Vysoké učení technické v Brně
Fakulta informačních technologií

Databázové systémy – Projekt
Část 5. – Dokumentace schématu databáze

Zadání č. 61 (IUS) – Kavárenský povaleč

29. dubna 2021 Ondřej Ondryáš (xondry02), František Nečas (xnecas27)

Obsah

1 Úvod

Navržený systém navazuje na zadání č. 61 (Kavárenský povaleč) z předmětu IUS, cílem je tedy vytvořit systém pro sdílení informací o kavárnách, nabízených nápojích a pořádaných degustačních (cuppingových) akcích.

2 Konceptuální modely (1. část projektu)

V systému vystupují 4 aktéři, pro modelování případů použití je použito zobecnění. Základem je *Uživatel*, který může informace o kavárnách primárně prohlížet, také si může rezervovat místa na cuppingových akcích. *Registrovaný uživatel* může navíc psát recenze na kavárny a akce, přidávat k recenzím komentáře a hodnotit kavárny. Dále také může vytvořit v systému novou kavárnu, čímž se pro tuto kavárnu stane *majitelem* a může jiným uživatelům přidat v této kavárně roli zaměstnance. *Zaměstnanec kavárny* může měnit informace o nabízené kávě a majitel může spravovat cuppingové akce v kavárně. Use case diagram je uveden v příloze A.

ER diagram je uveden v příloze B. Uživatel, reprezentovaný typem entity `User` může uvést do systému i údaje o oblíbeném způsobu přípravy kávy, oblíbeném druhu kávy nebo počtu šálků, které denně vypije. Tyto údaje jsou záměrně modelovány jako atributy typu entity, protože tyto informace mohou nabývat prakticky libovolné textové hodnoty a nemusí mít žádnou vazbu na nabídky konkrétních kaváren. Naopak informace o oblíbené kavárně je realizována vztahy mezi entitami typu `User` a entitami typu `Cafe` – jako oblíbenou kavárnu tak uživatel může nastavit pouze jednu z kaváren, které jsou uloženy v systému. Po převodu na schéma databáze byl do typu entity `User` byl doplněn ještě otisk hesla, z hlediska konceptuálního modelu systému to ale není podstatné.

Uživatel může psát recenze ke kavárnám (reprezentovaným typem entity `Cafe`), na které další uživatelé mohou reagovat vlastní recenzí, případně reakci hodnotit palcem nahoru nebo palcem dolů. Udělované palce nahoru a dolů jsou modelovány typem entity `ReviewRating` (její primární klíč je složený z cizích klíčů ukazujících na identifikátor uživatele a hodnocené recenze). Typ entity `Review` navíc obsahuje atributy `points_up` a `points_down`, ve kterých jsou číselně uloženy počty obou druhů hodnocení. Tyto hodnoty vycházejí z počtu entit `ReviewRating`, ale mít je uložené rovnou u jednotlivých recenzí je při následné práci s těmito entitami efektivnější. Hodnoty těchto atributů může později udržovat například databázový trigger.

Kromě detailních recenzí je také možné kavárny bodově hodnotit, to modelujeme typem entity `CafeRating` (i zde je primární klíč složen z cizích klíčů, tento typ entity v podstatě představuje vazbu mezi dvěma jinými entitami). Kavárny nabízejí kávy připravované z různých směsí kávových zrn, přičemž kávová zrna mají rozličné vlastnosti, například kyselost, odrůdu nebo původ. O kavárně je dále třeba znát její otevírací hodiny, adresu a kapacitu. Za pozornost zde stojí atribut `average_rating`, který podobně jako bodová hodnocení u recenzí funkčně vychází z entit `CafeRating`, ale z implementačního hlediska je efektivnější jej udržovat pomocí databázových triggerů a ukládat jej jako souhrnnou hodnotu ke kavárnám samotným.

Zaměstnanecký vztah uživatele ke kavárně modelujeme typem entity `UserCafeRole`; tyto entity mají atribut `is_owner`, který vyjadřuje, že je zaměstnanec také majitelem a má tedy oprávnění provádět některé další akce.

Majitel kavárny může v kavárně uspořádat cuppingové akce, na kterých mohou registrovaní účastníci ochutnat různé druhy kávy. Ke cuppingové akci se váže nabídka káv skrz typ entity `CuppingEventOffer`, který vyjadřuje vztah M:M mezi akcí `CuppingEvent` a nabízenou kávou `CoffeeBlend` a umožňuje doplnit další informace. Uživatelé také mohou na tyto akce psát recenze podobně jako recenze k samotné kavárně. Protože jsou recenze na kavárnu a na akci velmi podobné koncepty, jsou modelovány jako specializace `CuppingReview` a `CafeReview`, tyto specializace jsou odlišeny rozdílnými vazbami (v důsledku pak cizími klíči), jsou disjunktní a totální.

Rezervace na cuppingové akce jsou modelovány pomocí typu entity `EventReservation`. Vazba *uživatel tvoří rezervaci* má členství 0, protože rezervace může být vytvořena i neregistrovaným uživatelem. Primární klíč je složen z atributů `email` a cizího klíče `event_id`, což je důvod, proč je v modelu explicitně tento cizí klíč uveden, ačkoliv vyplývá ze vztahu. Může se zdát, že je atribut `email` v případě vazby s uživatelem redundantní, ale je to záměr – do rezervace takto může být uveden jiný kontaktní e-mail, což může být žádoucí.

