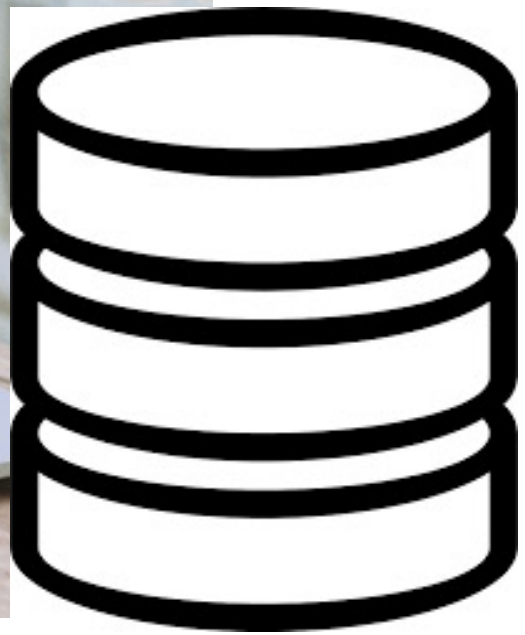
Comp 5400 – Hybrid and Relational Database

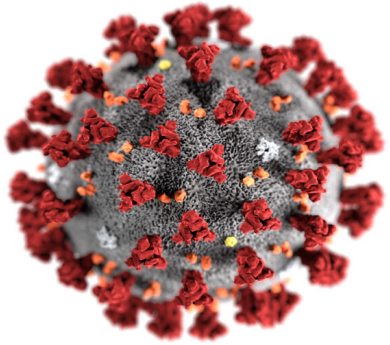
Project Presentation

