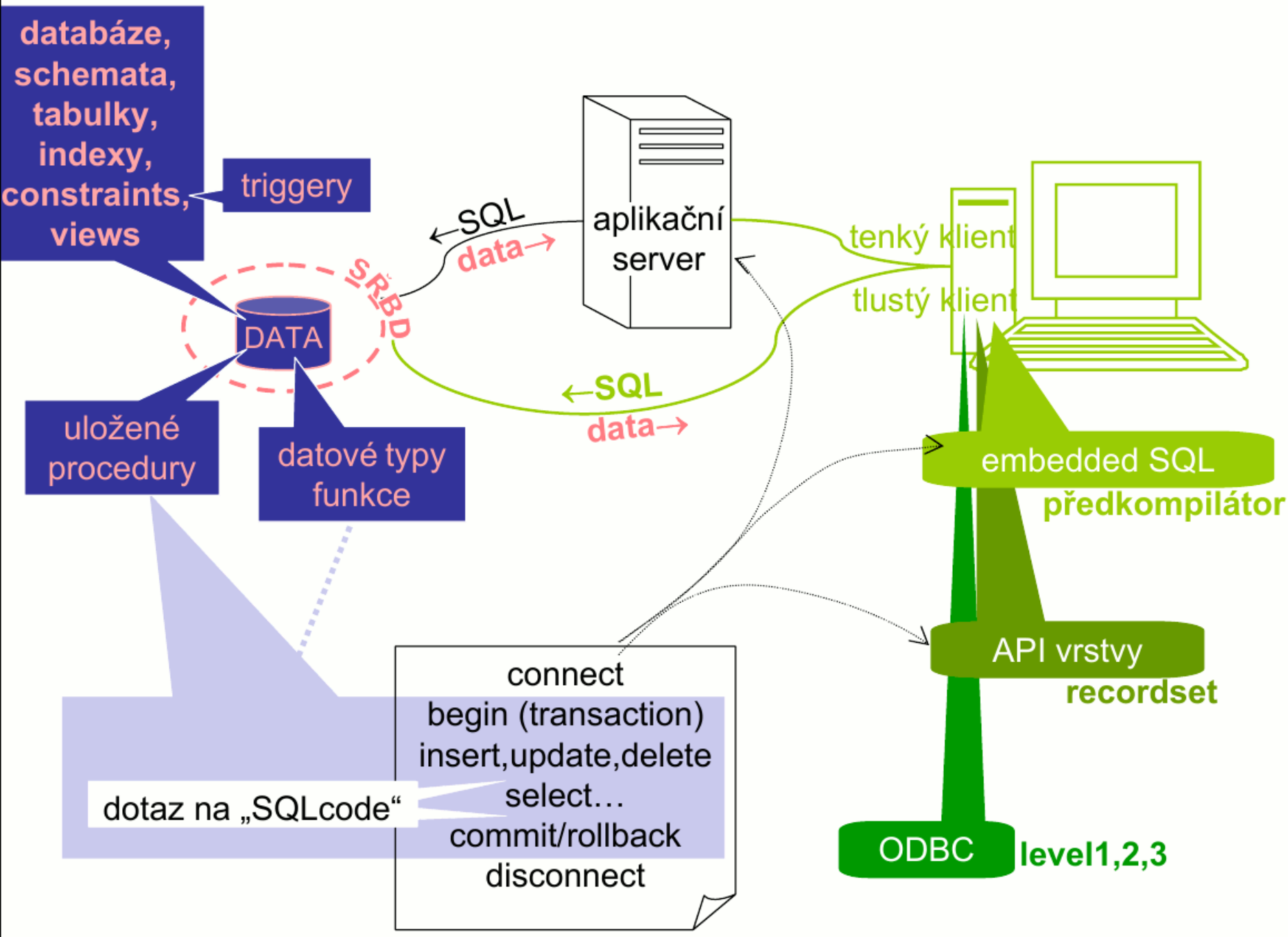
Databáze II

2. přednáška

Helena Palovská
palovska@vse.cz

SQL a aplikace


databáze,
schemata,
tabulky,
indexy,
constraints,
views
triggery
DATA
SRBD
uložené
procedury
datové typy
funkce
SQL
data
aplikační
server
tenký klient
tlustý klient
SQL
data
embedded SQL
předkompilátor
API vrstvy
recordset
ODBC
level1,2,3
connect
begin (transaction)
insert,update,delete
select…
commit/rollback
disconnect
dotaz na „SQLcode“