3 Návrh databázového schématu (2. část projektu)

Konceptuální model ve formě ER diagramu byl standardním postupem převeden na schéma databáze. Za zmínku zde stojí transformace atributu `Cafe.opening_hours` na tabulku `OpeningHours` s cizím klíčem ID kavárny pro zajištění 1NF (primární klíč této tabulky je složený a skládá se z ID kavárny, dne v týdnu a počátku otvírací doby pro možnost uložení více časových bloků v jeden den). Vztahy M:N následně byly převedeny pomocí vazebních tabulek.

Vztah generalizace/specializace mezi `Review` a `CuppingReview`, resp. `CafeReview` byl v databázi realizován pomocí dvou tabulek (jednu pro každou ze specializací), které obsahují veškeré atributy (zploštění). Tento přístup můžeme použít, protože se jedná o specializaci *totální*, v systému se tedy nemůže vyskytovat obecná recenze. Specializace je také *disjunktní*, modelování pomocí jedné společné tabulky by ale přinášelo další problémy: recenze mohou reagovat samy na sebe, muselo by tedy být zajištěno, že recenze na kavárnu bude opět reagovat jen na recenzi na kavárnu a podobně. V tomto systému navíc nevadí, když stejné primární klíče budou použity v `CuppingReview` i `CafeReview` a námi zvolený přístup je efektivnější než ukládaní do jediné tabulky. Pravdou ale je, že vytváří nutnost duplikovat některé pokročilé objekty, například proceduru pro změnu hodnocení (viz 4.2).

Oproti konceptuálnímu modelu proběhlo také několik dalších drobných změn:

- Typ entity `User` byl přejmenován na `SUser` aby nedocházelo ke kolizi s klíčovým slovem SŘBD.
- Entitám typu `SUser` byl přidán atribut `password` pro otisk hesla.
- Některým vazbám (specializací) tabulek `Review` bylo sníženo členství na 0. To se projevilo tak, že cizí klíče `made_by_user_id` a `reacts_to_id` jsou označeny jako *nullable*. Důvodem je zachování recenzí při odstranění uživatelského účtu, který je vytvořil.
- Entitám typu `CoffeeBlend` byl přidán atribut `roasters` pro uvedení pražírny.
- Entitám typu `CoffeeBean` byl přidán atribut `altitude` pro uvedení nadmořské výšky, což je ve světě výběrové kávy podstatný údaj o surovině. Odebrány byly atributy `[short_]description` a vztah s entitami `CoffeeBlend` byl změněn na 1:M – z praktického pohledu totiž nedává smysl udržovat zvlášť seznamy zrn a blendů z nich (to by se mohlo hodit například u informačního systému pražíren, ale určitě ne kaváren).

Výsledné schéma databáze je ilustrováno v příloze C.

4 Pokročilé objekty schématu databáze (4. část projektu)

Podle zadání byly implementovány uložené procedury a funkce, triggery a pohledy, které reprezentují některé akce, které by mohl provádět reálný systém (klientská aplikace) a jejichž logika může být výhodně přenesena na SŘBD. V následujících kapitolách budou tyto objekty podrobněji popsány.

4.1 Pohledy

Ve schématu se nacházejí následující pohledy:

`SUserLimited` umožňuje získávat data o uživatelích, ale nedovolí přístup k jejich heslům[1].

`CafeOwner` provádí jednoduchý dotaz, který vybírá uživatele, kteří mají k nějakým kavárnám vztah s nastaveným příznakem `is_owner`, a přidává k nim ID daných kaváren.

Materializovaný pohled `UpcomingCuppingEvent` vybírá nastávající cuppingové akce, ke kterým přidává informaci o počtu zbývajících míst na této akci.

První dva zmíněné pohledy jsou triviální a souvisí s případem užití popsaným v kapitole 4.4.

Materializovaný pohled `UpcomingCuppingEvent` by byl využit zejména v případu užití *Zobrazit seznam plánovaných cupping akcí* – poskytuje data například pro stránku s přehledem nejbližších cuppingových akcí, na které je žádoucí zobrazovat vybrané informace o události spolu s obsazeností. Právě obsazenost tento materializovaný pohled výrazně komplikuje, neboť vyžaduje komplexní dotaz, který ji spočítá podle záznamů v tabulce `EventReservation`.

Kvůli komplexitě dotazu není možné tento pohled aktualizovat `ON COMMIT`, tedy automaticky při změně záznamů v bázových tabulkách. To ale nevadí, protože návrh počítá s tím, zásahy do tabulky rezervací, které mění počet míst, budou spravovány výhradně příslušnou procedurou (viz 4.2), která pohled vždy obnoví.

Kromě aktualizací při změně volného počtu míst je nutná aktualizace také každý den, protože dotaz pohledu obsahuje klauzuli `WHERE event_date > CURRENT_DATE`, která filtruje pouze akce, které teprve proběhnou. To by měla zajišťovat klauzule `REFRESH ON DEMAND START WITH (SYSDATE) NEXT (SYSDATE + 1) WITH ROWID`; pod definicí pohledu je v komentáři uvedeno také alternativní řešení využívající plánovač SŘBD.

Ještě efektivnějším by mohlo být použití materializovaného pohledu s protokolem. Poddotaz z našeho řešení bychom oddělili do nového materializovaného pohledu `EventSeatsTaken`. Na tabulce `EventReservation` bychom vytvořili protokol materializovaného pohledu. Ten slouží ke zrychlení agregačních funkcí: při operacích nad tabulkou se do něj ukládají speciální záznamy o určených sloupcích, pomocí kterých je možné v materializovaném pohledu agregační funkci nahradit triviálním výpočtem (například k existující sumě se akorát přičtou nové hodnoty), místo aby musela funkce projít celou původní tabulku. Nový materializovaný pohled by pak mohl být nastaven jako `REFRESH FAST ON COMMIT ENABLE QUERY REWRITE`. Měli bychom tak automaticky se aktualizující materializovaný pohled obsahující vždy aktuální záznam o počtu zarezervovaných míst. Pohled `UpcomingCuppingEvent` by pak sice musel být stále aktualizován ručně, ale jeho provádění by bylo výrazně rychlejší.

