VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií

Databázové systémy

2016/2017

Projekt 4. a 5. část - Poslední SQL skript, dokumentace a finální obhajoba

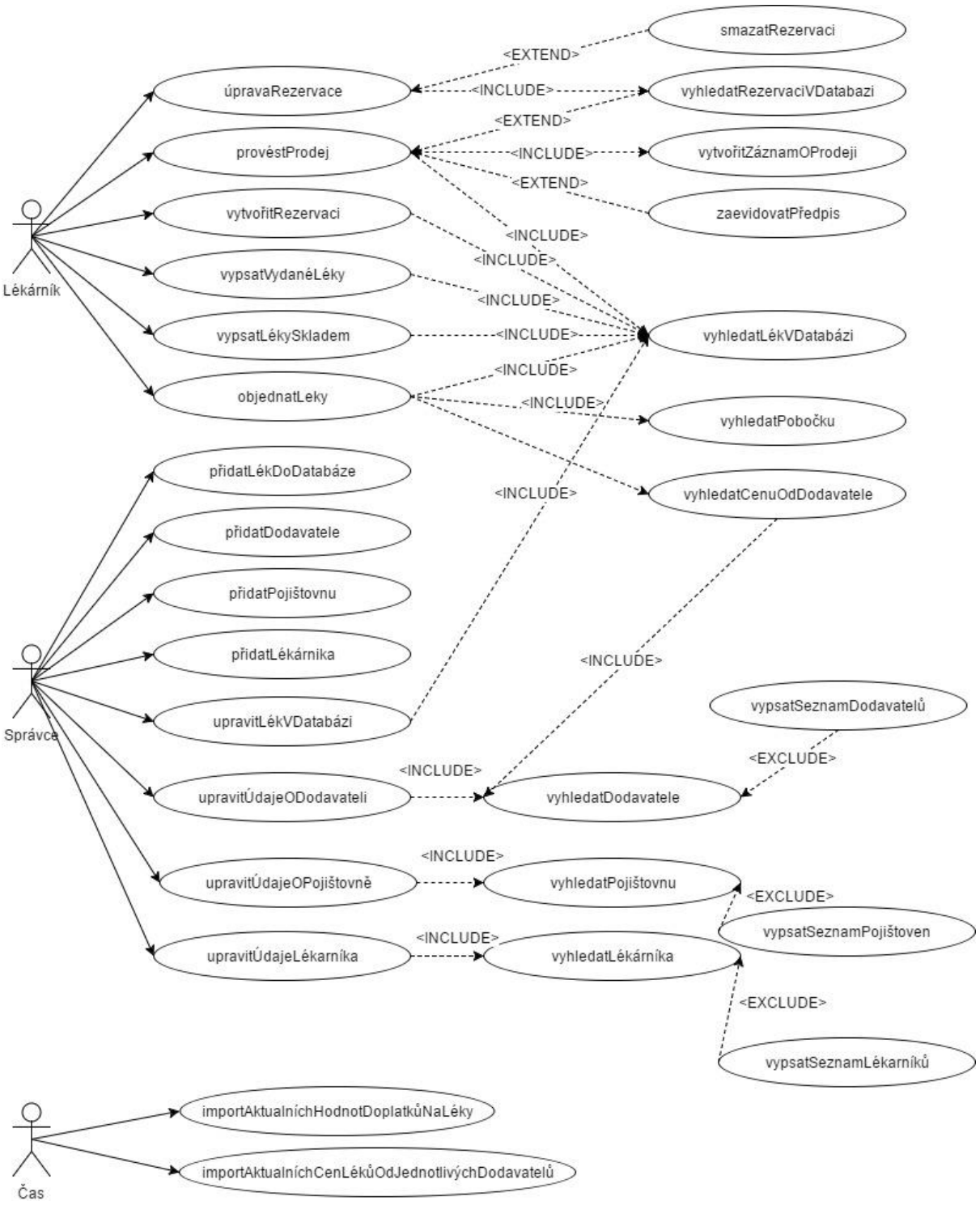
Zadání č. 35 – Lékárna

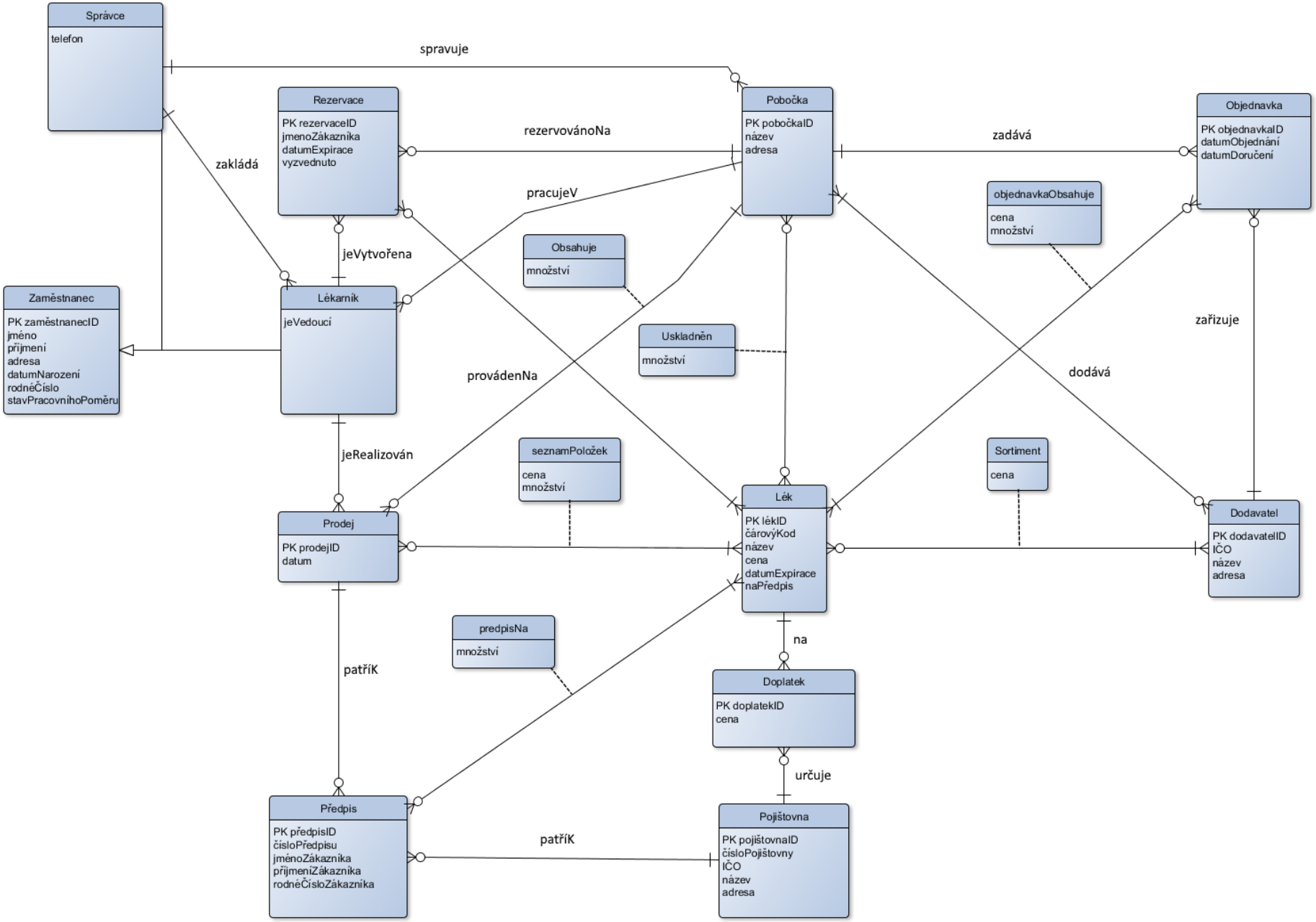
Matěj Mlejnek (xmlejn04)
Vojtěch Meluzín (xmeluz04)

