Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Realizace modelu akciového trhu a automatického hraní do aplikace DSGEgame

Plzeň, 2017 Pavel Straka

Místo této strany bude zadání práce.

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 15. května 2017

Pavel Straka

Poděkování

Rád bych touto cestou poděkoval panu Ing. et Ing. Jiřímu Pešíkovi, autorovi DSGEgame, za jeho ochotu věnovat čas schůzkám, kde mně objasnil detaily, jak jsou některé funkcionality aplikace implementovány a poskytl cenné rady, jak do stávající podoby aplikace implementovat nová rozšíření. Můj vděk patří také vedoucímu práce, panu JUDr. Ing. Davidu Martinčíkovi, Ph.D. za systematické vedení práce a pomoc s návrhem podoby modelu akciového trhu a automatického hraní subjektů. V neposlední řadě děkuji panu Ing. Martinu Zímovi, Ph.D., konzultantovi práce, za možnost častých schůzek, kde mně poskytl mnoho rad ohledně optimalizace výkonu databáze a formální podoby práce.

Abstract

Name: Realization of the stock market model and automatic playing into DSGEgame application

DSGEgame is specialized economic experimental software. It is a simulation game developed on Faculty of Economics of University of West Bohemia. The software is implemented as object-oriented PHP using MySQL database. This thesis aims to present extension of DSGEgame for stock market model and automatic playing. In addition of implementation of these extensions, database optimisation was also performed. Theoretical part of the paper describes the theory of stock markets and automatic playing and also describes the techniques for performance optimisation of MySQL database. The practical part is based on the theoretical part and describes design and implementation of software extensions and realized database optimisation.

Keywords: DSGE game, stock market, automatic playing, MySQL database optimization

Abstrakt

Název: Realizace modelu akciového trhu a automatického hraní do aplikace DSGEgame

DSGEgame je specializovaný software, umožňující provádění ekonomických experimentů velkého rozsahu. Jedná se o simulační hru, vyvinutou na Fakultě ekonomické Západočeské univerzity v Plzni. Software je realizován ve skriptovacím jazyce PHP, využívající principy objektově orientovaného programování a data jsou uložena v MySQL databázi. Předložená práce prezentuje rozšíření DSGEgame o model akciového trhu a systém automatického hraní subjektů. Kromě realizace těchto rozšíření byla provedena optimalizace databáze této aplikace. V úvodní části práce je popsána problematika akciových trhů a automatického hraní a popsány postupy optimalizace výkonu MySQL databáze. Na tomto teoretickém základu staví praktická část, popisující návrh a realizaci rozšíření hry a provedenou optimalizaci databáze aplikace.

Klíčová slova: DSGEgame, akciový trh, automatické hraní subjektů, optimalizace výkonu MySQL databáze

Obsah

Seznam obrázků

Seznam tabulek

1 Úvod

Experimentální ekonomie představuje významnou oblast ekonomické teorie. Její závěry jsou postaveny na studiu lidského chování za laboratorních podmínek v situacích, které ve zjednodušené formě imitují reálné tržní situace. Ve skutečném světě nejsme schopni odděleně zkoumat chování určitého subjektu, protože nemůžeme vyloučit vliv jiných subjektů a událostí v ekonomice na jeho chování - nejsme schopni zajistit stálost podmínek. Výhodou experimentálního přístupu je, že toto umožňuje - ekonomické prostředí je možné zafixovat a zkoumat chování vybraných subjektů trhu za daných podmínek [22].

Specializovaným softwarem, umožňujícím provádění ekonomických experimentů velkého rozsahu, je *DSGEgame*. Jedná se o simulační hru, vyvinutou na Fakultě ekonomické Západočeské univerzity v Plzni. Kromě oblasti experimentální ekonomie se DSGEgame uplatňuje také v edukačním procesu - slouží k podpoře výuky ekonomie na FEK ZČU. Zdrojový kód DSGEgame má v současnosti necelých 20 tis. řádků. Aplikace je dostupná na webové adrese `www.dsgegame.zcu.cz` [10].

Prvotní verze DSGEgame byla uvedena v roce 2010. Ve snaze nabídnout možnost simulace dalších reálných tržních situací je aplikace stále rozšiřována a zdokonalována. V současné podobě ale aplikace neobsahuje možnost obchodování na akciových trzích. Jednodušší administraci hry a řešení problému s neaktivitou části hráčů by umožnil systém automatického hraní subjektů. Zkušenosti s aplikací ukazují, že při zapojení velkého množství hráčů má aplikace problémy s výkonem.

Tato diplomová práce si tedy klade následující cíle:

1. navrhnout rozšíření DSGEgame o akciový trh a systém automatického hraní subjektů,
2. implementovat navrhnutá rozšíření a ověřit je v rámci pilotního projektu,
3. analyzovat databázi stávající aplikace a provést její optimalizaci.

V analytické části práce je popsána stávající podoba aplikace, problematika akciových trhů a automatického hraní a popsány postupy optimalizace výkonu MySQL databáze. Na tomto základu staví praktická část, popisující návrh a realizaci rozšíření hry a provedenou optimalizaci databáze. Uživatelská dokumentace je přílohou práce.

2 Stávající aplikace DSGEgame

2.1 DSGE modely

2.1.1 Základní popis

DSGE[1] modely, neboli dynamické stochastické modely všeobecné rovnováhy v současnosti představují dominantní makroekonomický[2] přístup. Tento typ modelu je aktuálně používán řadou centrálních bank a jiných ekonomických institucí k predikcím vývoje ekonomiky a k řízení měnové (monetární) a rozpočtové (fiskální) politiky [10, 24].

„Monetární politika je součástí a nástrojem hospodářské politiky. Je souhrnem opatření a zásad, které mají prostřednictvím měnových nástrojů prosazovat plnění měnových cílů. Monetární politika je nástroj centrální banky a jejím základním cílem je hlídání a aktivní ovlivňování míry znehodnocení peněz - inflace. V české republice plní funkci centrální banky Česká národní banka (ČNB).“ [20]

„Fiskální (rozpočtová) politika je nástrojem hospodářské politiky v rukou vlády. Fiskální politika se zabývá utvářením jak příjmové stránky rozpočtu (daně, cla, sociální pojištění), tak výdajovou stránkou. Rozpočet je schvalován ve formě zákona a musí být schválen Parlamentem.“ [14]

DSGE modely jsou principiálně postaveny na mikroekonomických[3] základech a popisují chování ekonomiky jako celku, který vznikne interakcí jednotlivých ekonomických subjektů. Základem DSGE modelů je tedy neustálé optimalizační chování subjektů. Chování subjektů je založeno na tzv. *deep parameters*, které jsou neměnné vůči změně hospodářské politiky. Díky tomu jsou DSGE modely robustní vůči tzv. Lucasově kritice[4].

Jako dynamické jsou modely označovány, protože do nich vstupuje čas, chování subjektu v minulosti má vliv na jeho situaci a možnosti volby v bu-

[1] Dynamic Stochastic Generic Equilibrum

[2] *„Makroekonomie zkoumá chování ekonomiky jako celku.“* [27, str. 15]

[3] *„Mikroekonomie se zabývá chováním jednotlivých subjektů jako jsou trhy, firmy a domácnosti.“* [27, str.15]

[4] Lucasova kritika je koncept formulovaný americkým ekonomem Robertem Lucasem v roce 1976. Lucas kritizoval především keynesiánské makroekonomické modely, které při predikcích změny ekonomického vývoje nezahrnovaly reakci ekonomických subjektů na tyto změny [24].

doucnosti. Kromě omezujících podmínek či okolí je tedy subjekt ovlivňován také svými vlastními předchozími rozhodnutími.

Jako stochastické jsou modely označovány, protože jejich nedílnou součástí jsou tzv. stochastické šoky. Pomocí těchto šoků jsou modelovány náhodné veličiny, které ovlivňují ekonomiku, ale nejsou endogenní součástí ekonomického systému a neumíme je tudíž vysvětlit pomocí ostatních ekonomických veličin. Příkladem těchto veličin jsou technologické šoky (ovlivňující produktivitu kapitálu, práce a tím i řadu další veličin) nebo ropné šoky (prudký vzrůst ceny ropy) [10, 24].

První dynamický model všeobecné rovnováhy formuloval anglický filosof a matematik Frank Plumpton Ramsey roku 1928. Přestože byl Ramseyův model deterministický (tedy ne stochastický), obsahoval řadu metod a přístupů, které jsou používány dodnes [24].

Subjekty, vystupující v modelech

Ekonomické subjekty, zastoupené v modelech jsou:

- spotřebitel (domácnost),
- výrobce (firma),
- vláda (stát),
- centrální banka.

Spotřebitelé a výrobci se snaží za daných podmínek (omezení) a v daném časovém horizontu maximalizovat svou účelovou funkci. V případě spotřebitelů jde o maximalizaci užitku[5] v rámci rozpočtového omezení, v případě výrobců pak o maximalizaci zisku v rámci omezení produkční funkcí[6] a technologií. Vláda provádí fiskální a centrální banka monetární politiku [10].

2.1.2 Walrasův aukcionář

Pro konstrukci ekonomického modelu s dobrovolnou tržní směnou je potřeba zvolit typ aukce, která určuje cenu zboží a výrobních faktorů na jednotlivých

[5] *„Užitek znamená uspokojení. Přesněji řečeno vyjadřuje, jak spotřebitelé hodnotí různé statky a služby. V teorii poptávky říkáme, že lidé maximalizují svůj užitek, což znamená, že vybírají soubor spotřebních statků, který nejvíce preferují.“* [27, str. 84]

[6] *„Produkční funkce vyjadřuje maximální množství výstupu, které může být vyrobeno při daném množství vstupů. Je definována pro daný stav technologické znalosti.“* [27, str. 108]

trzích a objem transakcí, ke kterým dochází. Typ aukce rozhoduje o tom, jaká množství a za jaké ceny budou na daném trhu obchodována.

Pro DSGE modely je vhodné zvolit aukci s využitím Walrasova aukcionáře. Její výhodou je, že je matematicky snadno řešitelná. V realitě se tento typ aukcí sice příliš často nevyskytuje, ale lze ukázat, že výsledky reálně používaných typů aukcí konvergují k výsledku, který dává Walrasova aukce [24].

Princip Walrasovy aukce

Uvažujme, že na trhu máme výrobce, nabízející identické zboží, a spotřebitele, ochotné směňovat toto zboží za nějakou komoditu. Poptávající (spotřebitel) nemá žádné preference ohledně prodejce (výrobce) a naopak. Walrasův aukcionář pak funguje na následujícím principu:

1. Zjistí individuální nabídky[7] od všech nabízejících subjektů a individuální poptávky[8] od všech poptávajících subjektů.

2. Na základě individuálních nabídek a poptávek sestaví tržní nabídku (agregací individuálních nabídek) a tržní poptávku (agregací individuálních poptávek).

3. Určí rovnovážnou cenu, kdy se nabízené množství rovná poptávanému množství.

4. Každému nabízejícímu vezme množství zboží, odpovídající jeho individuální nabídce pro tržní cenu a toto zboží rozdělí mezi poptávající tak, aby každý z nich dostal množství odpovídající jeho individuální poptávce pro tržní cenu.

5. Od každého poptávajícího získá množství komodity, odpovídající násobku rovnovážné ceny a množství nakoupeného zboží. Každý výrobce obdrží množství komodity, odpovídající násobku rovnovážné ceny a prodaného množství zboží.

V případě Walrasovy aukce nedochází k žádnému párování mezi nabízejícími a poptávajícími subjekty. Walrasova aukce umožňuje obchodování

[7] *„Nabídková funkce (a nabídková křivka) určitého statku vyjadřuje vztah mezi tržní cenou a množstvím statku, které budou výrobci za jinak stejných podmínek ochotni vyrábět a prodávat.“* [27, str. 51]

[8] *„Za jinak stejných podmínek existuje mezi tržní cenou statku a jeho poptávaným množstvím konkrétní vztah. Tento vztah mezi cenou a množstvím se nazývá poptávková funkce neboli poptávková křivka.“* [27, str. 46]

pouze za jedinou rovnovážnou cenu, stejnou pro všechny subjekty. Dva subjekty se nemohou domluvit a provést transakci za jinou cenu [24].

2.2 Popis hry

DSGEgame je simulační hra postavená na modelech DSGE. Stávající verze hry umožňuje vytváření modelu čtyřsektorové ekonomiky - ve hře jsou zastoupeni spotřebitelé, výrobci, vláda a je simulován zahraniční obchod. DSGEgame nachází uplatnění ve vědecké činnosti (v oblasti experimentální ekonomie) a v edukačním procesu (podpoře výuky). *„Je určena pro širokou skupinu hráčů, kteří budou v pravidelných intervalech provádět určitá rozhodnutí a tím ovlivňovat jak vlastní bodové hodnocení, tak i makroekonomické veličiny ekonomiky.“* [23, str. 7]

2.2.1 Základní principy hry

Hra se hraje v jednotlivých kolech, která představují určitá období ve skutečné ekonomice. Čas začátků kol stanoví administrátor hry vzhledem k předpokládaným časovým možnostem hráčů (čím kratší jsou kola, tím častěji jsou hráči nuceni se hře věnovat). Délka kola bývá obvykle stanovena na 3 dny, kola se uzavírají v úterý, čtvrtek a sobotu dopoledne. Takové nastavení je vhodné, protože hráči pak nejsou nuceni hrát o víkendu. Délka kola ve hře se definuje pro integrační celek.

Každému hráči DSGEgame je administrátorem přidělen některý typ subjektu. Oproti subjektům, zastoupeným v modelech DSGE (uvedené v kapitole 2.1.1) ve hře vystupuje navíc podnikatel (kombinuje roli spotřebitele a výrobce). Je možné, aby jeden hráč obsluhoval více subjektů (i různého typu). Dále může ve hře vystupovat také vláda nebo centrální banka, zpravidla ovládaná administrátorem hry.

Ekonomika je jakási autonomní geografická oblast. Každý subjekt, vystupující ve hře, je v daném čase přiřazen k jedné ekonomice (je jejím rezidentem), v průběhu hry může mezi ekonomikami migrovat, je-li to povoleno, ovšem jen v rámci integračního celku (skupina jedné nebo více ekonomik). Ve hře může vystupovat více měn, v každé ekonomice se platí jednou měnou. Pokud je ve hře nastaveno více měn, je potřeba vytvořit měnové trhy, na nichž se určují měnové kurzy. Pro každou ekonomiku je možné nastavit daně, které budou vybírány.

Ke směně mezi subjekty dochází na trzích. Trhy jsou součástí integračních celků, na trhu tak mohou být realizovány obchody mezi rezidenty různých ekonomik. Pro každý trh jsou definovány druhy subjektů, které na něj

mají přístup (zvlášť se rozlišuje právo poptávat a právo nabízet). Ve hře mohou existovat tyto trhy:

- trh spotřebního zboží,
- trh kapitálového zboží,
- trh práce,
- trh obligací,
- trh úvěrů,
- trh úspor,
- měnový trh.

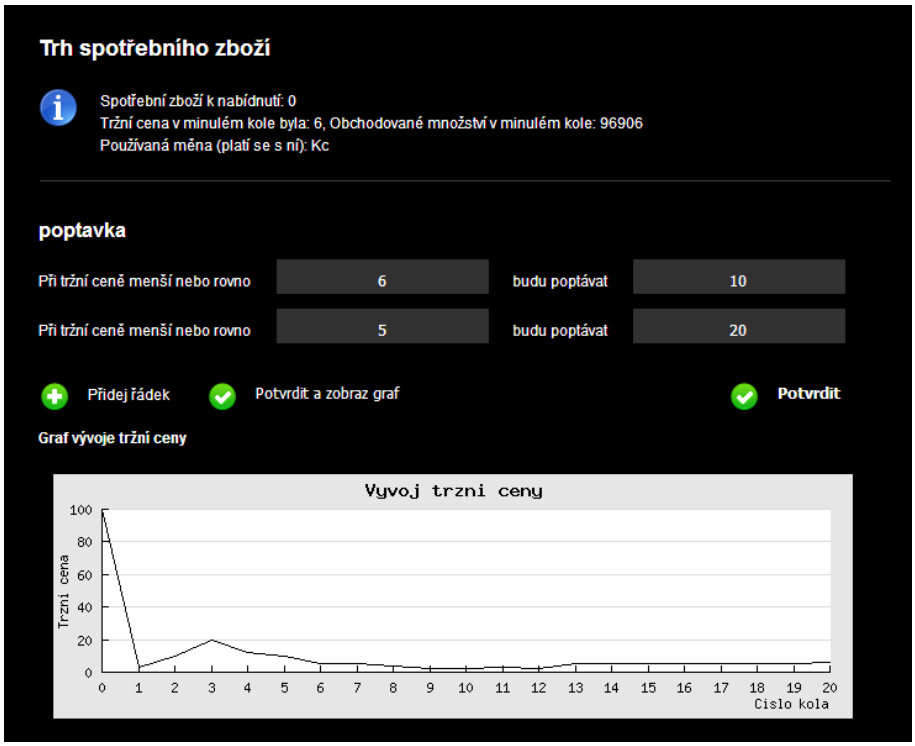
2.2.2 Průběh hry

Ve hře existují dva druhy vyráběných statků - kapitálové a spotřební zboží. Kapitálové zboží slouží jako výrobní faktor, spotřební zboží je spotřebováváno hráči. Na začátku hry obdrží všichni hráči stejné množství spotřebního a kapitálového zboží a peněz.

V průběhu kola může hráč uskutečnit několik rozhodnutí:

- Jak využít aktuální výši hotovosti - může za ni nakoupit spotřební či kapitálové zboží, práci nebo ji nabídnout na trhu finančních aktiv a získat tak úrok.
- Kolik hodin bude pracovat a zda bude pracovat pro sebe, nabídne svou práci na trhu práce či zvolí kombinaci předchozích možností, nebo zda bude preferovat volný čas (nebude pracovat vůbec).
- Zda své výrobní faktory vloží do kapitálového či spotřebního zboží nebo zvolí kombinaci předchozích možností (poměrově).

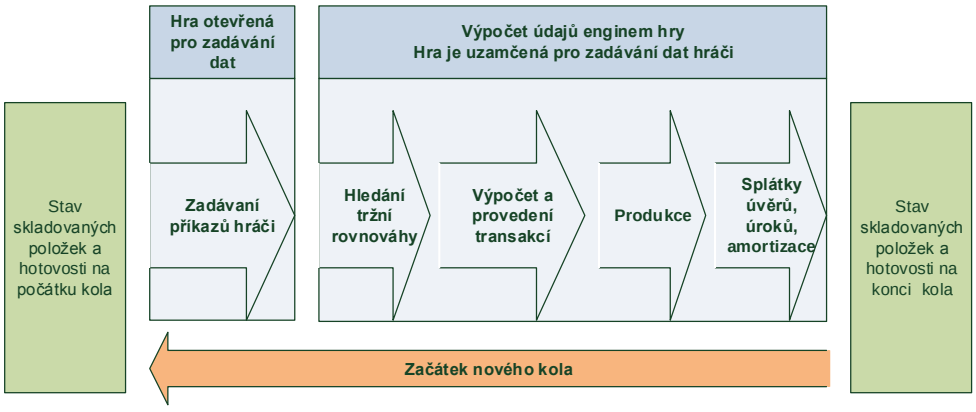
V průběhu kola hráči zadávají příkazy - výše popsaná rozhodnutí a individuální nabídku/poptávku na jednotlivých trzích, kam mají umožněn přístup. Individuální nabídka respektive poptávka určuje, jaké množství statku obchodovaného na konkrétním trhu bude daný hráč nabízet resp. poptávat v určitých cenových intervalech. Při nízké ceně statku chce hráč poptávat vysoké množství, při vysoké ceně statku pak nízké množství [23]. Obrázek 2.1 zobrazuje formulář pro zadání poptávky na trhu spotřebního zboží.