Toto řešení se bohužel ukázalo jako nefunkční. SŘBD při spuštění procedury pro aktualizaci rezervací `CafeUtils.make_reservation` zahlásil, že při aktualizaci protokolu materializovaného pohledu nemůže správně

[1] Pomocí systému oprávnění se sice dají nastavovat práva ke sloupcům, ale nelze tak omezit čtení: `https://docs.oracle.com/database/121/SQLRF/statements_9014.htm#SQLRF55029`

serializovat transakci. Z časových důvodů jsme tedy použili původní, méně efektivní verzi, ale návrh s optimálnější verzí přikládáme v příloze F.

4.2 Procedury

Skript obsahuje definice dvou procedur a jedné funkce. Všechny jsou deklarovány v pojmenovaném balíčku `CafeUtils`.

`set_fav_drink` nastavuje textový atribut `favourite_drink` uživatele s daným ID na hodnotu atributu `name` nabídky některé z kaváren `DrinkOffer`. Pokud zadaný uživatel neexistuje, nic se nestane. Pokud neexistuje zadaný nápoj, nastaví se hodnota NULL. Tato procedura je vcelku triviální a využití by mohla mít například ve formě akce po stisknutí tlačítka „Nastavit jako oblíbený nápoj“ v detailu kavárny.

`set_cafe_review_rating` přidává nebo aktualizuje hodnocení *recenze* kavárny. Tato procedura v podstatě obaluje složitější příkaz `MERGE` serializovatelnou transakcí a synchronizační logikou (popsáno dále).

`make_reservation` kompletně řeší vytváření, úpravy a mazání rezervací na cuppingové akce.

Poslední zmiňovaná procedura (respektive funkce) je komplexnější. Přebírá jako parametr ID události, e-mail (tyto dvě položky tvoří klíč rezervací), telefonní číslo a počet rezervovaných míst. Pokud je požadovaný počet míst nula, znamená to zrušení případné rezervace. Jinak je získán celkový počet míst události, od kterého jsou odečteny už zarezervovaná místa. Zde především demonstrujeme možné použití kurzoru, kterým procházíme tabulku `EventReservation` s informacemi o rezervovaných počtech míst; tuto funkcionalitu by ale bylo možné nahradit i dotazem s agregační funkcí. Pokud je počet zbývajících míst dostatečný, pokusíme se najít uživatele podle zadaného e-mailu (abychom případně mohli naplnit volitelný cizí klíč rezervace) a vložíme novou rezervaci, nebo aktualizujeme už existující. Navíc se chování liší podle toho, jestli se v už existující rezervaci změnil počet míst, nebo ne. Funkce vrací BOOLEAN hodnotu TRUE, pokud byla operace provedena úspěšně, FALSE se vrací například pokud neexistuje událost s daným ID. Zároveň funkce vyvolá obnovení materializovaného pohledu `UpcomingCuppingEvent`.

Celá tato logika musí být nutně atomická a serializovatelná – dokud provádíme operace upravující počty míst, nemůžeme dopustit, aby se data o rezervacích jakkoliv změnila (došlo by k *nekonzistentní analýze*). Toho dosahujeme několika způsoby. Cílový způsob použití této funkce je takový, že bude izolovaně zavolána v reakci například na stisk tlačítka uživatelem. Můžeme si tedy dovolit použití `PRAGMA AUTONOMOUS_TRANSACTION`, což izoluje tělo funkce od transakce, ze které byla zavolána, a může si sama spravovat, potvrzovat nebo rušit transakce. To také dělá: před prováděním jakýchkoliv dotazů je spuštěna transakce pomocí `SET TRANSACTION ISOLATION LEVEL SERIALIZABLE`, což zajišťuje uspořádatelnost operací případných souběžných transakcí. Může nastat situace, kdy SŘBD nebude schopen uspořádatelnost zajistit, v takovém případě vyvolává výjimku ORA-08177. Z tohoto důvodu je celé tělo funkce obaleno cyklem, který se pokouší transakci provést: pokud se to podaří, funkce vrací TRUE; pokud je zachycena zmíněná výjimka, proběhne rollback a cyklus se vrací na začátek a pokusí se požadované operace provést znovu v nové transakci.

Popsaný princip je použit i v proceduře `set_cafe_review_rating`, jejíž použití je zamýšleno obdobně. V sekci „demonstrace procedur“ v závěru skriptu je také demonstrováno, že v některých situacích, jež jsou v kontextu zamýšleného použití těchto procedur nestandardní, například při nepotvrzené transakci manipulující se stejnými daty, může dojít k deadlocku.

Pro praktické použití by bylo vhodné vytvořit duplikát procedury `set_cafe_review_rating`, který by prováděl stejné akce nad tabulkami reprezentujícími recenze cuppingových akcí a jejich hodnocení; protože by však šlo o prostý „*copy-paste*“, rozhodli jsme se jím skript zbytečně neprodlužovat.

4.3 Triggery

V rámci řešení byly vytvořeny čtyři triggery. Jeden z nich dle zadání zajišťuje generování primárního klíče v tabulce `Cafe` a tedy napodobuje funkcionalitou původně přítomné `GENERATED ALWAYS AS IDENTITY`[2]. Další dva implementované triggery zajišťují automatickou aktualizaci atributů, které jsou v konceptuálním modelu vyznačeny šedou barvou, tedy `average_rating` a `points_up`, resp. `points_down`.