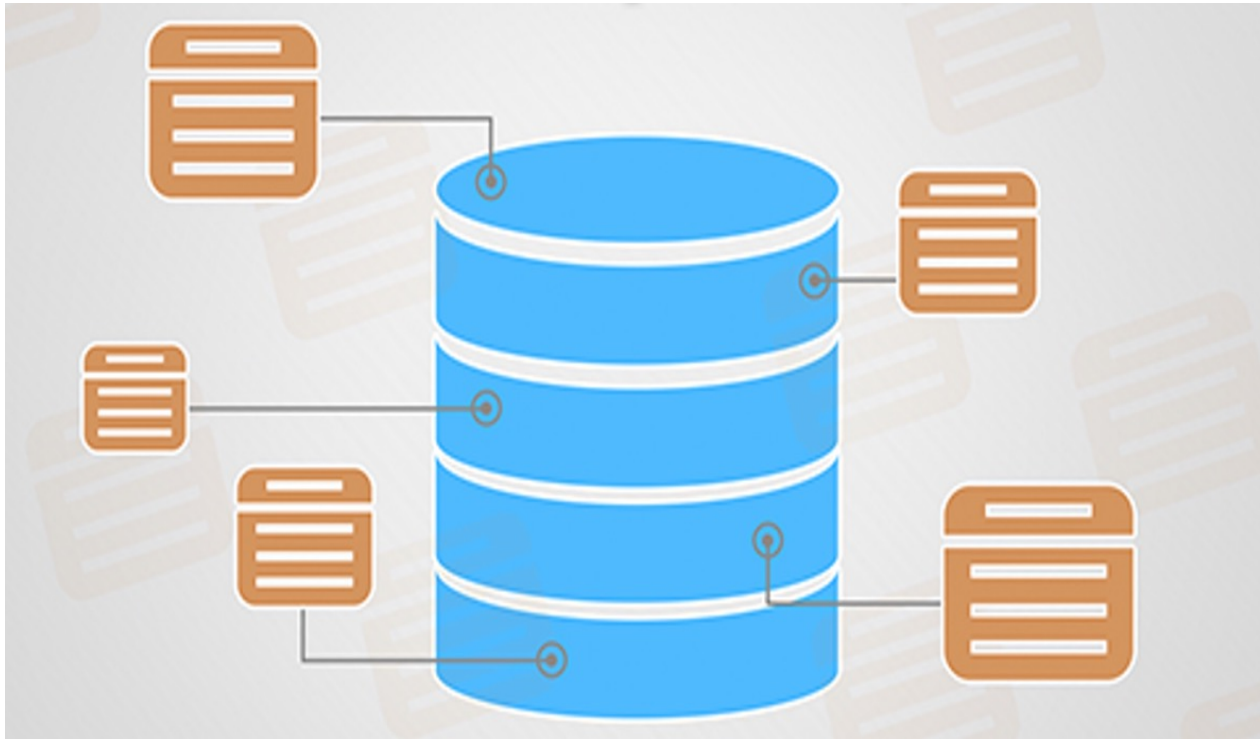
by

Eze Amadi

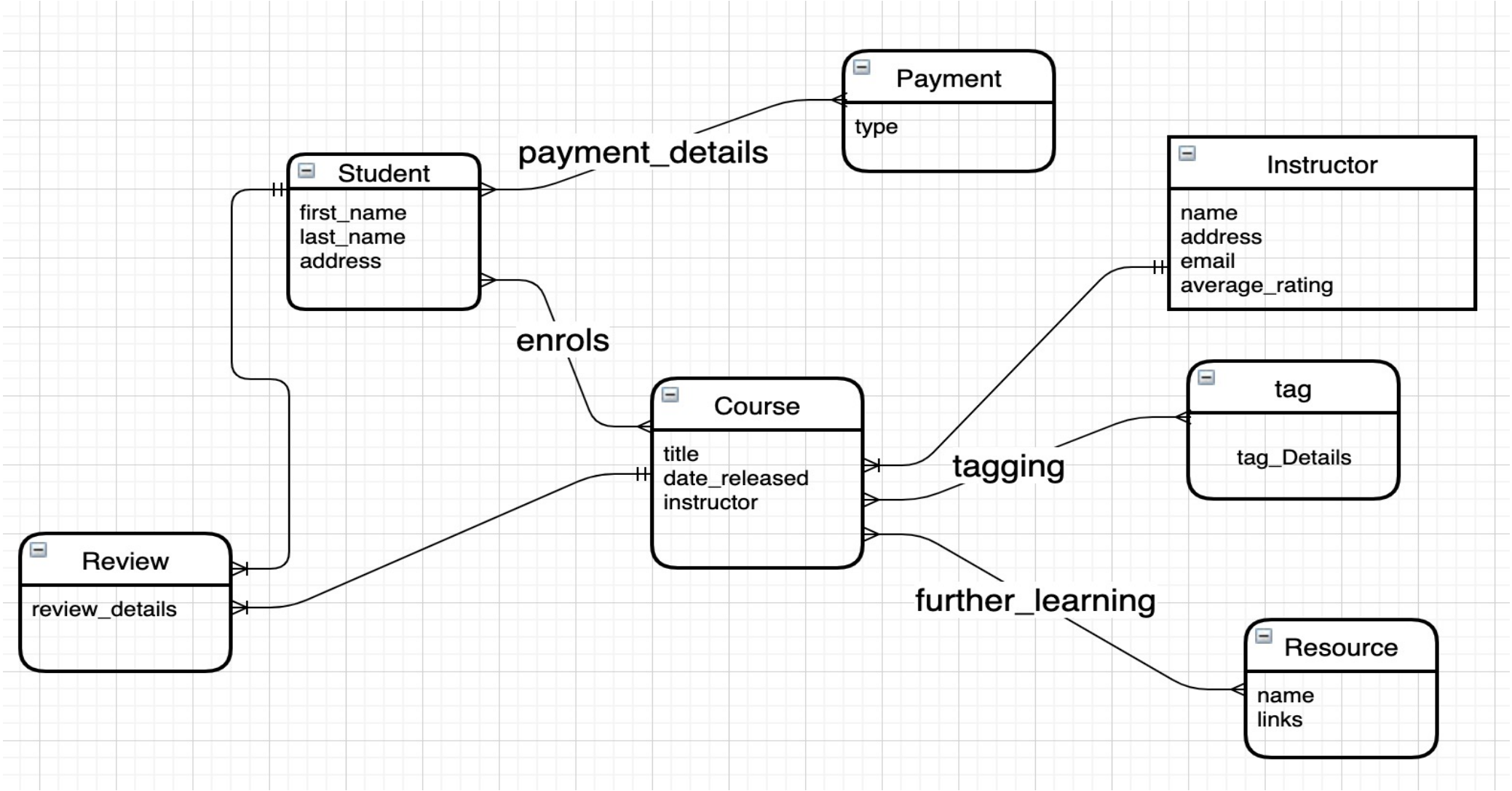


This is a database for an online learning platform – UltiLearn.