Program přednášky

- Řízení transakcí v SQL
- Integritní omezení v SQL
- Triggery a uložené procedury
- Zpracování množin záznamů

Řízení transakcí v SQL

Transakce

- Jednotka zpracování dat – ACID
  - Atomická: má se provést vše nebo nic. Pokud nelze dokončit, má se navrátit stav zpracovávaných dat před začátek transakce
  - Udržuje databázi v konzistentním stavu. Před dokončením transakce mohou být data „špatně“ – neodrážejí fakta tak, jak jsou – ale po potvrzení jsou „správně“.

pokračování

Transakce

- Jednotlivé transakce jsou navzájem izolovány: „nevědí“ o nepotvrzených stavech dat měněných jinou transakcí.
- Změny provedené potvrzenou transakcí jsou trvalé

- ACID zkratka:
  Atomicity, Consistency, Isolation, Durability

pokračování

Řízení transakcí v SQL

- Určení začátku transakce
  - BEGIN, event. BEGIN TRANSACTION
  - režim může být nastaven tak, že jednotlivé příkazy jsou automaticky považovány za samostatnou transakci
    (obvykle při interaktivním zadávání příkazů)
  - po ukončení předchozí transakce někdy automaticky začíná následující

pokračování

Řízení transakcí v SQL

- Izolace transakcí závisí na nastavení
  - TRANSACTION ISOLATION LEVEL
    - Serializable
      úplná izolace
    - Repeatable reads
      mohou nastat *phantom reads*
      (opakovaný SELECT vrací ev. jinou množinu řádků)
    - Read committed
      čten je poslední potvrzený stav dat, během transakce se může měnit - *non-repeatable reads*
    - Read uncommitted
      mohou být čtena nepotvrzená data – *dirty reads*

pokračování

Řízení transakcí v SQL

- Nastavení TRANSACTION ISOLATION LEVEL volíme v aplikaci,
  ovlivňuje následující transakce v této ní.
  - Ovlivňuje, jaký zámek je prováděn na data čtená (příkazem SELECT) transakcí.
  - Ovlivňuje, zda je pro transakci vedená dočasná kopie čtených dat.
  - → *Ovlivňuje rychlost zpracování.*

pokračování

Řízení transakcí v SQL

- Zámky pro zápis
  - implicitně provedeny na každá měněná data (INSERT, UPDATE, DELETE příkazy)
  - lze provést explicitně SELECT ... FOR UPDATE
  - lze zamknout i celou tabulku či pohled LOCK TABLE (nezamyká pro čtení)

pokračování

Řízení transakcí v SQL

- Ukončení transakce
  - COMMIT – požadavek na trvalé uložení změn
    - pokud se transakce dostala do stavu chyby (návratový kód některého příkazu sdělil chybu), pak COMMIT není proveden, je nutný ROLLBACK
  - ROLLBACK – požadavek vrátit změny před začátek transakce
    - pokud nastala chyba (viz výše)
    - pokud stav dat nevyhovuje

pokračování

Řízení transakcí v SQL

- Uvolňování zámků
  - zámky pro zápis jsou uvolněny s ukončením transakce
  - zámky pro čtení uvolňovány dle nastavení TIL:

TIL	zápis	čtení	čtení rozsahu
Read Uncommitted	S	S	S
Read Committed	C	S	S
Repeatable Read	C	C	S
Serializable	C	C	C

S – po příkazu, C – po ukončení transakce

Deadlock – uváznutí

1. transakce:

```
update ucty
  set stav=stav-100
  where cislouctu=501;
```

```
update ucty
 set stav=stav+100
 where cislouctu=1230;
```

2. transakce:

```
update ucty set
stav=stav+200 where
cislouctu=1230;
update ucty
set stav=stav-200
where cislouctu=501;
```

Deadlock

- Uváznutí zámků mezi jednotlivými transakcemi
- SŘBD obvykle mají proces detekce deadlocků,
  násilně ukončí některou z transakcí
  dle vlastních pravidel

Strategie zamykání v aplikacích

- Proti riziku deadlocku
  - pro všechny aplikace se určí pořadí tabulek a pořadí řádků v tabulkách, v transakcích se zamykají v data v tomto pořadí
- Pro serializaci
  - V transakci nejprve zamknout všechny potřebné objekty
  - Pak teprve manipulační příkazy

