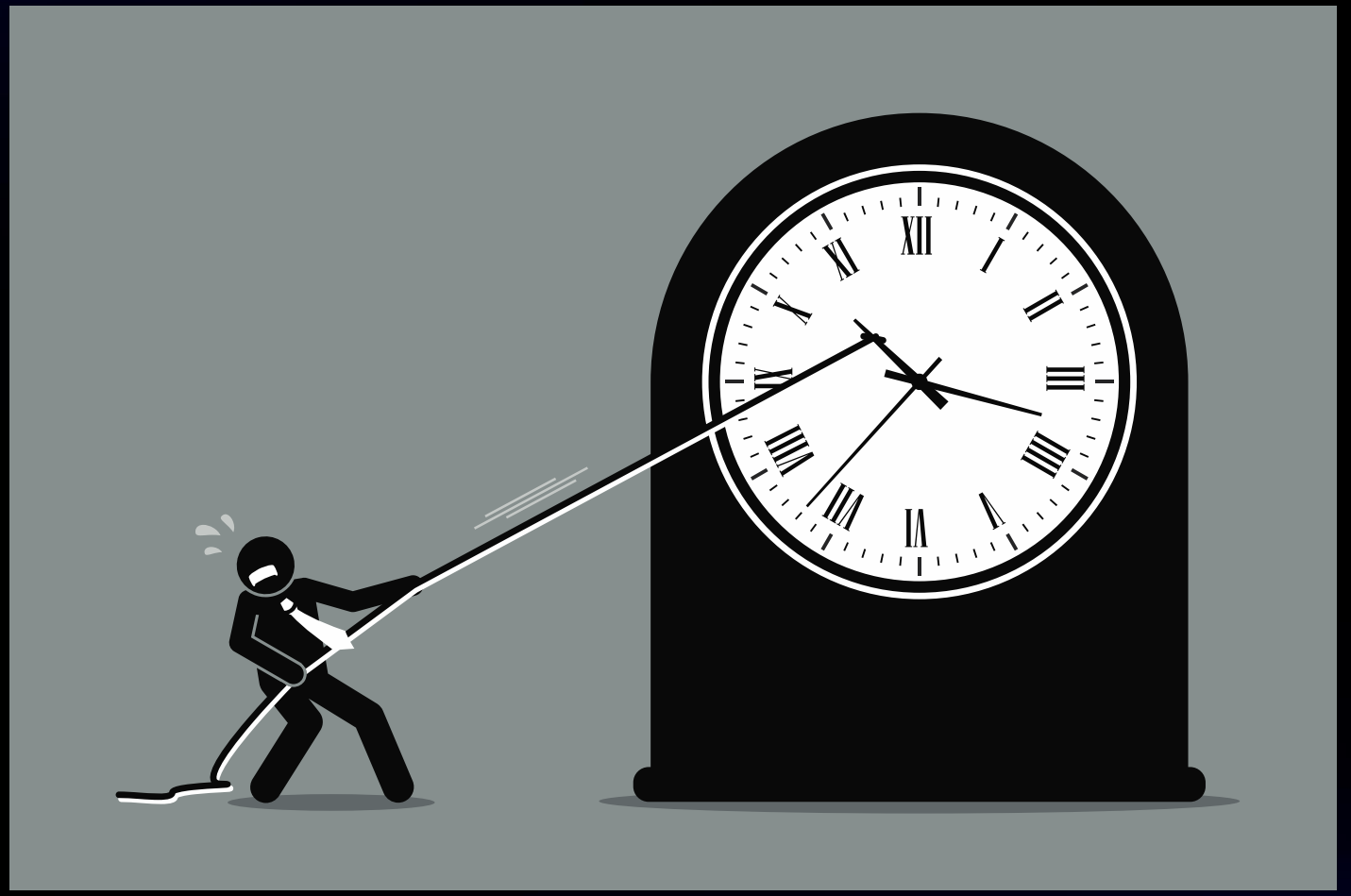






© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Rewind the clock



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Rewind the clock

- Define primary key on each table
  - Avoid running out of integer value → use bigint as primary key
  - Django ORM already does that
- Remove foreign key (FK) check constraints from DB in prod
  - Enforce referential integrity at application layer
  - Include test case(s) with FK constraints added to the DB
- Choose smallest data type to represent desired range of values
  - Use Boolean (PG) or tinyint (MySQL) for binary flag: 0 instead of “No”, 1 instead of “Yes”
  - Use enum instead of strings for status flag: “received”, “charged”, “delivered”

