1. Operácie relačný algebry a ich mapovanie do SQL

(σ) - SELEKCIA - $\sigma_p(r)$ - vyberie n-tice, ktoré spĺňajú predikát.

(∏) - PROJEKCIA - $\prod_{A1, A2, A3}(r)$ - Vyberie stĺpce, ktoré spĺňajú predikát, duplikáty sú automaticky odstránené.

(U) - ZJEDNOTENIE - r U s - Spojí dané relácia za podmienok, že majú rovnaký počet atribútov a domény atribútov sú kompatibilné. Duplikáty sú automaticky odstránené.

(-) - ROZDIEL - r - s - Nájde n-tice, ktoré sa nachádzajú v (r) ale nenachádzajú sa v (s).

(X) - KARTEZIÁNSKY SÚČIN - r X s - Kombinuje informácie (r) a (s).

(ρ) - PREMENOVANIE - $\rho_x(E)$ - Výsledok výrazu E bude mať meno x.

2. Integritné obmedzenie, súvislosti s transakčním spracovaním, indexy na cudzích kľúčoch

Integritné ombedzenia zachovávajú databázu v konzistentom stave.

ACID transakcia:

- Atomic - Buď prejde celá alebo sa odvolá
- Consistent - Na konci nie je porušené žiadne obmedzenie
- Isolated - Operácie sú izolované od ostatných transakcií
- Durable - Po ukončení sú dáta uložené trvalo

NOT NULL - Nesmie byť prázdne.

UNIQUE - Stĺpec ma unikátne hodnoty v celej tabuľke.

PRIMARY KEY - Primárny kľúč tabuľky, je unikátny a jednoznačne určuje n-ticu.

REFERENCES - Hodnota stĺpca je hodnotou primárneho kľúča inej tabuľky..

CHECK - Kontrola, či vkladaná/modifikovaná hodnota spĺňa predikát.

FOREIGN KEY - využíva references. teda odkazuje na hodnotu v inej tabuľke.

3. Užívateľské dátové typy, dedičnosť, možnosti prístupu, možnosti indexovania.

Index - štruktúra nad stĺpcami tabuľky, ktorá pomáha k efektívnejšiemu prístupu na jej riadky v prípade, že bol použitý “uplatiteľný” predikát (=, <, >, BETWEEN). Pri jednom prechode tabuľkou je použitý maximálne jeden index. RDBMS prechádza primárne indexovou štruktúrov a na základe nej získava odkazy na riadky.

Užívateľské dátové typy(ADT) - Podobne ako v objektovom programovaní, sú to objekty s určitými atribútmi.

Dedičnosť - Viacnásobná/Jeden nadradený typ. Všetky atribúty dedí z nadradeného typu + pridáva ďalšie, ktoré sa vyššie už nenachádzajú. ORACLE nepodporuje viacnásobnú.

Možnosti prístupu - Obecne sa nedá pristupovať na celý ADT, lebo nie všetky rozhrania ho podporujú, preto vyberáme len jeho jednotlivé složky.

Možnosti indexovania - Neindexuje sa štandardne, ale používajú sa deterministické funkcie, ktoré ako argument berú ADT a vracajú “indexovateľný” typ.

4. Indexovacia metóda B-Tree, typy predikátov, ktoré efektívne vyhodnocuje.

Je základná vyhľadávacia metóda implementovaná vo všetkých DB strojoch. Je to stromová štruktúra, v ktorej uzly uchovávajú viac kľúčov z úplne usporiadanej množiny s týmito vlastnosťami:

- Každý uzol má maximálne (n) potomkov.
- Každý uzol, s výnimkou koreňa a listov, minimálne (n/2) potomkov.
- Koreň má minimálne 2 potomkov, ak nie je list.
- Všetky listy sú na rovnakej úrovni
- Nelistový uzol s (k) potomkami obsahuje (k-1) kľúčov.
- Pre kľúče (key.i až key.k) platí, že sú usporiadané vzostupne
- ukazateľ p.i ukazuje na uzol, ktorého všetky kľúče sú z intervalu (key.i, key.i+1)

Efektívne využitie má napr. pri hľadaní prvku s konkrétnou hodnotou alebo hľadanie prvku v intervale hodnôt.

5. Indexovacia metóda R-Tree, typy predikátov, ktoré efektívne vyhodnocuje.

Analógia k B-tree, kľúče sú 2D(obecne nD) obdĺžniky.