Integritní omezení v SQL

SQL constraints

- PRIMARY KEY, UNIQUE
- CHECK()
- NOT NULL
- DEFAULT
- FOREIGN KEY ... REFERENCING

Entitní integrita

- Zajistit, aby žádný entitní výskyt nebyl v databázi dvakrát
  - PRIMARY KEY
    - sémantický, tj. business identifikátor
      např. rodné číslo
    - umělý, nevýznamový
      např. automaticky generované ID
      nezajistí entitní integritu
  - UNIQUE
    - alternativní business identifikátor
    - zajištění entitní integrity, když je primární klíč umělý

Integrita vztahů

- Aby žádný výskyt vztahu nebyl v databázi dvakrát. V tabulce vztahu:
  - PRIMARY KEY (kombinace cizích klíčů na role)
  - UNIQUE (když je použit umělý ID)
- Další business pravidlo pro daný vztah
  - UNIQUE
    - např. rozvrh (nelze jeden objekt na dvou místech najednou, nelze dva objekty na jednom místě najednou)

Doménová integrita

- Zajistit, aby vložené hodnoty odpovídaly doméně
  - CHECK ()
    - např. CHECK (Známka IN (1,2,3))

Pravidla pro jeden řádek

- Aby hodnoty v řádku tabulky splňovaly danou podmínku
  - CHECK()
    - např CHECK(ČasZačátku<ČasKonce)
    - Pozor na podmínky týkající se systémového času – v budoucnu bude čas jiný!

NOT NULL, DEFAULT

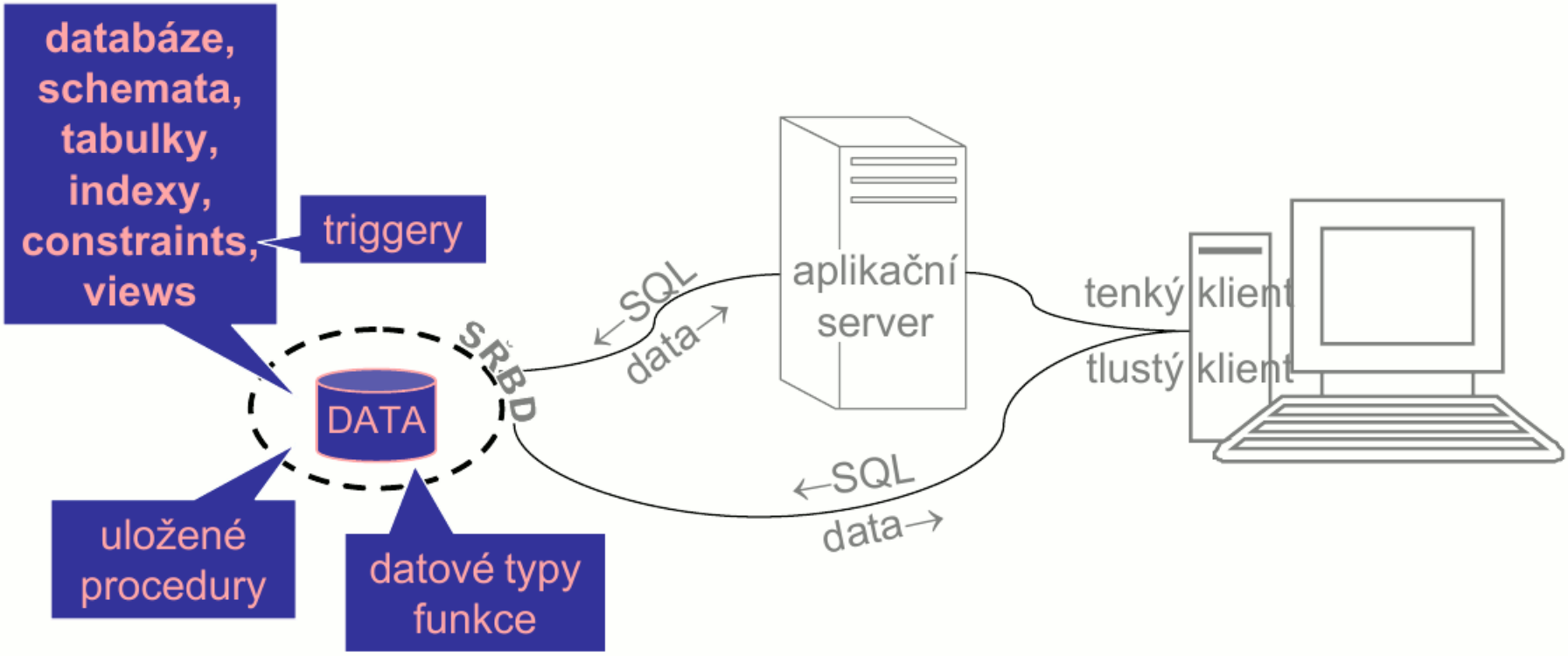
- Povinná hodnota
  - např. jméno klienta
  - počet objednaných kusů (v objednávce)
- Lze definovat dafaultní hodnotu
  - například „nezařazen“