aws

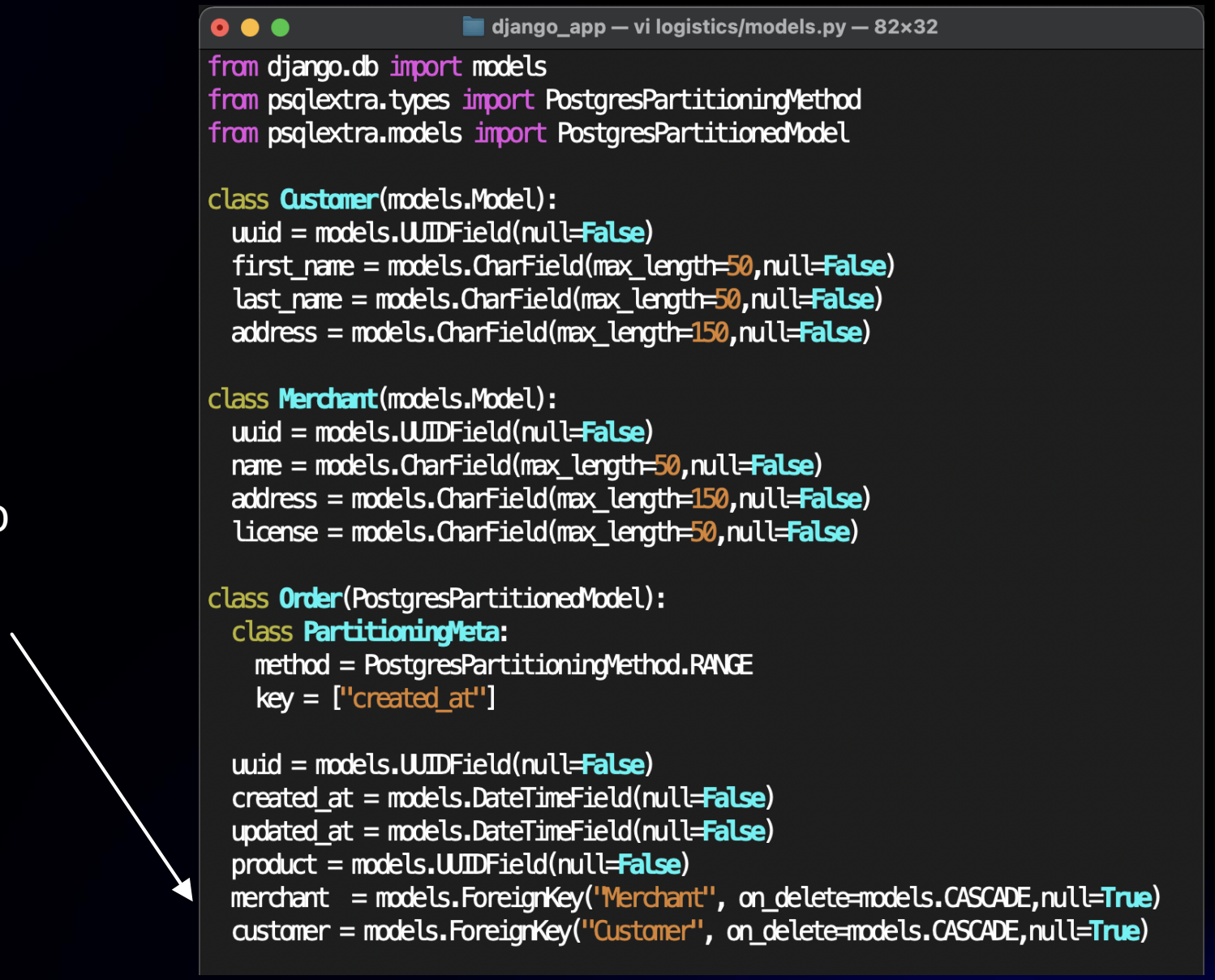
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Make your FKs removable

