12. Tétel

a) Adatbázisrendszerek II.

A PL/SQL alapjai: típusok, változók, konstansok, vezérlési szerkezetek

- A PL/SQL segítségével összetett eljárásokat és adatbázis-funkciókat lehet megvalósítani (Procedural Language)

- Típusok: előre definiált és felhasználó által létrehozott

- pl. INTEGER, VARCHAR2 vagy RECORD, OBJECT

- Változók: DECLARE kulcsszóval pl. x INTEGER := 10;

- Konstansok: CONSTANT kulcsszóval pl. PI CONSTANT NUMBER := 3.14159;

- Vezérlési szerkezetek:

- IF THEN ELSE (pl. IF condition THEN statement1; ELSIF .... ELSE END

- CASE (pl. CASE variable WHEN value1 THEN statement1; ... ELSE statement3; végén END CASE;

- LOOP: egy végtelen ciklust indít, amely addig fut, amíg ki nem lépünk belőle (pl. LOOP EXIT WHEN END LOOP) - do while-nak felel meg

- FOR LOOP: meghatározott számú ismétlést hajt végre (pl. FOR i IN 1..10 LOOP END LOOP;)

- WHILE LOOP: elől tesztelős WHILE ciklus

SQL utasítások elhelyezésének és használatának lehetőségei a PL/SQL-ben

- Ismétlés:

- DML (adatok: beszúrás, frissítés, törlés, lekérdezés)

- DDL (adatbázis objektumokat lehet létrehozni, módosítani vagy törölni)

- DQL (csak lekérdezés adatbázisból)

- DCL (hozzáférést vagy jogosultságot kezelni az adatbázisba)

- BEGIN END blokk közé (pl. DBMS\_OUTPUT.PUT\_LINE(')

- Cursorok: lekérdezési eredmények sorainak kezelésére

- Implicit kurzor: az Oracle automatikusan kezeli, nem kell explicit módon deklarálni vagy megnyitni, akkor használjuk amikor csak egyetlen sorra vagy adatblokkra van szükségünk

- Explicit kurzor: deklarálni kell, megnyitni, kezelni és bezárni, általában akkor hasznosak ha egy lekérdezés több soros eredményt ad vissza, amelyeket soronként kell feldolgozni

- létrehozása: CURSOR employee\_cursor IS -> itt valamilyen SELECT

- megnyitása: OPEN

- általában a cursor rekordjait FETCH-eljük valamilyen LOOP-on keresztül, a CURSORT a program végén mindig le kell zárni CLOSE

- kurzor attribútumok:

- %FOUND: Igaz, ha a FETCH sikeresen beolvasott egy sort

- %NOTFOUND

- %ROWCOUNT: a beolvasott sorok számát adja vissza

- %ISOPEN: igaz ha a kurzor nyitva van

A PL/SQL program felépítése

- DECLARE, BEGIN, END

- Alprogramok: különálló egységet képeznek és újrafelhasználhatóak

- Típusai: Eljárások (PROCEDURE) és Függvények (FUNCTION)

- CREATE [OR REPLACE] PROCEDURE procedure\_name -> új sorba IS (ide jönnek a deklarálások -> BEGIN aztán végén END procedure name;

- Függvénynél ugyanez csak FUNCTION és az IS felett RETURN return\_datatype, majd a BEGIN blokkba lesz RETURN return\_value és END function\_name;

Tárolt eljárások, tárolt függvények összehasonlítása, csomagok készítésének célja és lehetőségei

- tárolt: ha az adatbázisban vannak definiálva és ott is futnak.

- PROCEDURE: általában adatmódosító műveleteket, tranzakciókezelést, adatbetöltést stb. (adatmanipuláció után COMMIT;)

- FUNCTION: lekérdezésket, összetett számításokat és érték kalkulációkat hajtanak végre (RETURN)

- Triggerek: Olyan tárolt eljárások, amelyek automatikusan futnak, amikor bizonyos események bekövetkeznek egy táblában (pl. INSERT, UPDATE, DELETE).

- csomagkészítés segítségével összefoghatjuk és rendszerezhetjük az adatbázis objektumainkat,

újrafelhasználható, hatékony, strukturáltság

- máshogy megfogalmazva: logikailag összefüggő tárolt eljárások, függvények, típusok és változók gyűjteménye. Két részből áll: a specifikációból és a törzsből.