Odevzdaná implementace má bohužel značné problémy s efektivitou, neboť po každé změně přepočítává znovu průměr, resp. body od začátku, bez využití předchozí uložené informace. Toto řešení je ovšem jediné možné, protože trigger procházející jednotlivé řádky příkazu a využívající předchozích výpočtů za určitých situací kolidoval s `ON DELETE CASCADE`. Triggery využívající efektivnější řešení je možné najít v příloze G. Další možností, jak tento problém řešit, by bylo omezit přímý přístup k hodnotám těchto atributů a vytvořit si další procedury, které budou zajišťovat udržování těchto hodnot v konzistentním stavu a které budou tvořit jediné „rozhraní“ pro úpravu dat v těchto tabulkách.

Poslední implementovaný trigger je určen k zjednodušení sekvenčního vkládání otevíracích hodin. Případem užití pro tuto funkcionalitu je vytvoření nové kavárny v systému, při kterém majitel chce typicky postupně zadat i otevírací dobu pro jednotlivé dny. Tento trigger to usnadňuje a není nutné ručně specifikovat den, ten je dopočítán jako následující den od posledního vloženého dne (tedy pokud nejvyšší vložený den je středa, budou nová data vložena jako čtvrtek).

4.4 Oprávnění pro druhého člena týmu

V skriptu se nachází příkazy, které jednomu ze členů týmu (xnecas27) udělují práva ke čtení tabulek o kavárnách a jejich hodnoceních, ale explicitně mu berou práva ke čtení tabulek o cuppingových akcích. Naopak jsou mu udělena práva ke čtení materializovaného pohledu `UpcomingCuppingEvent` a ke spouštění procedur z balíčku `CafeUtils` – tímto způsobem je tedy umožněno uživateli přistupovat k datům o cuppingových akcích „řízeným způsobem“. Také nemůže přímo číst tabulku uživatelů, ale může číst a upravovat záznamy z pohledu `SUserLimited`.

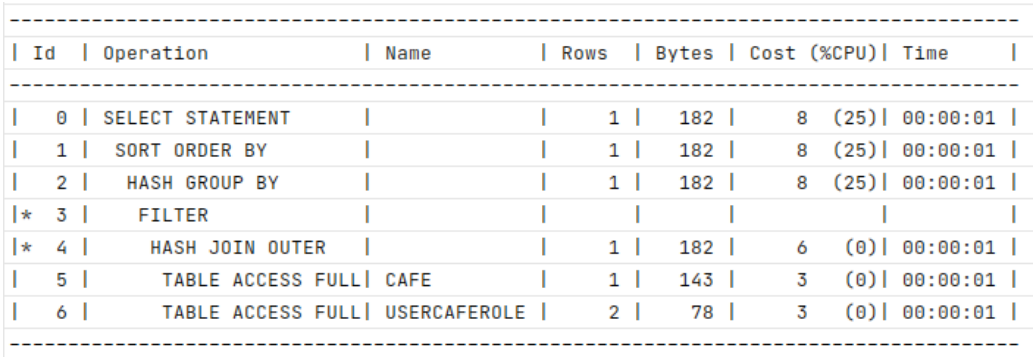
Tento uživatel tedy reprezentuje *částečného administrátora* systému, který může modifikovat některé informace kaváren a uživatelů, ale nemůže zasahovat do existujících recenzí a nemůže provádět zásahy do tabulek cuppingových akcí, které by mohly narušit konzistenci požadovaných dat.

S tímto rámcem oprávnění může tento uživatel například spustit klientskou aplikaci, kterou jsme implementovali v rámci rozšíření, viz kapitola 5.

4.5 Optimalizace dotazu

Pro ukázku optimalizace dotazu pomocí indexu jsme si vybrali dotaz počítající počet majitelů pro jednotlivé kavárny. Při tomto dotazu se dělá spojení přes tabulky `Cafe` a `UserCafeRole` a vybírají se ty n-tice, kde uživatel je majitelem. Prostor pro optimalizaci tohoto dotazu jsme viděli ve faktu, že většina uživatelů nebude majiteli, a tedy bez optimalizace se bude procházet velké množství řádků, kde ovšem pouze pár bude zahrnuto do výsledku (velmi podobné principu řídkému pole). Ideální by tedy bylo rychle nalézt pouze relevantní položky ve spojení tabulek a z toho důvodu jsme zavedli index nad tabulkou `UserCafeRole` a jejími atributy `user_id`, `cafe_id` a `is_owner`. Jak je možné vidět na obrázku 1, celková cena dotazu se díky indexu snížila, protože nebylo nutné dělat pomalé spojení, ale místo toho byl využit vytvořený index.

[2] Ve srovnání s touto klauzulí nezajišťuje, že se znovu nepoužije ID, které už bylo dříve použito, pokud dojde ke smazání posledního záznamu.



Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	182	8 (25)	00:00:01
1	SORT ORDER BY		1	182	8 (25)	00:00:01
2	HASH GROUP BY		1	182	8 (25)	00:00:01
* 3	FILTER					
* 4	HASH JOIN OUTER		1	182	6 (0)	00:00:01
5	TABLE ACCESS FULL	CAFE	1	143	3 (0)	00:00:01
6	TABLE ACCESS FULL	USERCAFEROLE	2	78	3 (0)	00:00:01

(a) Plán před optimalizací

Id	Operation	Name	Rows	Bytes	Cost (%CPU)	Time
0	SELECT STATEMENT		1	182	6 (34)	00:00:01
1	SORT ORDER BY		1	182	6 (34)	00:00:01
2	HASH GROUP BY		1	182	6 (34)	00:00:01
* 3	FILTER					
4	NESTED LOOPS OUTER		1	182	4 (0)	00:00:01
5	TABLE ACCESS FULL	CAFE	1	143	3 (0)	00:00:01
* 6	INDEX FULL SCAN	IS_OWNER_INDEX	1	39	1 (0)	00:00:01

(b) Plán po optimalizaci

Obrázek 1: Optimalizace dotazu

5 Rozšíření – klientská aplikace

Jako rozšíření nad rámec zadání jsme se rozhodli implementovat klientskou aplikaci – *scraper* pro naplnění databáze reálnými daty. Využili jsme jazyk Python a ORM (Object Relational Mapper) framework `sqlalchemy`, čímž jsme vyzkoušeli další možný přístup k práci s databázemi. Skript data stahuje z veřejné webové stránky Restu.cz[3].