Obrázek 2.1: Individuální poptávka hráče na trhu spotřebního zboží - zdroj: vlastní zpracování

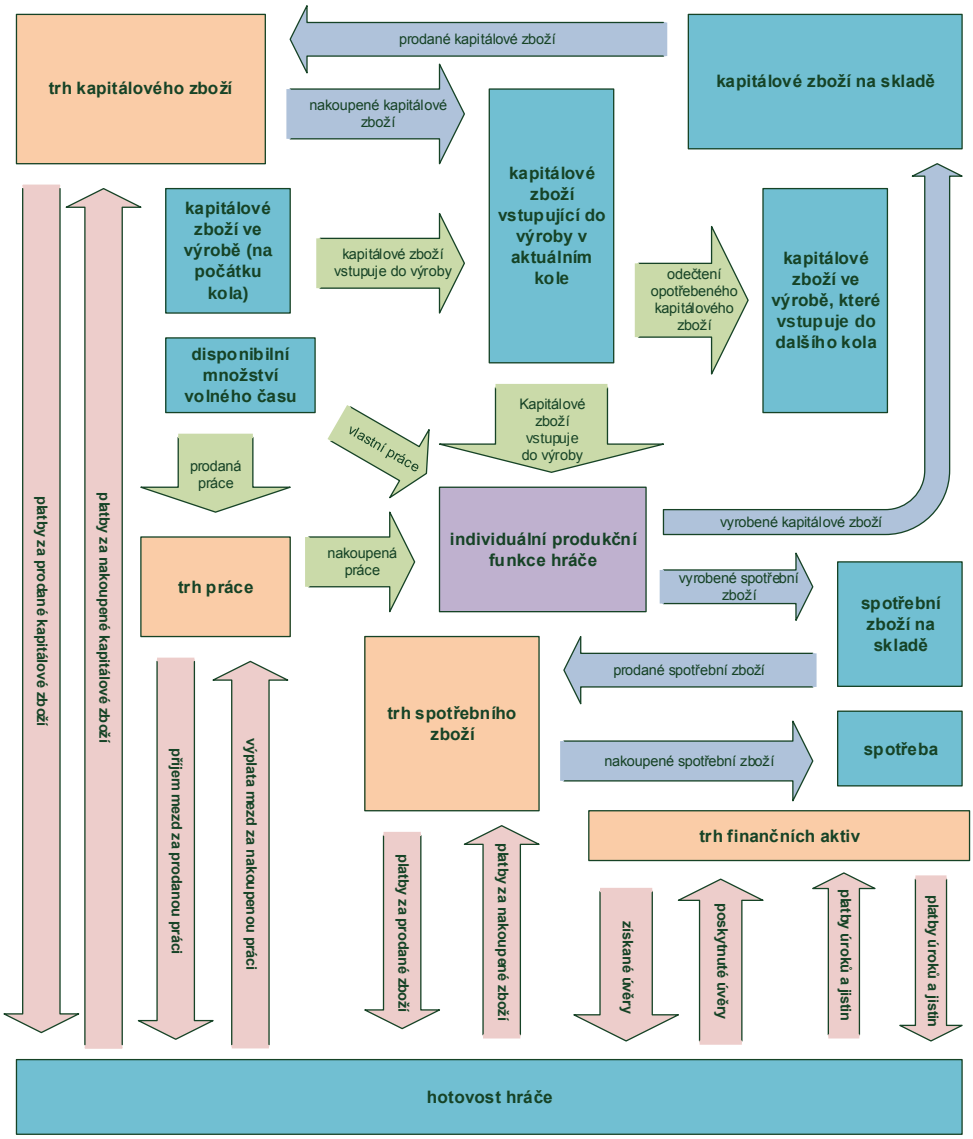
Proces obchodování na trzích je založen na principu Walrasova aukcionáře (popsán v kapitole 2.1.1).

Na konci kola je administrátorem specifikovanými algoritmy provedena analýza zadaných příkazů, vypočteny potřebné údaje (např. tržní ceny na jednotlivých trzích) a provedena realizace zadaných příkazů [10, 23]. Zjednodušený průběh událostí kola zobrazuje obrázek 2.2.



Obrázek 2.2: Průběh času ve hře - zdroj: [23]

Výpočtový proces na konci kola realizuje přesuny položek mezi hráči. Přesuny v DSGE modelu jsou z velké části realizovány prostřednictvím trhů. Např. při nákupu na trhu dochází k přesunu peněz od kupujících ve prospěch prodávajících a současně k toku zboží ke kupujícímu. Dále je třeba uvažovat platby úroků a jistin z úvěrů, poskytnutých v předcházejících kolech. Mezi množstvím kapitálového zboží vloženého do výroby a výsledným stavem kapitálového zboží na konci kola dochází k odečtu části kapitálového zboží vlivem amortizace [23]. Popsané operace jsou znázorněny na obrázku 2.3.



Obrázek 2.3: Toky peněz, zboží a výrobních faktorů během kola - zdroj: [23]

2.2.3 Cíl hry

Administrátor hry definuje způsob hodnocení hráčů - zadává vzorec výpočtu bodů (vzorec je odvozen z ekonomických funkcí užitku), o kterém jsou hráči informováni. Jednotlivým subjektům je přiřazeno počáteční vybavení (stavové proměnné na začátku hry) a účelové funkce. Jejich cílem je tyto účelové funkce maximalizovat v časovém horizontu trvání hry. Od toho se odvíjí bodové hodnocení hráčů [10, 23].

Hodnocení hráčů

Bodové hodnocení hráčů je založeno na základě jejich aktivity v každém kole. Body jsou přidělovány na základě následujících kritérií:

- spotřeba spotřebního zboží v jednotlivých kolech (diskontováno v čase),
- množství volného času v jednotlivých kolech (diskontováno v čase),
- záporné peněžní zůstatky.

2.3 Realizace stávající aplikace

Jedná se o webovou aplikaci, realizovanou ve skriptovacím jazyce PHP[9], využívající principy objektově orientovaného programování. Data jsou uložena v MySQL databázi. Objektová struktura aplikace odpovídá struktuře modelů DSGE. Objekty reprezentují individuální tržní nabídky a poptávky. Funkci Walrasova aukcionáře zastává `Správce trhu`. Dále existují subjekty, které spravují další aspekty ekonomiky hry - např. daně, měnový kurz, výrobu atd. Objekty jsou vázané buď na konkrétní ekonomiku nebo integrační celek v závislosti na tom, jestli spravují agendu jedné ekonomiky nebo celého integračního celku. Např. `Správce trhu` je vázaný na integrační celek, protože na jeden trh mohou mít přístup subjekty z více ekonomik.

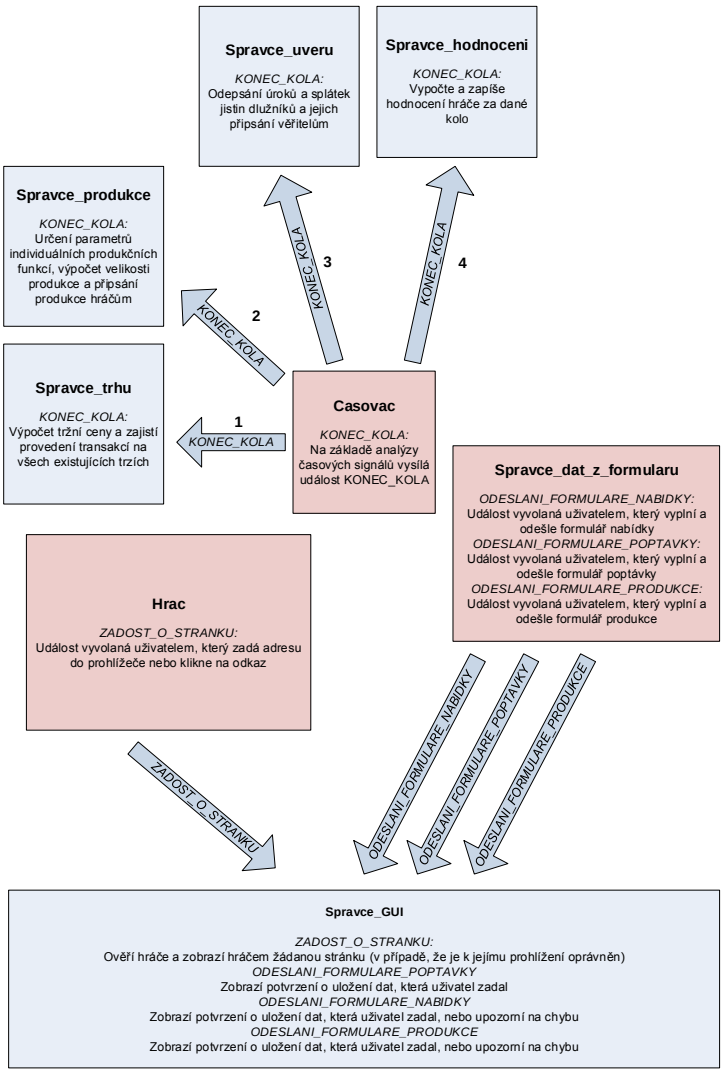
Aplikace používá jako hlavní uživatelský skript `index.php`. Ten se připojí k databázi, zařídí obsluhu všech možných generovaných událostí, zpracování událostí a vygenerování HTML[10] kódu (HTML kód je generován správcem GUI[11]). K přihlašování do hry se používá systém Orion. Uzavírání kola obstarává skript `cron_dsge_file.php`, který je zaheslovaný a běžný uživatel jej tak nemůže zavolat.

[9] Hypertext Preprocessor
[10] HyperText Markup Language
[11] Graphical User Interface

2.3.1 Objektový model

Objekty aplikace používají pro komunikaci systém událostí a ovladačů, které na tuto událost reagují. Implementace tohoto systému komunikace je postavena na modelu Davida Fellse [30].

Obrázek 2.4 zobrazuje objekty, mající schopnost generovat (zobrazeny červenou barvou) nebo reagovat (zobrazeny modrou barvou) na události. Události jsou symbolizovány šipkami, čísla zobrazená u šipek udávají pořadí obsluhy ovladačů v případě, že je na jednu událost jednoho objektu navěšeno více ovladačů.



Obrázek 2.4: Zjednodušený objektový model - zdroj: [23]

2.3.2 Systém událostí a ovladačů

Pokud dojde k vyvolání události, třída, ve které k vyvolání došlo, o tom posílá zprávu systému řízení událostí. Ten zajišťuje, že zprávu o události obdrží každá třída, která o ni má zájem. Každý objekt, který má možnost generovat události, má vlastní datovou strukturu - frontu událostí typu FIFO[12] (jednotlivé položky jsou ve frontě řazeny v pořadí, ve kterém byly vloženy). Ve frontě jsou všechny události, které může třída vyvolat.

Reakci tříd na události zajišťují ovladače. Na každou událost může být napojeno libovolné množství ovladačů. Ty jsou ukládány do datové struktury fronta ovladačů (opět struktura typu FIFO). Ovladač reprezentuje akci, ke které dojde v důsledku vyvolání události - je navázán právě na jednu třídu, schopnou reagovat na událost. V případě vyvolání události zajistí ovladač provedení konkrétní činnosti - volání metody třídy, na kterou je ovladač navázán.

Princip fungování systému událostí a ovladačů zobrazuje obrázek 2.5. Objekt `Časovač` v pravidelných intervalech vyvolává událost `Konec kola`. Objekty `Správce trhu` a `Správce produkce` na událost `Konec kola` reagují. Posloupnost akcí po vyvolání události je znázorněna zelenými šipkami:

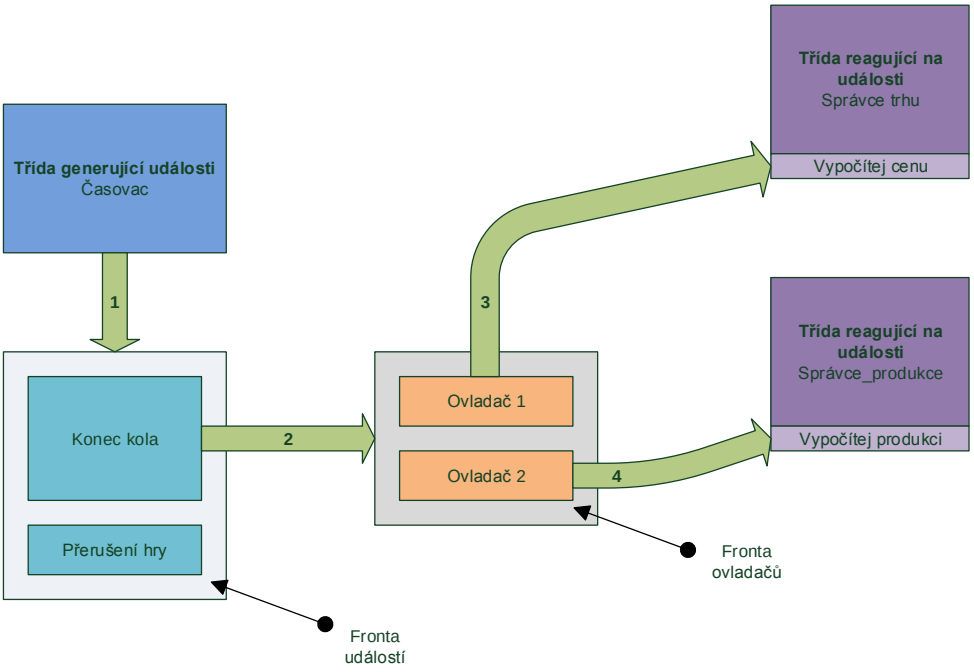
1. Šipka 1: Objekt `Časovač` vyvolává událost `Konec kola`.

2. Fronta událostí nalezne událost `Konec kola`. Fronta ovladačů této události obsahuje dva ovladače. Událost vydává frontě ovladačů příkaz, aby aktivovala všechny ovladače v ní zařazené (šipka 2).

3. Šipka 3: `Ovladač 1` provádí nadefinovanou činnost - volá objekt `Správce trhu` a předává mu informaci, aby spustil metodu `Vypočítej cenu`.

4. Šipka 4: Ovladač 2 provádí nadefinovanou činnost - volá objekt `Správce produkce` a předává mu informaci, aby spustil metodu `Vypočítej produkci` [23].

[12] First In First Out



Obrázek 2.5: Průběh vyvolání události - zdroj: [23]

3 Akciový trh a automatické hraní

3.1 Akciový trh

3.1.1 Akcie a akciová společnost

„Akciovou společností je společnost, jejíž základní kapitál je rozvržen na určitý počet akcií.“ [1] Akcie (stock, share) je cenný papír, se kterým jsou spojena práva akcionáře (majitele, držitele akcie) podílet se na řízení akciové společnosti, jejím zisku a likvidačním zůstatku v případě zániku společnosti. Akciová společnost je nejrozšířenějším způsobem podnikání v tržní ekonomice. Za porušení svých závazků odpovídá společnost celým svým majetkem, akcionář za závazky společnosti neručí [1, 16, 19].

Základní pojmy

- **Nominální hodnota (jmenovitá hodnota) akcie** představuje podíl na majetku akciové společnosti. Součet nominálních hodnot všech akcií je roven výši základního jmění (základního kapitálu) společnosti.
- **Tržní cena akcie (kurz akcie)** představuje cenu, za kterou se akcie obchoduje na akciovém trhu. Je ovlivněna ekonomickými, politickými a psychologickými faktory - prosperitou dané společnosti, kvalitou managementu společnosti, situací a perspektivou v daném oboru a ekonomice, spekulativními tlaky apod.
- **Dividenda** je podíl na zisku společnosti plynoucí z vlastnictví dané akcie a odhlasovaný pro dané období valnou hromadou akcionářů [16].

Emise akcií

Akciová společnost vydává na veřejném trhu cenných papírů (na burze) či mimoburzovním trhu své akcie s cílem získat kapitál pro další rozvoj nebo s cílem zvýšit vlastní prestiž a důvěryhodnost. Z hlediska akciové společnosti jsou emise akcií výhodné zejména pro dlouhodobé investice, protože akciový kapitál nemusí být akciovou společností splacen, ani z něho nemusí být poskytována pravidelná dividenda, zejména při nepříznivém hospodářském vývoji. Nevýhodou emisí akcií na veřejném trhu je, že stávající akcionáři

ztrácejí část kontroly nad rozhodováním o společnosti, což může ohrožovat původní záměry firmy.

Společnosti, s jejichž akciemi se obchoduje na veřejném trhu, musí zpravidla plnit rozsáhlou informační povinnost. To zvyšuje jejich transparentnost a důvěryhodnost. Vyšší důvěryhodnost pak příznivě ovlivňuje vztahy vůči věřitelům - společnosti jsou schopny prosadit si výhodnější úvěrové podmínky. Akciová společnost, s jejímiž akciemi se obchoduje na burze, má také do jisté míry zajištěnou poptávku po svých akciích, takže může případně vydat další akcie a získat tak další kapitál.

Jestliže společnost vydává na veřejném trhu akcie poprvé, hovoříme o primárních emisích (IPO[1]). Pokud společnost vydala své akcie na veřejném trhu cenných papírů již dříve a nyní vydává další, mluvíme o sekundárních emisích (SPO[2]) [13].

Práva akcionáře

Právo akcionáře podílet se na řízení společnosti Primárním právem akcionáře je právo podílet se na řízení akciové společnosti. Akcionář je oprávněn zúčastnit se valné hromady, hlasovat na ní a požadovat na ní vysvětlení záležitostí týkajících se společnosti. Počet hlasů akcionáře na valné hromadě je určen jmenovitou hodnotou akcie - každý akcionář má tolik hlasů, kolik odpovídá podílu jmenovité hodnoty jeho akcií na základním jmění akciové společnosti [12, 19].

Právo akcionáře na dividendy Druhým základním právem akcionáře je jeho právo na podíl z té části zisku, kterou valná hromada podle výsledku hospodaření schválila k rozdělení. Výše dividendy akcionáře se určuje poměrem jmenovité hodnoty akcií [12, 19].

Právo akcionáře na podíl na likvidačním zůstatku Dalším právem akcionáře je právo na podíl na likvidačním zůstatku společnosti. Po uspokojení všech věřitelů se likvidační zůstatek rozděluje mezi akcionáře v poměru odpovídajícím jmenovité hodnotě jejich akcií. Pokud likvidační zůstatek nestačí k úhradě jmenovité hodnoty akcií, dělí se na část připadající vlastníkům prioritních akcií a na část připadající vlastníkům ostatních akcií [12, 19].

[1] Initial Public Offering
[2] Secondary Public Offering

Orgány akciové společnosti

Valná hromada Nejvyšším orgánem akciové společnosti je valná hromada, na které akcionáři vykonávají své právo podílet se na řízení společnosti. Pokud stanovy společnosti neurčí jinak, rozhoduje valná hromada většinou přítomných akcionářů [1, 19].

Představenstvo *„Statutárním orgánem společnosti je představenstvo.“* [1] Přísluší mu obchodní vedení společnosti - zajišťuje řádné vedení účetnictví, předkládá valné hromadě ke schválení účetní závěrku a návrh na rozdělení zisku nebo úhradu ztráty. Neurčí-li stanovy společnosti jinak, má představenstvo tři členy, kteří jsou voleni a odvoláváni valnou hromadou. Představenstvo volí a odvolává svého předsedu [1].

Dozorčí rada *„Dozorčí rada dohlíží na výkon působnosti představenstva a na činnost společnosti.“* [1] Členové dozorčí rady se zúčastňují valné hromady a pověřený člen dozorčí rady ji seznamuje s výsledky její činnosti [1].

3.1.2 Klasifikace akcií

Klasifikace akcií podle podílu na základním kapitálu

- Akcie se jmenovitou hodnotou: Podíl akcionáře na základním kapitálu, na hlasovacích právech, právu na dividendě a likvidačním zůstatku je dán poměrem jmenovité hodnoty a celkového základního kapitálu.
- Kusové akcie: Tyto akcie nemají jmenovitou hodnotu - představují stejné podíly na základním kapitálu společnosti (na jednu kusovou akcii připadá jeden hlas, ledaže stanovy společnosti připouští vydání kusových akcií s různou váhou hlasů). Jestliže akciová společnost vydá kusové akcie, nemůže vydat ani mít vydány akcie se jmenovitou hodnotou [1, 26].

Klasifikace akcií podle formy

- Akcie listinné (fyzické listiny): Listinné akcie mohou být vydány jako:
  - jednotlivé akcie,
  - hromadné listiny, kde jedna listina zastupuje více akcií a lze ji „rozměnit“.

- Akcie zaknihované: Zaknihované akcie existují jen jako evidenční záznamy. Vést evidenci zaknihovaných cenných papírů může pouze licencovaný subjekt, většinou se využívá služeb centrálního depozitáře. V ČR jeho úlohu plní CDCP[3] [1, 26].

Klasifikace akcií podle způsobu převodu

Akcie může mít formu cenného papíru na řad (na jméno) nebo na doručitele (na majitele).

- Akcie na jméno: Je vydávána na jméno určité osoby (fyzické nebo právnické). Výhodou je lepší ochrana proti odcizení. Pokud společnost vydala akcie na jméno, vede seznam akcionářů, do nějž zapisuje označení druhu akcie, její jmenovitou hodnotu, jméno a bydliště (nebo sídlo) akcionáře a číslo bankovního účtu. Stanovy společnosti mohou určit, že pro zaknihované akcie je seznam akcionářů nahrazen evidencí u CDCP.

  V listinné podobě je na akcii uvedeno jméno vlastníka a převádí se rubopisem[4] a fyzickým předáním. V zaknihované podobě se převádí smlouvou a registrací převodu u emitenta (nebo CDCP). Převoditelnost akcií na jméno lze omezit stanovami společnosti a vázat tento převod na souhlas některého z orgánů společnosti.

