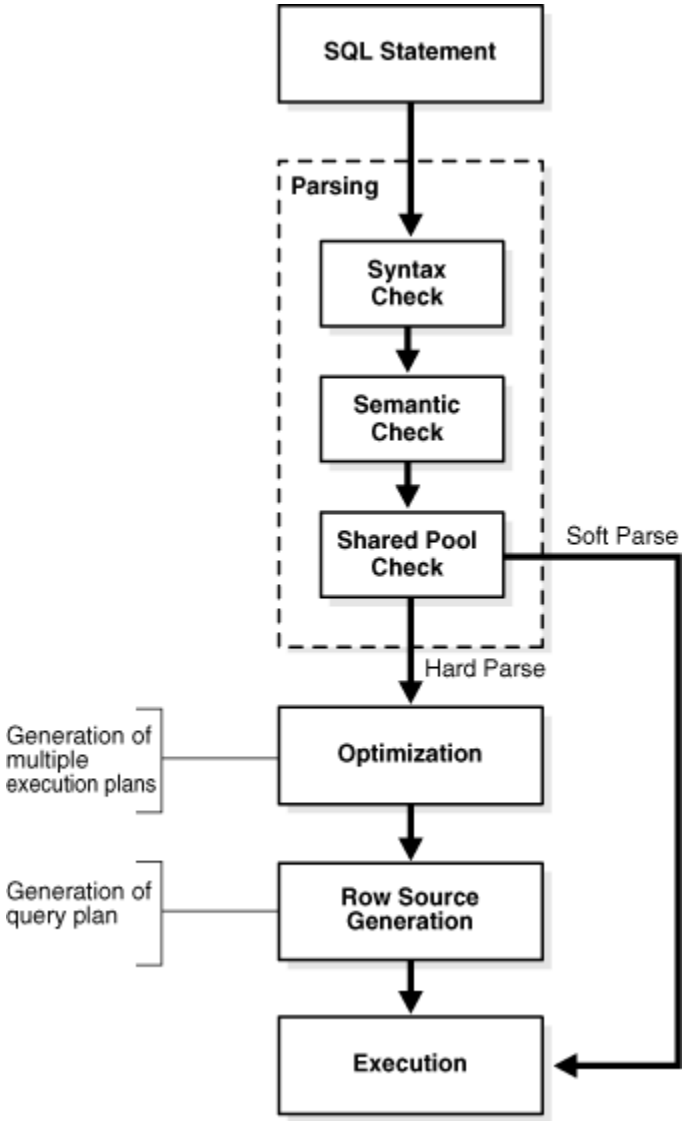

ELTE | IK

INFORMATIKAI KAR

# Adatbázisok 2.

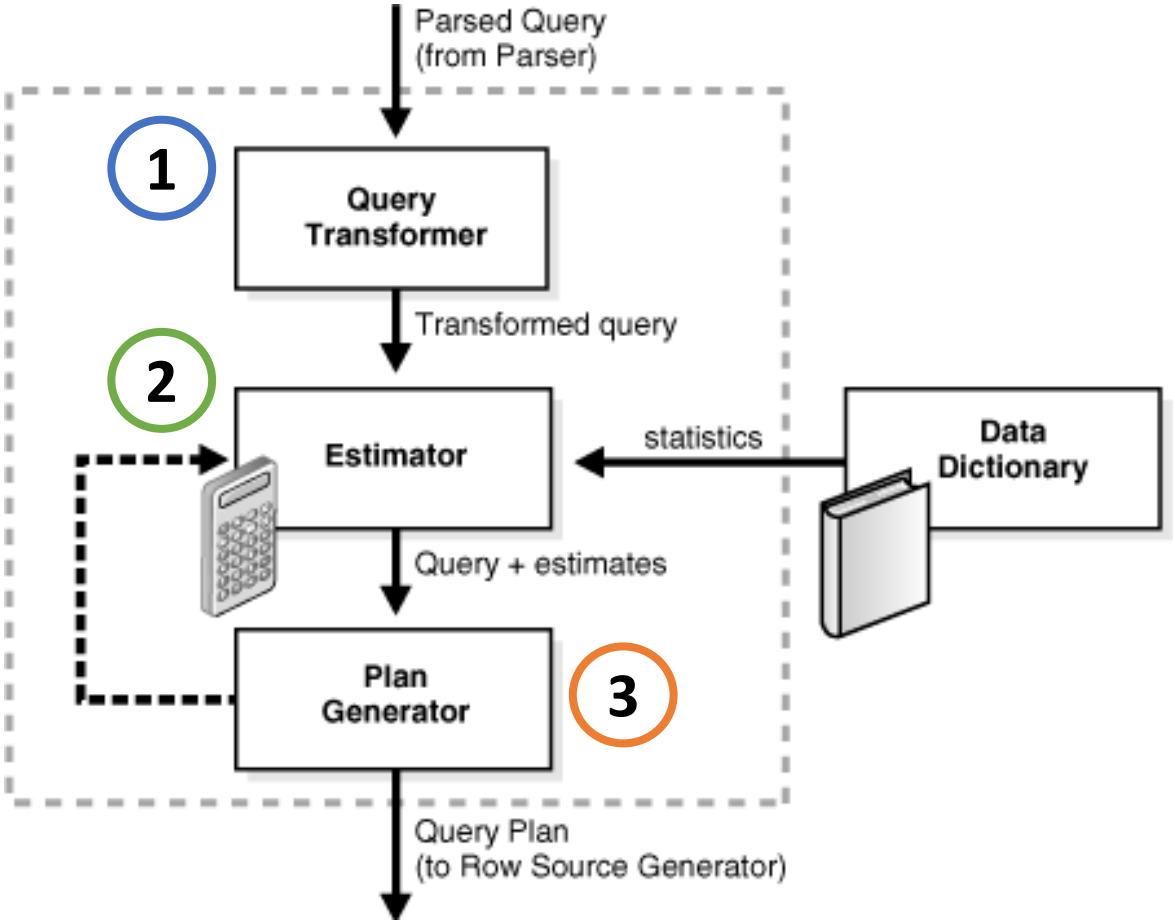
8. gyakorlat

# SQL utasítás végrehajtása

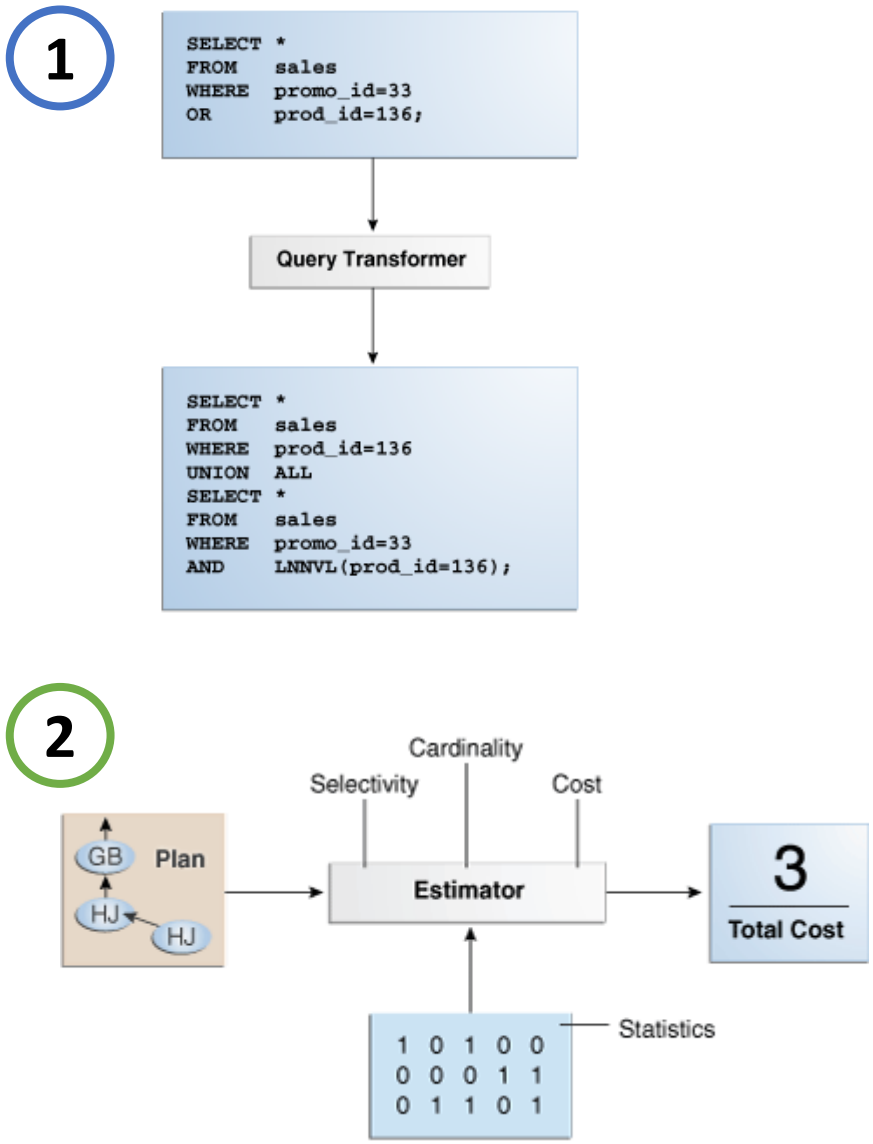



1. SQL utasítás
2. Szintaktikai ellenőrzés
   (jól írtuk a kulcsszavakat?)
3. Szemantikai ellenőrzés
   (van értelme, léteznek a táblák?)
4. Létezik ehhez a lekérdezéshez terv?
   (ha igen, hajtsuk végre)
5. Végrehajtási tervek elkészítése
   (optimalizáció, kiválasztás)
6. Lekérdezési terv elkészítése
   (iteratív program)
7. Végrehajtás

# Optimalizáció lépései



# Optimalizáció lépései




ELTE | IK

# Join végrehajtása: nested loop

- Két ciklus van: külső ciklus az egyikhez, belső ciklus a másikhoz.
- Használat: kis táblákon működik jól, ha van index a join attribútumra.

```
1 FOR erow IN (select * from employees where X=Y) LOOP
2   FOR drow IN (select * from departments where erow is matched) LOOP
3     output values from erow and drow
4   END LOOP
5 END LOOP
```

