1. Adott relációs adatbázissal egyenértékű NoSQL adatbázis létrehozása. A relációs adatbázis legalább 3 táblás legyen, a táblák között legyenek kapcsolatok. A NoSQL adatbáziskezelő rendszer a tanultak bármelyike lehet. Beadandó a relációs adatbázis leírása, a NoSQL megfelelőjének leírása + 5 db lekérdezés a NoSQL adatbázison. Ez utóbbi megvalósítható parancssoros (preferált) és grafikus felületen is. A lekérdezések értelmes funkciót valósítsanak meg.  
**1. Könyvek dokumentum**  
{

\_id: ObjectId("...")

cim: "A kém, aki megérdemelte a meghalást",

kiadas\_eve: 1953,

szerzok:   
[

ObjectId("..."),

ObjectId("...")

]

}

**2. Szerzők dokumentum**

{

\_id: ObjectId("..."),

nev: "John le Carré",

szuletesi\_ev: 1931

}

**Lekérdezések**

1. Könyv címének lekérdezése szerző alapján:

db.konyvek.find({ szerzok: ObjectId("...") }, { cim: 1, \_id: 0 })

1. Szerző nevének lekérdezése könyv címe alapján:

db.szerzok.find({ \_id: { $in: [ObjectId("..."), ObjectId("...")] } }, { nev: 1, \_id: 0 })

1. Egy adott könyv adatainak lekérdezése:

db.konyvek.findOne({ \_id: ObjectId("...") })

1. Azon könyvek lekérdezése, melyek kiadási éve az adott év felett van:

db.konyvek.find({ kiadas\_eve: { $gt: 2000 } })

1. Könyv címének és kiadási évének lekérdezése adott szerző neve alapján:

db.konyvek.aggregate  
([

{ $lookup: { from: "szerzok", localField: "szerzok", foreignField: "\_id", as: "szerzok" } },

{ $unwind: "$szerzok" },

{ $match: { "szerzok.nev": "John le Carré" } },

{ $project: { cim: 1, kiadas\_eve: 1, \_id: 0 } }

])

2. NoSQL adatbázis létrehozása importálással vagy parancsokkal, majd csatlakozás valamilyen lehetséges programnyelv (pl. Python) segítségével, és lekérdezések (legalább 3) vagy CRUD-műveletek (legalább 3) megvalósítása programkódból. Beadandó a NoSQL adatbázis létrehozásának leírása, az adatbázis leírása + a programkód.

**# Létrehozás**

def konyv\_hozzaad(cim, szerzo, kiado, megjelenes\_datuma, mufaj):

konyvek\_gyujtemeny = adatbazis["konyvek"]

konyv = {

"cim": cim,

"szerzo": szerzo,

"kiado": kiado,

"megjelenes\_datuma": megjelenes\_datuma,

"mufaj": mufaj

}

konyvek\_gyujtemeny.insert\_one(konyv)

**# olvasás**

def szerzo\_alapjan\_konyvek(szerzo\_neve):

konyvek\_gyujtemeny = adatbazis["konyvek"]

return konyvek\_gyujtemeny.find({"szerzo": szerzo\_neve})

**# frissítés**

def konyv\_cimenek\_frisselese(regi\_cim, uj\_cim):

konyvek\_gyujtemeny = adatbazis["konyvek"]

konyvek\_gyujtemeny.update\_many({"cim": regi\_cim}, {"$set": {"cim": uj\_cim}})

**# törlés**

def konyv\_torlese(cim):

konyvek\_gyujtemeny = adatbazis["konyvek"]

konyvek\_gyujtemeny.delete\_one({"cim": cim})

**# Tesztelés**

if \_\_name\_\_ == "\_\_main\_\_":

**# Könyv létrehozása**

konyv\_hozzaad("Harry Potter", {"nev": "J.K. Rowling"}, {"nev": "Bloomsbury"}, "1997-06-26", "Fantasy")

**# J.K. Rowling könyveinek kiolvasása**

jk\_rowling\_konyvei = szerzo\_alapjan\_konyvek({"nev": "J.K. Rowling"})

for konyv in jk\_rowling\_konyvei:

print(konyv)

**# Könyv címének módosítása**

konyv\_cimenek\_frisselese("Harry Potter", "Harry Potter és a bölcsek köve")

**# Könyv törlése**

konyv\_torlese("Harry Potter és a bölcsek köve")