- Akcie na majitele: Držitel této akcie je pro akciovou společnost anonymní (dokud neuplatní právo s akcií spojené). Výhodou je snadná obchodovatelnost na sekundárních trzích. Akcie na majitele je neomezeně převoditelná. V listinné podobě stačí fyzické předání (není třeba rubopis), v zaknihované podobě se převádějí smlouvou a registrací převodu u CDCP [1, 19].

Akcie podle práv s nimi spojených

- Kmenové (obyčejné) akcie: Jako kmenové akcie bývají označovány ty druhy akcií, které svým majitelům zaručují „standardní práva“ (právo podílet se na řízení společnosti, na zisku společnosti a likvidačním zůstatku společnosti) [25].

- Prioritní akcie: S prioritními akciemi je spojeno přednostní právo týkající se podílu na zisku nebo na likvidačním zůstatku společnosti. Před-

[3] Centrální depozitář cenných papírů

[4] *„Též zvaný „indosament“ nebo „žíro“. Způsob písemného převodu cenného papíru. Písemným projevem majitele cenného papíru se převádějí práva na jinou osobu (nového věřitele). Vyznačuje se zpravidla na rubu cenného papíru.“* [7]

nostní nárok na dividendu může spočívat v tom, že ze zisku společnosti budou vypláceny dividendy náležející prioritním akciím a teprve pak akciím ostatním, nebo v tom, že pro tyto dividendy bude ze zisku vyhrazeno vyšší procento, než kolik náleží podle poměrů jejich jmenovitých hodnot k jmenovitých hodnotám ostatních akcií. Pokud enení ve stanovách společnosti určeno jinak, jsou tyto akcie vydávány bez hlasovacího práva [12].

- Zaměstnanecká akcie: Tyto akcie bývají prodávány pouze zaměstnancům akciové společnosti, a to za výhodnějších podmínek než kmenové akcie. Proto nesmějí být převáděny na jiné osoby než pouze na jiné zaměstnance téže společnosti. Bývá u nich praktikována zásada, že při odchodu z firmy mají jejich majitelé povinnost odprodat je akciové společnosti. V případě odchodu do důchodu si je mohou ponechat, avšak po jejich smrti jsou dědicové povinni nabídnout je akciové společnosti k odkupu [25].

3.1.3 Akciové analýzy

Akciové analýzy podrobně rozebírají faktory, které mají vliv na cenu akcií.

Fundamentální analýza

„Fundamentální analýzu lze považovat za nejkomplexnější druh akciové analýzy, jež se v investiční praxi používá při přípravě zásadních investičních rozhodnutí.“ [25, str. 99] Předpokládá, že každá akcie má v daném okamžiku určitou „vnitřní hodnotu“ (teoretickou cenu) a že tato teoretická cena se liší od aktuální tržní ceny, za níž je akcie obchodována na akciovém trhu. Pokud je vnitřní hodnota akcie vyšší než její kurz, je akcie považována za podhodnocenou a naopak pokud je vnitřní hodnota akcie vyšší než její kurz, je považována za nadhodnocenou. Předmětem zkoumání fundamentální analýzy je hledání podhodnocených akcií k nákupu a nadhodnocených k prodeji [3, 25].

Fundamentální analýzu je možné provádět na několika úrovních. Globální (makroekonomická) analýza zkoumá ekonomiku jako celek a zabývá se zkoumáním vztahů mezi vývojem ekonomických makroagregátů (inflace[5],

[5] *„Inflace označuje růst celkové cenové hladiny. Míra inflace je definována jako míra změny celkové cenové hladiny a měří se takto: Míra inflace v roce t = (cenová hladina v roce t - cenová hladina v roce t-1) / (cenová hladina v roce t-1) * 100.“* [27, str. 440]

hospodářského růstu[6], úrokových sazeb atd.) na ceny akcií. Cílem odvětvové (oborové) analýzy je rozpoznat a charakterizovat nejvýznamnější specifika jednotlivých odvětví a následně prognózovat perspektivy jejich budoucího vývoje [3, 25].

Technická analýza

Technická analýza je založená na výzkumu ceny akcie. Předpokládá, že v ceně jsou obsaženy všechny informace. Pomocí modelování časových řad se snaží identifikovat jednotlivé vývojové trendy a z nich následně vyvozovat budoucí vývoj kurzů akcií [25].

Psychologická analýza

Psychologická analýza vychází z předpokladu, že akciové trhy jsou pod silným vlivem masové psychologie burzovního publika, které tím, že působí na účastníky trhu, ovlivňuje úroveň kurzů. Předmětem zkoumání není samotný kurz, ale chování investorů [3, 25].

Finanční analýza

Z finančních výkazů akciové společnosti se počítají různé poměrové ukazatele, které jsou následně používány v rámci akciových analýz.

Ukazatele rentability O výkonnosti společnosti informují investora ukazatele rentability. Nejpoužívanějšími jsou [3]:

- ROA[7] (výnos z aktiv[8])

$$\frac{\text{čistý zisk}}{\text{celková aktiva}} = \frac{\text{čistý zisk}}{\text{vlastní jmění + cizí zdroje}} \tag{3.1}$$

- ROE[9] (výnos z vlastního jmění)

$$\frac{\text{čistý zisk}}{\text{vlastní jmění}} \tag{3.2}$$

[6] *„Růst celkové produkce dané země. Obvykle bývá měřen roční mírou růstu reálného HDP.“* [27, str. 736]

[7] Return On Assets

[8] *„Hmotný majetek či nehmotný nárok, který má ekonomickou hodnotu. Hlavní případy aktiv jsou budovy, zařízení, půda, patenty, autorská práva a finanční nástroje, například peníze či obligace.“* [27, str. 732]

[9] Return On Equity

Ukazatele zadluženosti O úrovni finančního rizika vypovídají ukazatele zadluženosti, např. [3]:

- debt ratio (celková zadluženost aktiv)

$$\frac{\text{cizí zdroje}}{\text{celková aktiva}} = \frac{\text{cizí zdroje}}{\text{vlastní jmění + cizí zdroje}} \tag{3.3}$$

- leverage ratio (zadluženost vlastního jmění, finanční páka)

$$\frac{\text{cizí zdroje}}{\text{vlastní jmění}} \tag{3.4}$$

Investiční ukazatele O potenciální výnosnosti investorem vložených prostředků informují investiční ukazatele [3, 16].

- DPS[10] (dividenda na akcii)

$$\frac{\text{dividendy}}{\text{počet akcií}} \tag{3.5}$$

- EPS[11] (zisk na akcii)

$$\frac{\text{zisk}}{\text{počet akcií}} \tag{3.6}$$

- dividendový výnos (běžná výnosnost akcie, dividend yield)

$$\frac{\text{dividenda na akcii}}{\text{tržní cena akcie}} \tag{3.7}$$

- akciový výnos (celková výnosnost akcie, earnings yield)

$$\frac{\text{zisk na akcii}}{\text{tržní cena akcie}} \tag{3.8}$$

3.2 Automatické hraní

Ve hře vystupují 4 druhy subjektů (popsány v kapitole 2.1.2). Systém automatického hraní připadá v úvahu pro každý z nich, ovšem pro každý z jiného důvodu.

[10] Dividend Per Share
[11] Earnings Per Share

3.2.1 Automatické hraní spotřebitelů, podnikatelů a výrobců

Spotřebitelé, výrobci (a podnikatelé) bývají ovládáni studenty. V praxi se prokázalo, že pro průběh hry je problematické, když velká část studentů přestane hrát. V tom případě dochází k fluktuaci obchodovaného množství na trzích a vychylování tržních cen. Takové chování je nerealistické. Stabilizovat obchodovaná množství na trzích by bylo možné existencí automatu simulujícího hru neaktivních hráčů.

3.2.2 Automatické hraní vlády

Vláda bývá ovládána administrátorem hry. Existence automatu hrajícího podle předem zadaných pravidel by administraci hry usnadnila.

Vláda provádí fiskální politiku (popsána v kapitole 2.1.1). V ekonomických modelech bývá rozhodování vlády definováno jako rovnice. Studie Ministerstva financí prezentující DSGE model české ekonomiky s názvem HUBERT [33] definuje následující rovnici udávající deficit vlády.

$$D_t = GE_t - GR_t \tag{3.9}$$

Proměnná $GR^{12}{}_t$ reprezentuje celkové vládní příjmy (Government Revenues) a je definována následovně.

$$GR_t = PIT_t + CIT_t + VAT_t + EXCISE_t, \tag{3.10}$$

kde

- $PIT^{13}{}_t$ jsou příjmy na daních od fyzických osob (Personal Income Tax),
- $CIT^{14}{}_t$ jsou příjmy na daních od právnických osob (Corporate Income Tax),
- $VAT^{15}{}_t$ jsou příjmy z DPH[16] (Value Added Tax),
- $EXCISE$ jsou příjmy ze spotřební daně.

[12] Government Revenues
[13] Personal Income Tax
[14] Corporate Income Tax
[15] Value Added Tax
[16] Daň z přidané hodnoty

Proměnná $GE^{17}{}_t$ reprezentuje vládní výdaje (Government Expenditures) a je definována následovně.

$$GE_t = G_t + G_t^s, \tag{3.11}$$

kde

- G_t jsou vládní výdaje na statky a služby (Government Expenditures),
- G_t^s jsou vládní transfery, jejichž účelem je zvýšit příjem určité skupiny obyvatel - např. seniorů nebo nezaměstnaných (Social Benefit).

Model HUBERT následně uvádí rovnici definující pravidlo pro uzavření modelu.

$$G_t = (1 - \phi_g)\bar{G}_{t-1} + \phi_g G_{t-1}, \tag{3.12}$$

kde

- G_t jsou vládní výdaje (Government Expenditures),
- $\bar{G}_t$ jsou rovnovážné výdaje.

Rovnovážné výdaje $\bar{G}_t$ jsou odvozeny z rovnice 3.9, která je položená rovno nule.

$$D_t = GE_t - GR_t = 0 \tag{3.13}$$

Zpoždění o jedno období v rovnici $(t-1)$ reprezentuje zpoždění informací. Parametr ϕ_g udává rychlost konvergence veřejných financí.

3.2.3 Automatické hraní centrální banky

Stejně jako vláda, bývá i centrální banka ovládána administrátorem hry. Existence automatu hrajícího podle předem zadaných pravidel by administraci hra usnadnila i v tomto případě.

Centrální banka provádí monetární politiku (popsána v kapitole 2.1.1). Chování centrální banky se často řídí podle tzv. Taylorova pravidla. Jedná se o exaktně kvantifikovatelný vztah mezi inflací, hospodářským růstem a monetární politikou centrální banky. Nástroj vychází z reakcí na odchylky inflace a růstu produkce z požadovaných cílových úrovní. Vztah mezi požadovanou výší úrokové sazby, inflační a růstovou mezerou definoval v roce 1993 ekonom John Taylor následovně:

$$i = i^* + a(\pi - \pi^*) + b(y - y^*), \tag{3.14}$$

kde

[17] Government Expenditures

- i je požadovaná úroková sazba,
- i^* rovnovážná úroková sazba,
- π míra inflace,
- π^* inflační cíl,
- y růst HDP[18],
- y^* cíl růstu HDP.

Parametry a a b určují váhu inflační mezery $(\pi - \pi^*)$ a růstové mezery $(y - y^*)$. Nastavení váhy těchto parametrů určuje soustředění centrální banky (na inflaci nebo na hospodářský růst).

Prostřednictvím Taylorova pravidla je možné určit požadovanou krátkodobou úrokovou sazbu pro danou ekonomiku. Tuto sazbu centrální banka zvyšuje nad rovnovážnou úroveň, jestliže inflace roste nad inflační cíl nebo pokud aktuální hospodářský růst převyšuje jeho potenciální úroveň. Požadovaná úroková sazba vypočtená Taylorovým pravidlem určuje budoucí vývoj monetární politiky centrální banky [31].

[18] *„Hodnota veškeré finální produkce vyrobené v dané zemi za jeden rok.“* [27, str. 736]

4 Analýza optimalizace databáze aplikace

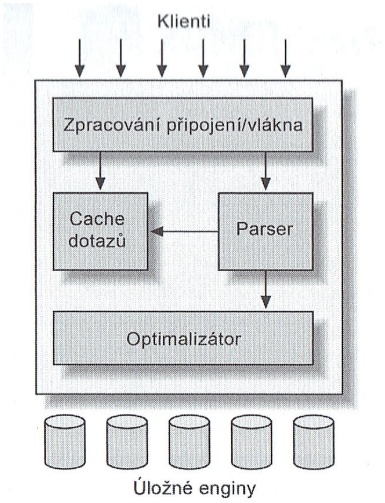
4.1 Použitý SŘBD

Databáze aplikace je postavena na SŘBD[1] MySQL.

MySQL je multiplatformní relační databázový systém, vytvořený švédskou firmou MySQL AB, nyní vlastněný společností Sun Microsystems, dceřinnou společností Oracle Corporation. Jedná se o nejpopulárnější open-source databázi. Díky svému výkonu, spolehlivosti a snadnému použití se databáze MySQL stala přední volbou pro použití ve webových aplikacích včetně např. Facebooku, Twitteru nebo Youtube. Základem pro veškerou komunikaci v rámci databáze je samozřejmě SQL[2]. MySQL zahrnuje SQL server, klientské programy pro přístup k serveru, nástroje pro správu a programovací rozhraní [2, 4]. Poslední verzí MySQL k dubnu 2017 je 5.7 [21]. Verzí používanou v aplikaci DSGEgame je 5.5.54-0.

Logická architektura MySQL

Obrázek 4.1 zobrazuje logický pohled na architekturu MySQL.



Obrázek 4.1: Logický pohled na architekturu MySQL serveru - zdroj: [28]

[1] Systém řízení báze dat
[2] Structured Query Language

První vrstva (nahoře) obsahuje služby, které nejsou jedinečné pro MySQL. Tyto služby obsluhují většinu potřebných nástrojů klient/server, které jsou založeny na síti, např. zpracování připojení, autentizace, bezpečnost atd.

Ve druhé vrstvě se nachází valná část mozku MySQL - kód pro rozbor (parsing), analýzu a optimalizaci dotazu a pro všechny zabudované funkce (např. pro datum a čas, matematické výpočty nebo šifrování). MySQL provádí rozbor dotazů proto, aby vytvořil interní stromovou strukturu (parse tree), pak aplikuje veškeré optimalizace: Může dotaz přepsat, určit pořadí, v němž bude číst tabulky, zvolit, které indexy použije atd. Na této úrovni se nachází také veškerá funkcionalita, která se poskytuje prostřednictvím úložných enginů - uložené procedury, triggery a pohledy.

Třetí vrstva obsahuje úložné enginy (storage engines). Úložné enginy mají na starosti ukládání a získávání dat uložených v MySQL. Každý úložný engine má své přednosti i své stinné stránky. Server s úložnými enginy komunikuje prostřednictvím jejich API[3]. Tato rozhraní skrývají rozdíly mezi jednotlivými úložnými enginy a činí je na vrstvě dotazů velmi transparentními [28].

Úložným enginem používaným v databázi aplikace DSGEgame je InnoDB. Engine InnoDB byl navržen pro zpracování transakcí a v současné době patří mezi nejpopulárnější úložné enginy pro jejich ukládání. Jeho výkonnost a možnosti automatického zotavení po havárii ho nicméně učinily populárním i pro potřeby netransakčních úložišť [28].

4.2 Stávající datový model

Následující kapitola popisuje nejdůležitější tabulky databáze, ERA[4] model je součástí přílohy práce.

`aktivita_subjektu` Evidence aktivit jednotlivých subjektů. Eviduje se zadání poptávky nebo nabídky na trhu či zadání příkazu výroby a datum a čas provedení této aktivity. Vazba na tabulku `subjekty`.

`casy_koncu_kol` Evidence časových okamžiků, ke kterým se váže uzavření kola hry v jednotlivých integračních celcích, vazba na tabulku `integracni_celky`.

`druhy...` Série číselníků, sloužící pro správu hodnot, kterých mohou příslušné tabulky nabývat. Seznam tabulek:

[3] Application Programing Interface

[4] Entity Relationship Attributes

- `druhy_dani_a_transferu`,
- `druhy_popisu_puvodu_toku`,
- `druhy_subjektu`,
- `druhy_trhu`.

`ekonomiky` Evidence ekonomik, vazba na tabulku `integracni_celky`.

`hraci` Evidence hráčů.

`integracni_celky` Evidence integračních celků.

`meny` Číselník měn.

`nabidky` Evidence zadaných nabídek, vazba na tabulku `subjekty`, `trhy` a `meny`.

`poptavky` Evidence zadaných poptávek, vazba na tabulku `subjekty`, `trhy` a `meny`.

`prikazy_vyroby` Evidence zadaných příkazů výroby, vazba na tabulku `subjekty`.

`puvody_toku` Evidence typu původu toků komodit na jednotlivých trzích nebo v jednotlivých ekonomikách, vazba na tabulku `trhy` a `ekonomiky`.

`rezidenti` Evidence přiřazení subjektů k jednotlivým ekonomikám, vazba na tabulku `ekonomiky` a `subjekty`.

`rezidenti_vek` Evidence věku rezidentů - tabulka se využívá v případě, že hra je spuštěna s OLG[5] modelem. V takovém případě subjekty typu spotřebitel a podnikatel v průběhu hry stárnou a v čase nastaveném administrátorem hry odcházejí do penze - pak nemají umožněn přístup na trh práce a pobírají starobní důchod. Tabulka je navázána na tabulku `subjekty`.

`subjekty` Evidence subjektů, vazba na tabulku `hraci` a `integracni_celky`.

[5]Overlapping Generations

toky... Série tabulek pro evidenci generovaných toků komodit. Vazba na tabulky `subjekty`, `meny` a `puvody_toku`. Seznam tabulek:

- `toky_kapitaloveho_zbozi_sklad`,
- `toky_kapitaloveho_zbozi_vyroba`,
- `toky_lidskeho_kapitalu`,
- `toky_obligaci`,
- `toky_penez`,
- `toky_prace`,
- `toky_spotrebniho_zbozi`.

trhy Evidence trhů ve hře, vazba na tabulku `integracni_celky`.

vyvoj_trznich_cen Evidence tržních cen na jednotlivých trzích v jednotlivých kolech, vazba na tabulku `trhy`.

ziskane_body Evidence získaných bodů jednotlivými subjekty v uplynulých kolech hry, vazba na tabulku `subjekty`.

4.3 Detekce problematických dotazů

Prvním krokem v rámci optimalizace databáze byla příprava mechanismu, evidujícího čas provádění SQL dotazů. Díky tomu je možné sledovat, které dotazy se provádějí dlouhou dobu. Byla vytvořena třída `SQLLog`, obsahující dvě metody. Metoda `query_count_time()` funguje následovně:

- Zaznamená čas (voláním funkce `microtime(true)`, jež vrací UNIX timestamp v mikrosekundách).
- Vykoná dotaz, předaný metodě jako parametr (voláním funkce `mysql_query()`).
- Zaznamená čas.
- Odečtením času před provedením dotazu od času po jeho provedení získá dobu provádění dotazu. Pokud tato doba přesáhne nastavenou

hodnotu[6], je volána metoda `write_time_executing_query_to_db()`. Ta ukládá do tabulky `sql_log` záznam v následující podobě: *SQL dotaz - doba provádění - datum a čas volání dotazu.*

id	dotaz	cas	datum_cas
373594	SELECT id_subjektu, id_meny, SUM(h...	0.70301	2016-12-06 09:27:37
121203	SELECT DISTINCT id_subjektu FRO...	0.611027	2016-12-03 11:11:35
399129	SELECT SUM(hodnota_toku) AS mnoz...	0.545738	2016-12-06 09:28:46

Tabulka 4.1: Log doby provádění SQL dotazů - zdroj: vlastní zpracování

Metoda `query_count_time()` zároveň ověřuje, zda vykonání daného dotazu nevrátilo chybu. Jestliže ano, vloží záznam do tabulky `sql_errors` v následující podobě: *SQL dotaz - chybová hláška - datum a čas volání dotazu.* Díky tomu je možné evidovat chybné SQL dotazy.

dotaz	chyba	datum_cas
SELECT marze FROM ma...	You have an error in your SQ...	2017-03-26 08:17:19
SELECT marze FROM ma...	You have an error in your SQ...	2017-03-26 08:17:25

Tabulka 4.2: Tabulka evidující chybné SQL dotazy - zdroj: vlastní zpracování

Pro volání veškerých SQL dotazů v kódu aplikace byla dosud používána metoda `mysql_query()`. Na všech místech kódu, kde byla tato funkce volána, byla nahrazena voláním metody `query_count_time()`. Díky tomu je možné o všech dotazech rozhodnout, zda jejich provádění trvá dlouhou dobu a případně je evidovat.

