Jegyzőkönyv

Webes adatkezelő környezetek

Féléves feladat

Könyvesbolt

Készítette: Orosz Péter
Neptun kód: WO02D7
Dátum: 2025. December

Tartalomjegyzék

Bevezetés **2**

A feladat leírása **2**

1. Könyvesbolt **2**

1.1. Az adatbázis ER modell tervezése 2

1.2. Az adatbázis konvertálása XDM modellre 3

1.3. Az XDM modell alapján XML dokumentum készítése 4

1.4. Az XML dokumentum alapján XMLSchema készítése 4

Bevezetés

A feladat leírása

1 Könyvesbolt

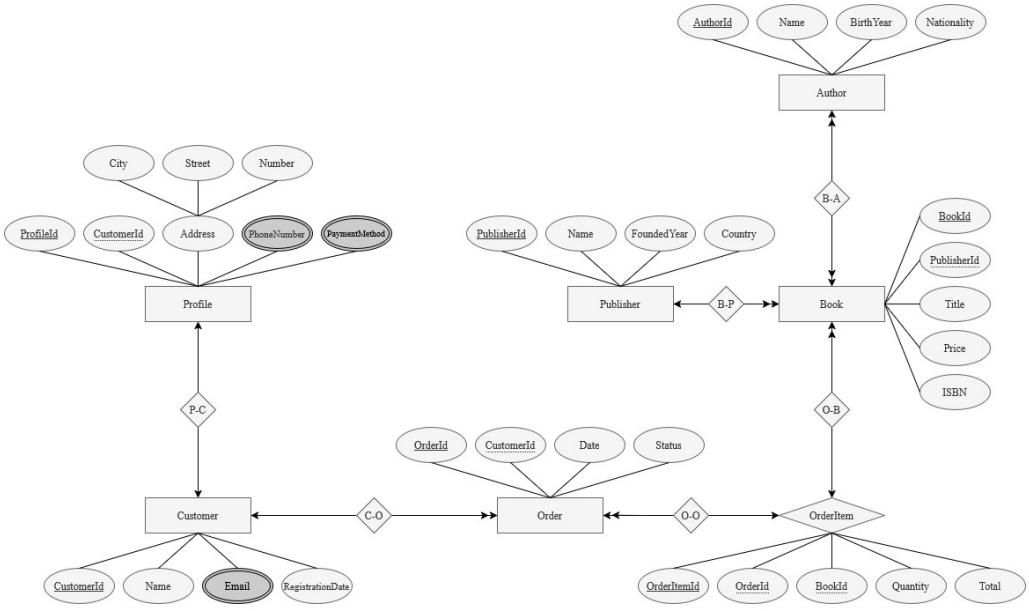
1.1 Az adatbázis ER modell tervezése

Az adatbázis szerkezetét az ER diagram szemlélteti (lásd az 1. ábrát), amelyben az entitások, attribútumok és kapcsolatok kerültek megtervezésre. Az ER modell célja, hogy áttekinthetően ábrázolja az adatbázis logikai felépítését, a főbb adatcsoportokat és azok összefüggéseit.

Az ER modell tervezése során először azokat a főbb adatcsoportokat (entitásokat) azonosítottam, amelyek a könyváruház működéséhez szükségesek: vevők (Customer), profilok (Profile), rendelések (Order), rendelési tételek (OrderItem), könyvek (Book), szerzők (Author) és kiadók (Publisher). Ezeket a valós üzleti folyamatok alapján választottam ki, figyelembe véve, hogy milyen információkat kell tárolni és milyen kapcsolatokat kell kezelni közöttük.

Az entitásokhoz hozzárendeltem a szükséges attribútumokat, például a vevőhöz nevet, email címet és regisztrációs dátumot, a könyvhöz címet, árat és ISBN-t. A kapcsolatok meghatározásánál ügyeltem arra, hogy a valós kapcsolatok (pl. egy vevő több rendelést is leadhat, egy könyvet több szerző is írhat) helyesen jelenjenek meg az adatmodellben. Az összetett kapcsolatokhoz (pl. Book-Author) kapcsolótáblát (B-A) terveztem.

Az ER diagram elkészítéséhez először papíron vázlatot készítettem, majd digitálisan, diagramkészítő eszközzel rajzoltam meg a végleges változatot, amelyet az alábbi ábrán mutatok be.



1. ábra. Az adatbázis ER diagramja

Entitások és attribútumaik

Entitás	Főkulcs	Idegen kulcs(ok)	Attribútum(ok) (típus)
Profile	ProfileId	CustomerId	Address: City (egyszeres) Address: Street (egyszeres) Address: Number (egyszeres) PhoneNumber (többszörös) PaymentMethod (többszörös)
Customer	CustomerId	–	Name (egyszeres) Email (többszörös) RegistrationDate (egyszeres)
Order	OrderId	CustomerId	Date (egyszeres) Status (egyszeres)
OrderItem	OrderItemId	OrderId, BookId	Quantity (egyszeres) Total (egyszeres)
Book	BookId	PublisherId	Title (egyszeres) Price (egyszeres) ISBN (egyszeres)
Author	AuthorId	–	Name (egyszeres) BirthYear (egyszeres) Nationality (egyszeres)
B-A	–	BookId, AuthorId	–
Publisher	PublisherId	–	Name (egyszeres) FoundedYear (egyszeres) Country (egyszeres)

1.2 Az adatbázis konvertálása XDM modellre

Az ER diagram alapján elkészült az XDM modell (lásd a 2. ábrát), amely az adatszerkezetet XML-re optimalizált formában mutatja be. Az XDM modell segíti az XML dokumentum struktúrájának megtervezését, figyelembe véve az adatok hierarchiáját és összefüggéseit.