- (M) maximálny počet kľúčov v uzle.
- (m <= M/2) minimálny počet kľúčov v uzle

Definícia:

- Každý uzol obsahuje minimálne (m) a maximálne (M) kľúčov, pokiaľ nie je koreň.
- Kľúče v R-Tree sú obdĺžniky s ukazateľmi na potomkovské uzly, v listoch obdĺžniky.
- Pre potomkovské uzly platí, že ich kľúče, sú vo vnútri rodičovského obdĺžniku
- Listy stromu majú rovnakú úroveň.
- Koreň obsahuje minimálne 2 kľúče pokiaľ nie je list.

Efektívne využitie má pri viac dimenzionálnych dátach.

6. Indexovacia metóda, GIST, jej abstraktné metódy.

GIST - Generalized index search tree, je zobecnením, predchádzajúcich metód, stromová štruktúra má rovnakú organizáciu ako R-Tree, nachádza sa napr. v PosgreSQL.

- Uzly sú tvorené n-ticami tvaru (key, nextNode), kde key je instancia ľubovoľného typu a nextNode je nasledujúci uzol (môže byť NULL pre list).
- Uzol obsahuje maximálne (N) kľúčov a keď nie je koreň minimálne (M<=N/2) kľúčov.

Jedná sa o akúkoľvek triedu, obsahujúcu definíciu kľúčov a operátorov, ktorá implementuje nasledujúce metódy.

- Consistent(key.i, query) - Vracia False pokiaľ je zaručené, že query na danom kľúči, nemôže byť pravdivý.
- Union{key.i} - vracia zjednotenie KEY, pre ktoré platí Consistent(key.i, q) -> Consistent(KEY, q) pre každý kľúč key.i a pre ľubovoľný možný dotaz q.
- Penalty(key.1, key.2) - vracia mieru rozšírenia key.1 o key.2
- PickSplit(node) - rozdelí sadu kľúčov daného uzlu na dve sady K1 a K2, tak aby pre všetky možné dotazy q v čo najväčšej miere: Consistent(uK1, q) vylučovala pravdivosť Consistent(uK2, q) a symetricky v R-Tree je mierou obsah prieniku obdĺžnikov (uK1)n(uK2).
- Same - operátor rovnosti
- Compress/Decompress - binárna podoba uzla vo file systéme.

Algoritmus Search:
vstup: Search(Node, query)
výstup: n-tica všetkých kĺúčov, pre ktoré platí Consistent(key.i, query)

1. Ak je Node list, daj na výstup všetky key.i, pre ktoré platí Consistent(key.i, query).
2. Ak nie je Node list, prejde algoritmom všetkých potomkov určené uzlami key.i, pre ktoré platí Consistent(key.i, query).

Algoritmus Insert:
vstup: Insert(Node, key)

1. Ak Node nie je list:
    a. Vyber z Node taký kľúč key.i, pre ktorý je Penalty(key.i, key) minimálny.
    b. Polož key.i = Union({key.i, key})
    c. Algoritmus na následníka key.i
2. Ak Node je list vlož key do Node.
    a. Ak nie je prekročená maximálna kapacita uzlu v GiST potom koniec inak b)
    b. Neck K1 a K2 sú dve sady kľúčov z PickSplit(Node). Vytvor nové uzly node1 a node2 z K1 a K2.
        i. Ak Node nie je koreň, nech P je rodičovský uzol Node, inak vytvor nový prázdny koreň P.
        ii. Odstráň z P ten kľúč, ktorého následníkom je Node a vlož do neho kľúče (Union(K1), node1), (Union(K2), node2) a opakuj krok 2.a) pre uzol P.

Indexy tohoto typu sú využiteľné napr. kockových n-dim dotazoch.

7. Techniky spojovania tabuliek (inner/outer join), exekučná stratégia prechádzania

Inner Join - Spája tabuľky na základe spoločného indexu, vracia len spoločné hodnoty
Outer Join - Spája tabuľky na základe spoločného indexu, vracia z prvej tabuľky všetko a z druhej iba spoločné hodnoty. Robí sa pomocou (+) operátora vo WHERE klauzule.

- Nested loops (vnorené cykly):
    - Prechádza sa celá "vonkajšia" tabuľka a hľadá sa zhodný kľúč v tabuľke "vnútornej". Podľa existencie indexu vo vnútornej tabuľke sa hľadá zhodný riadok podľa indexu, alebo opäť plným priechodom.
- Sort merge join:
    - Usporiada obe tabuľky (vrátené riadky) podľa kľúča, ktorým tabuľky spájame, v prípade existencie indexu sa použije vhodný index. Potom prechádza obe usporiadané tabuľky a vracia kombináciu riadkov s rovnakými kľúčmi.
- Hash join:
    - Pre menšiu z tabuliek sa z kľúčov vytvorí hash tabuľka. Potom sa prechádza väčšia tabuľka a hľadá sa zhodný kľúč v hash tabuľke.