- AS kulcsszó a PROCEDURE neve után, és IS kulcsszó minden mindkettőnél kidolgozás előtt

A tárolt alprogramok paraméterezési lehetőségei, az egyes lehetőségek jellemzői

- IN: alprogram bemenő paramétere, csak olvasható az alprogramban (bemeneti értékek átadásánál)

- OUT: kimenő, az alprogram módosíthatja (visszatérési értékeknél vagy hibaüzeneteknél)

- IN-OUT: bemenő és kimenő paraméterként is használható (értékek kiszámítása és visszaadásánál)

- DEFAULT: alapértelmezett értéket adhatunk meg vele

Kivételek kezelés PL/SQL-ben

- ugyanúgy hibakezelésre

- DECLARE blokkba EXCEPTION részbe írjuk

- sajátot is írhatunk, de vannak beépítettek pl. NO\_DATA\_FOUND, TOO\_MANY\_ROWS, ZERO\_DIVIDE

- kivételt kézi kiváltásra a RAISE utasítással dobhatunk

b) Bevezetés a számítógépi grafikába

Raszteres algoritmusok: DDA és MidPoint szakaszrajzoló

- Raszteres:

- sorokból (scanlines) és oszlopokból (pixelekből) rendezett képpontokból áll

- egy pixelnek ismerjük a koordinátáját és a színét

- rugalmatlan, gyenge képnagyítás, lépcsőhatás

- DDA:

- Lényege, hogy dx és dy értéke a nagyobb elmozdulás irányában 1 legyen

- probléma: valós aritmetikát és kerekítést alkalmaz

- Midpoint:

- Bresenham nevéhez

- javítja a DDA hátrányát -> egész számokat használ

- d döntési változó bevezetése, ha d>=0 akkor d\_NE, ha kisebb akkor d\_E

- két y koordináta között megkeresni melyik van kisebb távolságra az egyenestől

MidPoint körrajzoló algoritmusok

- félkör -> negyed -> nyolcad tükrözéssel

- A hosszúság kiszámításnál valós értékekkel számol amit ki lehet javítani programtranszformációval

- a D döntésiváltozót H döntésiváltozóra cseréljük D-1/4 -> 1 - R

Cohen-Sutherland vágó algoritmus

- Cohen és Sutherlandről kapta a nevét, egy repülőgép szimulátorhoz kellett a vágás

- egy meghatározott ablakban szakaszok megjelenítése

- A síkot 9 egyenlő részre osztjuk a téglalap segítségével: 2db vízszintes és 2db függőleges vonallal

- Majd ezeket 4 bites kódokkal látjuk el:

- 1. bit 1 ha a felső vízszintes felett van (0)

- 2. bit 1 ha az alsó vízszintes alatt van

- 3. bit 1 ha a jobb függőlegestól jobbra van

- 4. bit 1 ha a bal függőlegestől balra van

- Ha a szakasz két végpontja 0000 akkor nincs szükség vágásra, megrajzolhatjuk a szakaszt

- Ha mindkettő azonos helyen 1-es akkor azért nincs szükség vágásra mert nincs a megjelenített képernyőn

- Ha különböző helyen 1-es, akkor vágásra van szükség

- külön számítás vízszintes és függőleges élek esetén

- véges sok lépéssel folytatjuk

- Több dimenzióra pl.: 3 -nál téglatesttel osztjuk fel 27 egyenlő részre és 6 bites kódok

Paraméteres görbék

- A síkban mozgó pont egy görbét ír le

Ha minden t időpillanatban meghúzzuk az op vektort, és ezt r(t)-vel

jelöljük, akkor I intervallumon értelmezett vektorfüggvényt kapunk.