Na základě analýzy logu SQL dotazů, jejichž provádění trvá dlouhou dobu, byly provedeny úpravy, popsané v následujících kapitolách. Byla provedena úprava schématu a indexů databáze, úprava problémových SQL dotazů a také úpravy na straně aplikace.

4.4 Optimalizace schématu a indexování

Schéma a indexy je třeba navrhnout s ohledem na SQL dotazy, které budou spouštěny. Optimalizace schématu databáze znamená často kompromisy, například přidání indexů za účelem urychlení získávání dat, zpomalí aktuali-

[6]Byl hledán kompromis, jaký mezní čas zvolit, aby na jedné straně byla k dispozici relevantní data o době provádění dotazů a na druhé straně, aby nebylo zaznamenáváno příliš velké množství dotazů, protože se jedná o drahou operaci [28]. Postupem času se tato hodnota ustálila na 0,01 sekundy.

zace dat. A naopak - denormalizované schéma sice může jisté druhy SQL dotazů urychlit, ale zpomalit jiné [28].

4.4.1 Nastavení indexů tabulek

Indexování je nejdůležitějším nástrojem, který je pro zrychlování dotazů k dispozici. K dispozici jsou i jiné techniky (např. optimalizace datových typů sloupců nebo restrukturalizace dotazů), ale všeobecně největšího rozdílu ve výkonnosti se dá docílit správným používáním indexů. Nepoužíváme-li indexy, je plýtvání časem, pokud se snažíme zvýšit výkonnost jinými prostředky. Je vhodné nejprve použít indexy, abychom markantně zvýšili výkonnost a teprve poté zjišťovat, zda by ještě mohli pomoci jiné prostředky [11].

Indexy jsou datové struktury, které pomáhají efektivně získat data. Nabývají na důležitosti s rostoucím množstvím dat. Malé databáze s malou zátěží mívají často slušný výkon i bez řádných indexů, s narůstajícím objemem ukládaných dat však může výkon databáze velmi rychle klesnout [28].

Zisky plynoucí z indexů

Obrázek 4.2 zobrazuje vzorovou tabulku před přidáním indexu.

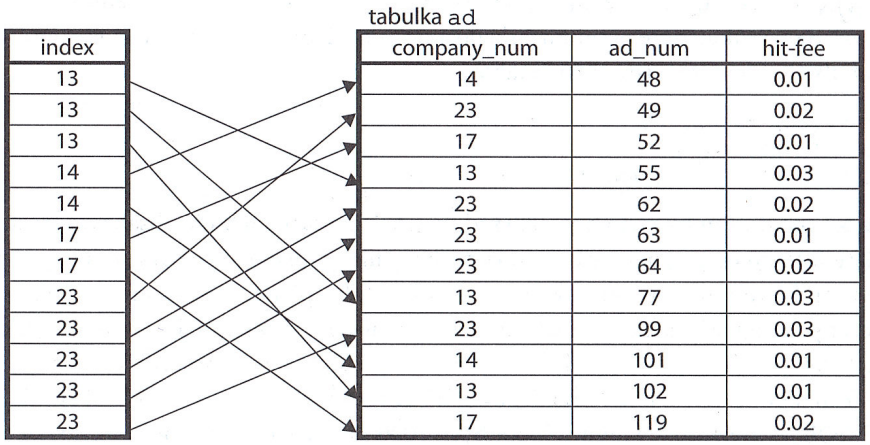
tabulka ad

company_num	ad_num	hit-fee
14	48	0.01
23	49	0.02
17	52	0.01
13	55	0.03
23	62	0.02
23	63	0.01
23	64	0.02
13	77	0.03
23	99	0.03
14	101	0.01
13	102	0.01
17	119	0.02

Obrázek 4.2: Vzorová tabulka - zdroj: [11]

Neindexovaná tabulka je neuspořádaná kolekce řádků. Pokud chceme najít záznam, je nutné projít všechny řádky tabulky, abychom zjistili, zda odpovídají požadované hodnotě. To znamená, projít tabulku od začátku

do konce, což je velmi neefektivní, je-li tabulka velká a záznamů, vyhovujících výběrovému kritériu, jen několik [11]. Obrázek 4.3 zobrazuje stejnou tabulku, ovšem s přidaným indexem pro sloupec `company_num`.

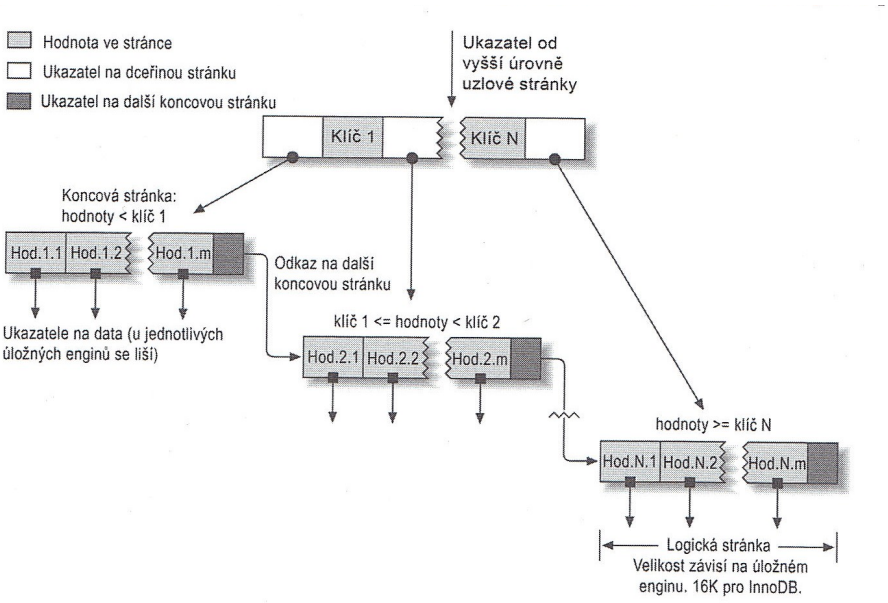


Obrázek 4.3: Vzorová tabulka s indexem - zdroj: [11]

Index obsahuje položku pro každý řádek tabulky, přičemž položky indexu jsou seřazené podle hodnot ve sloupci `company_num`. Díky tomu už není nutné prohledávat celou tabulku řádek po řádku, využije se index. Dejme tomu, že hledáme např. záznamy firmy číslo 13: Začneme prohledávat index a najdeme tři řádky této firmy. Pak narazíme na řádek s firmou číslo 14, což je hodnota vyšší, než ta, kterou hledáme. Díky tomu, že jsou hodnoty indexu seřazené, víme, že další souhlasný řádek nenajdeme a prohledávání tak můžeme ukončit. Jedno zvýšení efektivity tedy získáme tím, že víme, kde souhlasné řádky končí (a zbytek můžeme přeskočit). Další zvýšení efektivity spočívá v tom, že existují poziční algoritmy pro nalezení první vyhovující položky, takže se nemusí sekvenčně prohledávat od začátku indexu - např. binární prohledávání (popsáno dále) je o mnoho rychlejší než prohledávání sekvenční [11].

B-Tree indexy

Úložný engine InnoDB používá pro své indexy strukturu B-Tree. Abstraktní reprezentaci indexu B-tree zobrazuje obrázek 4.4.



Obrázek 4.4: Index vybudovaný na struktuře B-tree - zdroj: [28]

B-Tree index urychluje přístup k datům, protože úložný engine nemusí projít celou tabulku, pokud má najít požadovaná data. Místo toho začne na kořenovém uzlu (který na výše uvedeném obrázku není vidět). Sloty v kořenovém uzlu obsahují ukazatele (pointers) na dceřiné uzly a úložný engine postupuje podle těchto ukazatelů. Správný ukazatel hledá tak, že prochází hodnoty ve stránkách uzlu, který definuje horní a dolní mez hodnot v dceřiných uzlech. Nakonec úložný engine buď určí, že požadovaná hodnota neexistuje, nebo úspěšně dosáhne koncové stránky (leaf page, list stromu). Koncové stránky jsou speciální, protože obsahují ukazatele na indexovaná data, nikoliv ukazatele na jiné stránky. Výše uvedený obrázek zobrazuje pouze jednu uzlovou stránku a její koncové stránky, mezi kořenem a koncovými stránkami však může být mnoho úrovní uzlových stránek (větví stromu) [28].

Použití indexů

MySQL používá indexy pro urychlení vyhledávání řádků v souladu s požadavky klauzule `WHERE` nebo odpovídajících řádků z jiných tabulek, jak je popsáno výše. Za účelem zvýšení výkonnosti používá indexy ještě v dalších typech operací:

- V dotazech, ve kterých se vyskytuje funkce `MIN()` nebo `MAX()`, se dá, je-li sloupec indexován, najít jeho nejmenší respektive nejvyšší hodnota velmi rychle.

- Indexy mohou být využity k urychlení operací řazení a seskupování v klauzulích `ORDER BY` a `GROUP BY` [11].

Nevýhody indexování

Přestože v praxi výhody indexů obvykle značně převažují jejich nevýhody, je vhodné na ně upozornit:

- Indexy zrychlují získávání dat, ale zpomalují operace vkládání, odstraňování a aktualizace hodnot indexových sloupců. Dochází k tomu proto, že zápis záznamu neznamená jen zápis řádku do souboru dat, ale také změnu všech indexů.

- Index zabírá místo na disku. Kvůli tomu je možné dosáhnout horní meze velikosti tabulky dříve, než kdyby v tabulce žádné indexy neexistovaly [11].

Volba indexů

Jak uvádí DuBois [11], vodítka, jak identifikovat sloupce, které jsou vhodnými kandidáty na indexování, jsou následující:

- Indexují se sloupce, používané pro vyhledávání, seskupování nebo řazení. Jinými slovy nejlepšími kandidáty pro indexování jsou sloupce, které se v dotazech objevují v klauzulích `WHERE`, sloupce uváděné v klauzulích spojení (`JOIN`) nebo v klauzulích `GROUP BY` či `ORDER BY`.

- Je vhodné používat jedinečné indexy. Indexy totiž fungují lépe u sloupců, které mají jedinečné hodnoty a hůře u sloupců, které mají hodně duplicitních hodnot.

- Využíváme výhod levých prefixů. Pokud vytvoříme složený index tvořený n sloupci, vytvoříme ve skutečnosti n indexů, které bude moci MySQL využívat. Složený index slouží jako několik indexů, protože pro hledání řádků se dá použít jakákoli sada sloupců indexu zleva. Takové sadě sloupců se říká *levý prefix*. Dejme tomu, že máme tabulku se složeným indexem tvořeným sloupci `město`, `stát` a `PSČ`. Řádky se v indexu řadí podle `stát/město/PSČ`, takže jsou také automaticky seřazeny podle `stát/město` a podle `stát`. Díky tomu může MySQL z tohoto indexu těžit i tehdy, pokud v dotazu specifikujeme jen řazení podle států nebo jen podle států a měst. Naopak MySQL nemůže tento index využívat v případě, kdy při vyhledávání nejsou zahrnuty sloupce nacházející se nejvíce vlevo.

- Je třeba zvolit rozumný počet indexů. Řídit se heslem „čím více, tím lépe“ by byla v případě počtu indexů chyba. Každý dodatečný index zabere další prostor na disku a sníží výkonnost zapisovaných operací. Při modifikaci tabulek se indexy musí zaktualizovat a čím více indexů máme, tím déle to trvá. Je tedy vhodné udržovat jen tolik indexů, kolik je skutečně potřeba. Pokud uvažujeme o přidání dalšího indexu do tabulky, která už nějaký index má, zkontrolujeme, zda plánovaný index není levým prefixem nějakého existujícího vícesloupcového indexu. Pokud ano, index přidávat nemusíme, protože už jej vlastně máme.

- Je třeba brát v úvahu typ porovnávání. Indexy se obecně používají pro operace `<`, `<=`, `=`, `>=`, `>`, `BETWEEN` a pro operace `LIKE`, je-li ve vzorku jako prefix literál [11].

4.4.2 Volba vhodných datových typů sloupců

Výběr vhodného datového typu pro je klíčový faktor pro získání vysokého výkonu. Prvním krokem při rozhodování, jaký datový typ pro daný sloupec použít, je určení vhodné všeobecné třídy typu: číslo, řetězec, datum a čas atd. Dalším krokem je volba konkrétního typu. Jednotlivé datové typy se odlišují rozsahem hodnot, který do nich lze uložit, fyzickým prostorem, který zabírají (na disku a v paměti) a přesností, kterou povolují [28].

Základní pravidla pro volbu vhodného datového typu podle Schwartze [28] jsou:

- Menší datové typy jsou vhodnější. Pro reprezentaci dat je vhodné používat ten nejmenší datový typ, který lze k uložení dat užít. Menší datové typy jsou obvykle rychlejší, protože zabírají méně místa na disku, v paměti, i v CPU cache. Také obvykle potřebují ke zpracování menší množství cyklů CPU. Na druhou stranu je třeba dát pozor, aby nedošlo k podcenění rozsahu hodnot, které jsou potřeba ukládat, protože pozdější zvyšování rozsahu datového typu na několika místech schématu může být strastiplnou operací.

- Jednodušší datové typy jsou vhodnější. Operace nad jednoduššími datovými typy obvykle vyžadují méně cyklů CPU. Například porovnávání celých čísel je méně náročné než porovnávání znaků, protože porovnávání znaků komplikují různé znakové sady a kolace (pravidla určující pořadí znaků v dané řeči).

- Použití sloupců definovaných jako `NOT NULL` je vhodnější. Optimalizovat dotazy, odkazující na sloupce, ve kterých může být hodnota `NULL`

je totiž pro MySQL obtížnější. Tato skutečnost komplikuje indexy i porovnávání hodnot. Sloupec, do něhož se může ukládat hodnota `NULL`, potřebuje více úložného prostoru, pokud se takový sloupec indexuje, pro každou položku se požaduje jeden bajt navíc. Jestliže je potřeba do tabulky uložit informaci o tom, že daná hodnota chybí, je vhodné zamyslet se nad jiným způsobem reprezentace této skutečnosti (např. 0, prázdný řetězec nebo nějaká speciální hodnota).

Dopad změn sloupce s atributem `NULL` na `NOT NULL` na výkon je obvykle malý. Pokud tedy pracujeme s již existujícím schématem a nejsme si jisti, že tato změna nezpůsobí žádné problémy, můžeme toto nastavení ponechat. Jestliže ovšem navrhujeme schéma od začátku a teprve uvažujeme nad tím, které sloupce budeme indexovat, je vhodné nepovolovat v těchto sloupcích `NULL` [28].

Celá čísla

MySQL podporuje standardní číselné typy dle SQL `INTEGER (INT)` a `SMALLINT`. Navíc přidává podporu `TINYINT, MEDIUMINT` a `BIGINT`. Celočíselné typy mohou mít nepovinný atribut `UNSIGNED`, který nepovoluje záporná čísla a zdvojnásobuje horní mez ukládání kladných čísel [18]. Požadovaný úložný prostor a rozsah datových typů zobrazuje následující tabulka.

Dat.typ	Paměť	Min.hodnota	Max.hodnota
	(byty)	(`UNSIGNED`)	(`UNSIGNED`)
`TINYINT`	1	-128	127
		0	255
`SMALLINT`	2	-32 768	32 767
		0	65 535
`MEDIUMINT`	3	-8 388 608	8 388 607
		0	16 777 215
`INT`	4	-2 147 483 648	2 147 483 647
		0	4 294 967 295
`BIGINT`	8	-9 223 372 036 854 775 808	9 223 372 036 854 775 807
		0	18 446 744 073 709 551 615

Tabulka 4.3: Požadovaný úložný prostor a rozsah datových typů - zdroj: [18]

Reálná čísla

MySQL podporuje tzv. exaktní a neexaktní datové typy. Typy `FLOAT` a `DOUBLE` jsou neexaktní, podporují přibližné výpočty se standardními operacemi pohyblivé řádové čárky. `FLOAT` požaduje 4 bajty úložného prostoru, `DOUBLE` 8 bajtů.

Typ `DECIMAL` podporuje přesné matematické operace (od MySQL 5.0 výše) a je určen k ukládání exaktních čísel s desetinnými místy. `DECIMAL` požaduje 4 bajty pro každých 9 číslic (před i za desetinnou čárkou).

Typy `FLOAT` a `DOUBLE` obvykle zaberou méně místa než `DECIMAL` (při ukládání stejného rozsahu hodnot). Typ `DECIMAL` znamená kromě dodatečných požadavků na místo také dodatečné náklady na výpočty, tento typ je tak vhodné používat jen v případě, pokud jsou opravdu potřeba přesné výsledky matematických operací pro čísla s desetinnými místy [15, 17, 28].

Řetězcové typy

MySQL podporuje velké množství řetězcových datových typů s mnoha variantami. Hlavní dva řetězcové typy jsou `VARCHAR` a `CHAR`.

`VARCHAR` je nejběžnější řetězcový datový typ, ukládá znakové řetězce proměnlivé délky. Díky tomu může zabírat méně místa než typy s pevnou délkou (protože zabírá jen tolik místa, kolik je nutné) a přispívá tak k vyššímu výkonu.

Naopak `CHAR` má pevnou délku, což znamená, že MySQL vždy alokuje dostatek úložného prostoru pro specifikovaný počet znaků. Použití tohoto typu je vhodný tehdy, chceme-li ukládat velmi krátké řetězce nebo když všechny hodnoty mají velmi podobnou délku [28].

Typy pro datum a čas

Pro ukládání data a času nabízí MySQL dva velmi podobné datové typy: `DATETIME` a `TIMESTAMP`.

Typ `DATETIME` může obsahovat velké rozpětí hodnot, od roku 1001 až do roku 9999 s přesností na jednu sekundu.

Typ `TIMESTAMP` ukládá počet sekund, které uplynuly od půlnoci 1.ledna 1970 greenwichského hlavního času (GMT) - stejně jako časová značka UNIXU. `TIMESTAMP` používá jen 4 bajty úložného prostoru, má proto menší rozpětí než `DATETIME`: od roku 1970 po část roku 2038.

Obecně řečeno - pokud je možné použít typ `TIMESTAMP`, použijme ho, protože z hlediska spotřeby místa je efektivnější [28].

4.4.3 Výběr identifikačního sloupce

Vhodný výběr datového typu pro identifikační sloupec (identifier column) je velmi důležitá věc. Tyto sloupce jsou totiž častěji než jiné sloupce porovnávány s jinými hodnotami (například při spojování tabulek) a častěji se v nich i vyhledává. Bývají také používány v jiných tabulkách jako cizí

klíče. Volbou datového typu pro identifikační sloupec stanovujeme také datový typ sloupce pro všechny tabulky, které budou spojeny prostřednictvím relací. Datové typy sloupců, přes které se realizuje spojení tabulek, by totiž měly souhlasit přesně, včetně vlastností jako `UNSIGNED`. Kombinace datových typů může mít kromě negativních dopadů na výkon vliv také na výskyt různých chyb při implicitních konverzích během porovnávacích operací. Úložný engine InnoDB vytvoření cizích klíčů, pokud datové typy sloupců přesně nesouhlasí, ani neumožňuje [28].

Vhodné je indexovat krátké hodnoty. Menší index znamená méně zabraného místo na disku, méně vstupních a výstupních operací s diskem a porovnávání kratších hodnot je rychlejší. Rozumné je zvolit nejmenší velikost, která pojme požadované rozpětí hodnot (s rezervou pro budoucí růst). Nejlepší volbou pro identifikátory bývají celá čísla, jejich porovnávání je rychlé a pracují s `AUTO_INCREMENT`. Naopak použití řetězcových typů pro identifikátory je lepší se vyhnout, protože zabírají více místa v paměti a jsou obecně pomalejší než celočíselné typy [11, 28].

4.5 Optimalizace výkonu SQL dotazů

Optimalizace schématu a indexů, popsaná v předchozí kapitole, je jedna z nutných podmínek získání vysokého výkonu. Pracovat pouze se schématem a indexy ale nestačí, důležité jsou také dobře navržené dotazy. Se špatně navrženými dotazy ani optimálně navržené schéma mnoho nesvede. [11]

4.5.1 Optimalizace přístupu k datům

Základní příčinou špatného výkonu dotazu bývá to, že pracuje s příliš velkým množstvím dat. Většina špatně optimalizovaných dotazů se dá upravit tak, aby dotaz přistupoval k menšímu množství dat. K tomu je potřeba zjistit, zda aplikace nezískává více dat, než je nezbytně nutné - některé dotazy požadují více dat, než potřebují (přistupuje se k příliš mnoha řádkům nebo příliš mnoha sloupcům tabulek) a nepotřebná data poté bez užitku odhodí. To znamená více práce pro MySQL server, vyšší režii na síti a vyšší spotřebu paměti a zdrojů CPU na aplikačním serveru. Typické chyby jsou:

- Načítání více řádků, než je nutné. Příklad: v aplikaci chceme zobrazit 10 nejnovějších článků pro zpravodajský web. `SELECT` dotazem načteme všechny články z databáze a zobrazíme jen prvních 10. Zde je nejlepším řešení přidat do dotazu klauzuli `LIMIT`.