Why do we need the DB to check if a customer exists with a FK?
The app already knows the customer ID

Relax your DB integrity checks in prod when you need to scale

Enforce integrity check in test environment only



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Choose smallest data type

Scenario

status column

Possible values: “received”, “charged”, “delivered”

Number of rows: **8 billion**

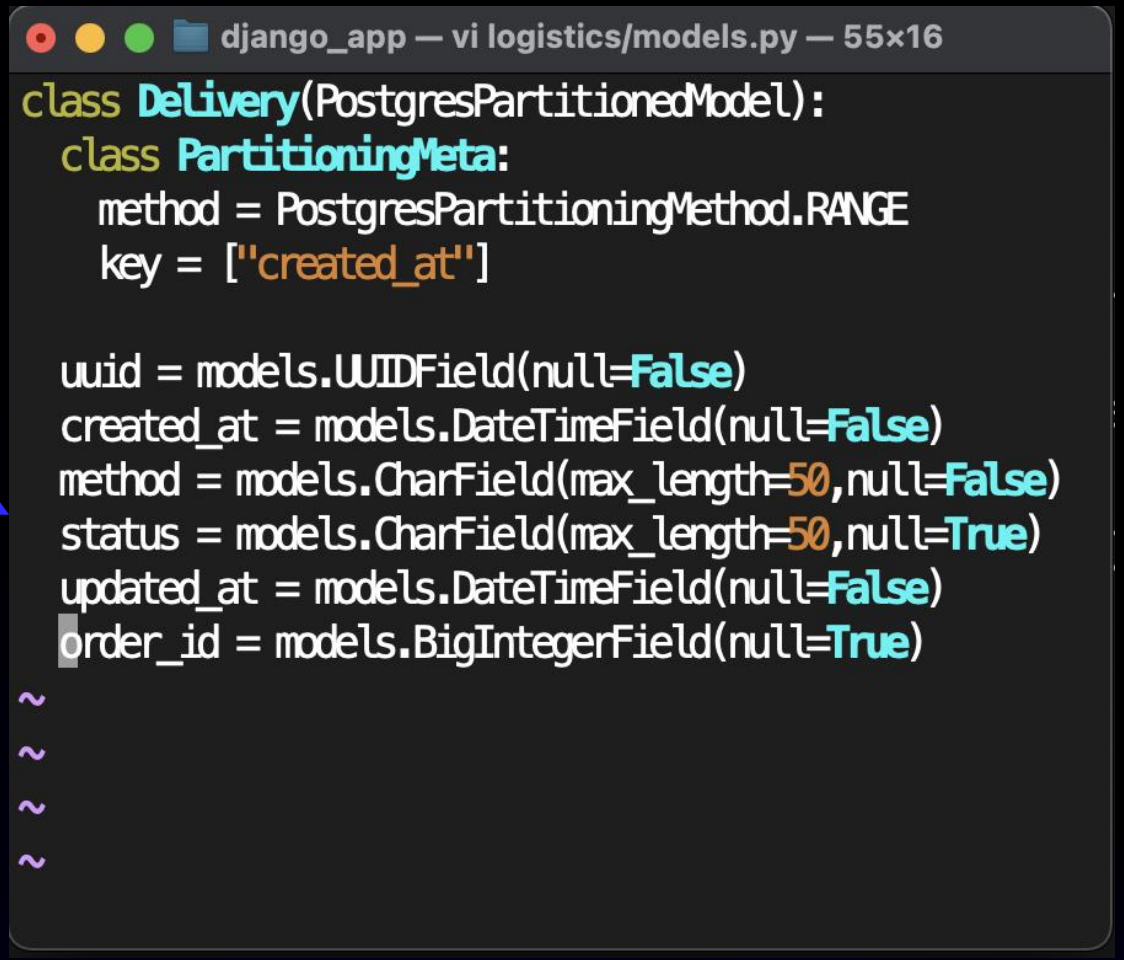
Average size of value

SELECT (length(‘received’) + length(‘charged’) + length(‘delivered’)) / 3;