Brno, 30. dubna 2017

Zadání:

Vaším úkolem je vývoj informačního systému lékárny. Lékárna vydává občanům léky jak na předpis, tak i bez předpisu. U léků vydávaných na předpis může část ceny hradit zdravotní pojišťovna. Některé léky se vydávají pouze na předpis. U léků na předpis musí systém evidovat informaci o rodném čísle zákazníka a pojišťovně, které bude částka za daný lék fakturována. Systém musí umožnit evidenci vydávaných léků, import aktuálních hodnot příspěvků na léky od zdravotních pojišťoven (může se čas od času měnit) a musí evidovat množství zboží na skladě. Léky jsou identifikovány číselným kódem či názvem. Firma
má několik poboček, informační systém tedy musí být schopen zjistit množství zboží na skladě
na zvolené pobočce, stejně tak se eviduje i prodané zboží vzhledem k dané pobočce. Požaduje-li zákazník zboží, které momentálně není na dané pobočce k dispozici, může si jej zarezervovat. Náš sortiment neobsahuje zboží, u kterého bychom si potřebovali pamatovat, kdo si jej koupil, u prodaného zboží si tedy nepamatujeme detaily o zákaznícovi, stejně tak v případě rezervací poslouží pouze jméno zákazníka. Informační systém bude obsluhován prozatím pouze zaměstnanci lékárny. Doplatky pojišťoven na jeden lék se různí, stejně tak jako cena léku prodaného bez předpisu. Evidujte i dodavatele lékárny, jakož i jejich ceny pro dané zboží. Objednávky u těchto dodavatelů již modelovat nemusíte. Předpokládejte, že není třeba evidovat zaměstnance, je irelevantní, kdo prodej zboží realizoval.


Lékárník
úpravaRezervace
provéstProdej
vytvořitRezervaci
vypsatVydanéLéky
vypsatLékySkladem
objednatLeky
smazatRezervaci
vyhledatRezervaciVDatabazi
vytvořitZáznamOProdeji
zaevidovatPředpis
vyhledatLékVDatabázi
vyhledatPobočku
vyhledatCenuOdDodavatele
<EXTEND>
<INCLUDE>
Správce
přidatLékDoDatabáze
přidatDodavatele
přidatPojišťovnu
přidatLékárnika
upravitLékVDatabázi
upravitÚdajeODodavateli
upravitÚdajeOPojišťovně
upravitÚdajeLékarníka
vyhledatDodavatele
vyhledatPojišťovnu
vyhledatLékárníka
vypsatSeznamDodavatelů
vypsatSeznamPojišťoven
vypsatSeznamLékarníků
<EXCLUDE>
Čas
importAktualníchHodnotDoplatkůNaLéky
importAktualníchCenLékůOdJednotlivýchDodavatelů


Správce
telefon
spravuje
Rezervace
PK rezervaceID
jmenoZákazníka
datumExpirace
vyzvednuto
rezervovánoNa
Pobočka
PK pobočkaID
název
adresa
zadává
Objednavka
PK objednavkaID
datumObjednání
datumDoručení
zakládá
pracujeV
objednavkaObsahuje
cena
množství
jeVytvořena
Obsahuje
množství
Zaměstnanec
PK zaměstnanecID
jméno
příjmení
adresa
datumNarození
rodnéČíslo
stavPracovníhoPoměru
Lékarník
jeVedoucí
Uskladněn
množství
zařizuje
provádenNa
dodává
jeRealizován
seznamPoložek
cena
množství
Sortiment
cena
Prodej
PK prodejID
datum
Lék
PK lékID
čárovýKod
název
cena
datumExpirace
naPředpis
Dodavatel
PK dodavatelID
IČO
název
adresa
predpisNa
množství
na
patříK
Doplatek
PK doplatekID
cena
určuje
Předpis
PK předpisID
čísloPředpisu
jménoZákazníka
příjmeníZákazníka
rodnéČísloZákazníka
patříK
Pojišťovna
PK pojišťovnaID
čísloPojišťovny
IČO
název
adresa