Referenční integrita

- Aby odkazy cizích klíčů nebyly do prázdna
  - Omezuje vkládané hodnoty do slave klíče
  - Pro mazání master klíčů (ON DELETE):
    - RESTRICT (defaultní)
    - CASCADE
    - SET NULL
  - Pro UPDATE master klíčů bývá obvykle RESTRICT

http://krokodata.vse.cz/DM/RefInt

Objekty SQL databáze



Triggery a uložené procedury

Co je trigger

- Spouštěč při nějakém typu události
- Databázový trigger může být spouštěn
  - pokusem o INSERT
  - pokusem o UPDATE
  - pokusem o DELETE
- To, co trigger dělá, se považuje za součást transakce, která danou operaci přikazuje

Triggery

Užití pro:

- Integritní omezení
- Vytváření odvozených datových objektů
  - Repliky, archivy, vypočtená pole
- Akce ven
  - E-mail, zaslání objednávky…

Množství možných triggerů na jednu tabulku je omezené, zatěžují transakce!

Struktura triggeru

událost INSERT|UPDATE|DELETE

před/po BEFORE|AFTER (event.|INSTEAD)

tabulka … (u UPDATE výběr polí)

…jméno pro referenci na vkládaný/měněný/mazaný řádek

podmínka …

akce …

pro FOR EACH ROW | STATEMENT

Integritní omezení triggerem - příklad

Titul (ISBN, název, …)

Svazek (signatura, titul→Titul.ISBN,…, umístění)

Vypujcka (ctenar, svazek, od, do)

Rezervace (ctenar, titul, od, do)

Titul není možno rezervovat, když existuje volný svazek tohoto titulu.

pokračování příkladu

událost: INSERT do tabulky Rezervace

podmínka: „existuje volný svazek“

```
EXISTS SELECT * FROM Svazek s
LEFT OUTER JOIN Vypujcka v ON
(v.svazek=s.signatura AND v.do IS NULL)
WHERE s.titul=NEW.titul
AND v.svazek IS NULL
```

akce: vyvolat chybu, vrátit „nepovolit“

...málo efektivní

pokračování příkladu

Svazek (signatura, titul→Titul.ISBN,…,
umístění, je_volný:A/N)

podmínka: „existuje volný svazek“
EXISTS SELECT * FROM Svazek s
WHERE s.titul=NEW.titul
AND s.je_volný

…trigger na údržbu odvozeného pole je_volný

pokračování příkladu

událost: UPDATE pole do v tabulce Vypujcky

podmínka: -

akce: „u toho svazku napiš, že je volný“

```
UPDATE Svazek SET je_volny=T
WHERE Svazek.signatura=NEW.svazek
```