- Načítání všech sloupců. Použití příkazu `SELECT *` je sice pohodlné, ale nevhodné. Získávaní všech sloupců přináší zvýšené nároky na I/O, paměť a režii CPU pro server. Navíc se tím mohou potlačit různé optimalizace, jako např. pokrývající indexy. Vždy je tedy třeba zamyslet se, data z kterých sloupců jsou skutečně potřeba a tyto sloupce v dotazu vyjmenovat.
- Načítání všech sloupců z několika spojených tabulek. Např. chceme-li získat všechny knihy od George Orwella, je nevhodné psát dotaz takto:

```
SELECT
       *
  FROM books
  JOIN books_authors ON books.author_id = books_authors.author_id
  JOIN authors ON books_authors.authors_id = author.author_id
 WHERE author.surname = 'Orwell'
   AND author.name = 'George'
```

  Takový dotaz totiž vrátí všechny sloupce ze všech tří tabulek. Vhodnější je dotaz formulovat takto:

```
SELECT
       books.*
  FROM books ...
```

  Díky tomu se načtou data jen z tabulky `books`. Ještě lepší variantou ovšem zůstává vyjmenování sloupců v dotazu [11].

4.5.2 Restrukturalizace dotazů

Cílem restrukturalizace dotazů je nalézt alternativní způsoby pro získání požadovaných výsledků. Dotaz je často možné přepsat tak, že vrátí totožné výsledky při nižších nákladech. Nemusí se ale nutně jednat o stejnou výslednou sadu získanou z MySQL. Pokud to bude mít příznivý dopad na efektivitu, je někdy možné získat vyšší výkon použitím dotazu, vracející odlišné výsledky, které jsou na straně aplikace následně převedeny na požadované výsledky [28].

4.5.3 Hromadné vkládání dat

Dokumentace MySQL doporučuje pří jednorázovém vkládání velkého množství řádků do InnoDB tabulek následující kroky:

- Vypnutí autocommit módu. Pokud je autocommit mód zapnutý, je po každém `INSERT` příkazu prováděn záznam do logu. Vypnutí autocommit módu se provede následující sérií příkazů:

```
SET autocommit=0;
... SQL příkazy pro import ...
COMMIT;
```

- Vypnutí kontroly unikátnosti. Pokud tabulka obsahuje sekundární unikátní klíče, je možné zrychlit import dat dočasným vypnutím kontroly unikátnosti. U velkých tabulek pak dochází k velké úspoře I/O disku, protože InnoDB může použít change buffer pro zápis sekundárních indexů v dávce. Je ovšem potřeba se ujistit, že vkládaná data neobsahují duplicitní hodnoty. Vypnutí kontroly unikátních klíčů se provede takto:

```
SET unique_checks=0;
... SQL příkazy pro import ...
SET unique_checks=1;
```

- Vypnutí kontroly cizích klíčů. Pokud tabulka obsahuje cizí klíče, je možné zrychlit import dat dočasným vypnutím kontroly cizích klíčů. U velkých tabulek pak dochází k velké úspoře I/O disku.

```
SET foreign_key_checks=0;
... SQL příkazy pro import ...
SET foreign_key_checks=1;
```

- Vkládání řádků v pořadí primárního klíče. Při vkládání velkého množství dat je takový způsob vkládání rychlejší. Tabulky InnoDB používají clusterovaný index[7], díky čemuž je použití dat v pořadí klíče rychlé [5].

4.5.4 Vkládání výchozích hodnot

Při vkládání záznamů příkazem `INSERT` je často možné vložit data bez specifikování výchozích hodnot. V tom případě je vhodné nechat vložit data MySQL a nespecifikovat je. Tento přístup vede ke kratším dotazům, což znamená, že MySQL server může strávit méně času zpracováním dotazu [29].

[7]Každá InnoDB tabulka má speciální index, nazývaný clusterovaný. Pokud má tabulka definovaný primární klíč, InnoDB jej použije jako clusterovaný index. Pokud ne, je jako clusterovaný index použit první unikátní index. Jestliže není definovaný ani unikátní index, generuje InnoDB skrytý clusterovaný index - unikátní šestibajtovou hodnotu zvanou `rowid` [6].

5 Optimalizace databáze aplikace

5.1 Optimalizace schématu a indexování

5.1.1 Nastavení indexů tabulek

Aktuální nastavení indexů respektovalo většinu pravidel uvedených v kapitole 4.4.1. Byly indexovány vhodné sloupce, používány jedinečné indexy a počet indexů odpovídal tomu, kolik jich bylo potřeba. U několika dotazů ale nebylo využíváno výhod levých prefixů a zvýšení výkonu bylo možné docílit přidáním tzv. pokrývajících indexů (covering indexes).

Využívání výhod levých prefixů

Analýza doby provádění SQL dotazů ukázala pomalé zpracování následujícího dotazu. Dotaz slouží k načítání poptávaného respektive nabízeného množství při jednotlivých cenách na jednotlivých trzích pro daný subjekt.

```
SELECT
       p.cislo_kola,
       p.mezni_cena,
       p.poptavane_mnozstvi,
       t.nazev_trhu
  FROM poptavky p,
       trhy t
 WHERE p.id_subjektu = 'straka92'
   AND p.id_trhu = t.id_trhu
 ORDER BY p.cislo_kola ASC,
          t.nazev_trhu,
          p.mezni_cena DESC;
```

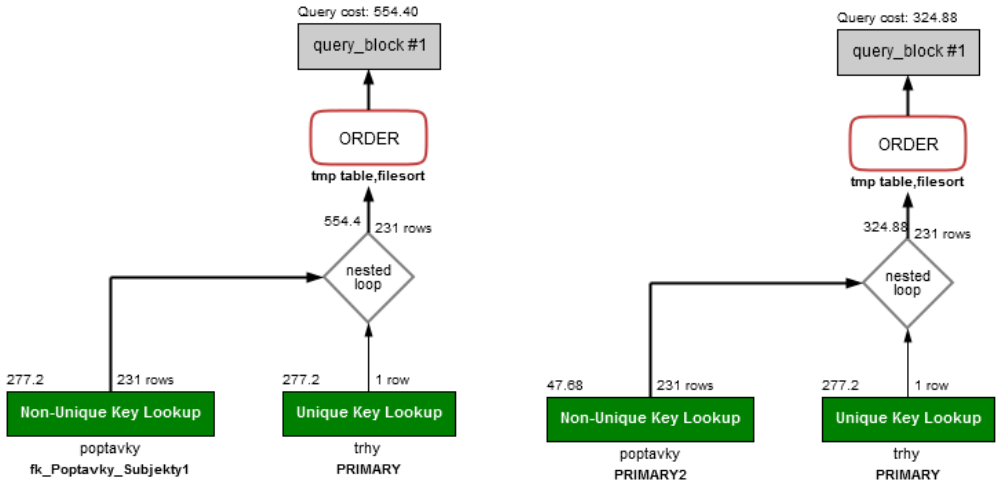
Index tabulek `poptavky` a `nabidky` vypadal následovně:

```
PRIMARY KEY (‘cislo_kola‘,‘id_subjektu‘,‘id_trhu‘,‘mezni_cena‘)
```

Pro výše zmíněný dotaz není při tomto indexu možné využít jeho levý prefix. Proto byl index upraven následovně:

```
PRIMARY KEY (‘id_subjektu‘,‘cislo_kola,‘id_trhu‘,‘mezni_cena‘)
```

Obrázek 5.1 zobrazuje Visual Explain Plan[1] výše uvedeného dotazu před a po úpravě.



(a) (b)

Obrázek 5.1: Plán vykonání dotazu před (a) a po (b) provedení úpravy - zdroj: vlastní zpracování

Vidíme snížení nákladů (Query cost) na vykonání výše uvedeného dotazu na cca 58%.

Index v podobě `PRIMARY KEY('cislo_kola','id_subjektu', ...)` vystupoval ve více tabulkách databáze. Nad těmito tabulkami ovšem byly podobně jako výše volány dotazy provádějící spojení přes sloupec `id_subjektu` a tím pádem docházelo ke stejné situaci jako výše - nemohl se využít levý prefix. Proto byla obdobná úprava pořadí sloupců v indexu provedena také pro tabulky:

- `rezidenti`, `rezidenti_vek`,
- `toky_kapitaloveho_zbozi_sklad`, `toky_kapitaloveho_zbozi_vyroba`, `toky_penez`, `toky_lidskeho_kapitalu`, `toky_obligaci`, `toky_prace`, `toky_spotrebniho_zbozi`.
- `ziskane_body`.

[1]Visual Explain Plan ukazuje vizuální reprezentaci příkazu MySQL `EXPLAIN`. Ten poskytuje informace o tom, jak MySQL provádí dotaz. Visual Explain Plan je vhodným nástrojem pro analýzu nákladů na vykonání dotazu. Ty jsou zobrazeny v horní části pod označením Query cost [32].

Přidání pokrývajících (covering) indexů

Indexy slouží k rychlému a efektivnímu vyhledání požadovaných řádků. MySQL však umí indexy používat i tak, že získá data sloupce přímo z indexu (a nemusí tak číst celý řádek). Indexu, který obsahuje (neboli pokrývá) všechna data, potřebná k uspokojení dotazu, se říká pokrývající index. Pokrývající indexy jsou velmi mocným nástrojem a mohou dramaticky pozvednout výkon. Čtení jen indexu a ne dat je výhodné z těchto důvodů:

- Položky indexu bývají mnohem kratší než plná délka řádku, takže pokud bude MySQL číst pouze index, bude přistupovat k výrazně menšímu množství dat.
- Většina úložných enginů cachuje indexy lépe než data.
- Pokrývající indexy jsou užitečné zejména pro tabulky InnoDB (kvůli jeho clusterovaným indexům). Sekundární indexy InnoDB totiž ve svých koncových uzlech obsahují hodnoty primárního klíče řádku. Jinými slovy, sekundární index, který pokrývá dotaz, nemusí podniknout žádné další indexové vyhledávání v primárním klíči [28].

Zejména nad tabulkami toků (`toky_penez`, `toky_prace` atd., více v kapitole 4.2) a nad tabulkou `ziskane_body` jsou často volány dotazy typu `SELECT SUM(sloupec)`. Tyto dotazy slouží k načítání stavu zásob hráče (suma toků) nebo pro generování žebříčku hráčů (suma bodů hráče). Dotaz vypadá např. takto:

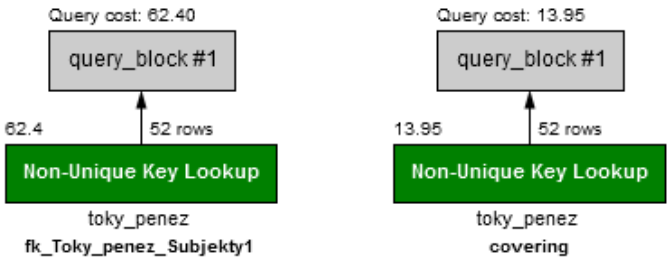
```
SELECT
       SUM(hodnota_toku) AS mnozstvi_penez
  FROM toky_penez
 WHERE id_subjektu = 'straka92';
```

Pro výše zmiňované tabulky toků byly nastaveny pokrývající indexy. Příkaz pro přidání pokrývajícího indexu[2] např. pro tabulku `toky_penez`:

```
ALTER TABLE `toky_penez`
ADD KEY `covering` (`id_subjektu`,`hodnota_toku`);
```

Obrázek 5.2 zobrazuje Visual Explain Plan výše uvedeného dotazu před a po úpravě.

[2]Dle dokumentace MySQL jsou `INDEX` a `KEY` synonyma. Toto označení bylo zavedeno z důvodu kompatibility s jinými databázovými systémy. [8].



(a) (b)

Obrázek 5.2: Plán vykonání dotazu před (a) a po (b) provedení úpravy - zdroj: vlastní zpracování

Vidíme snížení nákladů (Query cost) na vykonání výše uvedeného dotazu na cca 22%.

5.1.2 Volba vhodných datových typů sloupců a identifikačního sloupce

U většiny tabulek byly používány vhodné typy sloupců - byly používány malé a jednoduché datové typy a sloupce byly definovány jako `NOT NULL`. Také identifikační sloupce byly definovány dle doporučení uvedených v kapitole 4.4.3.

Pro celočíselné hodnoty byl v tabulkách databáze ve většině případů používán datový typ `INT`. Změna datového typu sloupců z `INT` na `TINYINT` nepřinesla znatelné zrychlení, nebyla tedy realizována.

Nevhodně zvolené datové typy byly u následující sloupců: v tabulkách `subjekty` a `meny` byl pro identifikační sloupec (`id_subjektu` respektive `id_meny`) použit datový typ `VARCHAR`. Jak je uvedeno v kapitole 4.4.3, nejlepší volbou pro identifikátory je celé číslo. Proto byly datové typy těchto sloupců upraveny následovně:

- Datový typ sloupce `id_subjektu` byl upraven na `SMALLINT UNSIGNED`. Rozsah typu `TINYINT UNSIGNED` 255 by vzhledem k počtu subjektů nebyl dostatečný, naopak rozsah typu `SMALLINT UNSIGNED` 65 535 již dostačovat bude.

- Datový typ sloupce `id_meny` upraven na `TINYINT UNSIGNED`, rozsah 255 je pro počet měn naprosto dostačující.

Protože byla úprava prováděna za běhu hry (v průběhu zimního semestru), bylo potřeba připravit algoritmus, který textovým identifikátorům (subjektu či meny) přiřadí číselný identifikátor a v rámci celé databáze nahradí textový identifikátor číselným. Textový identifikátor subjektu a měny bylo však potřeba zachovat pro výpis uživateli ve hře (např. aby hráč viděl, že používaná měna v dané ekonomice je „koruna“ a nikoli „11“). Proto bylo nutné tabulku `subjekty` rozšířit o sloupec `nazev_subjektu`, Tabulka `meny` již sloupec `nazev_meny` obsahovala (ale jeho hodnota byla shodná s hodnotou ve sloupci `id_meny`). Na všech místech v kódu, kde se uživateli vypisují hodnoty, bylo nutné provést úpravu, aby místo číselné hodnoty byla stále vypisována textová hodnota.

id_subjektu	**id_druhu_subjektu**	**id_integracniho_celku**	**login_hrace**
acingros	podnikatel	2016ZS	acingros
adus09	podnikatel	2016ZS	adus09
alenac	podnikatel	2016ZS	alenac
alim	podnikatel	2016ZS	alim
astankov	podnikatel	2016ZS	astankov

Tabulka 5.1: Struktura tabulky subjekty před úpravou - zdroj: vlastní zpracování

id_subjektu	**nazev_subjektu**	**id_druhu_subjektu**	**id_integracniho_celku**	**login_hrace**
1	acingros	podnikatel	2016ZS	acingros
2	adus09	podnikatel	2016ZS	adus09
3	alenac	podnikatel	2016ZS	alenac
4	alim	podnikatel	2016ZS	alim
5	astankov	podnikatel	2016ZS	astankov

Tabulka 5.2: Struktura tabulky subjekty po úpravě - zdroj: vlastní zpracování

Další nevhodně nastavené datové typy sloupců byly v tabulkách `aktivita_subjektu` a `prikazy_vyroby`. V obou tabulkách byl datový typ sloupce `cislo_kola` chybně definován jako `VARCHAR` a v obou tabulkách byl tedy upraven na `TINYINT UNSIGNED`.

5.2 Optimalizace výkonu SQL dotazů

V SQL dotazech často nebyly vyjmenovány požadované sloupce, ale byla použita hvězdička, což je, jak je uvedeno v kapitole 4.5.1 nevhodné.

Všechny dotazy obsahující hvězdičku byly proto přepsány - hvězdička byla nahrazena výčtem sloupců tabulek. Jednalo se o celkem náročný úkon, protože bylo potřeba prostudovat PHP kód pracující s výsledky SQL dotazu a zkoumat, hodnoty z jakých sloupců tabulky jsou v kódu používány. Pokud by některá z hodnot používaných v kódu nebyla dotazem načtena, vedlo by to k chybnému chování aplikace.

Příklad původní podoby dotazu:

```
SELECT
       *
  FROM nabidky
 WHERE cislo_kola = 1
   AND id_subjektu = 'straka92'
   AND id_trhu = 17;
```

Příklad nové podoby dotazu:

```
SELECT
       cislo_kola,
       id_subjektu,
       id_trhu,
       mezni_cena,
       id_meny,
       nabizene_mnozstvi
  FROM nabidky
 WHERE cislo_kola = 1
   AND id_subjektu = 'straka92'
   AND id_trhu = 17;
```

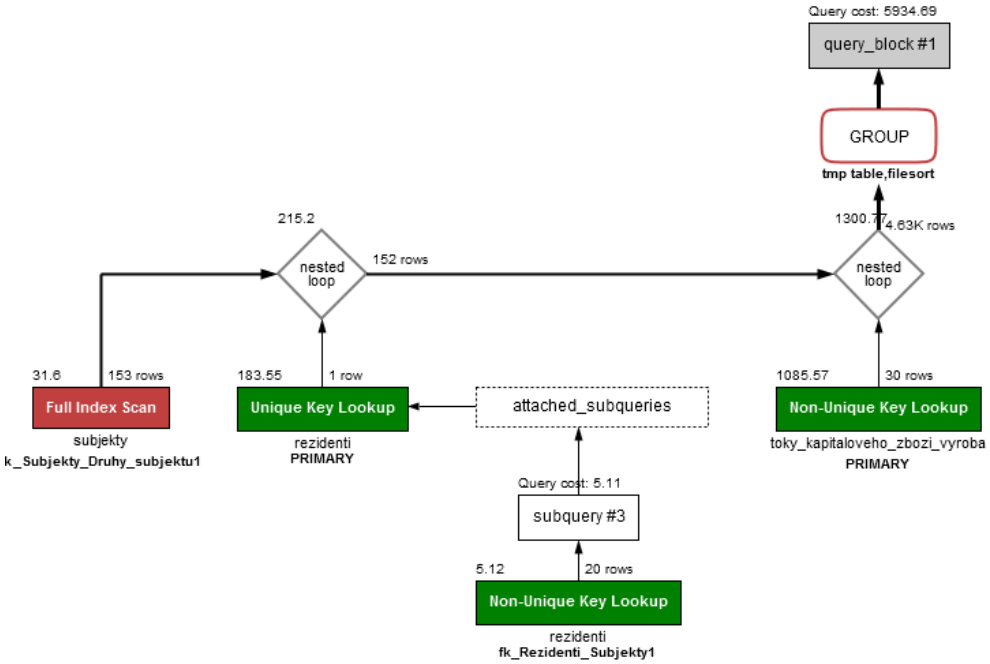
5.2.1 Restrukturalizace dotazů

Čtyři SQL dotazy, které analýza doby provádění dotazů odhalila jako problémové, byly přeformulovány tak, aby vracely stejný výsledek s nižšími náklady.

SQL dotaz č.1 Tento dotaz vrací množství kapitálového zboží ve výrobě v daném kole pro jednotlivé subjekty.

```
SELECT
       SUM(t.hodnota_toku) AS mnozstvi_kapitaloveho_zbozi_vyroba,
       t.id_subjektu
  FROM toky_kapitaloveho_zbozi_vyroba t
 WHERE t.id_subjektu IN
 (SELECT DISTINCT
         r.id_subjektu
    FROM subjekty s,
         rezidenti r
   WHERE s.id_subjektu = r.id_subjektu
     AND r.id_ekonomiky = 'kem'
     AND r.cislo_kola =
  (SELECT
          MAX(cislo_kola)
     FROM rezidenti r2
    WHERE r2.id_subjektu = s.id_subjektu
  )
 )
GROUP BY id_subjektu;
```

Obrázek 5.3 zobrazuje plán vykonání tohoto dotazu.

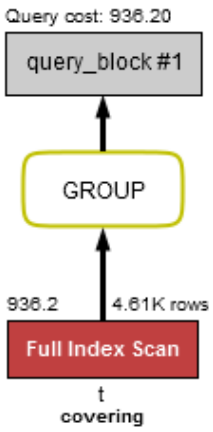


Obrázek 5.3: Plán vykonání původního dotazu - zdroj: vlastní zpracování

Volání dvou poddotazů v původní podobě dotazu je náročná operace. Klauzuli `IN` je možné nahradit klauzulí `EXISTS`. Dotaz lze přepsat do této podoby:

```
SELECT
       SUM(t.hodnota_toku) AS mnozstvi_kapitaloveho_zbozi_vyroba,
       t.id_subjektu
  FROM toky_kapitaloveho_zbozi_vyroba t
 WHERE EXISTS
 (SELECT
         *
    FROM rezidenti r
   WHERE r.id_ekonomiky = 'kem'
     AND r.cislo_kola =
  (SELECT
          MAX(cislo_kola)
     FROM rezidenti r2
    WHERE r.id_subjektu = r2.id_subjektu
  )
 )
GROUP BY t.id_subjektu;
```

Obrázek 5.4 zobrazuje plán vykonání upraveného dotazu. Vidíme snížení nákladů vykonání dotazu na cca 16%.

