1. A. Ön egy olyan vállalat informatikusa, amely több-felhasználós operációs rendszerre fejleszt alkalmazásokat. Az alkalmazás memóriaigénye miatt Önnek át kell alakítani a fizikai memória kiosztását. Ismertesse, hogy milyen memória definíciós eljárásokat ismer, és magyarázza el a statikus és dinamikus memória allokáció közötti különbségeket!

Információtartalom vázlata

– A memória definiálás lehetséges módszerei

– A memória allokálás technikái (bit térkép, memória ellenőrző blokk) – Allokációs stratégiák dinamikus allokációhoz

– A virtuálismemória-kezelés elve.

– A virtuális memória-kezelés alapproblémái (Betöltési, Elhelyezési, Helyettesítési és Visszatöltési problémák)

A memória definiálás lehetséges módszerei

1. **Statikus memória definiálás**:
    - Az alkalmazás indulásakor rögzített memóriaterületet rendel a programhoz.
    - A foglalás a program teljes futási ideje alatt fennáll.
    - Előny: Egyszerűbb implementáció és stabilitás.
    - Hátrány: Nem hatékony, ha a foglalt memóriaterület nincs teljesen kihasználva.
2. **Dinamikus memória definiálás**:
    - A memóriafoglalás a program futása közben történik igény szerint.
    - A foglalt memóriaterület felszabadítható, amikor már nincs rá szükség.
    - Előny: Hatékonyabb memóriafelhasználás.
    - Hátrány: Nagyobb kockázat a memória-kezelési hibákra, pl. szivárgásokra.

A memória allokálás technikái

1. **Bit térkép (Bitmap):**
    - A memória kisebb egységekre oszlik, amelyek egy biten keresztül vannak nyilvántartva.
    - A bit értéke (0 vagy 1) jelzi, hogy az adott egység foglalt vagy szabad.
    - Előny: Egyszerű szerkezet.
    - Hátrány: Lassú lehet nagy memória esetén.

2. **Memória ellenőrző blokk (Memory Control Block):**
    - A memóriaterületeket blokkokra bontják, amelyeket egy táblázatban tartanak nyilván.
    - A blokkok tartalmazzák a kezdőcímet és a méretet.
    - Előny: Hatékonyabb memória-kezelés nagyobb területek esetén.
    - Hátrány: Többlet adatok tárolása miatt némi memóriahasználat.

Allokációs stratégiák dinamikus allokációhoz

1. **Első találat (First Fit):**
    - Az első olyan szabad blokkot foglalja le, amely elég nagy az igényekhez.
    - Gyors, de töredezettséget okozhat.
2. **Legjobb illeszkedés (Best Fit):**
    - A legkisebb olyan blokkot választja, amely még megfelelő.
    - Csökkentheti a töredezettséget, de lassabb.
3. **Legrosszabb illeszkedés (Worst Fit):**
    - A legnagyobb szabad blokkot választja, hogy minél nagyobb szabad blokkok maradjanak.
    - Kevésbé hatékony hosszútávon.

A virtuális memória-kezelés elve

- A virtuális memória kezelése azt az illúziót kelti a felhasználó számára, mintha egy hatalmas, folyamatos memória állna rendelkezésre, miközben a fizikai memória (RAM) korlátozott lehet. Az operációs rendszer (OS) felelős a virtuális és fizikai címek közötti átalakításért, így a programok a virtuális címekkel dolgoznak, miközben az OS a megfelelő fizikai helyeket kezeli. Ez lehetővé teszi a programok számára, hogy a virtuális címeket kényelmesen válasszák ki, míg a fizikai címek dinamikusan, a futás során lesznek leképezve. A virtuális memória előnyei közé tartozik, hogy a virtuális és fizikai címek elkülönülnek, így a programok nem érzékelik a fizikai memória változásait, valamint a memóriavédelem is könnyebben megszervezhető, mivel a programok csak saját virtuális címterületükhöz férhetnek hozzá.
- **Virtuális memória:** Lehetőséget ad arra, hogy a programok több memóriát lássanak, mint amennyi fizikailag rendelkezésre áll.
- Az operációs rendszer a virtuális címeket fizikailag hozzárendeli a memória lapjaihoz.
- **Lapozás (Paging):** A memória kisebb egységekre (lapokra) bontva van kezelve, amelyeket szükség szerint cserél ki a fizikai memóriával.

A virtuális memória-kezelés alapproblémái

1. **Betöltési probléma (Loading):**
    - A program futásához szükséges lapokat kell betölteni a memóriába.