= 8 bytes

Cost of storage

- VARCHAR(50): 8 bytes * 8 billion rows = 64 GB
- Enum (MySQL: 1–2 bytes): 1 bytes * 8 billion rows = 8 GiB



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Use connection pooling

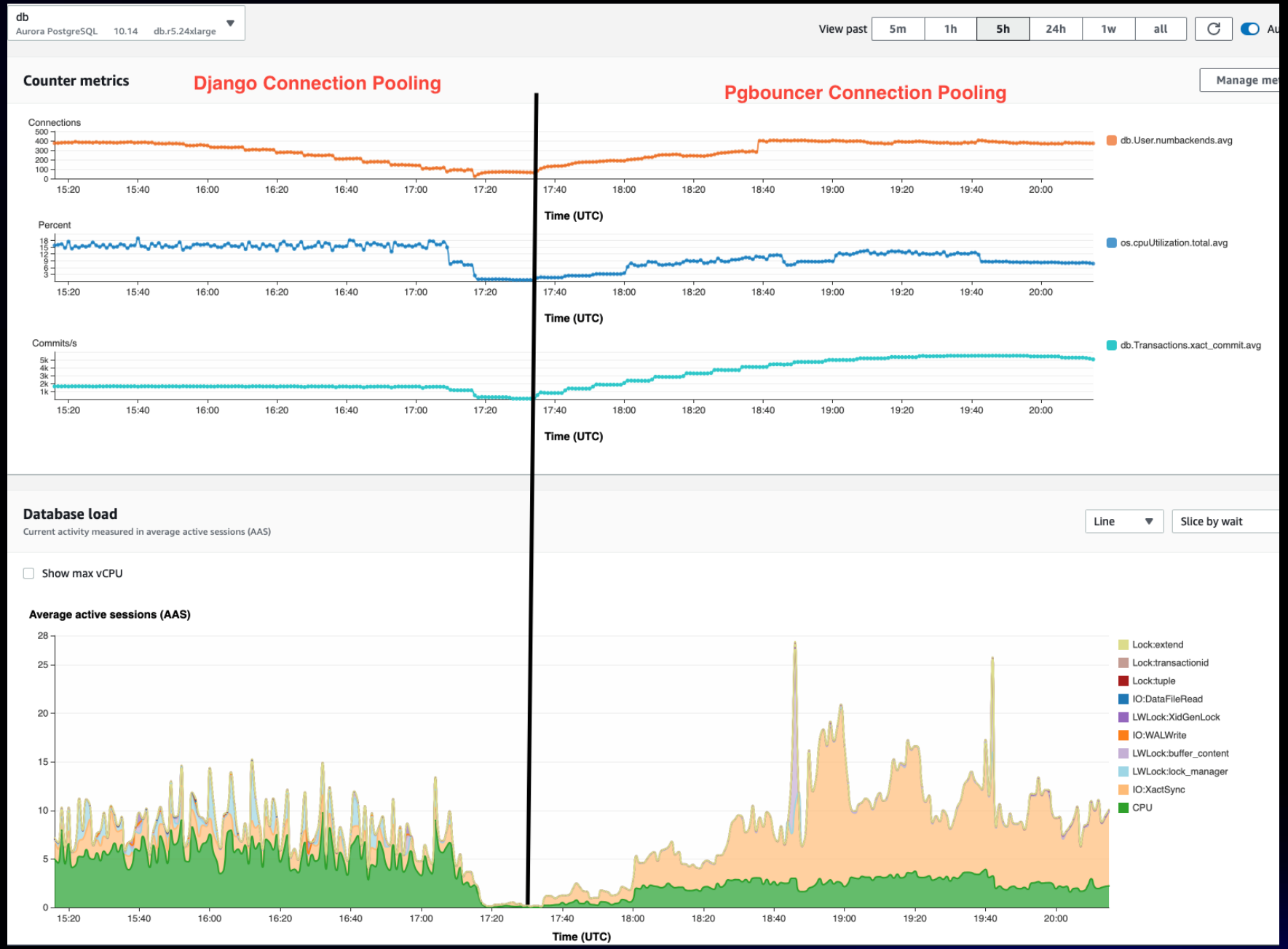
- New connection is expensive
- High number of connections increases context switches
- Your database is happy when it has #conn/#vcpu
- Django ORM connection is not persistent by default CONN_MAX_AGE=0
- Implement connection pooling solution