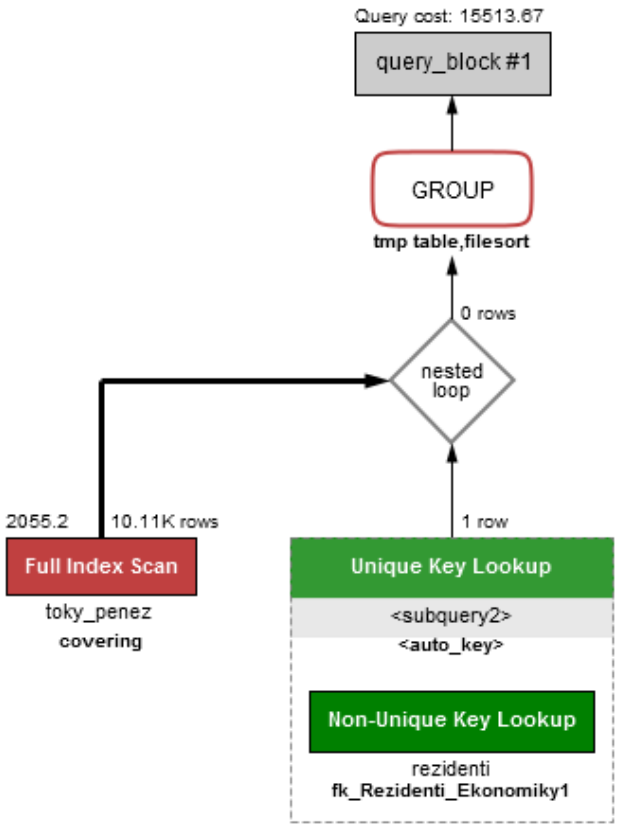


Obrázek 5.4: Plán vykonání upraveného dotazu - zdroj: vlastní zpracování

SQL dotaz č.2 Tento dotaz vrací sumu toků peněz v daném kole pro jednotlivé subjekty.

```
SELECT
       t.id_subjektu,
       t.id_meny,
       SUM(t.hodnota_toku) AS hodnota_toku
  FROM toky_penez t
 WHERE t.id_subjektu IN
 (SELECT
         id_subjektu
    FROM rezidenti r
   WHERE r.id_ekonomiky = 'kem'
 )
GROUP BY t.id_subjektu,
         t.id_meny
```

Obrázek 5.5 zobrazuje plán vykonání tohoto dotazu.

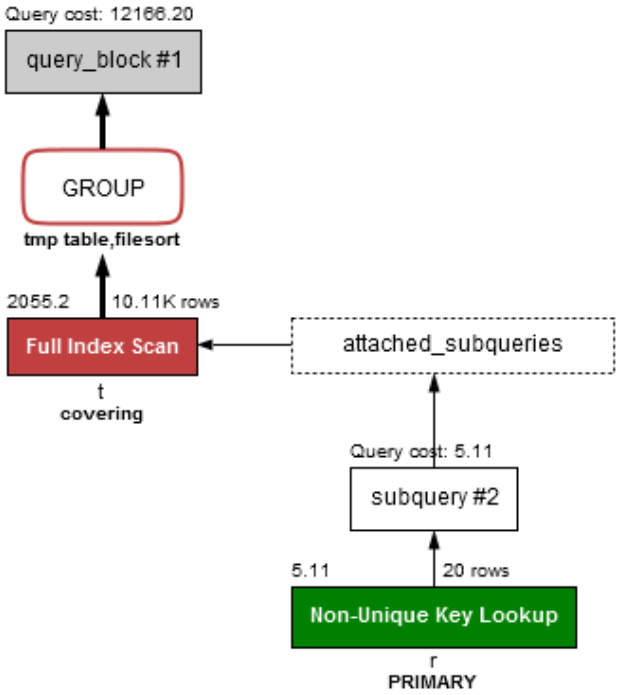


Obrázek 5.5: Plán vykonání původního dotazu - zdroj: vlastní zpracování

Dotaz je možné přepsat do této podoby (opět nahrazením klauzule `IN` klauzulí `EXISTS`):

```
SELECT
       t.id_subjektu,
       t.id_meny,
       SUM(t.hodnota_toku) AS hodnota_toku
  FROM toky_penez t
 WHERE EXISTS
 (SELECT
         *
    FROM rezidenti r
   WHERE r.id_ekonomiky = 'kem'
     AND t.id_subjektu = r.id_subjektu
 )
GROUP BY t.id_subjektu,
         t.id_meny
```

Plán vykonání upraveného dotazu zobrazuje obrázek 5.6. Vidíme snížení nákladu na vykonání dotazu na cca 78%.

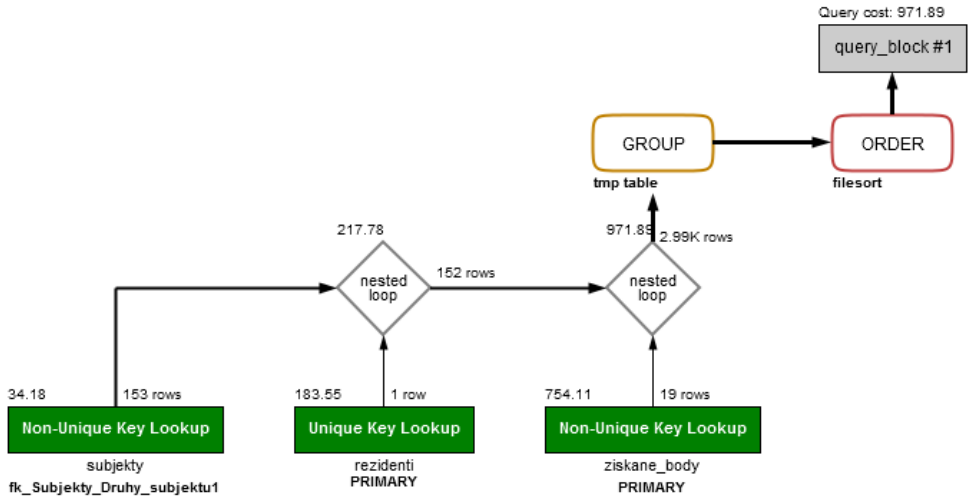


Obrázek 5.6: Plán vykonání upraveného dotazu - zdroj: vlastní zpracování

SQL dotaz č.3 Dotaz vrací žebříček subjektů dané ekonomiky podle počtu získaných bodů.

```
SELECT
       zb.id_subjektu,
       SUM(zb.body_ziskane_v_kole) AS body_ziskane_v_kole
  FROM ziskane_body zb,
       subjekty s
 WHERE s.id_subjektu = zb.id_subjektu
   AND s.id_druhu_subjektu = 'podnikatel'
   AND zb.id_subjektu IN
 (SELECT
        id_subjektu
    FROM rezidenti
   WHERE id_ekonomiky = 'kem'
     AND cislo_kola = 20
 )
GROUP BY zb.id_subjektu
ORDER BY body_ziskane_v_kole DESC
```

Obrázek 5.7 zobrazuje plán vykonání tohoto dotazu.

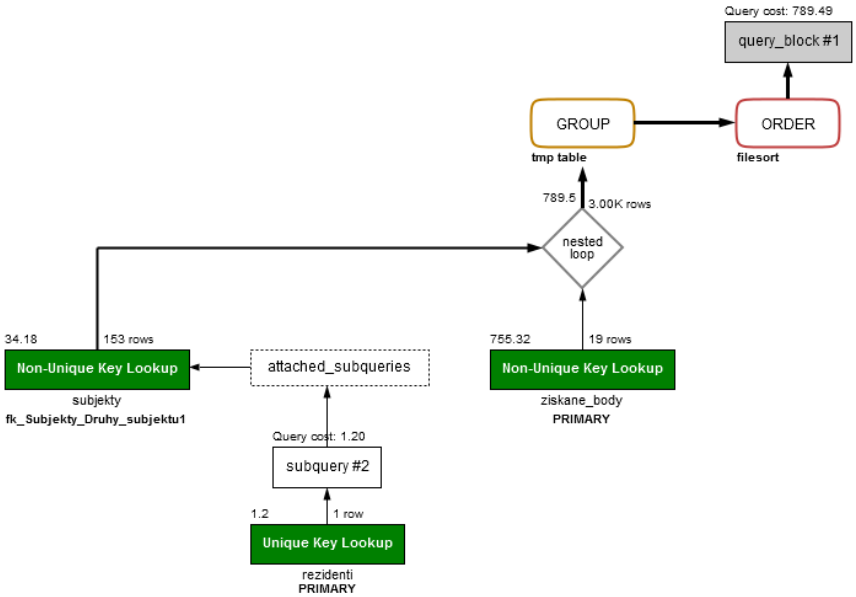


Obrázek 5.7: Plán vykonání původního dotazu - zdroj: vlastní zpracování

Klauzuli `IN` je opět možné nahradit klauzulí `EXIST` a dotaz pak přepsat následovně.

```
SELECT
       zb.id_subjektu,
       s.nazev_subjektu,
       SUM(body_ziskane_v_kole) AS body_ziskane_v_kole
  FROM ziskane_body zb,
       subjekty s
 WHERE s.id_subjektu = zb.id_subjektu
   AND s.id_druhu_subjektu = 'podnikatel'
   AND EXISTS
 (SELECT
         r.id_subjektu
    FROM rezidenti r
   WHERE r.id_ekonomiky = 'kem'
   AND r.cislo_kola = 20
   AND zb.id_subjektu = r.id_subjektu
 )
GROUP BY zb.id_subjektu
ORDER BY body_ziskane_v_kole DESC;
```

Plán vykonání upraveného dotazu zobrazuje obrázek 5.8. Vidíme snížení nákladu na vykonání dotazu na cca 81%.



Obrázek 5.8: Plán vykonání upraveného dotazu - zdroj: vlastní zpracování

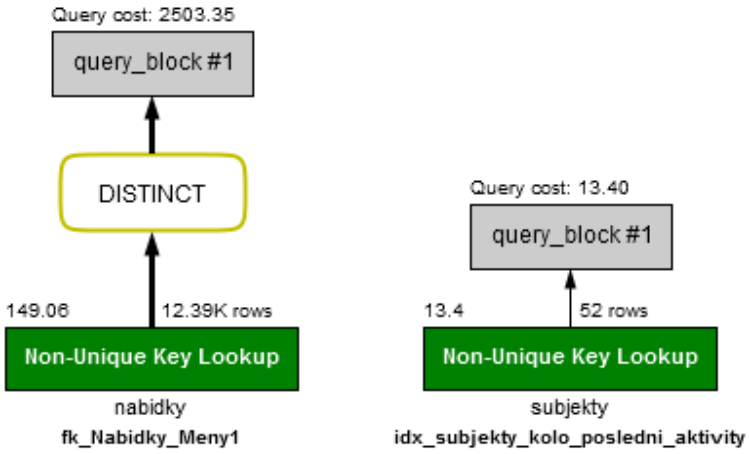
SQL dotaz č.4 Dotaz slouží pro získání seznamu subjektů, kteří v daném kole zadali na trhu nabídku respektive poptávku.

```
SELECT DISTINCT
       id_subjektu
  FROM nabidky
 WHERE cislo_kola = 13
   AND id_trhu = 17
   AND id_meny = 12;
```

Použití klauzule `DISTINCT` je drahé [28]. Aby bylo možné se mu vyhnout, byl zaveden následující mechanismus. Do tabulky `subjekty` byl přidán sloupec `kolo_posledni_aktivity`, na tomto sloupci byl vytvořen index. Při každém zadání poptávky nebo nabídky hráčem je do tohoto sloupce uloženo číslo aktuálního kola ve hře. Díky tomu je na konci kola možné získat seznam subjektů, kteří v daném kole zadali poptávku respektive nabídku. Na straně aplikace při běhu tržního mechanismu, zpracovávajícího tržní nabídky a poptávky je pak kontrolováno, zda daný subjekt nabízel či poptával právě na trhu, jehož obchody se zpracovávají. Původní dotaz vracel přímo seznam subjektů nabízejících či poptávajících na daném trhu, po použití upraveného dotazu tuto informaci nemáme, úspora nákladů je ovšem značná a toto omezení lze realizovat na straně aplikace. Upravený dotaz:

```
SELECT
       id_subjektu
  FROM subjekty
 WHERE id_integracniho_celku = '2016ZS'
   AND kolo_posledni_aktivity = 20;
```

Plán vykonání dotazu před a po úpravě zobrazuje obrázek 5.9, vidíme snížení nákladů na cca 0,5%.



(a) (b)

Obrázek 5.9: Plán vykonání dotazu před (a) a po (b) provedení úpravy - zdroj: vlastní zpracování

5.2.2 Hromadné vkládání dat při uzavření kola

Zpracování konce kola je nákladná operace, dochází zde k vkládání velkého množství záznamů.

Byly realizovány všechny výše uvedené úpravy, které doporučuje MySQL dokumentace, uvedené v kapitole 4.5.3. Kolo bylo nejprve 10x uzavřeno při starém nastavení a poté 10x při novém nastavení. Následně byly časy běhu skriptu pro uzavření kola před a po úpravách zprůměrovány. Dle tohoto měření došlo k poklesu času běhu skriptu pro uzavření kola na cca 13 %.

Výsledný kód skriptu pro uzavření kola (`cron_dsge_file.php`):

```
mysql_query("SET autocommit=0;");
mysql_query("SET foreign_key_checks=0;");
... původní kód pro zpracování konce kola ...
mysql_query("COMMIT;");
mysql_query("SET autocommit=1;");
mysql_query("SET foreign_key_checks=1;");
```

5.2.3 Vkládání výchozích hodnot

V případě tabulky `aktivita_subjektu` byl využit přístup, kdy při vkládání dat SQL příkazem `INSERT` není nutné specifikovat výchozí hodnoty. Tabulka `aktivita_subjektu` eviduje datum a čas, kdy daný subjekt provedl aktivitu. Původní SQL příkaz pro evidenci aktivity vypadal např. takto:

```
INSERT INTO
aktivita_subjektu (id_subjektu, cislo_kola, cas_aktivity)
VALUES (153,5,CURRENT_TIMESTAMP);
```

Datový typ sloupce `cas_aktivity` je `TIMESTAMP`. Tento typ má speciální vlastnost: při vložení nového řádku bez specifikace hodnoty pro sloupec typu `TIMESTAMP` MySQL standardně nastaví první sloupec typu `TIMESTAMP` na aktuální čas. Pravidla chování `TIMESTAMP` jsou nicméně komplikovaná a v různých verzích MySQL se mění, je tedy potřeba ověřit si, že chování bude takové, jaké očekáváme. Bývá vhodné zkontrolovat výstup příkazu `SHOW CREATE TABLE` [28]. Ten je pro tabulku `aktivita_subjektu` následující:

```
CREATE TABLE ‘aktivita_subjektu‘ (
  ‘id_subjektu‘ smallint(5) unsigned NOT NULL,
  ‘cislo_kola‘ tinyint(3) unsigned NOT NULL,
  ‘cas_aktivity‘ timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  KEY ‘fk_Aktivita_subjektu_Subjekty1‘ (‘id_subjektu‘),
  CONSTRAINT ‘fk_Aktivita_subjektu_Subjekty1‘
  FOREIGN KEY (‘id_subjektu‘) REFERENCES ‘subjekty‘ (‘id_subjektu‘)
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci’;
```

U sloupce `cas_aktivity` vidíme definici `DEFAULT CURRENT_TIMESTAMP`, což značí, že při vložení záznamu skutečně dojde k automatickému vyplnění aktuální časové známky. Klauzule `CURRENT_TIMESTAMP` ve výše uvedeném dotazu je tedy zbytečná. Upravený SQL příkaz pak vypadá např. takto:

```
INSERT INTO
aktivita_subjektu (id_subjektu, cislo_kola)
VALUES (153,5);
```

5.3 Optimalizace na straně aplikace

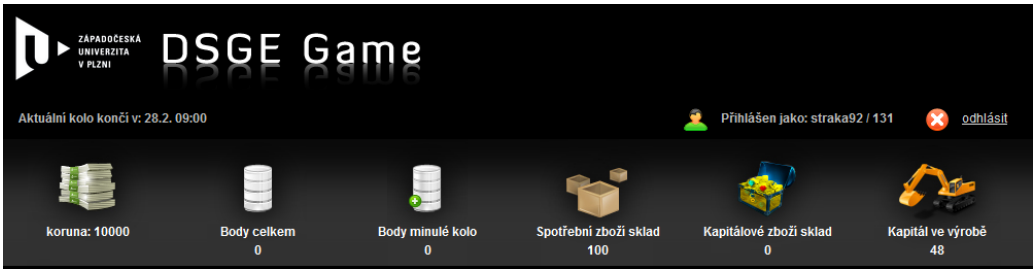
5.3.1 Cache hodnot neměnících se v průběhu kola

Přestože se hodnoty mnoha hráčských proměnných aktualizují jen na konci kola, tj. prakticky jednou za 2-3 dny, byly tyto hodnoty načítány při každém obnovení stránky. Jedná se o následující hodnoty:

- stav zásob hráče,
- právo přístupu na jednotlivé trhy,
- stav zásob hráče v návaznosti na přístup na daný trh.

Informace o stavu zásob hráče jsou zobrazeny v horizontálním pruhu v horní části stránky hry. Horizontální pruh zobrazuje následující hodnoty:

- množství držených peněz v jednotlivých měnách,
- stav bodů celkem,
- stav bodů v minulém kole,
- množství spotřebního zboží na skladě a ve výrobě,
- množství kapitálového zboží na skladě a ve výrobě.



Obrázek 5.10: Horizontální pruh - zdroj: vlastní zpracování

K načtení stavu zásob je potřeba zavolat čtyři SQL dotazy `SELECT SUM(hodnota)`. Pro každý druh subjektu je definováno právo poptávat (tabulka `prava_poptavat`) a nabízet (tabulka `prava_nabizet`) na jednotlivých trzích. Pokud má subjekt pro trh přiřazeno alespoň jedno z těchto práv, je daný trh zobrazen v menu a do formuláře trhu je mu umožněno vkládat tržní příkazy. V opačném případě uživatel možnost zadávat tržní příkazy nemá (v menu tento trh nemá a je ošetřena varianta, kdy uživatel do internetového prohlížeče zadá URL adresu skriptu zpracovávajícího tržní příkazy - přístup bude uživateli odepřen). K načtení práv zadávat tržní příkazy je potřeba volat 2 SQL dotazy (jeden do tabulky `prava_poptavat` a druhý do tabulky `prava_nabizet`).

Tato logika byla upravena tak, aby se hodnoty ukládaly do `SESSION`. Po každém obnovení stránky uživatele dochází k ověření, zda daná `SESSION` existuje. Pokud neexistuje, vytvoří se a naplní se příslušnými hodnotami. Kromě dat o stavu zásob je v `SESSION` uloženo také číslo kola, kdy byla naplněna. Toto číslo kola se ověřuje s aktuálním číslem kola - pokud je číslo kola

v `SESSION` nižší, tj. `SESSION` není aktuální, je `SESSION` znovu naplněna a číslo kola aktualizováno. Díky této úpravě bylo ušetřeno velké množství volaných dotazů, protože dotazy na stav zásob hráče se pro jednotlivé hráče volají maximálně jednou za kolo (za předpokladu, že nedojde k vymazání `SESSION` uživatelem).

Některé hodnoty byly načítány i v případě, že daný subjekt nemá právo přístupu na příslušný trh (např. je zbytečné dotazovat se na obligace, které subjekt vlastní, pokud tento subjekt na trh obligací nemá přístup). V těchto případech byly tedy přidány podmínky, omezující volání podobných SQL dotazů jen pro případ, kdy subjekt má na daný trh přístup.

6 Realizace akciového trhu a systému automatického hraní

6.1 Akciový trh

6.1.1 Návrh rozšíření

Pro rozšíření DSGEgame o akciový trh byly navrženy dvě varianty. Obě varianty mají společné následující skutečnosti. Obchodování na akciových trzích bude realizováno rozšířením již existujícího tržního mechanismu vycházejícího z Walrasova aukcionáře. Na jednom akciovém trhu se bude obchodovat s akciemi jednoho emitenta. Firmy budou reprezentovány subjekty vystupujícími ve hře jako výrobci. Ze zisku těchto subjektů budou vypláceny dividendy akcionářům - subjektům typu spotřebitel nebo podnikatel. Každý akcionář společnosti bude mít v jednotlivých kolech hry možnost hlasovat o tom, kolik procent ze zisku společnosti připadne na dividendy. Výsledné procento zisku se vypočte jako vážený průměr zadaných hodnot, kdy váha hlasu je určena poměrem počtu akcií držených akcionářem ku emitovanému počtu akcií. V obou variantách rozšíření budou emitovány kusové akcie.

