Projekt IDS
2020/2021

4. a 5. část - poslední SQL skript, dokumentace

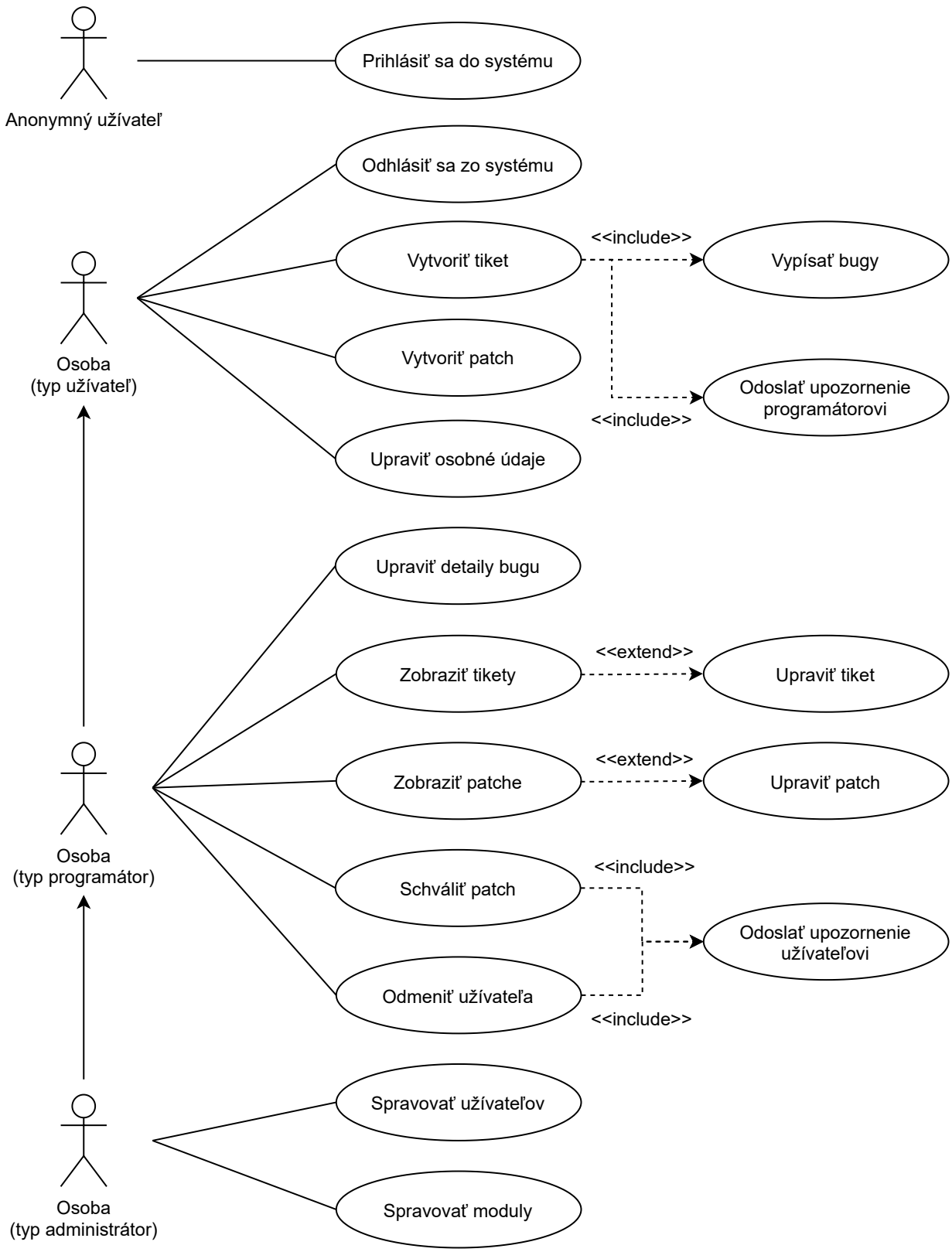
Adrián Kálazi, Kevin Lackó
xkalaz00, xlacko08