V praxi sa Oracle rozhoduje, ktorú funkciu použiť sám na základe obsahu a frekventovanosti kľúča.

8. Objekty VIEW, DML príkazy na VIEW, materializované VIEW.

View je dočasná tabuľka, na umožnenie lepšej práce s dátami.
SELECT - Rovnako ako s tabuľkami.
UPDATE - Všetky stĺpce sú jednoznačne priradene key-preserved tabuľkám, inak povedané, takým tabuľkám ktorých primárny kľúč je zároveň kľúčom vo view, príkaz mení riadky práve jednej tabuľky.
DELETE - riadky View odkazujú práve na jednu key-preserved tabuľku, z ktorej sú riadky zmazané
INSERT - Nesmie sa odvolávať na riadky patriace non-key-preserved tabuľky. Všetky vkladané stĺpce patria práve jednej key-preserved tabuľke.
Materializované view sú uložené výsledky SELECT dotazov, narozdiel od view dáta su reálne uložené a je možné ich obnovovať.

9. Where klauzula, agregačné funkcie, having fráza. Základné stratégie vyhodnotenia SQL dotazu.

Vyhodnotenie SQL dotazu:

1. Ak obsahuje SELECT WHERE klauzulu spracujú sa iba riadky, ktoré vyhovujú WHERE
2. Ak obsahuje SELECT GROUP BY klauzulu, vytvárajú sa skupiny podľa výrazov GROUP BY
3. Ak obsahuje SELECT HAVING klauzulu, potom sú vyradené tie skupiny, ktoré nespĺňajú HAVING

WHERE klauzula - realizuje operáciu SELEKCIE z relačnej algebry, Aj pre WHERE klauzulu platí pravidlo, že ľubovoľný výraz sa dá v SQL nahradiť jedno-riadkovým, jedno-stĺpcovým SELECT príkazom.

HAVING klauzula - používa sa k obmedzeniu výstupu na základe skupinových funkcií.

10. Transakčné príkazy, úrovne izolácie, odložené integritné obmedzenia, deadlock (príklad).

Transakčné príkazy sú - COMMIT, ROLLBACK (TO SAVEPOINT)
Úrovne izolácie:

- SERIALIZABLE - 2 transakcie menia jeden riadok, tak prvá prejde a druhá skončí s chybou
- READ COMMITTED - 2 transakcie menia jeden riadok, druhá počká, kým sa vykoná prvá(môže nastať deadlock)

odložené integritné obmedzenia - INITIALLY DEFERRED kontroluje sa až po commite.
deadlock:

```
CREATE TABLE I1 (cislo INT, slovo VARCHAR2(64), CONSTRAINT I1_PK PRIMARY KEY (cislo));
        INSERT INTO I1 VALUES (1,'A');
        INSERT INTO I1 VALUES (2,'B');
        COMMIT;

        sezení 1 - UPDATE I1 SET slovo='C' WHERE cislo=1;[OK]
        sezení 2 - UPDATE I1 SET slovo='D' WHERE cislo=2;[OK]
        sezení 1 - UPDATE I1 SET slovo='E' WHERE cislo=2;[OK - čeká] -- ##
        sezení 2 - UPDATE I1 SET slovo='F' WHERE cislo=1;[OK - čeká] -- DEADLOCK
```

11. Triggery, ich typy, spúštacie udalosti, zakázané príkazy SQL vnútri triggerov.

Trigger(spúštač) je procedúra, ktorá sa spustí pri výrazoch INSERT, UPDATE, DELETE, využíva sa na dynamickú úpravu tabuliek. Vnútri triggeru sa nesmú nachádzať transakčné príkazy COMMIT a ROLLBACK. Teoreticky by sa tam mohli vyskytnúť iba v prípade, že sa ku nim nikdy nedostaneme.

12. SQL injekcie a obrana proti nim.

Je nežiadúci vpád do DB útočníkom, s cieľom poškodiť alebo odcudziť dáta. Najlepší spôsob obrany, je zabezpečiť si všetky queries, takým spôsobom, aby do nich útočník nedokázal vložiť nepovolené znaky.