Varianty se liší v tom, zda se obchoduje s homogenními akciemi (varianta s investičním fondem) nebo konkrétními akciemi jednotlivých výrobců (varianta s přímým obchodováním).

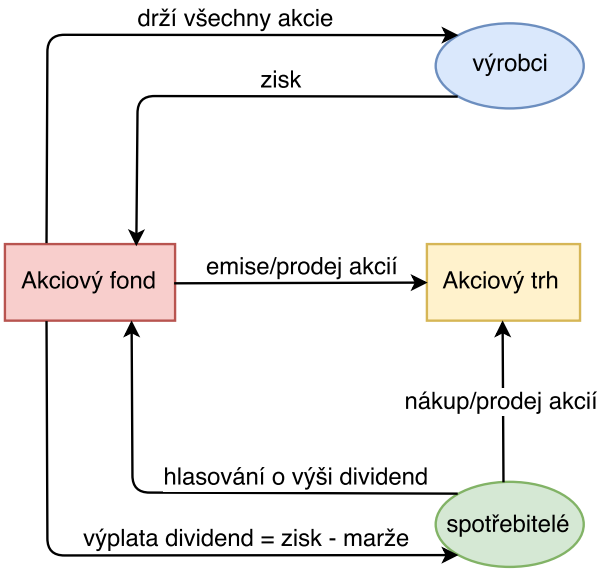
Varianta s investičním fondem

Nejprve popišme variantu s investičním fondem. Investiční fond bude ve hře vystupovat jako nový subjekt, ovládaný administrátorem nebo hráčem. Protože portfolio tohoto fondu budou tvořit pouze akcie, můžeme jej označovat jako akciový. Podle toho, zda ve hře vystupuje jeden (monopolní akciový fond) nebo více akciových fondů (komerční akciové fondy), můžeme tuto variantu ještě dále rozvětvit.

Monopolní akciový fond Akciový fond je v tomto případě vlastníkem všech akcií, tvořící základní kapitál všech firem v dané ekonomice. Vystupuje jako „stocks packer“, prostředník, který „zabalí“ akcie všech výrobců

do jedné a tím pádem jsou akcie homogenní. Ve hře tak nevystupují akcie různých společností, ale jen akcie fondu. Tím pádem existuje jediný akciový trh. Díky tomu je tato varianta pro hráče jednodušší a přehlednější a z tohoto důvodu bude pravděpodobně upřednostněna před variantou s přímým obchodováním. Smyslem není simulace ekonomiky, kdy všechny firmy vlastní stát, jde pouze o zjednodušení hry, zisk fondu neputuje do státního rozpočtu.

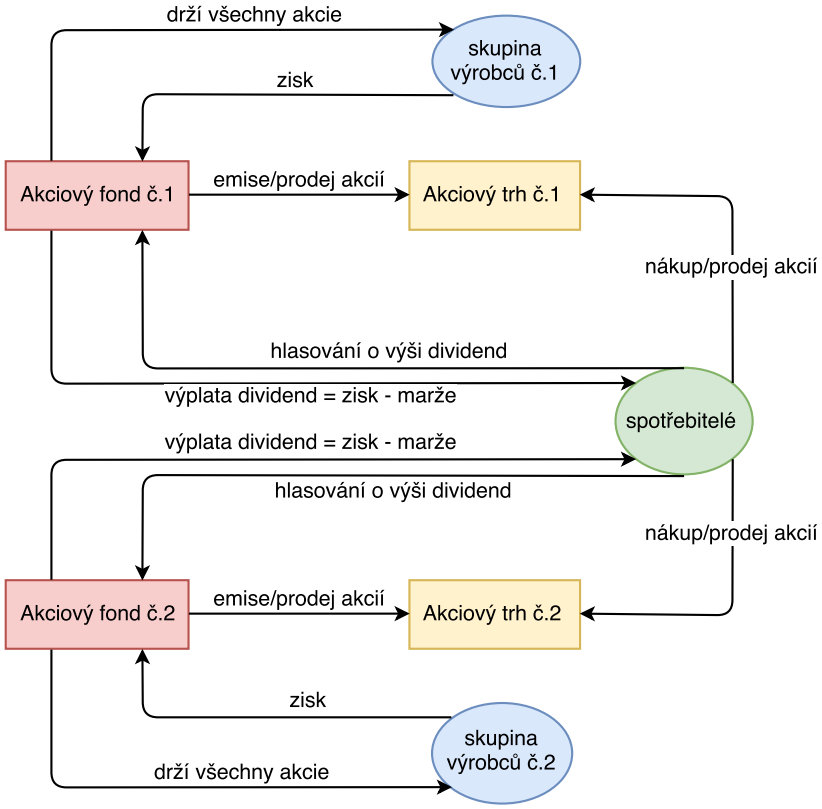
Subjekt hrající za akciový fond bude mít možnost nastavit výši marže fondu. Akcionáři odhlasují, kolik procent ze sumy zisku výrobců v daném kole připadne na dividendy, akciový fond tuto částku povýší o marži a povýšenou hodnotu vybere od výrobců. Akcionářům vyplatí tolik, kolik si odhlasovali, a marži si ponechá. Pro vyplácení dividend není nutné, aby byla suma zisku všech výrobců kladná. Dividendy jsou vypláceny ze zisku jednotlivých výrobců, pokud v kole nějaký vygenerují. Logiku fungování této varianty zobrazuje obrázek 6.1.



Obrázek 6.1: Schéma fungování varianty s monopolním investičním fondem - zdroj: vlastní zpracování

Komerční akciové fondy Fungování této varianty je analogické k variantě s monopolním akciovým fondem s tím rozdílem, že v tomto případě vystupuje v ekonomice více akciových fondů, které si navzájem konkurují. Každý fond je vlastníkem akcií části firem v ekonomice. Jednotlivé fondy se mohou diverzifikovat např. podle zaměření na akcie různé rizikovosti. Akcio-

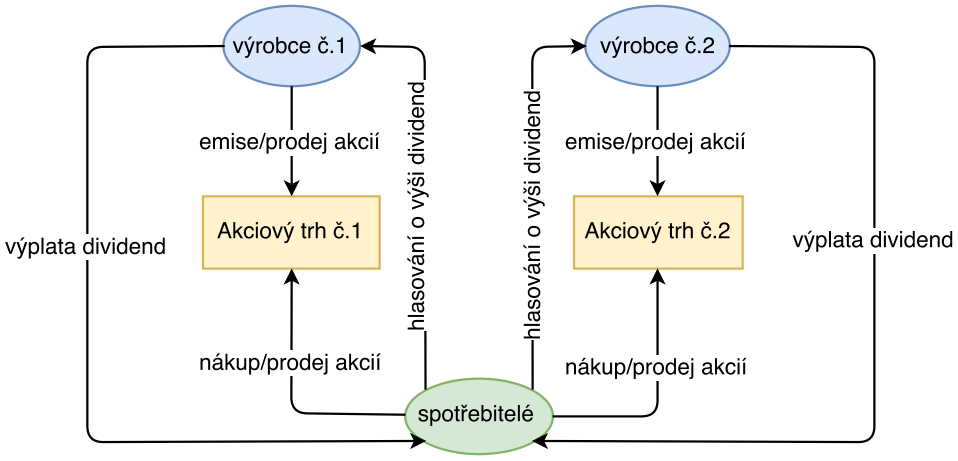
náři by si pak mohli vybírat fond podle svého vztahu k riziku. Pro každý fond by ve hře existoval jeden akciový trh. Logiku fungování této varianty zobrazuje obrázek 6.2. Tato varianta nebyla, vzhledem ke své složitosti pro hráče, realizována.



Obrázek 6.2: Schéma fungování varianty s dvěma komerčními investičními fondy - zdroj: vlastní zpracování

Varianta s přímým obchodováním

V této variantě se obchoduje s konkrétními akciemi jednotlivých výrobců. Ve hře existuje tolik akciových trhů, kolik ve hře vystupuje výrobců. Logiku fungování této varianty zobrazuje obrázek 6.3.



Obrázek 6.3: Schéma fungování varianty s přímým obchodováním - zdroj: vlastní zpracování

Emise akcií

Před startem hry nastaví administrátor hry výrobcům nebo akciovém fondu počáteční zásobu akcií. Dále je možné výrobcům a akciovému fondu nastavit právo dodatečné emise akcií. Toto právo dovoluje subjektům nabízet na akciovém trhu více akcií, než je jejich aktuální zásoba. V takovém případě je emitován počet akcií odpovídající rozdílu nabízeného množství a aktuální zásoby. Pokud subjekty toto právo nemají, použije se standardní omezení (mohou nabízet jen takové množství akcií, které aktuálně drží).

6.1.2 Popis realizace

Vytvoření akciového trhu

Aby ve hře mohly existovat akciové trhy, bylo potřeba provést následující kroky. Do tabulky `druhy_trhu`, která definuje, jaké typy trhů ve hře vystupují, byl vložen záznam *akciovy_trh*. Pro evidenci přiřazení emitenta k trhu byla tabulka `trhy` rozšířena o sloupec `id_emitenta` typu `SMALLINT UNSIGNED`, ve kterém bude ukládána hodnota `id_subjektu` z tabulky `subjekty` (tabulky byly svázány cizím klíčem). Administrační formulář pro správu trhů byl rozšířen o možnost zadávat `id_emitenta` k jednotlivým trhům.

Varianta s monopolním investičním fondem Pokud se jedná o variantu s monopolním investičním fondem, je dále potřeba vložit nový druh

subjektu *fond* do tabulky `druhy_subjektu` a vytvořit nový subjekt s tímto druhem. ID tohoto subjektu pak musí být přiřazeno do sloupce `id_emitenta` v tabulce `trhy`.

Varianta s přímým obchodováním U varianty s přímým obchodováním je nutné vytvořit několik subjektů druhu *vyrobce* (kolik, závisí na rozhodnutí administrátora hry), následně založit stejný počet akciových trhů a svázat trhy se subjekty prostřednictvím hodnoty ve sloupci `id_emitenta`.

Posledním krokem je přiřazení práv k přístupu na akciový trh jednotlivým druhům subjektů. Spotřebiteli je potřeba přiřadit právo poptávat i nabízet (záznam v tabulkách `prava_poptavat` a `prava_nabizet`). Stejná práva je nutné přiřadit druhu subjektu, jenž bude emitovat akce (fondu, pokud se jedná o variantu s akciovým fondem nebo výrobci, pokud se jedná o variantu s přímým obchodováním).

Po realizaci těchto kroků se akciový trh (případně akciové trhy) objevuje v menu DSGEgame a je možné na něm zadávat nabídky a poptávky stejným způsobem jako na ostatních trzích.

Toky akcií

Pro evidenci toků akcií (transakcí s akciemi) byla založena tabulka `toky_akcii` s těmito sloupci:

- `cislo_kola`: číslo kola, ve kterém byl tok realizován.
- `id_subjektu`: ID subjektu, jemuž je tok přiřazen.
- `id_emitenta`: ID emitenta akcie. Určuje, akcie jakého emitenta byla předmětem toku.
- `id_puvodu`: ID původu toku odpovídající záznamu v tabulce `puvody_toku`.
- `hodnota_toku`: hodnota toku (počet akcií, který byl předmětem toku).

Každý tok vkládaný do této tabulky musí odpovídat definovanému původu toku. Číselník `druhy_puvodu_toku` ukládá, jaké původy toků mohou existovat v rámci celé hry. Tabulka `puvody_toku` pak eviduje, jaké původy toků jsou přípustné pro jednotlivé trhy nebo ekonomiky.

Číselník `druhy_puvodu_toku` byl rozšířen o záznamy:

- *emise*,
- *marze_fondu*,

- *vyplata_dividendy*,
- *prijem_dividendy*.

Následně byly do tabulky `puvody_toku` vloženy tyto původy toků definované pro akciový trh. Kromě těchto záznamů byly vloženy ještě původy toků *nakup* a *prodej*.

Ukládání toků akcií do tabulky `toky_akcii` obstarává nově vytvořená třída `Tok_akcie`, vzniklá oděděním od třídy `Tok`. Vložení záznamu toku do tabulky zajišťuje její metoda `zapis_tok_do_databaze()` .

Emise a nabídka akcií

Právo výrobce či fondu emitovat akcie je evidováno pro jednotlivé ekonomiky. Pro ukládání těchto práv byly vytvořeny dva nové sloupce v tabulce `ekonomiky`: `pravo_emise_vyrobce` a `pravo_emise_fond`, které budou nabývat hodnot `1` (právo emise je přiřazeno) a `0` (není přiřazeno). Oba sloupce jsou typu `TINYINT UNSIGNED`. Pro načítání těchto hodnot v aplikaci byla třída `Informace_o_ekonomikach` rozšířena o metody `get_pravo_emise_fondu_v_ekonomice()` a `get_pravo_emise_vyrobce_v_ekonomice()`.

Platnost zadané nabídky na trhu ověřuje metoda `over_platnost_nabidky()` třídy `Subjekt`. Ta byla upravena tak, aby subjektu dovolovala nabízet libovolné množství akcií, pokud má přiřazeno právo emise akcií (bude-li nabízet více akcií, než má v držení, akcie se doemitují) a jen takové množství akcií, které má subjekt v držení, pokud toto právo nemá. Toho bylo docíleno následovně. Právo subjektu emitovat akcie je načteno metodou `get_pravo_emise_fondu_v_ekonomice()` nebo `get_pravo_emise_vyrobce_v_ekonomice()` (v závislosti na druhu subjektu). Pokud má subjekt právo přiřazeno, vrací metoda `true`. V opačném případě metoda ověřuje, zda je zásoba akcií prodejce vyšší než zadávané nabízené množství. Pokud ano, vrací metoda `true`, jinak `false`. Pro načtení zásoby akcií prodejce slouží metoda `get_pocet_akcii()` nově vytvořené třídy `Informace_o_akciovem_trhu`. Tato třída vrací různé informace souvisejících s akciovým trhem. Parametry metody `get_pocet_akcii()` jsou `id_subjektu` a `id_emitenta`. Pomocí SQL dotazu na tabulky `toky_akcii` zjistí metoda, kolik akcií daného emitenta subjekt drží a vrátí výsledek.

K nabídce akcií je možné využít formulář akciového trhu nebo v případě akciového fondu, který bude ovládán administrátorem hry, administrační formulář *Nahrání příkazu ze souboru*. Tento formulář umožňuje uložení velkého množství nabídek či poptávek najednou nahráním CSV souboru. Je možné jej využít pro zadání nabídky akciového fondu na začátku hry po celou dobu hry.

Obchodování na akciovém trhu

Pro obchodování na akciových trzích byl rozšířen již existující mechanismus. Obchodování na trzích obstarává třída `Spravce_trhu`. Tato třída reaguje na událost `KONEC_KOLA`, vysílanou objektem `Casovac` a zajišťuje výpočet tržní ceny a provedení transakcí na všech existujících trzích. V metodě `zobchoduj_polozky_na_vsech_trzich()` jsou načteny existující trhy a pro každý z nich je založena instance třídy `Trh`. Následně je pro každou instanci volána metoda `zobchoduj_polozky_na_trhu()`.

Úprava třídy Trh Třída byla rozšířena o atribut `$id_emitenta`[1]. V konstruktoru třídy je volána metoda `zjisti_udaje_o_trhu()`. Ta byla upravena tak, aby v případě, že se jedná o akciový trh, uložila do nového atributu třídy `id_emitenta` ID subjektu, jenž nabízí akcie na daném akciovém trhu (ID subjektu se získá z hodnoty sloupce `id_emitenta` v tabulce `trhy`). Metoda `zobchoduj_polozky_na_trhu()` načítá individuální nabídky a poptávky při vypočtené tržní ceně. Ty následně v cyklu prochází a generuje příslušné toky komodit a peněz (kladný tok komodity kupujícímu, záporný tok komodity prodávajícímu; kladný tok peněz prodávajícímu a záporný tok peněz kupujícímu). Metoda byla upravena následovně. Pokud se jedná o akciový trh, načítá se:

- Právo emise prodejce akcií. Podobně jako při ověřování platnosti nabídky subjektu je potřeba načíst informaci, zda daný prodejce má právo emitovat akcie. Ověřování platnosti nabídky subjektu je realizováno ve třídě `Subjekt`, kde jsou k dispozici informace o tom, jaké ekonomiky je tento subjekt rezidentem a jaký je jeho druh (v atributech `$id_ekonomiky` a `$id_druhu_subjektu`). Ve třídě `Trh`, v cyklu procházejícím individuální nabídky jednotlivých subjektů tyto informace k dispozici nejsou a musí se získat jinak. Pro načtení práva emise akcií slouží metoda `get_pravo_prodejce_emitovat_akcie()` třídy `Informace_o_akciovem_trhu`. Metoda přijme na vstupu ID subjektu. Na jeho základě zjistí druh daného subjektu. Pokud se nejedná o výrobce nebo akciový fond, vrací metoda `false` (jiné subjekty než akciový fond a výrobce nemají možnost emitovat akcie). V opačném případě metoda zjišťuje, jaké ekonomiky je subjekt v aktuálním kole rezidentem a následně v závislosti na druhu daného subjektu volá jednu z metod - `get_pravo_emise_fondu_v_ekonomice()` nebo `get_pravo_emise_vyrobce_v_ekonomice()` třídy `Informace_o_ekonomikach`. Metody načtou hodnotu sloupce `pravo_emise_fondu` nebo `pravo_emise_vyrobce`

[1]Znak „$“ uvozuje v PHP proměnnou, toto značení bude dále v textu používáno.

z tabulky `ekonomiky` pro danou ekonomiku a tuto hodnotu vrátí. Výsledek volání metody je navrácen metodou `get_pravo_prodejce_emitovat_akcie()`.

- Počet akcií prodejce. Počet akcií prodejce je načten prostřednictvím metody `get_pocet_akcii()` třídy `Informace_o_akciovem_trhu`.

Při procházení individuálních poptávek v metodě `zobchoduj_polozky_na_trhu()` je generován tok akcie s původem *nakup* založením instance třídy `Tok_akcie`. Během procházení individuálních nabídek metoda zjišťuje, zda nabízené množství není vyšší než počet akcií daného prodejce. Pokud ne, je generován jen tok akcie původu *prodej* se záporným znaménkem hodnoty toku. Pokud ano a prodejce má právo emitovat akcie, je navíc generován tok akcie původu *emise* s kladným znaménkem hodnoty toku.

Hlasování o výši dividend a jejich vyplácení

Správu hlasů o výši dividend a jejich vyplácení obstarává nově vytvořená třída `Spravce_dividend`.

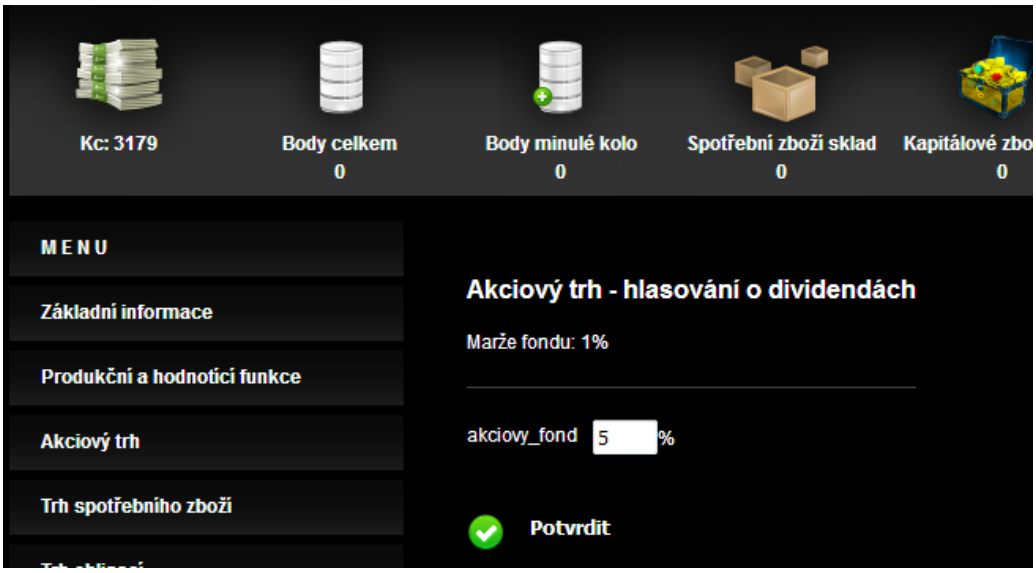
Hlasování o výši dividend Pro uchovávání záznamů o hlasování jednotlivých akcionářů byla vytvořena tabulka `volba_dividendy` s těmito sloupci:

- `cislo_kola`: číslo kola, ve kterém byl hlas zaevidován.
- `id_subjektu`: ID subjektu, jehož hlas je evidován.
- `id_emitenta`: ID emitenta, o rozdělení jehož zisku se hlasuje.
- `procento`: procento zisku emitenta, které má připadnout na dividendy.