Zadanie IUS č. 49 - Bug Tracker

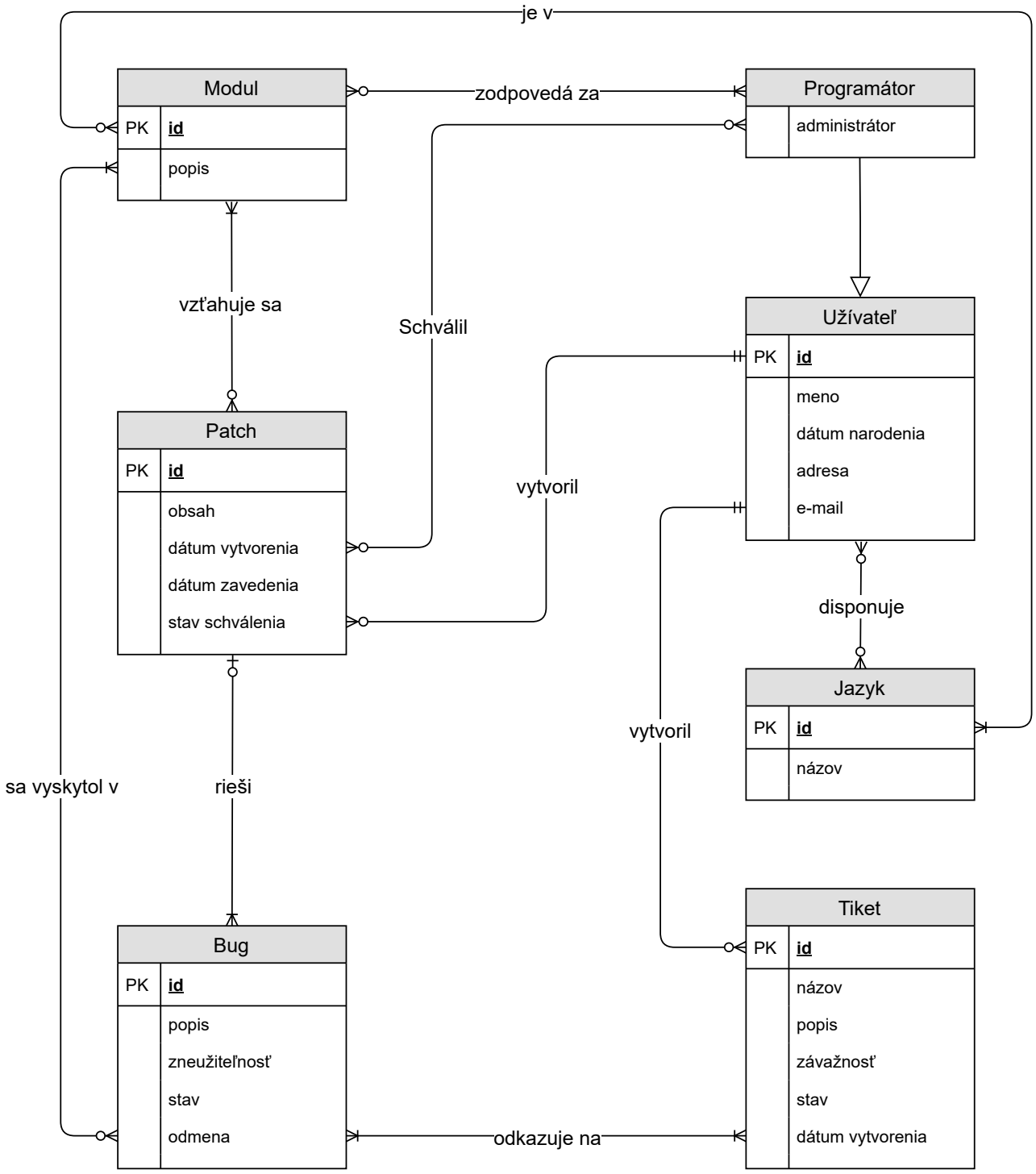
Vytvořte informační systém pro hlášení a správů chyb a zranitelností systému. Systém umožňuje uživatelům hlásit bugy, jejich závažnosti a moduly, ve kterých se vyskytly, ve formě tiketů. Tikety mohou obsahovat hlášení o více než jednom bugu a stejný bug může být zahlášen více uživateli. Bug může (ale nemusí) být zranitelností a v tomto případě zaevidujeme i potenciální míru nebezpečí zneužití této zranitelnosti. V případě zahlášení bugů, odešle systém upozornění programátorovi, který zodpovídá za daný modul, přičemž může odpovídat za více modulů. Programátor pak daný tiket zabere, přepne jeho stav na "V řešení"a začne pracovat na opravě ve formě Patche. Patch je charakterizován datem vydání a musí být schválen programátorem zodpovědným za modul, které mohou být v různých programovacích jazycích. Jeden Patch může řešit více bugů a současně řešit více tiketů a vztahuje se na několik modulů. Samotní uživatelé mohou rovněž tvořit patche. Takové patche však musí projít silnější kontrolou než jsou zavedeny do systému. Kromě data vytvoření patche rovněž evidujte datum zavedení patche do ostrého provozu. Každý uživatel a programátor je charakterizován základními informacemi (jméno, věk, apod.), ale současně i jazyky, kterými disponuje, apod. V případě opravení bugů, mohou být uživatele upozorněni na danou opravu a případně být odměněni peněžní hodnotou (podle závažnosti bugu či zranitelnosti).