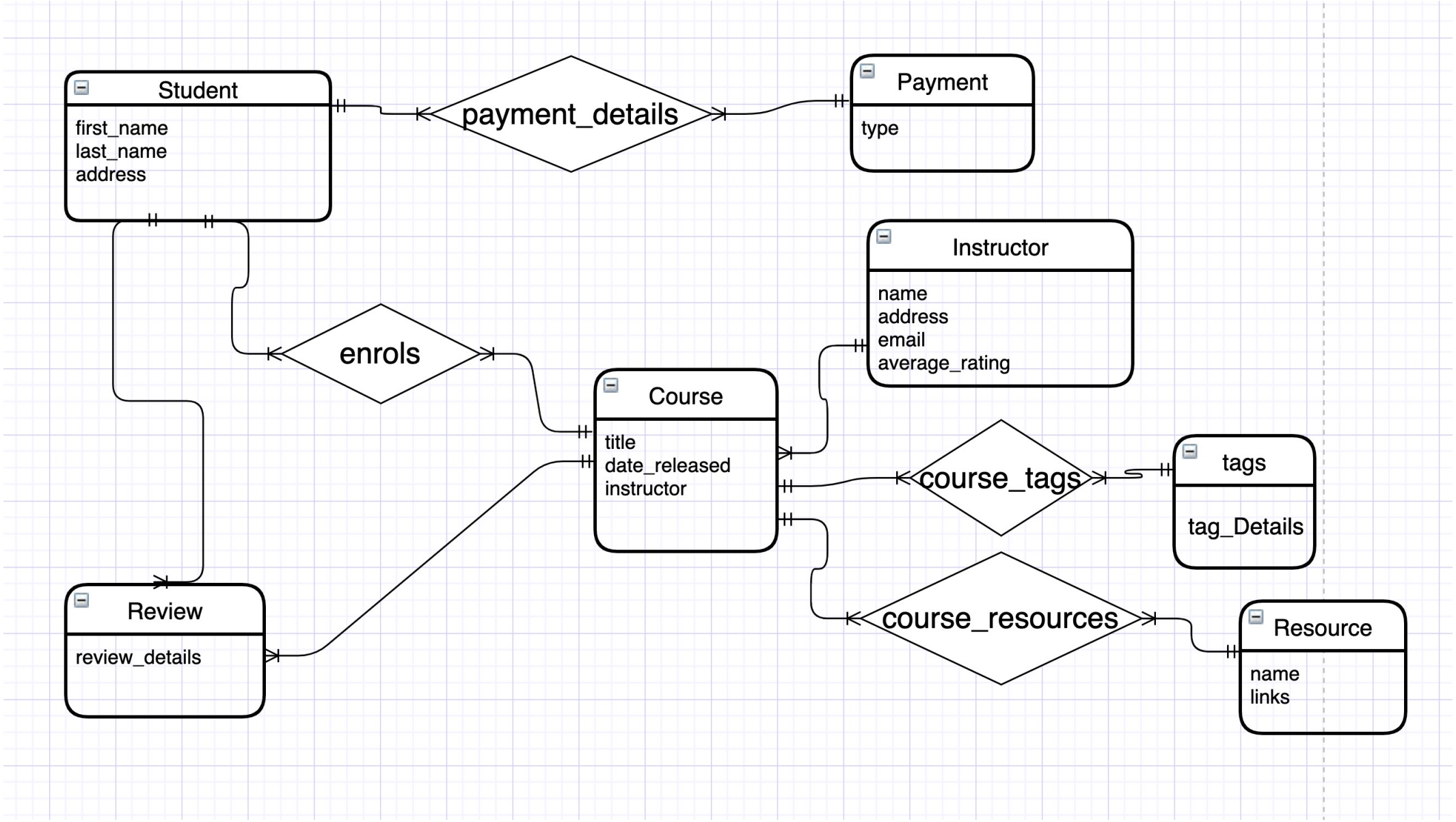




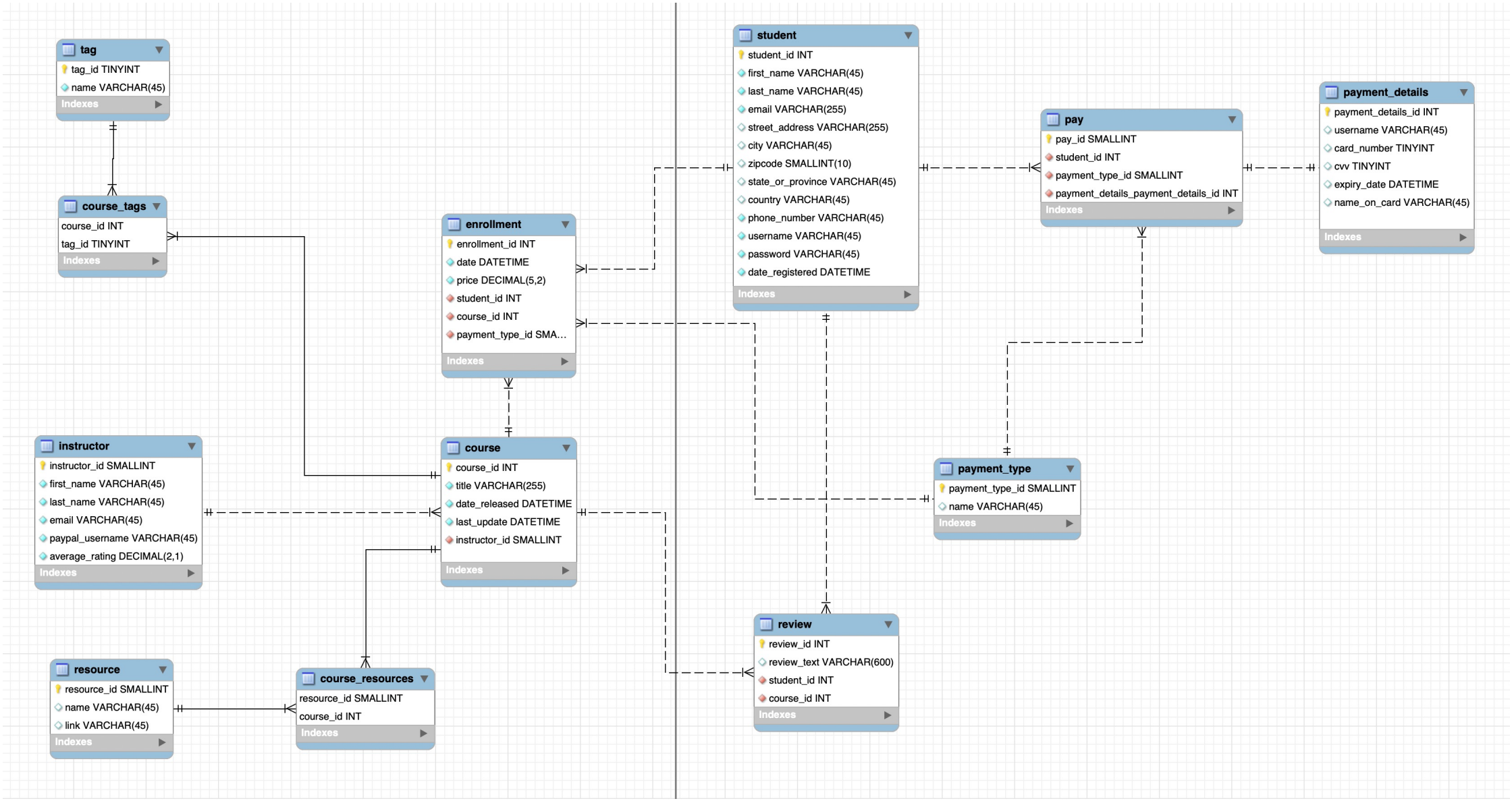
The conceptual schema of UltiLearn's database shows the proposed components.



The Entity-Relationship Diagram of the database shows some details of the entities and their relationships.



The full details of the tables shows the one-to-one, one-to-many and many-to-many relationships between the tables.



SQLAlchemy ORM was used to build the database with Jupyter Notebook.

In [1]:
```
from sqlalchemy import Table, Column, Integer, Numeric, String, CheckConstraint, desc
from sqlalchemy import DateTime, ForeignKey, Boolean, func
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import relationship, backref
from sqlalchemy.orm import sessionmaker
from sqlalchemy import create_engine
from datetime import date, datetime
import pandas as pd
import numpy as np
```

In [2]:
```
engine = create_engine('sqlite:///:memory:')
# engine = create_engine('sqlite:///OnlineLearningPlatform2.db')

# Create a session
Session = sessionmaker(bind = engine)
session = Session()

# Create the Base
Base = declarative_base()
```

This is an example of Entity type class with attributes and methods.