5.1 Požadavky pro spuštění

Aplikace je kompatibilní s Python verzí 3.6 a vyšší, pro spuštění doporučujeme využívat virtuální prostředí (vytvořitelné například pomocí `python3 -m venv venv`). Knihovny třetích stran jsou sepsány v souboru *requirements.txt*, konkrétně se jedná o knihovny `requests` pro stahování HTML stránek, `beautifulsoup4` pro jejich zpracování, `sqlalchemy` pro práci s databází a `cx_oracle` pro využití Oracle driveru v rámci `sqlalchemy`. Na některých systémech může být nutné také nainstalovat Oracle client pro zprovoznění `cx_oracle`, tato instalace je ovšem závislá na systému, viz `cx_oracle` dokumentace.

5.2 Implementace a spuštění

V souboru jsou nejdříve definovány třídy pomocí rozhraní `sqlalchemy` definující mapování na tabulky databáze. Definovali jsme zde pouze podmnožinu tabulek, kterou plníme daty, v komplexnější aplikaci by ovšem podobným způsobem byly dodefinovány i třídy pro zbylé tabulky (například pro akce a podobně).

[3]Tato data samozřejmě podléhají autorským právům. Naše aplikace je stahuje čistě za didaktickými účely a pro soukromé užití.

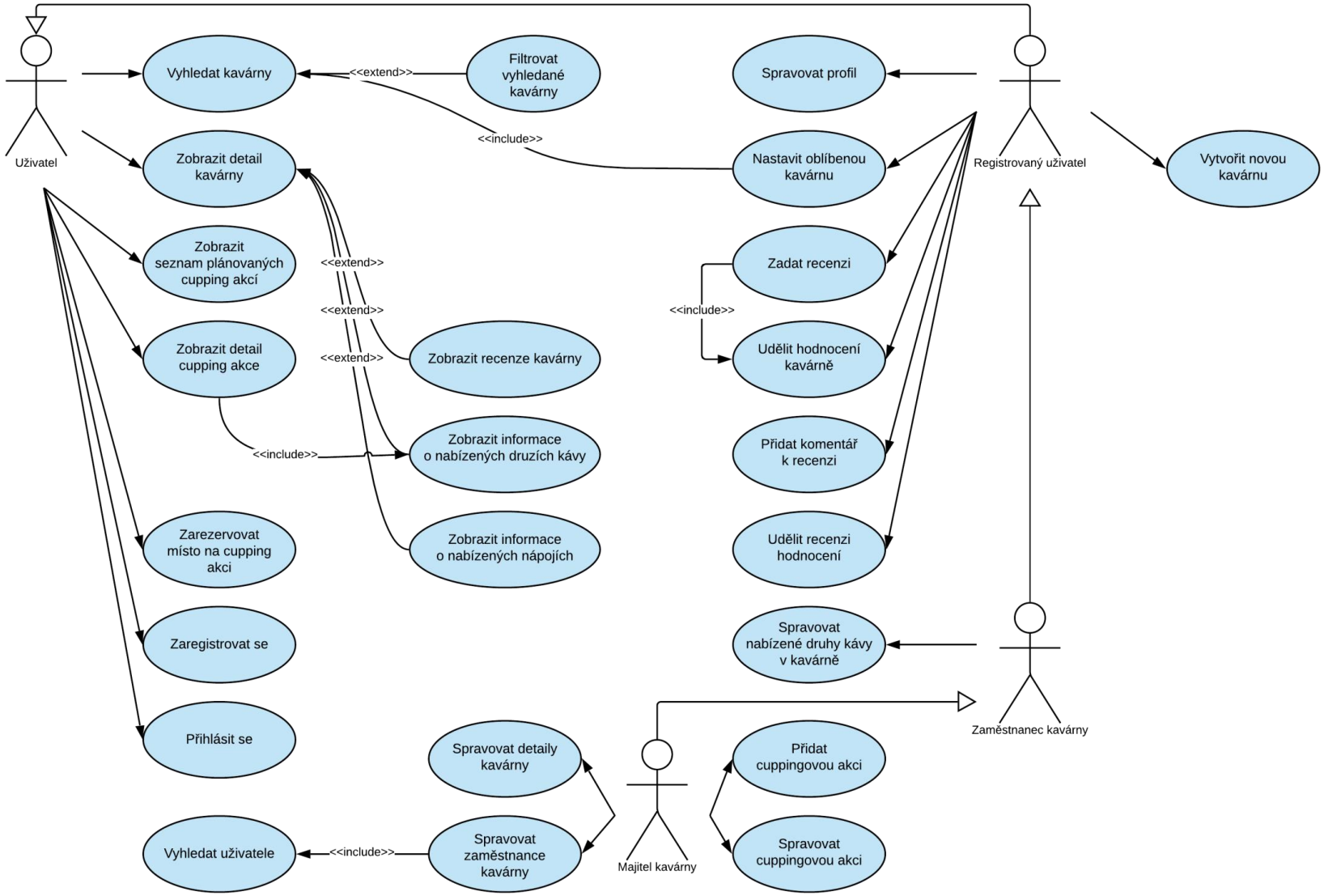
Pro samotné načítání dat je použita knihovna `requests` a analýza HTML je provedena pomocí knihovny `beautifulsoup4`. Protože odkazovaná stránka používá stránkování při zobrazování kaváren, v prvním kroku je zjištěn počet stránek, které musí skript projít. V následující fázi už jsou procházeny jednotlivé stránky a kavárny na nich uložené. Díky vhodné struktuře HTML stránky je možné jednoduše hledat relevantní informace, typicky pomocí tříd značek. Načítání jedné kavárny a veškerých informací o ní je zabezpečeno *transakcí*. O kavárně zjišťujeme její název, adresu a popis. Po přidání objektu kavárny do databáze je zavolána metoda `flush` pro doplnění ID kavárny, které je vyžadováno jako cizí klíč do dalších objektů. U kaváren také vybíráme pár recenzí a bodových hodnocení (ta jsou převáděna ze škály do 15 na škálu do 5 bodů). Lehce komplexnější zpracování vyžadují otevírací hodiny, protože některé kavárny mají otevírací dobu i přes půlnoc a pro použití v našem modelu dat je tedy nutné tento čas rozdělit na dva časové bloky (do konce jednoho dne a od začátku následujícího dne).