Obsah

1 Use case diagram



2 Entity relationship diagram



3 Popis výsledného skriptu

3.1 Drop existujúcich tabuliek

Pred vytvorením nových databázových objektov sa vykoná `DROP TABLE` pre každú vytvorenú tabuľku, `DROP SEQUENCE` pre sekvenciu a `DROP MATERIALIZED VIEW` pre materializovaný pohľad aby nedochádzalo k redefinícii už existujúcich tabuliek. Pri tabuľkách so závislosťami sa použije `CASCADE CONSTRAINTS` na odstránenie tabuľky a jej závislostí.

3.2 Vytvorenie nových tabuliek

Vytvoria sa tabuľky podľa špecifikácie ER diagramu zo sekcie 2. Pre vzťahy many to many sa vytvorili nové pomocné tabuľky. Pre vzťahy one to many sa iba pridali codzie kľúče pomocou `ALTER TABLE` po vytvorení oboch tabuliek vzťahu.

3.3 Triggery - vytvorenie

1. `ticket_id_gen` - Slúži na automatické generovanie hodnôt primárneho kľúča tabuľky `ticket` podľa sekvencie `ticket_seq`.
2. `patch_created_gen` - Podobný trigger-u 1, generuje hodnotu aktuálneho času do stĺpca `created` tabuľky `patch`.
3. `notify_programmer` - V prípade vytvorenia nového bugu notifikuje programátorov zodpovedných za modul v ktorom sa bug vyskytol. Notifikuje programátorov zodpovedných za modul ak je vytvorený nový bug.

3.4 Naplnenie databázy

Po definovaní triggerov sa databáza naplní ukážkovými dátami (v tejto časti sql skriptu je aj ukážka použitia triggerov).

3.5 Triggery - použitie

1. `ticket_id_gen` - Pred vykonaním `INSERT`-u do tabuľky `ticket` sa do stĺpca `id` vloží nasledujúca hodnota zo sekvencie.
2. `patch_created_gen` - Pred vykonaním `INSERT`-u do tabuľky `patch` sa do stĺpca `created` vygeneruje a vloží timestamp aktuálneho času.
3. `notify_programmer` - Po vykonaní `INSERT`-u sa odošle notifikácia programátorom (v našom programe implementované iba ako `DBMS_OUTPUT.PUT_LINE()` ale v produkčnej verzii sa môže pridať možnosť odosielania emailov cez SMTP server).

3.6 Procedúry

1. `p_display_ticket_info` - Zobrazí informácie o tickete (podľa id) a bugoch na ktoré odkazuje. Použije sa zavolaním `"p_display_ticket_info"(ticket_id);` kde `ticket_id` je id ticketu na zobrazenie.
2. `p_display_module_info` - Zobrazí informácie o module (podľa id), v akých jazykoch je implementovaný a kto zaň zodpovedá. Použije sa zavolaním `"p_display_module_info"(module_id);` kde `module_id` je id modulu na zobrazenie.

3.7 Explain plan a index

Pre ukážku EXPLAIN PLAN sme si zvolili SELECT doposiaľ neschválených patchov ktoré sa týkajú viac ako jedného modulu (u týchto patchov je možné predpokladať záujem viacerých programátorov). Tento príkaz spojí 2 tabuľky, používa agregačnú funkciu `COUNT` a klauzulu `GROUP BY` podľa zadania.

Pôvodný príkaz je neoptimálny z dôvodu dvoch vnorených cyklov (čo je vidieť z výstupu `EXPLAIN PLAN`).

Pre zrýchlenie príkazu sme zaviedli nový index `index_approved` na stĺpci `approved` tabuľky `patch`. Tento index používa B strom (default index u Oracle databáze).

Príkaz s použitím indexu je lepší lebo použije namiesto vnorených cyklov `HASH JOIN` čo je rýchlejšia operácia a teda značne zníži CPU cost príkazu hlavne pri práci s väčším objemom dát. Rozdiel je vidieť pri porovnaní výstupov z oboch volaní `EXPLAIN PLAN`.

3.8 Prístupové práva

Pre každú tabuľku a materializovaný pohľad je vykonaný príkaz `GRANT` s právami `ALL`. Pre procedúry postačí právo execute, preto je pre každú procedúru vykonaný príkaz `GRANT` s právom `EXECUTE`.

3.9 Materializovaný pohľad

Vytvorili sme materializovaný pohľad ktorý zobrazí všetky tickety, kto ich vytvoril a zoradí ich podľa statusu. Dáta v materializovanom pohľade sa neaktualizujú ak sa zmení hodnota v tabuľkách z ktorých je vytvorený.