In [4]:

```
# Student Class

class Student(Base):

    __tablename__ = 'student'

    __table_args__ = ({'extend_existing' : True})

    student_id = Column(Integer(), nullable=False, primary_key=True)
    first_name = Column(String(50), nullable=False, index=True)
    last_name = Column(String(50), nullable=False, index=True)
    email = Column(String(255), nullable=False,)
    street_address = Column(String(255))
    city = Column(String(45))
    zipcode = Column(Integer())
    state_or_province = Column(String(45))
    country = Column(String(45))
    phone_number = Column(String(45))
    username = Column(String(45), nullable=False)
    password = Column(String(45), nullable=False)
    date_registered = Column(DateTime, default=datetime.now)


    def __init__(self, student_id, first_name, last_name, email, street_address, city, zipcode,
                 state_or_province, country, phone_number, username, password, date_registered):

        self.student_id = student_id
        self.first_name = first_name
        self.last_name = last_name
        self.email = email
        self.street_address = street_address
        self.city = city
        self.zipcode = zipcode
        self.state_or_province = state_or_province
        self.country = country
        self.phone_number = phone_number
        self.username = username
        self.password = password
        self.date_registered = date_registered


    def printObj(self):

        print("Student ID " , self.student_id)
        print("First name is " , self.first_name)
        print("Last name is " , self.last_name)
        print("Username is " , self.username)
        print("Email address is " , self.email)
        print("Phone is " , self.phone_number)
        print("The Date Registered was ", self.date_registered)

    def conv_to_tup(self):

        return (self.student_id, self.first_name, self.last_name, self.email, self.street_address, self.city,
                self.zipcode, self.state_or_province, self.country, self.phone_number, self.username, self.password,
                self.date_registered)
```

Relationship type relations of the database shows the ForeignKeys and relationships with other tables.

In [8]:
```
# Enrollment Class

class Enrollment(Base):

    __tablename__ = 'enrollment'

    __table_args__ = ({'extend_existing' : True})

    enrollment_id = Column(String(10), primary_key=True)
    enrollment_date = Column(DateTime, nullable=False, default=datetime.now)
    price = Column(String(10), nullable=False,)
    student_id = Column(Integer(), ForeignKey('student.student_id'))
    course_id = Column(String(10), ForeignKey('course.course_id'))
    payment_type_id = Column(String(10), ForeignKey('payment_type.payment_type_id'))

    student = relationship('Student', backref=backref('enrollment'))
    course = relationship('Course', backref=backref('enrollment'))
    payment_type = relationship('Payment_Type', backref=backref('enrollment'))

    def __init__(self, enrollment_id, enrollment_date, price, student_id, course_id, payment_type_id):

        self.enrollment_id = enrollment_id
        self.enrollment_date = enrollment_date
        self.price = price
        self.student_id = student_id
        self.course_id = course_id
        self.payment_type_id = payment_type_id


    def printObj(self):

        print("Enrollment ID is " , self.enrollment_id)
        print("Enrollment Date is " , self.enrollment_date)
        print("Price for enrollment is " , self.price)


    def conv_to_tup(self):

        return (self.enrollment_id, self.enrollment_date, self.price, self.student_id, self.course_id,
                self.payment_type_id)
```

The different tables formatted in Pandas dataframe show the columns and rows of data in the database.

In [41]:
```
print("                                                STUDENTS TABLE")
get_data_and_display(Student, 'student_id')
```

```
                                                STUDENTS TABLE
```

Out[41]:

student_id	first_name	last_name	email	street_address	city	zipcode	state_or_province	country	phone_number	username	passw
1	Doyle	Sullivan	dsuli@mac.com	876 Pearl Dr	Old Bridge	94566	New Jersey	USA	257-555-0179	basketba283	V37X
2	Moses	Richards	mricha@gmail.com	7660 Maple St	Sheboygan	3714	Osaka	Japan	202-093-0112	bladerun71	/xcM7*n
3	Johnny	Cox	jocox@msn.com	158 Primrose Dr	Banning	22003	Lagos	Nigeria	211-345-0164	kidneyho72	$rZt2
4	Forrest	Miles	zilla@att.net	8029 East South Street	Lengton	46321	Missisipi	USA	930-826-0102	pianobass13	Xk6ZTv#
5	Kyle	Torres	ktorress@icloud.com	146 Somerset St	Port Orange	29841	Sao Paolo	Brazil	567-555-0122	toystoryweb9	{pf'r*6H
6	Dana	Morrison	mdana@comcast.net	8100 Randall Mill Circle	Charlottesville	15102	London	UK	025-982-0161	beerflora32	5FfpDw[