Nakonec je v souboru ukázka dotazu, který nad nově načtenými daty vybere kavárny s průměrným hodnocením alespoň 4, které mají právě teď otevřeno. Jedná se už o trochu složitější dotaz, který je ovšem v zápise pomocí `sqlalchemy` poměrně čitelný, pravděpodobně více než kdybychom jej měli psát v čistém SQL.

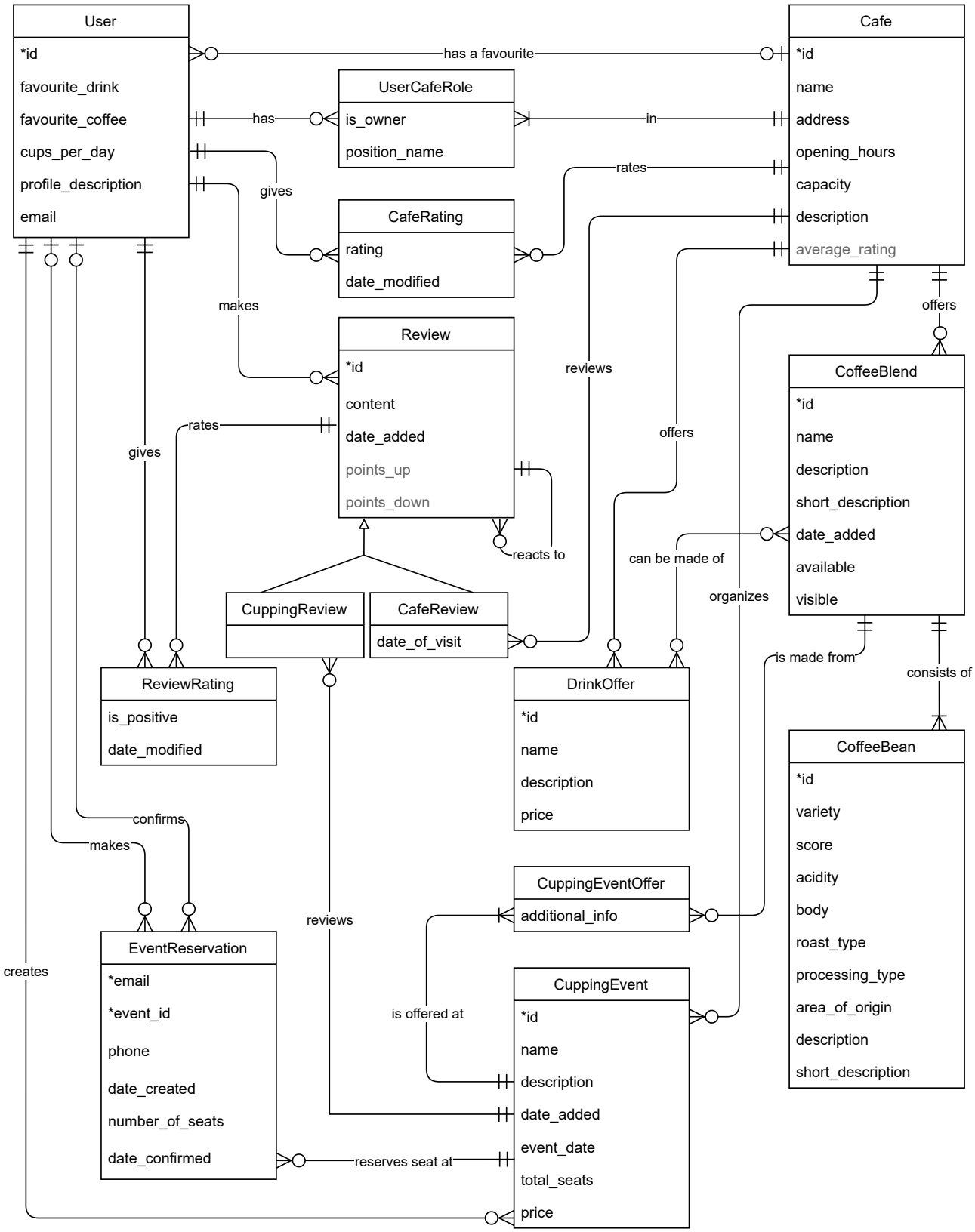
Pro spuštění je na začátku souboru nutné změnit uživatelské jméno a heslo a případně také na konci změnit současně používané schéma (nastavované pomocí příkazu *ALTER SESSION*), obě tyto lokace jsou označeny *TODO* komentářem. Aplikace v průběhu vypisuje data, která ukládá do databáze. V příloze D je uveden příklad výpisu pro dvě kavárny.