# Join végrehajtása: hash join

- A kisebb tábla join attribútumára hashmapet készít. Végigmegy a nagyobb tábla sorain, hasheli az attribútumot, és ha létezik akkor összekapcsolja.
- Használat: nagy táblákon is jól működik, de csak összekapcsolásnál.
- Nagyon hatékony ha a kisebb tábla befér a memóriába.

```
1  FOR small_table_row IN (SELECT * FROM small_table)
2  LOOP
3    slot_number := HASH(small_table_row.join_key);
4    INSERT_HASH_TABLE(slot_number,small_table_row);
5  END LOOP;
6
7  FOR large_table_row IN (SELECT * FROM large_table)
8  LOOP
9     slot_number := HASH(large_table_row.join_key);
10    small_table_row = LOOKUP_HASH_TABLE(slot_number,large_table_row.join_key);
11    IF small_table_row FOUND
12    THEN
13       output small_table_row + large_table_row;
14    END IF;
15 END LOOP;
```

# Join végrehajtása: sort merge join

- Ha nem rendezett a két tábla előbb rendezi őket
- Utána nested loop-hoz hasonlít, de átugorhat sorokat
- Használat: Nem csak egyenlőségvizsgálatnál használathó. Ha korábban már volt rendezés (vagy rendezve tárolunk) nem annyira költséges.