Az XDM modell elkészítésekor az ER diagram entitásait és kapcsolatait kellett úgy átalakítanom, hogy azok megfeleljenek az XML hierarchikus szerkezetének. Első lépésként meghatároztam, hogy mely entitások lesznek főbb gyökérelemek (pl. Customer, Book), és melyek lesznek beágyazott vagy hivatkozott elemek (pl. Order, OrderItem).

Az ER modell relációs szerkezetét át kellett alakítani úgy, hogy az XML-ben a kapcsolatok vagy beágyazással, vagy attribútumként történő hivatkozással jelenjenek meg. Például a rendelési tételeket (OrderItem) az adott rendelés (Order) alá helyeztem, és a könyvekre, szerzőkre hivatkozásokat attribútumként vagy külön elemként jelenítettem meg.

A modell készítése során törekedtem arra, hogy az adatok logikusan, könnyen feldolgozható módon jelenjenek meg az XML-ben, és a későbbi validálás is egyszerű legyen. A végleges XDM modellt szintén diagramkészítő eszközzel rajzoltam meg.

2. ábra. Az adatbázis XDM modellje

1.3 Az XDM modell alapján XML dokumentum készítése

Az XDM modellből kiindulva elkészült az XML dokumentum (lásd: `WO02D7_XML.xml`), amely az adatok tényleges tárolását biztosítja. Az XML fájlban az adatok a modellnek megfelelően hierarchikusan, jól strukturáltan jelennek meg.

Az XML dokumentum elkészítéséhez először az XDM modell szerkezetét követtem, és meghatároztam a főbb elemeket és azok attribútumait. Az egyes entitásokhoz tartozó adatokat mintaként töltöttem ki, ügyelve arra, hogy minden kötelező mező szerepeljen, és a kapcsolatok (pl. rendelés és rendelési tételek, könyvek és szerzők) is megjelenjenek.

Az XML szerkesztése során figyeltem arra, hogy a dokumentum jól olvasható, áttekinthető legyen, és megfeleljen az XDM modellben megadott hierarchiának. A dokumentumot Visual Studio Code-al készítettem el, majd ellenőriztem, hogy szintaktikailag helyes legyen.

```
<?xml version="1.0" encoding="UTF-8"?>
<BookStore>
  <Customer CustomerId="C001">
    <Name>John Doe</Name>
    <Email>john.doe@example.com</Email>
    <RegistrationDate>2022-01-15</RegistrationDate>
  </Customer>
  ...
</BookStore>
```

Az XML dokumentum szerkezete lehetővé teszi több vevő, profil, rendelés, szerző, kiadó és könyv kezelését, valamint az adatok közötti kapcsolatok megjelenítését.

1.4 Az XML dokumentum alapján XMLSchema készítése

Az XML dokumentum szerkezetének validálásához elkészült az XMLSchema (lásd: `WO02D7_XMLSchema.xsd`), amely meghatározza az elemek típusait, kötelező és opcionális mezőket, valamint az adatok közötti kapcsolatokat. Az XMLSchema biztosítja, hogy az XML dokumentum megfeleljen a kívánt szerkezeti és tartalmi követelményeknek.

Az XMLSchema készítése során az XML dokumentum szerkezetét vettem alapul, és minden főbb elemhez (pl. Customer, Book, Order) létrehoztam a megfelelő komplex típusokat. Meghatároztam, hogy mely elemek kötelezőek, melyek ismétlődhetnek (pl. több email cím), és definiáltam az attribútumokat is.

Az adatok helyességének biztosítására egyedi típusokat (simpleType) is létrehoztam, például az email címekhez reguláris kifejezést használtam, hogy csak érvényes formátumú email címek legyenek elfogadva. Hasonló módon jártam el a telefonszámok, fizetési módok és státuszok esetében is, ahol szükséges volt.

Az XMLSchema-t többször teszteltem az XML dokumentumra, hogy minden szerkezeti és tartalmi követelmény teljesüljön, és a validáció hibamentes legyen.

```
<xs:element name="Customer" maxOccurs="unbounded">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Name" type="xs:string"/>
      <xs:element name="Email" type="EmailType" maxOccurs="unbounded"/>
      <xs:element name="RegistrationDate" type="xs:date"/>
    </xs:sequence>
    <xs:attribute name="CustomerId" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>

<xs:simpleType name="EmailType">
  <xs:restriction base="xs:string">
    <xs:pattern value="[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}"/>
  </xs:restriction>
</xs:simpleType>
```

Az XMLSchema részleteiben megtalálhatók az egyedi típusdefiníciók, mint például az email címek, telefonszámok, fizetési módok és státuszok mintái, amelyek biztosítják az adatok helyességét és konzisztenciáját.