    - Kihívás: Gyors lapbetöltés biztosítása minimalizált késéssel.
2. **Elhelyezési probléma (Placement):**
    - A lapokat a fizikai memóriába kell helyezni.
    - Kihívás: Hatékony memóriahasználat biztosítása.
3. **Helyettesítési probléma (Replacement):**
    - Ha a memória tele van, egy létező lapot ki kell cserélni egy újra.
    - Stratlégiák: pl. FIFO, LRU (Legkevésbé használt).
4. **Visszatöltési probléma (Swapping):**
    - A korábban kivett lapokat újra be kell tölteni, ha szükségessé válnak.
    - Kihívás: Minimalizálni a visszatöltési időt.

A virtuális memória-kezelés alapproblémái 2.0

1. Betöltési probléma

- **Kérdés:** Mikor kell egy lapot betölteni a memóriába?
- A lapokat csak akkor töltjük be, amikor a processzornak szüksége van rá (lusta betöltés). Ez csökkenti a memória kihasználtságát és gyorsítja a program futását.

2. Elhelyezési probléma

- **Kérdés:** Hova kell a lap képét betölteni a memóriában?
- A rendszer eldönti, hogy a fizikai memóriában melyik keretben kapjon helyet az új lap.
- **Cél:** Minimalizálni a töredezettséget és gyorsítótárazási hatékonyságot elérni.

3. Helyettesítési probléma

- **Kérdés:** Ha nincs szabad memória, melyik lapot kell eltávolítani, hogy helyet biztosítsunk az újnak?
- **Cél:** Olyan lapot válasszunk, amely a legkisebb valószínűséggel lesz szükséges a közeljövőben.
- **Stratégiák:**
    - **Legrégebben használt (LRU):** Azt a lapot távolítjuk el, amelyhez a legrégebben nem nyúltunk.
    - **Legritkábban használt (LFU):** A legkevesebbszer használt lapot cseréljük ki.
    - **Első be, első ki (FIFO):** Az a lap kerül ki, amelyet legrégebben töltöttek be.
    - **Kombinált módszerek:** Az LRU és LFU kombinációi a hatékonyság növelése érdekében.

4. Visszatöltési probléma

- **Kérdés:** Mi történik, ha egy módosított lapot kell eltávolítani?
- **Módosított lapok:** Ezeket először vissza kell írni a háttértárra, ami időigényesebb.
- **Nem módosított lapok:** Ezek eltávolítása gyorsabb, mert nincs szükség írási műveletre.

- **Stratégia:** Először nem módosított lapokat választunk ki eltávolításra, és csak akkor nyúlunk a módosított lapokhoz, ha nincs más lehetőség.

Lokalitás elve

- **Lokalitás elve:** Az aktuális cím és a hozzá közeli címek a közeljövőben nagy valószínűséggel ismét hivatkozottak lesznek.
- **Szerző:** Bélády László, 1966.
- **Jelentősége:** Ez az elv a hatékony memória-kezelés alapja, mivel a memóriaműveletek optimalizálhatók azáltal, hogy a gyakran használt lapokat a memóriában tartjuk.
-

Összegzés

A memória allokáció és virtuális memória-kezelés megfelelő stratégiáinak kiválasztása kulcsfontosságú az alkalmazások hatékony futásához, különösen több-felhasználós operációs rendszerekben.

B. Ön részt vesz programozóként egy üzleti portál fejlesztésében. A portál egyik szolgáltatása az online értékesítés, amelyhez egy adatbázist kell kialakítani. A részletesen meghatározott követelmények és funkcionális elvárások alapján el kell készítenie a logikai adatmodellt. Fejtse ki az adatbázisok tervezésének módszereit és eszközeit!

Információtartalom vázlata

– Adatbázis-kezelés alapfogalmai

– Adatbázis kezelő rendszerek feladatai

– Adatmodell elemei és jellemzőik

– Egyed-Kapcsolat diagram

Adatbázis-kezelés alapfogalmai

Az adatbázisok olyan strukturált adattárolási rendszerek, amelyek lehetővé teszik nagy mennyiségű adat hatékony tárolását, kezelését és visszakeresését. Az adatbázis-kezelés célja az adatok rendszerezése, védelme, valamint az adatokhoz való gyors és egyszerű hozzáférés biztosítása.

