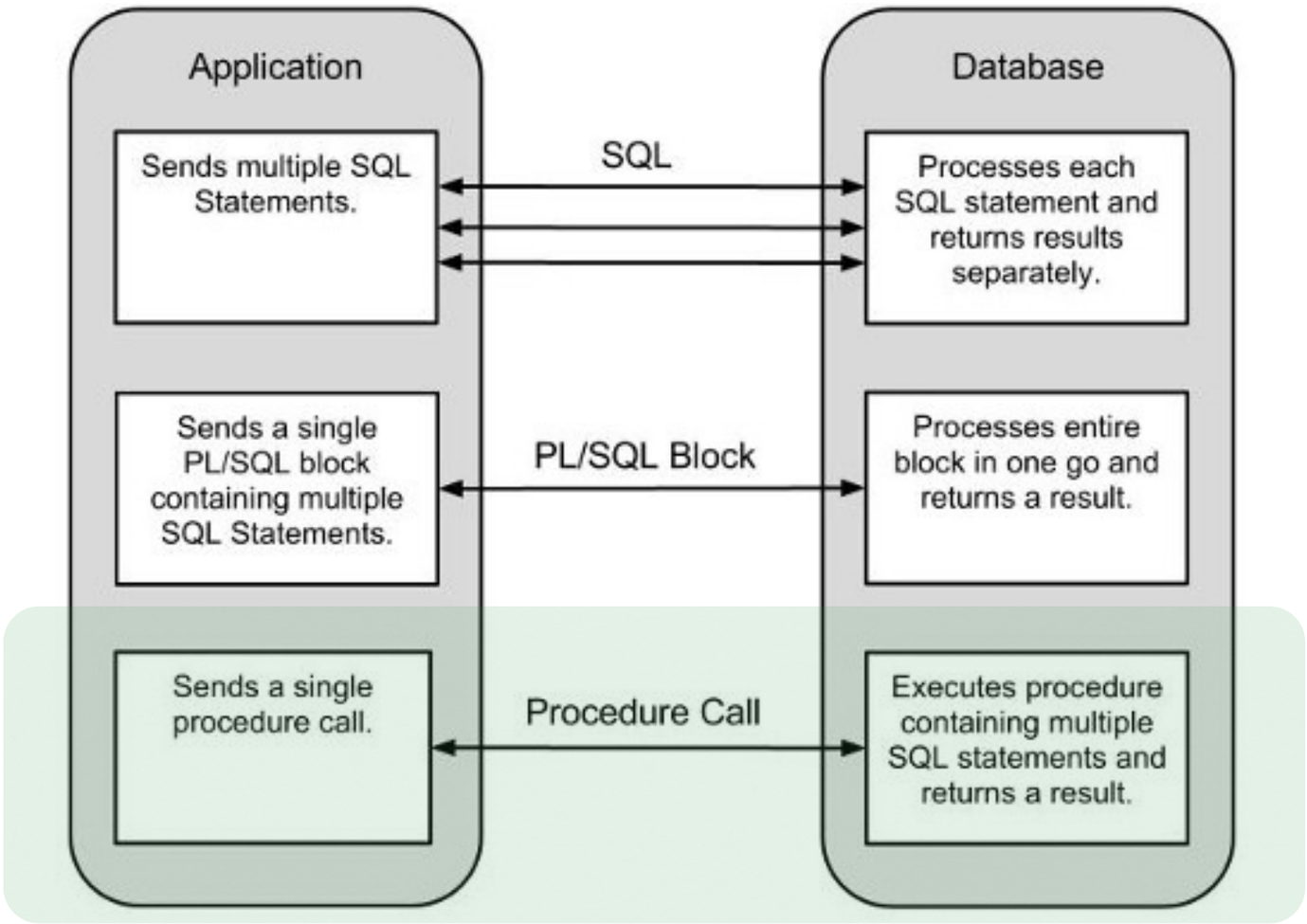
ORACLE DATABASE 12c

Advanced PL/SQL

Budácsik Attila

VANIO

PL/SQL szerepe



- sok DB hívás
- SQL-ek helye eloszott
- hálózati forgalom terhelése

- SQL-ek helye eloszott
- hálózati forgalom terhelése még mindig nagy

PL/SQL előnyei

- SQL integráció
- Minden SQL típust ismer
- Dinamikus SQL
- Modularizálás
- Teljesítmény
- Hordozhatóság
- OOP támogatás

Csomag tervezés

- Tervezd és készítsd el a Spec részt a Body előtt (publikus típusok, kurzorok, eljárások.. privát elemek a Body-ba kerüljenek)

- Használj privát eljárást vagy függvényt a logika elrejtéséhez

- Limitáld a kódokat BEGIN .. END között - ~60sor - több, kisebb, önálló eljárások - tesztelés, olvashatóság, újrafelhasználhatóság

- Közös elemek kiszervezése, pl.: csomagváltozóba = ne legyen kódismétlés!