- síkban és térben (x,y,z)

- megjelenítése egy görbének törött vonalakkal történik

Hermit ív, Bézier görbe, B-Spline

- Hermit ív

- kezdő és végpont (P0, P1)

- kezdő és végérintő (P0ˇ, P1ˇ)

- Célunk leírni az s(t) görbét a kontroll alakzatok és bizonyos bázisfüggvényei lineáris kombinációjaként

- bázis(1 t, t^2, t^3) -> függvényteret keressük

- Ahhoz hogy hullám legyen a görbében akkor a hermit íveket csatlakoztatni kell egymáshoz

- Bézier görbe

- hermite-ív rossz tulajdonságai:

- Az érintőkkel való definiálás körülményes

- Bizonyos helyzetekben a görbe metszi az értintőt

- Bizonyos helyzetekben a görbe metszi önmagát

- Célunk egy olyan görbe előállítása, mely a síkban (vagy térben) elhelyezett kontrollpontok által meghatározott kontrollpoligon által lesz definiálva.

- Renault és Citroen egymástól függetlenül alkották meg

- van egy mátrixos megoldás

- de casetljou algoritmus megoldása: geometriai adatok alapján, rekurzívan definiálja a görbét -

- B-Spline

- Bézier problémája, hogy a magas fokszám rugalmatlanná teszi a görbét, azaz nem követi kellőképpen a kontrollpoligont.

- Emellett nagy mennyiségű kontrollpont esetén a számítások erőforrásigénye is megnő.

- megoldás Spline görbék alkalmazása

- lényege, hogy alacsony fokszámú görbéket csatlakoztatunk egymáshoz, amelyek automatikusan folytonosan csatlakoznak

- görbe kirajzolása után azt látjuk, hogy a görbe nem érinti a végpontokat

- A görbék automatikusan csatlakoznak

- Nem látjuk a csatlakozási pontot, tehát a folytonosság automatikusan biztosított

- van Zárt B-Spline: a görbe a kontrollpontok konvex burkán belül marad (szemüveg)

Pont-transzformációk síkban, térben, homogén koordináták

- Egybevágósági transzformáció (eltolás, forgatás, tükrözés x tengely körül)

- Hasonlósági transzformáció (origó középpontú kicsinyítés és nagyítás)

- Affin transzformáció (skálázás, nyírás)

- Ezek mind geometriai transzformációk

- Windows-Viewport transzformáció:

- lényege, hogy egy a képsíkon lévő ablakot transzformál át egy másik, a képernyőn megjelenítendő ablakba

Tér leképezése síkra: párhuzamos és centrális vetítés, axonometria

Térbeli objektumok megjelenítése képernyőn

- egy 'n' dimenziós objektumot nála alacsonyabb dimenzióba transzformál

- Descartes-féle koordinátarendszer: R^3 -> R^2 transzformáció

- képsík [x,y]

- Párhuzamos vetítés: Térbeli pontokat párhuzamos egyenesek mentén vetítjük a síkra, méret- és formatorzítás nélkül.

- Centrális vetítés: Térbeli pontokat egy középpontból kiinduló egyenesek mentén vetítjük a síkra, távolságtól függő torzítással.

- Axonometria: Térbeli objektumokat három koordinátatengely mentén vetítjük a síkra, torzításmentes, de nem perspektivikus ábrázolásban.

Poliéderek megadása: Wire Frame modell, B-Rep adatstruktúra

- Wire Frame modell: csak csúcsokat és az összekötő éleket tartalmazza a poliédernek

- Boundry Representation:

- adatstruktúrát használ:

- geometriai (csúcspontok koordinátái, élek & lapok egyenlete)

- topológiai (lapok, élek és csúcspontok kapcsolata)

Poliéderek megjelenítése: hátsó lapok eltávolítása, Z-Buffer algoritmus. Flat-, Gouraud- és Phong árnyalás

- Zárt poliéderek megjelenítésénél -> nem látható lapok eltávolítása

- hogyan? normálvektor előállításával + skaláris szorzattal

- Z-Buffer algoritmus: Mélységértékek tárolása és összehasonlítása a pixel-színek megjelenítésére.

- Konstans árnyalás (flat-shading):

- minden lap egy pontjában kiszámítjuk a színértéket és színezünk

- ugrásszerű változások a színekben (megoldás: közös csúcspont normálisának átlaga)

- Gouraud árnyalás:

- háromszöglap minden csúcsában számítjuk a színinformációt (normálisokkal)

- élein csúcspontok színének vesszük

- eredmény: körvonal nem sima (látszik a közelítés) + Mach-sáv hatás

- Phong árnyalás

- színek helyett normálisokat interpolál

- probléma: képhatár (finomítással lehetséges megoldani)