In [42]:
```
print("                    INSTRUCTORS TABLE")
get_data_and_display(Instructor, "instructor_id")
```

```
                    INSTRUCTORS TABLE
```

Out[42]:

instructor_id	first_name	last_name	email	paypal_username
1	Melvin	Ortiz	mortiz@evt.edu	sunanchamir91
2	Sergio	Carroll	scarr@opt.com	hypernova64
3	Kathleen	Garner	katner.p@cpt.com	hypernova64
4	Gilberto	Patterson	gil.patt@ret.edu	harpsoprano00
5	Sophie	Clarke	sophclarke@tre.et	harprhythm76
6	Marcus	Moran	marmor45@gmail.com	clefflat6653
7	Colleen	Gordon	coll.gord@comp.edu	neose7en

Data manipulation of the data in the tables shows more information that can be obtained by querying the database.

In [54]:
```
## Calculate the average rating of each course

joined_course = session.query(Review.course_id, Course.title, func.count(Review.course_id).label("Number of Reviews"),
                              func.avg(Review.star_rating).label("Average Star Rating"))

joined_course = joined_course.join(Course).group_by(Review.course_id)

data = list()
for obj in joined_course:
    to_append = obj[:3] + (round(obj[3], 2),)
    data.append(to_append)

col = [Review.course_id, Course.title, func.count(Review.course_id), func.avg(Review.star_rating)]

df = pd.DataFrame(data=data, columns = col).astype(str)
df.set_index(Review.course_id)
df.style.hide_index()
```

Out[54]:

Review.course_id	Course.title	count(review.course_id)	avg(review.star_rating)
c1	Values Of Minority Queer Theory: Ideas In Conflict	3	4.33
c10	Birdwatching In Recent Times: Myth & Reality	1	3.0
c13	Sex, Community, And Family In The United States: Different Points Of View	1	4.0
c14	Here Come The Mass Extinctions: Topics In Modern Environmentalism	1	2.0
c15	Ad-Hoc Investigation Of The Liberated Female Revolution In The Modern Age	1	5.0
c2	Latino Ideas: The Big Picture	1	5.0
c3	Philosophy Of Westward Expansion: An Interdisciplinary Approach	1	5.0
c4	Masterpieces Of Western Italian Drama	1	3.0
c7	Masterpieces Of Populist Asian Folklore	1	5.0
c8	Horror & Infidelity In The 21st Century	2	3.5
c9	German Marxism Traditions In The Liberal World	2	3.5

Another example of data manipulation…

In [56]:
```
## How much has each instructor made

joined_course = session.query(Instructor.first_name.label('first name'),
                              Instructor.last_name.label('last name'),
                              func.sum(Enrollment.price).label("Total Amount Made"))

joined_course = joined_course.join(Course, Enrollment.course_id == Course.course_id
                                   ).join(Instructor, Course.instructor_id == Instructor.instructor_id
                                          ).group_by(Instructor.instructor_id
                                                     ).order_by(desc("Total Amount Made"))

data = list()

for obj in joined_course:
    to_append = obj[:2] + (round(obj[2], 2),)
    data.append(to_append)

col = [Instructor.first_name, Instructor.last_name, func.sum(Enrollment.price)]

df = pd.DataFrame(data=data, columns = col).astype(str)
df.set_index(Instructor.first_name)
df.style.hide_index()
```

Out[56]:

Instructor.first_name	Instructor.last_name	sum(enrollment.price)
Gilberto	Patterson	149.94
Sophie	Clarke	64.98
Melvin	Ortiz	55.96
Marcus	Moran	51.96
Kathleen	Garner	24.98
Sergio	Carroll	19.99
Colleen	Gordon	9.99

In conclusion, this project served as a good exercise in a deeper understanding of databases and SQLAlchemy ORM.

Any
Questions???