  Solutions: PgBouncer, ProxySQL, Amazon RDS Proxy



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Connection pooling in action



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Partition large tables

- Fit the app data size to database size
- Why do we need to store two years of old orders?
- Identify the column to partition (e.g., created_at in our Order or Delivery tables)
- Consider churning partitions to an OLAP database (like Amazon Redshift) to fit your hot data to the primary limits
- ORM supports partitioning
- Enhanced partitioning in PostgreSQL 13+ and MySQL 5.7+

aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Partition large tables with ORM

```
django_app — vi logistics/models.py — 61×16
class Delivery(PostgresPartitionedModel):
  class PartitioningMeta:
    method = PostgresPartitioningMethod.RANGE
    key = ["created_at"]

  uuid = models.UUIDField(null=False)
  created_at = models.DateTimeField(null=False)
  method = models.CharField(max_length=50,null=False)
  status = models.CharField(max_length=50,null=True)
  updated_at = models.DateTimeField(null=False)
  order_id = models.BigIntegerField(null=True)
~
~
```

```
postgres=> \d logistics_delivery
                                  Partitioned table "public.logistics_delivery"
   Column   |          Type          | Collation | Nullable |
------------+------------------------+-----------+----------+
 id         | bigint                 |           | not null | nextval('logistic
 uuid       | uuid                   |           | not null |
 created_at | date                   |           | not null |
 method     | character varying(50)  |           | not null |
 status     | character varying(50)  |           |          |
 updated_at | date                   |           | not null |
 order_id   | bigint                 |           |          |
Partition key: RANGE (created_at)
Indexes:
    "logistics_delivery_pkey" PRIMARY KEY, btree (id, created_at)
Number of partitions: 1 (Use \d+ to list them.)
~
```

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Consider multiple logical databases/schemas

- Start with logical databases – separate orders from deliveries
- Physical databases – separate readers from the writer instance
- Split the work to separate databases (e.g., orders_db and delivery_db)
- You can do it with ORM
- Add the database to the ORM database dictionary
- Use database router to simplify your code