Výsledek dotazu na části dat je možné vidět v příloze E, dotaz byl spuštěn ve středu v cca 19 hodin.

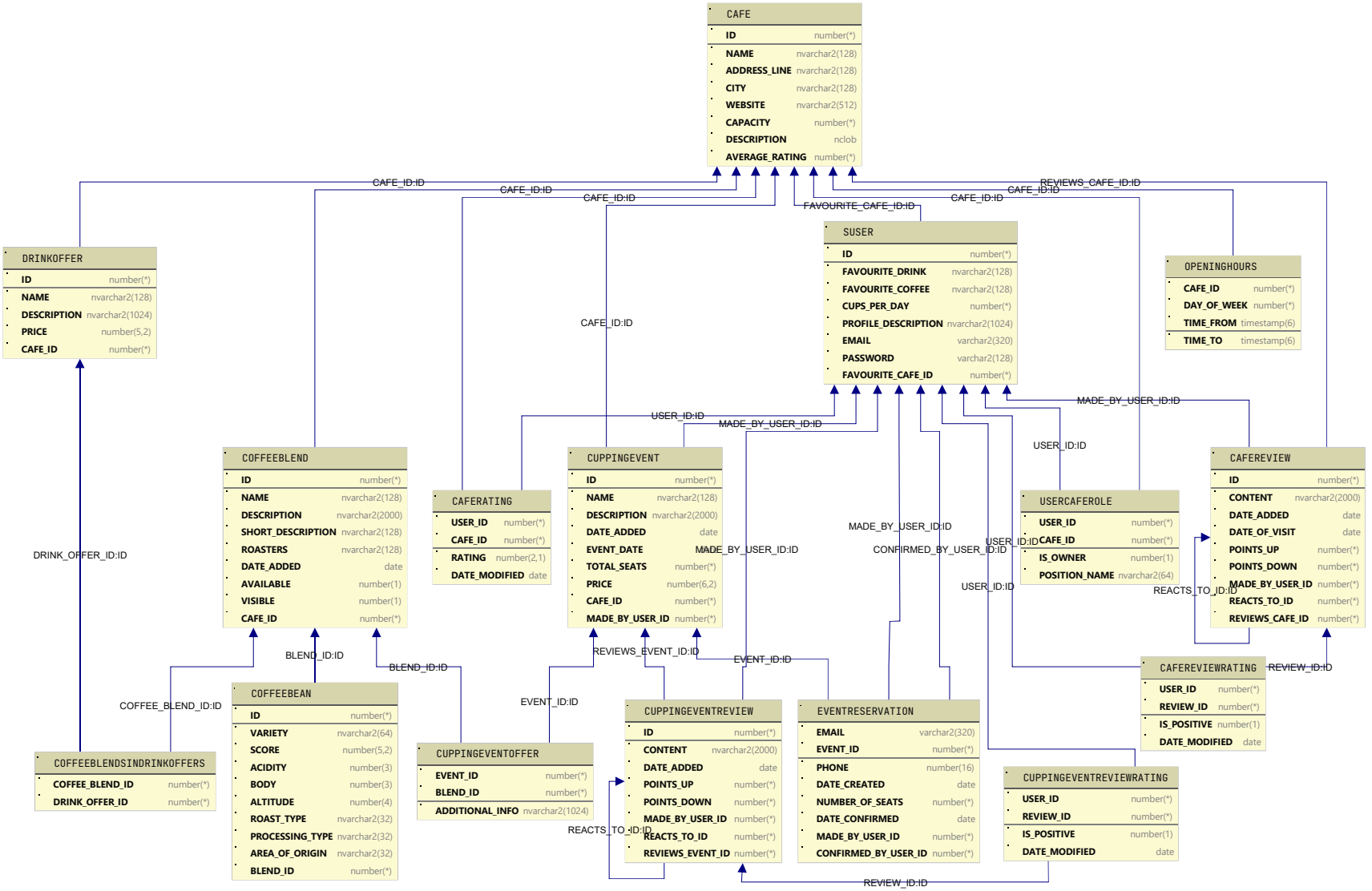
A Use Case diagram



B ER diagram


User
*id
favourite_drink
favourite_coffee
cups_per_day
profile_description
email
has a favourite
Cafe
*id
name
address
opening_hours
capacity
description
average_rating
UserCafeRole
is_owner
position_name
has
in
gives
CafeRating
rating
date_modified
rates
makes
Review
*id
content
date_added
points_up
points_down
reacts to
reviews
offers
CoffeeBlend
*id
name
description
short_description
date_added
available
visible
rates
gives
offers
can be made of
organizes
CuppingReview
CafeReview
date_of_visit
is made from
consists of
ReviewRating
is_positive
date_modified
DrinkOffer
*id
name
description
price
CoffeeBean
*id
variety
score
acidity
body
roast_type
processing_type
area_of_origin
description
short_description
confirms
makes
CuppingEventOffer
additional_info
reviews
EventReservation
*email
*event_id
phone
date_created
number_of_seats
date_confirmed
creates
is offered at
CuppingEvent
*id
name
description
date_added
event_date
total_seats
price
reserves seat at