```
 1 READ data_set_1 SORT BY JOIN KEY TO temp_ds1
 2 READ data_set_2 SORT BY JOIN KEY TO temp_ds2
 3
 4 READ ds1_row FROM temp_ds1
 5 READ ds2_row FROM temp_ds2
 6
 7 WHILE NOT eof ON temp_ds1,temp_ds2
 8 LOOP
 9     IF ( temp_ds1.key = temp_ds2.key ) OUTPUT JOIN ds1_row,ds2_row
10     ELSIF ( temp_ds1.key <= temp_ds2.key ) READ ds1_row FROM temp_ds1
11     ELSIF ( temp_ds1.key => temp_ds2.key ) READ ds2_row FROM temp_ds2
12 END LOOP
```

# Tábla elérés módjai

| Access Path | Heap-Organized Tables | B-Tree Indexes and IOTs | Bitmap Indexes |
|---|---|---|---|
| Full Table Scans | x | | |
| Table Access by Rowid | x | | |
| Sample Table Scans | x | | |
| Index Unique Scans | | x | |
| Index Range Scans | | x | |
| Index Full Scans | | x | |
| Index Fast Full Scans | | x | |
| Index Skip Scans | | x | |
| Index Join Scans | | x | |
| Bitmap Index Single Value | | | x |
| Bitmap Index Range Scans | | | x |
| Bitmap Merge | | | x |
| Bitmap Index Range Scans | | | x |