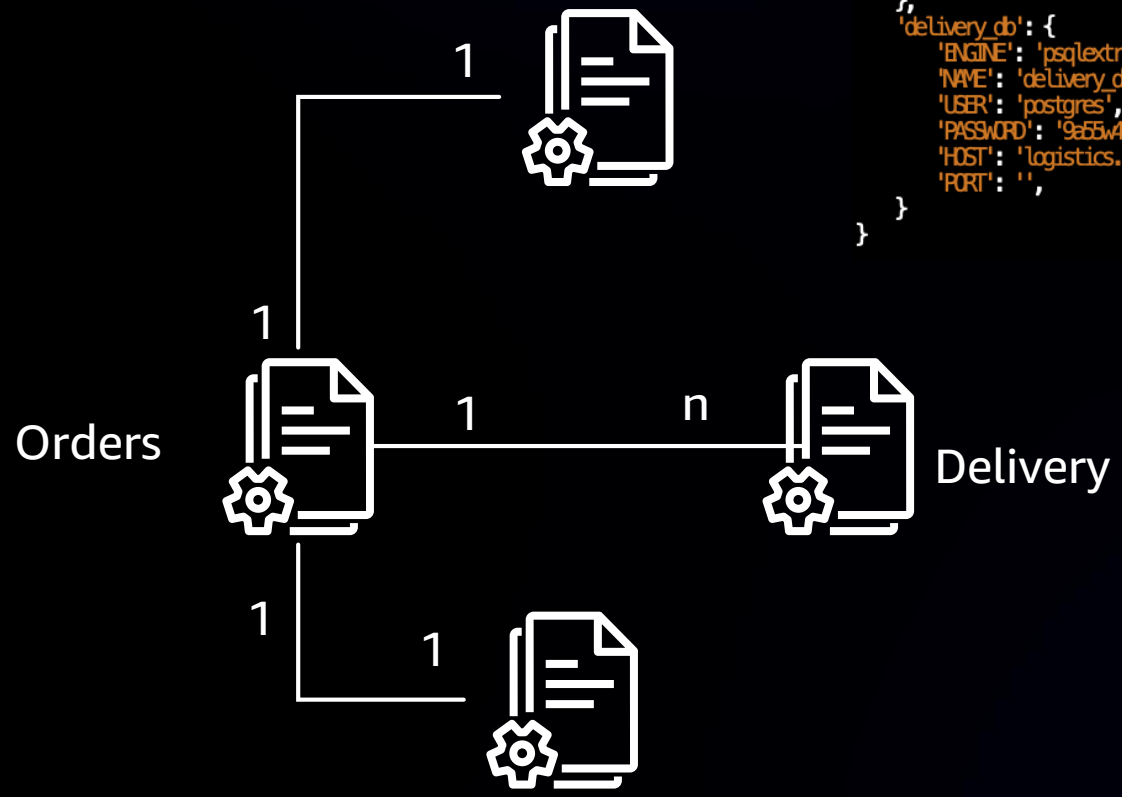
aws

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

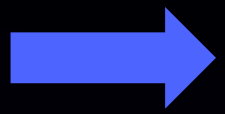
Multiple databases

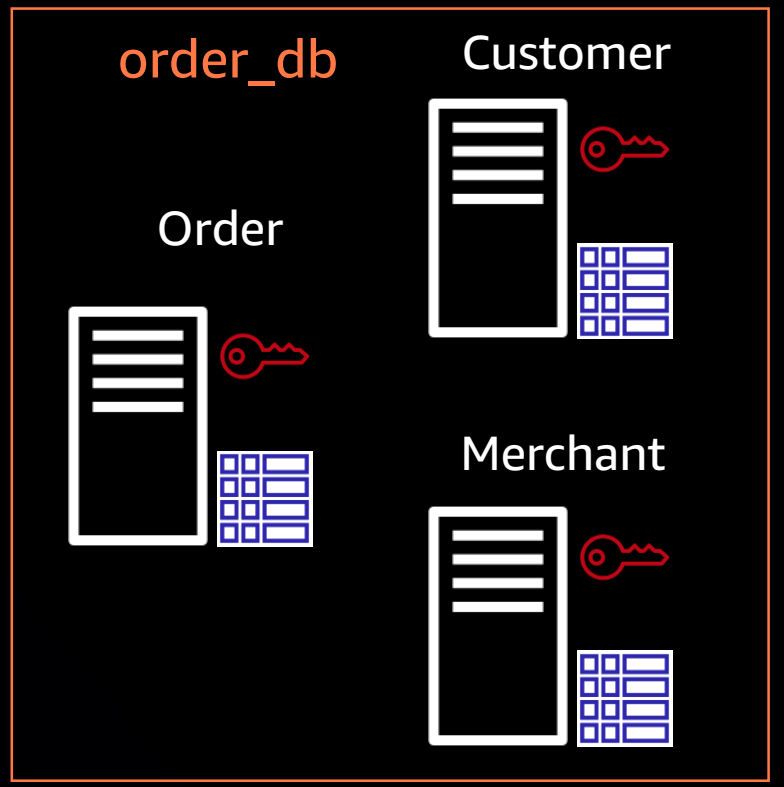
Database dictionary

```
DATABASES = {
    'order_db': {
        'ENGINE': 'psqlextra.backend',
        'NAME': 'order_db',
        'USER': 'postgres',
        'PASSWORD': '9a55w4d',
        'HOST': 'logistics.czk5ni6ofaxj.us-west-2.rds.amazonaws.com',
        'PORT': '',
    },
    'delivery_db': {
        'ENGINE': 'psqlextra.backend',
        'NAME': 'delivery_db',
        'USER': 'postgres',
        'PASSWORD': '9a55w4d',
        'HOST': 'logistics.czk5ni6ofaxj.us-west-2.rds.amazonaws.com',
        'PORT': '',
    }
}
```