pokračování příkladu

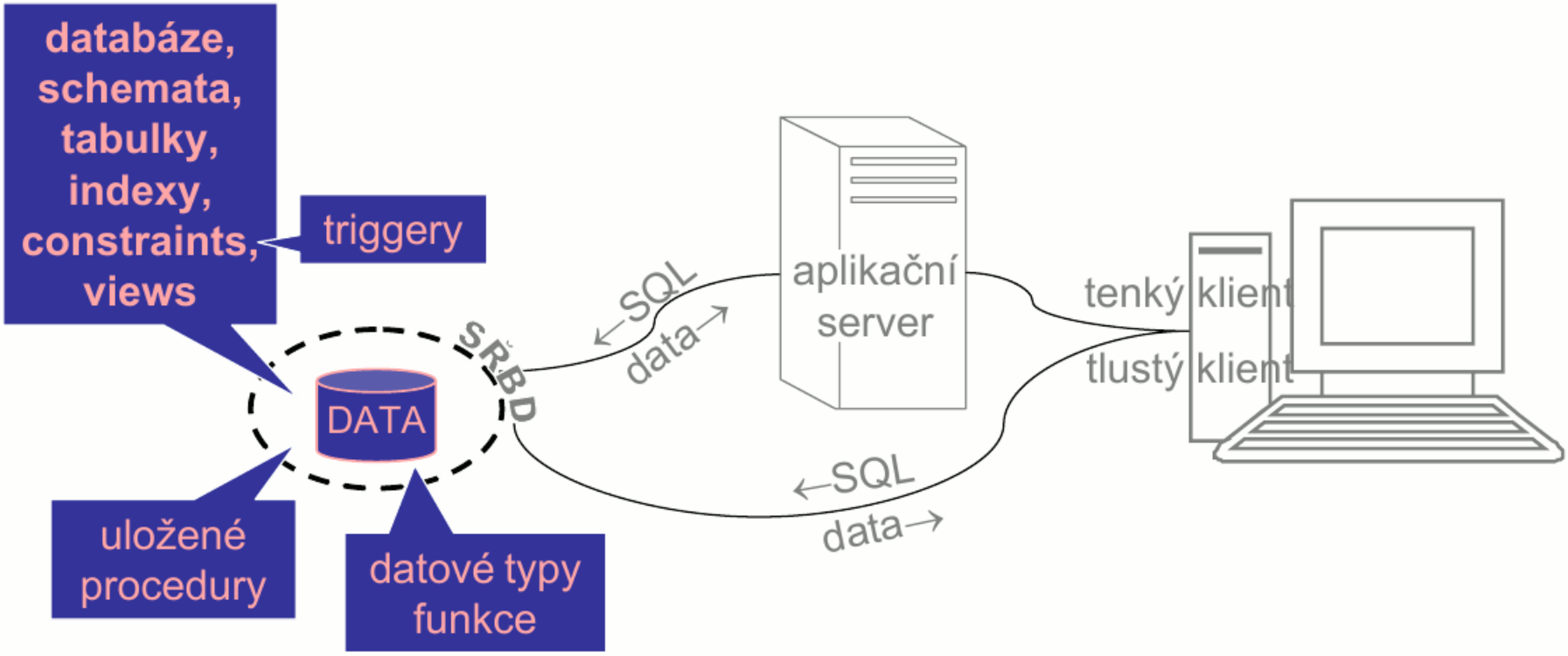
událost: INSERT do tabulky Vypujcky

podmínka: -

akce: „u toho svazku napiš, že není volný“

```
UPDATE Svazek SET je_volny=F
WHERE Svazek.signatura=NEW.svazek
```

Objekty SQL databáze



Uložené procedury

- Spustitelné programy spravované a prováděné databázovým serverem

Užití uložených procedur

- Jednotlivé typové SELECTy
  - (proti nebezpečí neoptimalizovaných SELECTů)
- Zapouzdřené manipulace s daty
- Těla triggerů
- Údržbové procedury