- Hibakezelés megtervezése az üzleti logika pontos ismeretével

PL/SQL tervezés I.

- Hibaérzékeny kódot tegyük külön blokkba
- Kiértékelési sorrent - PL/SQL motor rövidre zárja
- CASE használata IF (PL/SQL) vagy DECODE (SQL) helyett
- SAVE EXCEPTION
- Bulk limit, save PGA, de ne hard kódolva!
- Emeljük ki az alkalmazásban használt konstansokat
- Emeljük ki az alkalmazásban használt változókat: SUBTYPE
- Collection bejárásához: FIRST, LAST és NEXT

PL/SQL tervezés II.

- Soha ne ismételj SQL utasítást
- Minden SQL utasítás beágyazása
- Kód írásakor vegyük figyelembe, hogy az alap struktúra (adatbázis) változhat
- Kurzor eredményét soha ne külön deklarált változóba fetch-eljük
- Használjunk SELECT FOR UPDATE-et, a lockolt sorok minimalizásához
- Újabb bekérdezést spórolhatunk meg a RETURNING záradékkal
- NOCOPY

PL/SQL hangolás

- FORALL
- BULK COLLECT
- Implicit konverzió elkerülése
- Kontextus váltás elkerülése
- Munka kurzorokkal
- Hard pasring
- SELECT beágyazása

FORALL

- Tömeges SQL művelethez ne használjunk FOR-t!

- Instead of:

```
...
FOR i IN 1 .. 50000 LOOP
  INSERT INTO bulk_bind_example_tbl
    VALUES(...);
END LOOP; ...
```

- Use:

```
...
FORALL i IN 1 .. 50000
  INSERT INTO bulk_bind_example_tbl
    VALUES(...);
...
```

- Az első esetben 50000 context váltás történik. A művelet java részét ez teszi majd ki. A FOR futás 2.184ms, a FORALL futási ideje 828ms

BULK COLLECT

- FORALL tömeges küldésre
- BULK COLLECT tömeges fogadásra
- Collection deklarálás szükséges
- Növeli a lekérdezés teljesítményét
  - SELECT INTO
  - FETCH

Implicit konverzió elkerülése

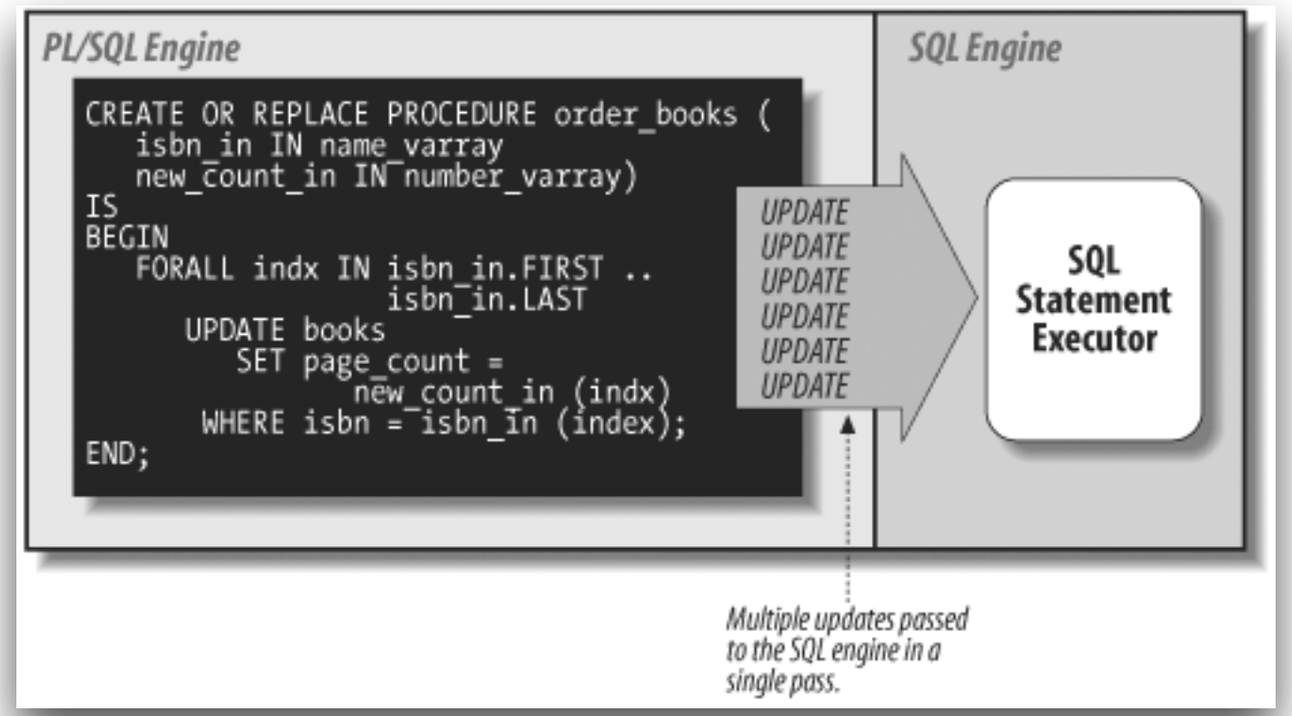
A PL/SQL minden esetben elvégzi az implicit konverziót az olyan típusok között, ahol ez lehetséges. Ennek elkerülése teljesítmény javulassal jár. Használd a következő függvényeket:

- TO_CHAR
- TO_NUMBER
- TO_DATE
- CAST

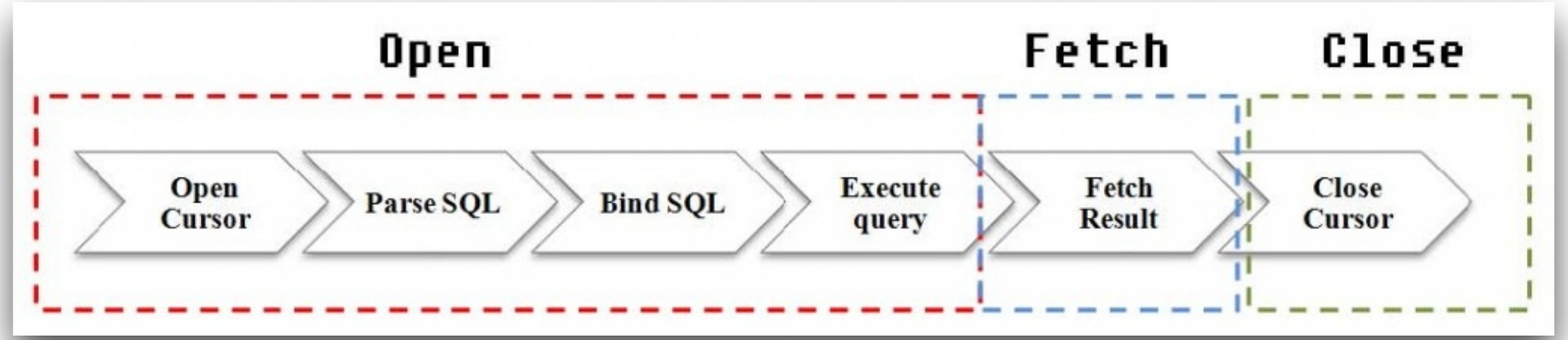
Explicit konverzió mindig gyorsabb
SUBTYPE-ok használata

Kontextus váltás elkerülése

FORALL



Munka kurzorokkal



- Két típusa van: implicit és explicit kurzor
- Manuáliasan vagy automatikusan? (OPEN, FETCH, CLOSE vagy FOR)
- kurzor_nev%ROWTYPE
- Paraméterezhető (hard coding elkerülése)
- REF CURSOR használata (dinamikus kurzor)

Hard parsing

Minden SQL-t parsol az Oracle (szintaktika és szemantika), majd minden különböző SQL utasításhoz létrehoz egy implicit kurzort

Hard parsing
Az adatbázisba először kerül be egy SQL utasítás, ami még nem található meg a shared pool-ban. Erőforrás igényesebb, mert újra be kell tölteni a memóriába

Soft parsing
A query megtalálható a shared pool-ban, és újra használható a futási terv is (?!)

Bind változók használatával növelhető az implicit kurzor újra felhasználhatósága

SELECT beágyazása

- **Eredmény cache-elve van**
  - Function Result Cache
  - Result Cache SQL hint
  - DETERMINISTIC
- **A select újra felhasználható**
  - Oracle database: soft parsing
  - Fejlesztő: egyszer kell megírni, paraméterezhető a függvény

Collections

- **Associative array**
  - **PL/SQL only**
  - Akármennyi eleme lehet ($-2_{31}+1$ től $+2_{31}-1$)

```
TYPE type_name IS TABLE OF element_type INDEX BY index_type;
```

- **Nested Table**
  - **SQL and PL/SQL**
  - Lehet tábla mező, felhasználható PL/SQL-ben, objektum attribútum ként..

- **Varrays**
  - **SQL and PL/SQL**
  - Fix elemű tömb. Hatékonyabb az előző két típusnál, mert előre le lesz foglalva a hely számára a memóriában

Collections

- Inicializálás (konstruktorral, FETCH-el vagy direkt hozzárendeléssel)

- Collection metódusok (FIRST, LAST, EXISTS, COUNT)

- Lekérdezés SELECT-el, TABLE(v1)

- Memoria felszabadítása:
  DBMS_SESSION.FREE_UNUSED_USER_MEMORY

Virtual Private Database

- Sor szintű biztonság
- Policy-k felállításával
- Példa: biztosítók (IGFB)
- Policies for SELECT, INSERT, UPDATE, and DELETE

Köszönöm!

attila.budacsik@vanio.hu