Popisy triggerů:

V našem řešení používáme pět druhý triggerů

1. Automatické přiřazení id
   Automaticky přiřadí privátnímu klíči další číslo v pořadí ze sekvence pro danou tabulku.

2. Kontrola množství
   Zkontroluje jetli je zadané množství větší jak 0. Pokud nedovolí přidat záznam a upozorní na chybu.

3. Kontrola ceny
   Zkontroluje jestli náhodou není zadaná cena menší než 0. Pokud ano nedovolí záznam se zápornou cenou přidat a upozorní na chybu.

4. Kontrola rodného čísla
   Zkontroluje podle předem daných pravidel správnost zadaného rodného čísla. Pokud není správné nedovolí záznam založit a upozorní na chybu.

5. Kontrola IČO
   Zkontroluje podle předem daných pravidel správnost zadaného IČO. Pokud není správné nedovolí záznam založit a upozorní na chybu.
   Zkontrolu

Procedury:

Zrušení rezervace(1):

zrus_rezervaci(id_rezervace)

Parametry:
id_rezervace = id rezervace

Funkce:
Odstraní rezervaci a její obsah.

Popis:
1. Zkontroluje existenci rezervace.
2. Odstraní obsah rezervace
3. Odstraní samotnou rezervaci

Objednání docházejících léku pro danou pobočku(2):

objednat_dochazejici_leky_na(pobocka_id,minimalni_mnozstvi)

Parametry:
pobocka_id = id pobočky
minimalni_mnozstvi = minimální množství pro každý lék na skladě

Funkce:
Doplnění zásob léků na pobočce, objednáním každého s nedostatečným počtem naskladě od nejlevnejšího dodavatele.

Popis:
Najde kolik léků je uskladněných pro danou pobočku. Pokud je množství pro jeden lék menší jak minimální zadané pmnožství, vytvoří objednávku dodavateli, který má nejlepší cenu pro daný lék, aby bylo na skladě alepoň minimální množství pro každý z léků.
1. Zkontroluje správnost vstupních parametrů
2. Vybere všechny léky, které nejsou na skladě v dostatečném množství
3. Najde nejlepšího dodavatele pro daný lék
4. Vytvoří pro každého dodavatele jednu objednávku obsahující všechny léky, které od něj budeme objednávat

Objednání potřebných léků pro danou rezervaci(3):

rezervace_objednat_leky(id_rezervace)

Parametry:

id_rezervace = id rezervace

Funkce:

Při vytváření rezervace doobjedná léky, kterých není dostatečné množství na skladě.

Popis:

Zkontroluje jestli pro danou rezervaci je na pobočce, kde je rezervace k vyřízení dost léků. Pokud nějaký chybí vytvoří objednávku od dodavatele s nejnižší cenou.

1. Zkontroluje správnost vstupních parametrů
2. Vybere všechny léky, které nejsou na skladě v dostatečném množství pro naši rezervaci
3. Najde nejlepšího dodavatele pro daný lék
4. Vytvoří pro každého dodavatele jednu objednávku obsahující všechny léky, které od něj budeme objednávat

Indexy:

Chce zjistit seznam léků s jejich množstvím

```
SELECT l.NAZEV "Nazev leku", NVL(SUM(uk.MNOZSTVI), 0) "Mnozstvi na skladu"
FROM  lek l
left outer join uskladnen uk on
   uk.LEKID = l.LEKID
GROUP BY l.NAZEV;
```