Výhody uložených procedur

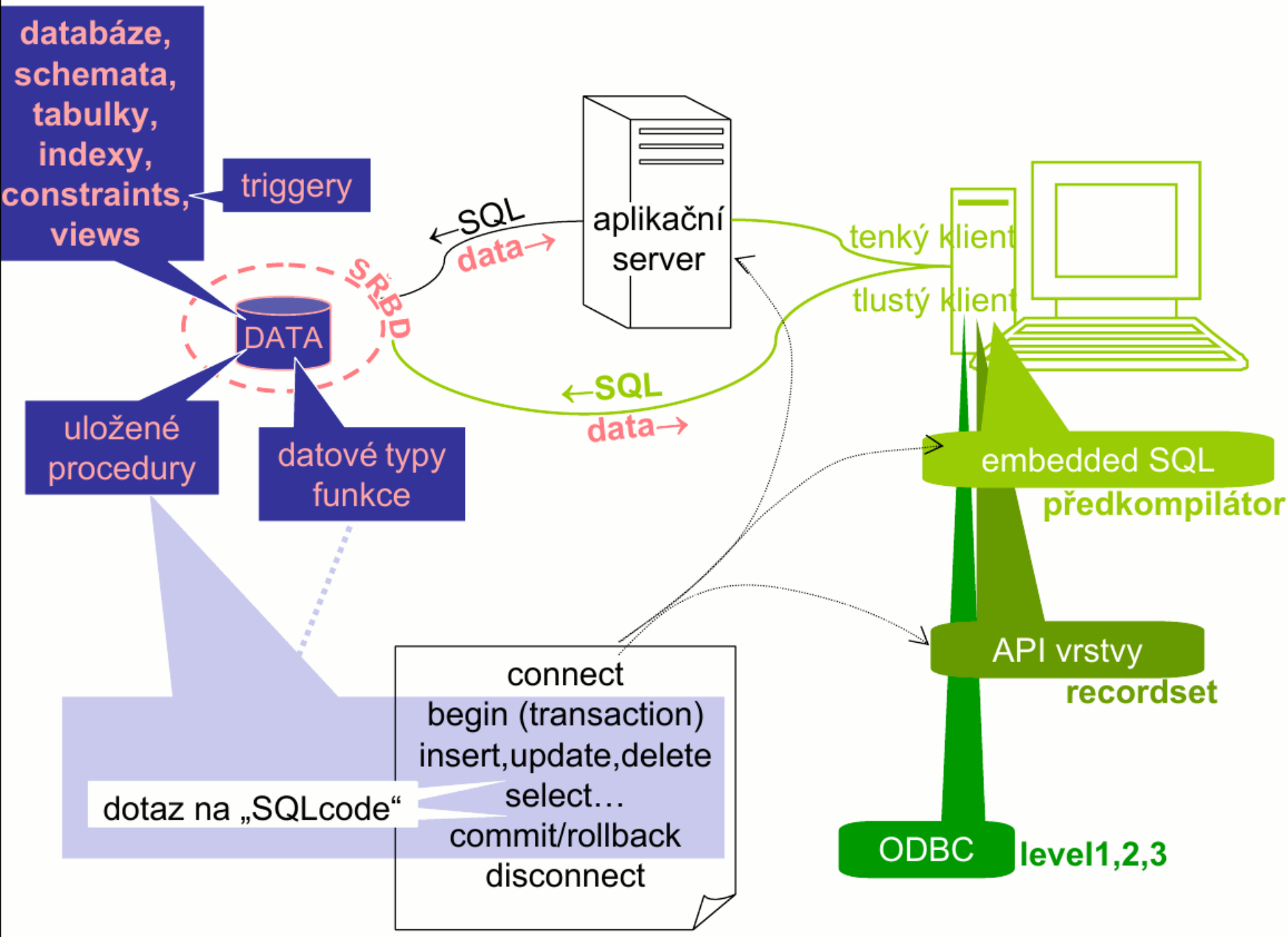
- Jsou spravované db serverem
  - Přístupová práva
- Zkompilované, prováděné db serverem
  - Rychlost, sdílení paměti
- Logika aplikace je na jediném místě
  - Snadná údržba (opravy, aktualizace)
  - Může programovat nejlepší programátor, jednou
- Lze omezit přístupová práva k datovým objektům
  - Bezpečnost, spolehlivost přístupu

Nevýhoda uložených procedur

- Zatěžují DB server
  - Proto jsou vhodné jsou hlavně pro „datově intenzivní procesy“
- Otázka, kam umístit aplikační logiku, musí být řešena s ohledem na rozmístění zátěže

Uložené procedury

- Napsané v SQL
  - Od verze 1999 je SQL úplný programovací jazyk
  - Proměnné…
- Napsané v jiném programovacím jazyce
  - Db server musí mít kompilátor toho jazyka


databáze,
schemata,
tabulky,
indexy,
constraints,
views
triggery
DATA
SRBD
uložené
procedury
datové typy
funkce
SQL
data
aplikační
server
tenký klient
tlustý klient
SQL
data
embedded SQL
předkompilátor
API vrstvy
recordset
ODBC
level1,2,3
connect
begin (transaction)
insert,update,delete
select…
commit/rollback
disconnect
dotaz na „SQLcode“

Databázové kurzory

- Databázový objekt poskytující přístup jednotlivým řádkům z množiny definované příkazem SELECT
- Funkcionalita poskytující „pohyb kurzorem“
  - následující (řádek)
  - event. předešlý, posun o n řádků...
  - event. cyklický průchod všemi řádky...

Databázové kurzory

- Odlišné režimy
  - pouze pro čtení/ pro zápis (v původní tabulce)
  - statické/dynamické
  - privátní/sdílené
  - jen dopředné/pohyb tam a zpět
  - po jednom řádku/dávkové čtení více řádků

Užití kurzorů

- Průchod množinou objektů k postupnému zpracovávání – manipulaci nějakých dat
- Znovupoužití výsledku SELECTu
  - uvážené kontrolované užití

API recordsets

- Objekt aplikace odstiňující od databázového kurzoru

SELECT prvních n řádků

- ŠRBD může efektivně optimalizovat

Hromadné UPDATE

- Riziko neúspěchu
  a nutnosti opakovat
- Volitelná strategie:
  - po dávkách,
    s evidencí úspěšných zpracování

Konec