C Konečné schéma databáze



D Výpis klientské aplikace (získávání dat)

```
Scraping data for Pizza Coloseum Brno on /coloseum-brno/
Pizza Coloseum Brno Vídeňská 223/1, Brno Brno
https://www.restu.cz/coloseum-brno/ 44
Vlastní rozvoz: https://www.coloseum.cz/vlastni-rozvoz.html
Pizza Coloseum Brno nabízí skvělé pochoutky v příjemném prostředí
s milou obsluhou. Místo Sharinghamu nově Pizza Coloseum.
0 2021-04-28 11:00:00 2021-04-28 20:00:00
1 2021-04-28 11:00:00 2021-04-28 20:00:00
2 2021-04-28 11:00:00 2021-04-28 20:00:00
3 2021-04-28 11:00:00 2021-04-28 20:00:00
4 2021-04-28 11:00:00 2021-04-28 20:00:00
5 2021-04-28 11:00:00 2021-04-28 20:00:00
6 2021-04-28 11:00:00 2021-04-28 20:00:00
Rating: 4.666666666666667
Review: Přišli jsme a i v tom největším tlaku, díky krizi, personál
pohotový milý. Neustále se vracíme. Děkujeme především kuchařům,
že podávají takový výkon. Děkujeme
Rating: 4.666666666666667
Rating: 4.333333333333334

Scraping data for Jedna Báseň wine & coffee bar
on /jedna-basen-wine-coffee-bar/
Jedna Báseň wine & coffee bar Sukova 2, Brno Brno
https://www.restu.cz/jedna-basen-wine-coffee-bar/
45
Jedna Báseň je kavárna a vinárna s kuchyní jedna báseň
v Brně - střed. Nabízí italskou kávu Vergnano, moravská
sudová i lahvová vína. Vaří převážně ze sezónních surovin
a myslí i na bezlepkáře a vegetariány. Pochutnáte si tu
na skvělých snídaních i poledním menu.
0 2021-04-28 08:00:00 2021-04-28 16:00:00
1 2021-04-28 08:00:00 2021-04-28 16:00:00
2 2021-04-28 08:00:00 2021-04-28 16:00:00
3 2021-04-28 08:00:00 2021-04-28 16:00:00
4 2021-04-28 08:00:00 2021-04-28 16:00:00
Rating: 4.333333333333334
Rating: 4.666666666666667
Rating: 4.0
```

E Výpis klientské aplikace (dotaz na data)

```
Pizza Coloseum Brno Vídeňská 223/1, Brno 4.57 2021-04-28 11:00:00
2021-04-28 20:00:00
Cafe Park Slovaňák Slovanské náměstí 1804/7 4.5 2021-04-01 08:00:00
2021-04-01 19:00:00
Vinárna Perkmistr Dusíkova 910/17, Brno 4.33 2021-04-28 11:00:00
2021-04-28 20:00:00
```

F Návrh efektivnějších materializovaných pohledů

```
DROP MATERIALIZED VIEW UpcomingCuppingEvent;
DROP MATERIALIZED VIEW EventTakenSeats;
DROP MATERIALIZED VIEW LOG ON EventReservation;

CREATE MATERIALIZED VIEW LOG ON EventReservation WITH ROWID, PRIMARY KEY
    (number_of_seats) INCLUDING NEW VALUES;

CREATE MATERIALIZED VIEW EventTakenSeats
    REFRESH FAST ON COMMIT ENABLE QUERY REWRITE
AS
SELECT EVENT_ID, SUM(NUMBER_OF_SEATS) TAKEN
FROM EventReservation
GROUP BY EVENT_ID;

CREATE MATERIALIZED VIEW UpcomingCuppingEvent
    (Cafe_ID, Cafe_Name, City, Rating, Event_ID, Event_Name,
     Event_Date, Description, Seats_Left, Seats_Total, Price)
    REFRESH ON DEMAND
AS
SELECT E.CAFE_ID,
       C.NAME,
       C.CITY,
       C.AVERAGE_RATING,
       E.ID,
       E.NAME,
       E.EVENT_DATE,
       E.DESCRIPTION,
       COALESCE(E.TOTAL_SEATS - ST.TAKEN, E.TOTAL_SEATS),
       E.TOTAL_SEATS,
       E.PRICE
FROM CUPPINGEVENT E
         JOIN CAFE C ON C.ID = E.CAFE_ID
         LEFT JOIN EventTakenSeats ST ON E.ID = ST.EVENT_ID
WHERE EVENT_DATE > CURRENT_DATE
ORDER BY EVENT_DATE DESC;
```

G Efektivnější triggery

```
CREATE OR REPLACE TRIGGER cafe_average
    BEFORE UPDATE OR INSERT OR DELETE
    ON CafeRating
    FOR EACH ROW
DECLARE
    rating_count INT;
    prev_average Cafe.average_rating%TYPE;
BEGIN
    SELECT COUNT(*) INTO rating_count FROM CafeRating WHERE CafeRating.cafe_id =
                                                            :NEW.cafe_id;
    SELECT average_rating INTO prev_average FROM Cafe WHERE Cafe.id =
                                                            :NEW.cafe_id;
    UPDATE Cafe
    SET average_rating = ((prev_average * rating_count) + :NEW.rating) /
                         (rating_count + 1)
    WHERE Cafe.id = :NEW.cafe_Id;
END;

CREATE OR REPLACE TRIGGER points_insert
    BEFORE INSERT
    ON CafeReviewRating
    FOR EACH ROW
BEGIN
    IF :NEW.is_positive = 0 THEN
        UPDATE CafeReview SET points_down = points_down + 1 WHERE
          CafeReview.id = :NEW.review_id;
    end if;
    IF :NEW.is_positive = 1 THEN
        UPDATE CafeReview SET points_up = points_up + 1 WHERE
          CafeReview.id = :NEW.review_id;
    end if;
END;
```