Fontos alapfogalmak:

- **Adat**: Strukturált információ, amelyet egy rendszerben tárolnak.
- **Adatbázis (Database)**: Olyan rendszerezett adatgyűjtemény, amelyet az adatok egyszerű visszakeresése és kezelése érdekében hoznak létre.
- **Adatbázis rendszer (DBMS - Database Management System)**: Az adatbázis-kezelő rendszer az eszköz, amely lehetővé teszi az adatbázis létrehozását, karbantartását és kezelését.
- **Relációs adatbázis**: Olyan adatbázis, amely az adatokat táblákban tárolja, és ezek között relációkat (kapcsolatokat) hoz létre.
- **SQL (Structured Query Language)**: Az adatbázisok kezelésére használt szabványos lekérdező nyelv.
- **Információ: Az adatokat értelmezett formában tároló és felhasználható tudás.**
- **Meta Adat**: A metaadat olyan adat, amely más adat jellemzésére szolgál.
- **Séma**: Az adatbázis struktúrája, amely meghatározza a táblák és azok kapcsolatait.
- **Konzisztencia**: Az adatbázis állapotának helyessége és érvényessége tranzakciók után.
- **Adatintegritás**: Az adatok helyessége, megbízhatósága és épsége.
- **Hivatkozási integritás**: Az idegen kulcsok érvényessége, biztosítva, hogy a kapcsolatok mindig létező rekordokra hivatkoznak.
- **Redundancia**: Az adatok felesleges ismétlődése az adatbázisban.
- **Kulcs**: Az egyed olyan tulajdonsága amely egyértelműen azonosítja bármely egyed-előfordulást
- **Gyenge egyed**: Ha egy egyednek nincs kulcsa. Ekkor kell lenni egy másik egyednek, amely ezzel kapcsolatban van és annak a tulajdonságaival együtt már az első egyedben is létezik kulcs. Az ilyen kapcsolatot meghatározó kapcsolatnak nevezzük.

Adatbázis-kezelő rendszerek feladatai

Az adatbázis-kezelő rendszerek (DBMS) a következő kulcsfontosságú feladatokat látják el:

1. **Adatok tárolása és rendszerezése:**
    - Nagy mennyiségű adat strukturált tárolása táblák formájában.
2. **Adatbiztonság biztosítása:**
    - Jogosultságkezelés, az adatokhoz való hozzáférés szabályozása.
3. **Adatvédelem:**
    - Az adatok integritásának és helyességének biztosítása.
4. **Adatok visszakeresése és manipulálása:**
    - Lekérdezések (SELECT), módosítások (INSERT, UPDATE, DELETE).

5. **Többfelhasználós hozzáférés kezelése:**
    - Egyidejű hozzáférések támogatása.
6. **Mentés és visszaállítás:**
    - Adatvesztés elleni védelem, adatmentés és helyreállítás biztosítása.

Adatmodell elemei és jellemzőik

Az adatmodell az adatok logikai szerkezetét és kapcsolatait határozza meg. Az adatmodell segíti az adatok strukturálását az adatbázisban, és biztosítja a hatékony működést.

Főbb elemek:

1. Egyed(Entity)

- **Definíció**: Elkülöníthető egységek, vagy objektumok, amelyeket az adatbázisban modellezni szeretnénk.
- **Példa**: Hallgató

  Alkalmazott kifejezések:

    - **Egyedtípus**: Az egyedek absztrakciója, a különböző egyedek közös jellemzőinek összessége.
    - **Egyedhalmaz**: A konkrét egyedek összessége (pl. az összes hallgató).
    - **Egyed-előfordulás**: Egy konkrét egyed, például egy konkrét hallgató.

2. Tulajdonság (Attribútum)

- **Definíció**: Az egyedeket leíró jellemzők vagy tulajdonságok.
- **Példa**: Hallgató neve

  Alkalmazott kifejezések:

    - **Tulajdonságtípus**: Az attribútum absztrakt típusa.
    - **Tulajdonságérték**: A konkrét érték, amelyet az attribútum képvisel (pl. egy hallgató neve).

  Speciális tulajdonságok:

    - **Összetett tulajdonság**: Olyan tulajdonság, amely több részattribútumból áll. **Példa**: Lakcím (település, irányítószám stb.)
    - **Többértékű tulajdonság**: Olyan tulajdonság, amelynek több értéke lehet. **Példa**: Keresztnév (több keresztnevet is tartalmazhat egy személy).

3. Kapcsolat

- **Definíció**: Két vagy több egyed közötti összefüggést ír le.