ORM







© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

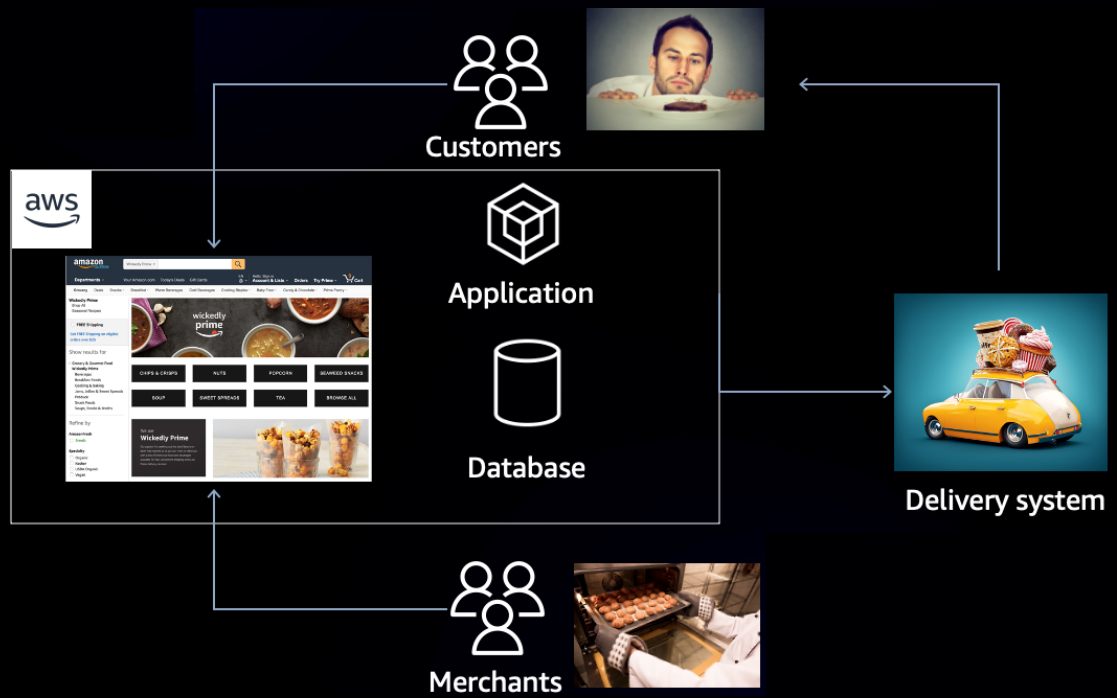
Infrastructure scaling

- Read scale-out: in-Region and cross-Region read replica
- Compute/memory
  - Choice of DB instance types: T3, M5, R5, R6, M6g, R6g
  - xlarge to 24xlarge, with up to 96 vCPU, 768 GB RAM
- Storage
  - Storage space auto scaling
  - Scale IOPS/throughput: gp2, io1, io2
  - Amazon Aurora limit: 128 TB
  - Amazon RDS limit: 64 TB