Bez indexů přistupujeme k oběma tabulkám přes *TABLE ACCESS FULL*, což znamená, že čteme celou tabulku. *HASH JOIN* znamená propojení dvou tabulek přes hash(spočítá se hash pro každý řádek menší tabulky a pak pro každý řádek větší tabulky, pak se tabulky prováží)

```
-------------------------------------------------------------------------------
| Id  | Operation          | Name      | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |           |     4 |   364 |     6  (34)| 00:00:01 |
|   1 |  HASH GROUP BY     |           |     4 |   364 |     6  (34)| 00:00:01 |
|*  2 |   HASH JOIN OUTER  |           |     4 |   364 |     5  (20)| 00:00:01 |
|   3 |    TABLE ACCESS FULL| LEK      |     4 |   260 |     2   (0)| 00:00:01 |
|   4 |    TABLE ACCESS FULL| USKLADNEN |     3 |    78 |     2   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

Výsledek selectu můžeme urychlit, když si vytvoříme index na tabulce uskladnen.

```
CREATE INDEX uskladnen_ix
   ON uskladnen (LEKID, MNOZSTVI);
```

Vidíme, že se použil vytvořený **index**. *INDEX FULL SCAN* znamená, že se hodnoty vybírají z **indexovaných sloupců**. Používá pouze jeden Input/Output blok a data jsou seřezena.

```
-------------------------------------------------------------------------------
| Id  | Operation          | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |              |     4 |   364 |     5  (40)| 00:00:01 |
|   1 |  HASH GROUP BY     |              |     4 |   364 |     5  (40)| 00:00:01 |
|*  2 |   HASH JOIN OUTER  |              |     4 |   364 |     4  (25)| 00:00:01 |
|   3 |    TABLE ACCESS FULL| LEK         |     4 |   260 |     2   (0)| 00:00:01 |
|   4 |    INDEX FULL SCAN  | USKLADNEN_IX |     3 |    78 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

Pokud budeme chtít select ještě **urychlit přidáme i index nad tabulku lek**, který nám prováže jména a idleků.

```
CREATE INDEX lek_ix
   ON lek (lekid, NAZEV);
```

```
-------------------------------------------------------------------------------
| Id  | Operation         | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |              |     4 |   364 |     4  (50)| 00:00:01 |
|   1 |  HASH GROUP BY    |              |     4 |   364 |     4  (50)| 00:00:01 |
|*  2 |   HASH JOIN OUTER |              |     4 |   364 |     3  (34)| 00:00:01 |
|   3 |    INDEX FULL SCAN| LEK_IX       |     4 |   260 |     1   (0)| 00:00:01 |
|   4 |    INDEX FULL SCAN| USKLADNEN_IX |     3 |    78 |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

Indexování a jeho výhody pocítíme hlavně při **velkém množství dat**.

Ukázka transakčního zpracování a uzamykání tabulek:

Máme dva uživatele. Uživatel A a B.

A aktualně vidí, že lék s id 21 má název Paralen.
>select nazev from B.lek WHERE lekid = 21;
Výpis: Paralen

A se rozhodně změnit název léku.
>UPDATE B.lek SET nazev = 'Paralen Ultra' WHERE lekid = 21;

Název léku se změnil a tabulka je uzamčená.
>select nazev from B.lek WHERE lekid = 21;
Výpis: Paralen Ultra

B se podívá na data.
*>select * from B.lek WHERE lekid = 21;*
Výpis: Paralen

B se rozhodne daný lék taky upravit, ale nepodaří se mu to do té doby, než bude tabulka odemknuta uživatelem **A**.
>UPDATE B.lek SET nazev = 'Paralen Max' WHERE lekid = 21;

A se rozhodne, že potvrdí změny
>commit;
Tabulka B.lek se odemče a **B** provede změny nad tabulkou

B nyní vidí
*>select * from B.lek WHERE lekid = 21;*
Výpis: Paralen Max

A ale změnu nevidí.
*>select * from B.lek WHERE lekid = 21;*
Výpis: Paralen Ultra

B potvrdí změnu
>commit;

A tak uvidí změnu, kterou udělal **B**
*>select * from B.lek WHERE lekid = 21;*
Výpis: Paralen Max