Kapcsolatok típusai:

- **Multiplicitás szerint**:
  - **Egy-egy (1-1)**: Minden egyedhez legfeljebb egy pár tartozik a másik egyedből és fordítva. **Példa**: Hallgató és Felhasználó kapcsolata (egy hallgatónak egy felhasználója van).
  - **Egy-sok (1-N)**: Egy egyedhez több pár tartozhat, de fordítva egy párhoz csak egy egyed tartozhat. **Példa**: Kurzus és Terem kapcsolata (egy kurzust egy teremben tartanak, de egy teremben több kurzus is lehet).
  - **Sok-sok (N-M)**: Mindkét egyedhez több pár tartozhat. **Példa**: Hallgató és Képzés kapcsolata (egy hallgató több képzésen is részt vehet).
- **Résztvevő egyedek száma szerint**:
  - **Bináris kapcsolat**: Két egyed vesz részt a kapcsolatban. **Példa**: Hallgató és Képzés kapcsolata.
  - **Sokágú kapcsolat**: Kettőnél több egyed vesz részt. **Példa**: Hallgató, VizsgaAlkalom és Kurzus kapcsolata.
  - **Rekurzív kapcsolat**: Egy egyed többször vesz részt a kapcsolatban. **Példa**: Kurzus saját magával is lehet „előfeltétele" kapcsolatban
- **Kapcsolati szerep**: A kapcsolatban részt vevő egyedek szerepe, amely segít a kapcsolat értelmezésében. **Példa**: A Kurzus esetén az „előfeltétel" szerep.
- **Kapcsolat attribútuma**: A kapcsolathoz rendelt tulajdonságok. **Példa**: A Kurzus és Terem közötti „helyszíne" kapcsolat időpont attribútuma.

Adatmodell típusok:

1. **Hierarchikus adatmodell:** Az adatok hierarchikus (fa) szerkezetben vannak szervezve.
2. **Hálós adatmodell:** Az adatok összetett, hálószerű kapcsolatokat alkotnak.
3. **Relációs adatmodell:** Az adatok táblákban vannak szervezve, amelyek között relációk léteznek.
4. **Objektum-orientált adatmodell:** Az adatok objektumként vannak ábrázolva, amelyek tulajdonságokat és viselkedéseket is tartalmaznak.

Egyed-Kapcsolat diagram (Entity-Relationship Diagram, ERD)

Az Egyed-Kapcsolat diagram egy grafikus módszer az adatbázis logikai szerkezetének ábrázolására. Az ERD az adatmodellezés egyik alapvető eszköze, amely segíti az adatbázis-tervezők és a fejlesztők közötti kommunikációt.

ERD elemei:

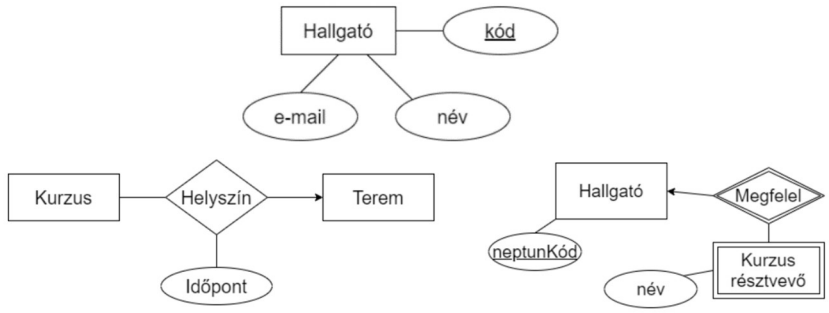
1. **Egyedek (Entities):** Téglalapokkal ábrázolva. Például:

- `Ügyfél`, `Rendelés`, `Termék`.

2. **Kapcsolatok (Relationships):** Rombuszokkal ábrázolva, amelyek összekötik az egyedeket. Például:
   - `Rendel`, `Tartalmaz`.
3. **Attribútumok (Attributes):** Ellipszisekkel ábrázolva, amelyek az egyedekhez vagy kapcsolatokhoz tartoznak. Például:
   - `Ügyfél neve`, `Rendelés dátuma`.

Példa ERD felépítésére:

- Egy online értékesítési adatbázis esetén:
  - Egyedek: `Ügyfél`, `Rendelés`, `Termék`.
  - Kapcsolatok:
    - Az `Ügyfél` kapcsolatban áll a `Rendelés`-sel (1:N kapcsolat).
    - A `Rendelés` kapcsolatban áll a `Termék`-kel (N:M kapcsolat).
  - Attribútumok:
    - `Ügyfél`: `ÜgyfélID`, `Név`, `Email`.
    - `Rendelés`: `RendelésID`, `Dátum`.
    - `Termék`: `TermékID`, `Név`, `Ár`.



Az adatbázisok tervezése az adatmodellezési módszerek (pl. ERD) használatán alapszik. Az adatbázis-tervezés során fontos az igények pontos meghatározása, a megfelelő adatmodell kiválasztása, és az adatkapcsolatok átgondolt kialakítása a hatékony működés érdekében.