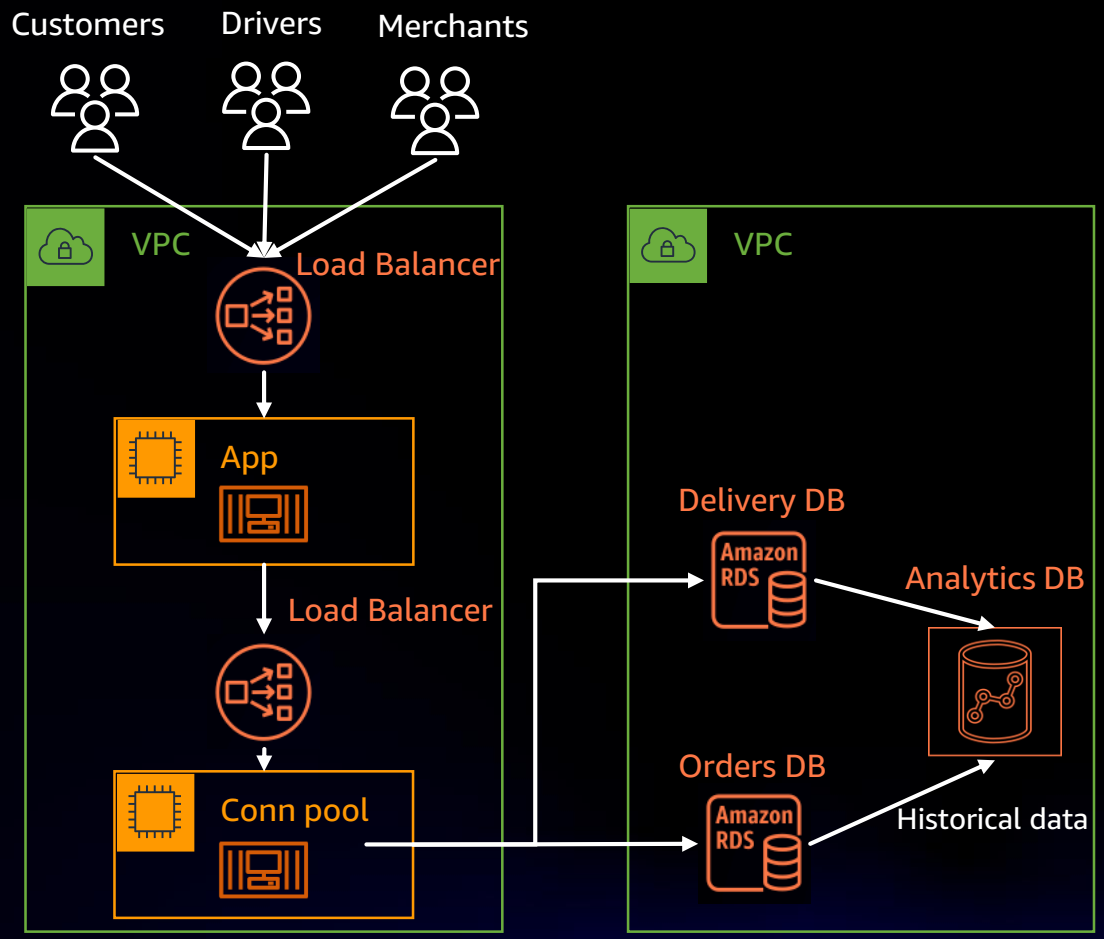
© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Problem and solution recap

The business model



The solution architecture



© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Key takeaways

- Know the ORM limitations – choose the smallest data type, minimize foreign keys, conn_max_age
- Allow the hot working set to live in memory – partitioning or multiple databases
- Reduce concurrency on the database – use connection pooling
- Use logical database/schema to keep independent data decoupled
- Try it yourself: bit.ly/3B1110p

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Yahav Biran

birayaha@amazon.com
@yahavbiran – Twitter

Wanda He

wanhe@amazon.com

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.