Na začátku hry je tabulka naplněna hlasy s nastavenou hodnotou pro sloupec `procento`. Naplnění realizuje metoda `vloz_pocatecni_hlasy_ve_volbe_dividend()`, která na vstupu přijímá ID ekonomiky, číslo kola, ID emitenta a procento. Metoda načte všechny subjekty, které jsou rezidenty dané ekonomiky v daném kole a vygeneruje SQL příkaz zapisující hlasy pro jednotlivé subjekty a jednotlivé emitenty akcií (každému subjektu je vygenerován záznam pro každého emitenta). Výchozí hodnota hlasu - procenta zisku emitenta, které má připadnout na dividendy je uložena v konstantě `DEFAULTNI_HLAS_FOND` respektive `DEFAULTNI_HLAS_VYROBCE` ve třídě `Spravce_dividend`.

V průběhu kola mají spotřebitelé možnost měnit svůj hlas prostřednictvím nově vytvořeného formuláře. Pro jeho vytvoření byla rozšířena třída `Spravce_GUI` a oděděním od třídy `Formular` vytvořena třída `Formular_volba_dividend`. Ve třídě `Spravce_GUI` byla vytvořena metoda `vloz_obsah_stranky_volba_dividend()`.

Ta načte pole hlasů příslušného subjektu pro aktuální kolo (v podobě klíč - hodnota: `id_emitenta - hlas`) pomocí metody `get_hlasy_ve_volbe_dividend()` třídy `Informace_o_akciovem_trhu`. Následně založí instanci třídy `Formular_volba_dividend`, které předá načtené pole hlasů. Třída `Formular_volba_dividend` generuje HTML formulář pro editaci hlasu akcionáře. Tento formulář má tolik řádků, kolik ve hře vystupuje akciových trhů, akcionář tak zadává, kolik procent ze zisku všech emitentů má připadnout na dividendy. Pro zpracování dat z tohoto formuláře byla třída `Spravce_dat_z_formularu` rozšířena o metodu `zpracovani_formulare_volby_dividend()`. Ta přijímá pole hodnot z formuláře, ověřuje, že jsou data korektně vyplněna a pokud ano, ukládá je do tabulky `volba_dividend`. Obrázek 6.4 zobrazuje podobu formuláře pro hlasování o výši dividend pro případ, že ve hře vystupuje jediný emitent akcií (v tomto případě akciový fond).



Obrázek 6.4: Formulář hlasování o výši dividend - zdroj: vlastní zpracování

Formulář načítá hodnotu hlasu akcionáře pro aktuální kolo. Na konci kola jsou hlasy spotřebitelů překopírovány pro další kolo: všechny záznamy v tabulce `volba_dividendy` jsou vloženy znovu s hodnotou ve sloupci `cislo_kola` povýšenou o 1.

Marže fondu V případě varianty s akciovým fondem má administrátor hry možnost definovat pro každé kolo výši jeho marže. K tomu slouží tabulka `marze_fondu` s těmito sloupci:

- `cislo_kola`: číslo kola, pro které je výše marže fondu nastavena.

- `id_fondu`: ID subjektu typu fond, pro který je výše marže nastavena.
- `marze`: výše marže fondu.

Marže fondu je zohledněna v hlasování akcionářů o výši dividend. Akcionáři odhlasují, kolik procent ze sumy zisku výrobců v daném kole připadne na dividendy, akciový fond tuto částku povýší o nastavenou výši marže a povýšenou hodnotu vybere od výrobců. Akcionářům vyplatí tolik, kolik si odhlasovali a zbytek si ponechá. Je nutné ošetřit, aby vážený průměr hlasů akcionářů povýšený o marži fondu nepřesáhl 100 %, protože více než 100 % zisku výrobců jim fond odebrat nemůže. Toho je docíleno následovně. Při editaci formuláře volby dividend se kontroluje, že zadaná hodnota + marže fondu není vyšší než 100 %. Stejná kontrola je aplikována při kopírování hlasů pro další kolo (pro případ, že by se výše marže fondu pro následující kolo lišila od té pro aktuální kolo). Není-li této kontrolní podmínce vyhověno, je v obou případech zadána nejvyšší možná hodnota, která nepřesahuje 100 %.

Pro správu nastavených výší marže fondu byl vytvořen administrační formulář *marze_fondu*. Oděděním od třídy `Formular_administratora` byla vytvořena třída `Formular_marze_fondu`. Tato třída generuje HTML formulář pro vkládání/editaci záznamů v tabulce `marze_fondu` a zobrazuje aktuálně uložené hodnoty v této tabulce. Pro zobrazení těchto hodnot byla třída `Informace_o_ekonomikach_xhtml_vystupy` rozšířena o metodu `generuj_tabulku_marze_fondu()`.

Vyplácení dividend Výpočet a vyplácení dividend se realizuje postupně po jednotlivých ekonomikách. Je načten zisk výrobců v dané ekonomice a jeho část připadající na dividendy rozdělena mezi akcionáře výrobce. Výpočet a vyplácení dividend obstarává třída `Spravce_dividend`. Ta reaguje na událost `KONEC_KOLA` vyvolanou třídou `Casovac`. V konstruktoru třída přijímá ID ekonomiky a číslo aktuálního kola. Vykonává následující kroky:

- Ověřuje, zda se jedná o variantu s akciovým fondem nebo s přímým obchodováním. K tomu slouží metoda `zjisti_variantu_akcioveho_trhu` třídy `Informace_o_akciovem_trhu`. Pokud se jedná o variantu s akciovým fondem, je ID subjektu ovládající akciový fond, přiřazeno do atributu třídy `$id_fondu`.
- Metodou `nacti_info_o_ekonomice()` načítá informace o ekonomice - ID měny, používané v ekonomice a výši marži fondu, pokud se jedná o variantu s akciovým fondem.
- Metodou `nacti_id_puvodu_toku_zisk()` načítá do atributu `$id_puvodu_zisk` ID původu toků, které budou započteny jako zisk výrobců v ekonomice.

- Načítá zisky firem pomocí metody `vypocti_zisky_firem()`. Ta SQL dotazem načítá sumu toků, jejichž původy odpovídají původům v poli `$id_puvodu_toku_zisk`, jejichž subjekt je druhu výrobce a jejichž subjekt je v aktuálním kole rezidentem dané ekonomiky. Výsledné zisky metoda ukládá do pole `$pole_zisku_firem`.

- Metodou `zjisti_procenta_na_dividendy()` generuje dvourozměrné pole hlasů jednotlivých akcionářů, kolik procent zisků jednotlivých firem má připadnout na dividendy. Výsledné pole má podobu: `$pole_procenta_na_div[id_akcionare][id_emitenta] = hlas`.

- Metodou `zjisti_vazeny_prumer()` zjišťuje vážený průměr hlasů akcionářů, kolik procent zisku firem má připadnout na dividendy a zároveň pole poměrů akcií jednotlivých akcionářů k emitovanému počtu akcií firmy. Metoda nejprve SQL dotazem načte sumu počtu vydaných akcií jednotlivých firem. Následně firmy prochází v cyklu a v každém průchodu cyklu iteruje všechny její akcionáře. V tomto cyklu zjišťuje poměr akcií jednotlivých akcionářů k emitovanému počtu akcií firmy. Poměr ukládá do pole podoby: `$pole_pomeru_poctu[id_emitenta][id_akcionare] = poměr akcií`. Toto pole je později využito v metodě `pripis_dividendy()`. Hlas akcionáře (získaný z pole `$pole_procenta_na_div`) je násoben poměrem akcií akcionáře k emitovanému počtu akcií firmy a ukládán do pole vážených průměrů pro jednotlivé firmy. Pole má následující podobu: `$pole_vazeny_prumer[id_emitenta] = vážený průměr`.

- Vypočítává výši dividend prostřednictvím metody `vypocti_dividendy()`. Ta iteruje `$pole_zisku_firem`. Pokud se jedná o variantu s přímým obchodováním, násobí metoda hodnotu zisku firmy váženým průměrem hlasů (získaným z pole `$pole_vazeny_prumer`). Pokud se jedná o variantu s akciovým fondem, násobí metoda hodnotu zisku fondu váženým průměrem hlasů povýšeným o marži fondu. Výsledkem volání metody je pole říkající, jaká částka peněz připadne na dividendy u jednotlivých firem v této podobě: `$pole_dividendy[id_emitenta] = částka`.

- Odepisuje vypočtené dividendy firmám a připisuje je akcionářům prostřednictvím metody `pripis_dividendy()`. Metoda nejprve iteruje přes pole dividend `$pole_dividendy` a inicializací třídy `Tok_penez` generuje tok peněz odečítající firmě částku připadající na dividendy. Pokud se jedná o variantu s akciovým fondem, generuje metoda navíc tok peněz s marží fondu. Následně metoda iteruje `$pole_pomeru_poctu` a každému akcionáři připisuje dividendu vypočtenou jako poměr akcií akcionáře ku počtu akcií firmy vynásobený částkou připadající na dividendy.

Vývoj zisku fondu

Jak je popsáno v kapitole 6.1.1, dividendy nejsou akcionářům vypláceny v závislosti na sumě zisku všech výrobců, ale jsou vypláceny ze zisku výrobců, kteří jej dosáhnou. Pro evidenci výše zisku fondu slouží nově vytvořená tabulka `vyvoj_zisku_fondu` s těmito sloupci:

- `cislo_kola`: číslo kola, ve kterém bylo tohoto zisku dosaženo,
- `id_fondu`: ID subjektu druhu fond, pro který je zisk evidován,
- `zisk`: zisk fondu.

Díky existenci této tabulky je později možné hodnoty zisku fondu snadno načítat. Jestliže by tabulka neexistovala, bylo by pro zobrazení těchto hodnot ve hře potřeba použít několik složitějších SQL dotazů (načíst zisk výrobců v jednotlivých kolech do pole, toto pole zisků iterovat, v každé iteraci vybírat jen zisky vyšší než 0 a ty přičítat do nějaké proměnné).

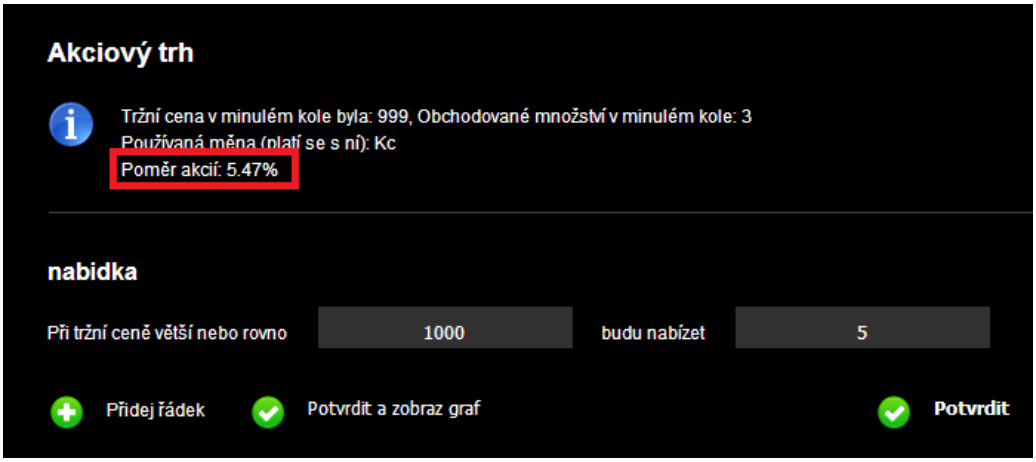
Pokud se jedná o variantu s akciovým fondem, je zisk fondu v daném kole ukládán do tabulky `vyvoj_zisku_fondu` metodou `zapis_zisk_fondu()` třídy `Spravce_dividend`.

Zobrazení údajů ve hře

Horizontální pruh a akciový trh v horizontálním pruhu přibyla informace o tom, jaký počet akcií jednotlivých emitentů hráč drží. Pro načtení počtu akcií akcionáře byla do třídy `Subjekt` přidána metoda `zjisti_mnozstvi_akcii()`, která ukládá pole počtu držených akcií jednotlivých emitentů do pole této podoby: `$drzene_akcie[$id_emitenta] = počet akcií`. Toto pole je stejně jako ostatní hodnoty v horizontálním pruhu ukládáno do `SESSION` a obnovuje se po uzavření kola (nebo vymazání `SESSION` hráčem). Pro zobrazení počtu akcií byla upravena metoda `generuj_horizontalni_pruh()` třídy `Subjekt`. HTML kód, který metoda generuje byl rozšířen o ikonu a hodnoty načtené z pole `$drzene_akcie`. Obrázek 6.5 zobrazuje podobu horizontálního pruhu v případě, že ve hře vystupuje jediný emitent akcií (v tomto případě akciový fond).

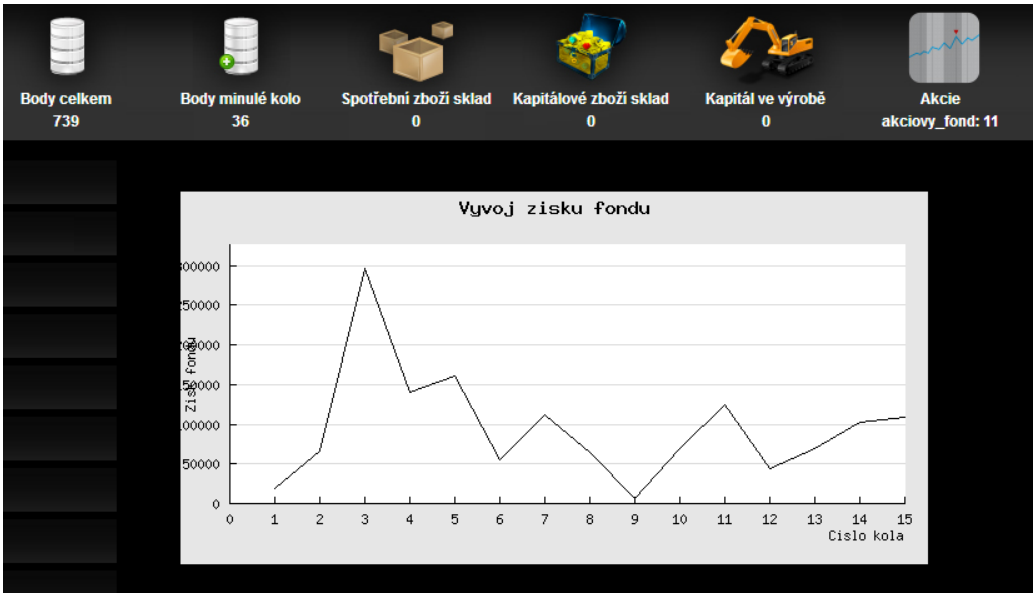
Obrázek 6.5: Horizontální pruh rozšířený o počet akcií v držení akcionáře - zdroj: vlastní zpracování

Na akciovém trhu přibyla informace o poměru akcií akcionáře k emitovanému počtu akcií firmy, jak ukazuje obrázek 6.6.



Obrázek 6.6: Akciový trh rozšířený o informaci o poměru akcií - zdroj: vlastní zpracování

Grafy zisku V menu hry přibyla položka *Akciový trh - graf vývoje zisku*. V závislosti na nastavení hry zobrazuje tato stránka graf vývoje zisku fondu nebo graf vývoje zisku výrobců.



Obrázek 6.7: Graf vývoje zisku fondu - zdroj: vlastní zpracování

Pro založení této stránky byla do třídy `Spravce_GUI` vložena nová metoda `vloz_obsah_stranky_akciovy_trh_grafy()`. Metoda generuje HTML kód pro zobrazení obrázku vygenerovaného pomocí třídy `JPGraph`. Této třídě jsou předány

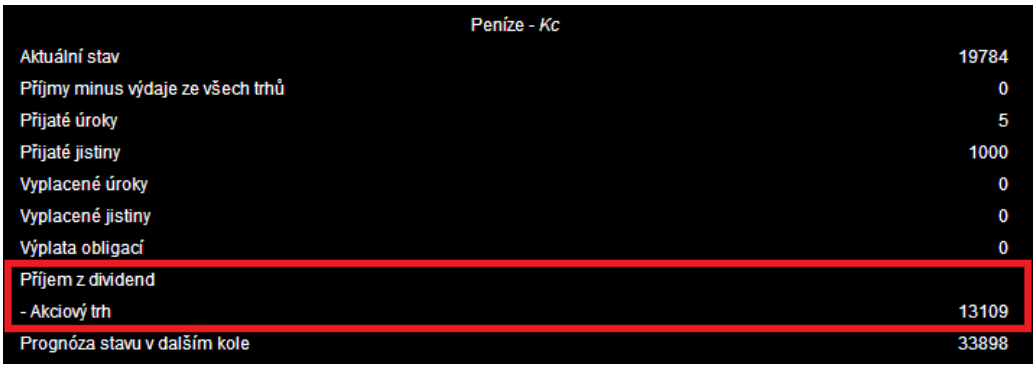
hodnoty zisku fondu nebo výrobců v jednotlivých kolech získané SQL dotazem. Obrázek 6.7 zobrazuje vzorovou podobu grafu zisku fondu.

Generovaný XLS soubor Hráči mají možnost vygenerovat XLS soubor s údaji o vývoji hry. Tento soubor byl rozšířen o informaci o vývoji zisku akciového fondu (pro variantu s akciovým fondem) nebo výrobců (pro variantu s přímým obchodováním). Pro realizaci tohoto rozšíření byla upravena třída `XLS_vyhodnoceni_kola`. Její metoda `vygeneruj_soubor()` byla rozšířena o SQL dotaz načítající zisk fondu nebo výrobců. Zisk fondu je zobrazen na jednom novém listě XLS souboru, zisk výrobců pak na jednotlivých listech XLS souboru (jeden list pro jednoho výrobce).

Prognóza výsledku pro zadané hodnoty Tato stránka vypočítává prognózu výsledků hráče (prognóza prodaného a nakoupeného množství na trzích, prognóza toků peněz a získaných bodů) na základě zadaných hodnot cen na jednotlivých trzích. Pokud hráč nezadá žádné hodnoty, jsou načteny hodnoty z minulého kola. Prognóza byla rozšířena o odhadovaný příjem z dividend. Metoda `vloz_obsah_stranky_prognoza()` byla upravena následovně. Iteruje přes pole akciových trhů získaných pomocí metody `get_akciove_trhy()` třídy `Informace_o_akciovem_trhu`. V každé iteraci pomocí metody `get_pocet_akcii()` načítá, kolik akcií, které se na daném trhu obchodují, má subjekt v držení. Dále načítá zadanou individuální nabídku a poptávku subjektu po akciích obchodovaných na daném akciovém trhu. Díky tomu je možné prognózovat, jaké množství akcií bude subjekt držet v příštím kole (akcie v držení + akcie poptávané - akcie nabízené). Dále metoda načte hodnotu ukazatele *dividenda na akcii* pomocí tohoto vzorce:

$$\text{dividenda na akcii} = \frac{\text{suma dividend vyplacených v minulém kole}}{\text{suma akcií v držení akcionářů v minulém kole}} \quad (6.1)$$

Díky tomu získá průměrnou výši dividendy vyplácenou na jednu akcii. Vynásobením dividendy na akcii prognózovaným množstvím akcií v držení subjektu získává metoda odhad příjmu z dividend. Prognózované hodnoty pro jednotlivé trhy jsou zobrazeny pod řádkem *Příjem z dividend*. Obrázek 6.8 zobrazuje podobu stránky *Prognóza* v případě, že ve hře vystupuje jediný emitent akcií (v tomto případě akciový fond).



Obrázek 6.8: Stránka *Prognóza* rozšířená o prognózu příjmu z dividend - zdroj: vlastní zpracování

6.2 Systém automatického hraní subjektů

6.2.1 Automatické hraní spotřebitelů, podnikatelů a výrobců

Návrh řešení

Na konci každého kola bude zjištěn počet aktivních a neaktivních hráčů v daném kole. Pro každý trh bude načteno pole s klíčem *cena* a hodnotou *suma množství poptávaného (respektive nabízeného) při této ceně*. Množství pro jednotlivé ceny budou vydělena počtem aktivních hráčů - tím získáme průměrné množství poptávané (respektive nabízené) jedním aktivním hráčem při dané ceně. Tato množství budou pak vynásobena počtem neaktivních hráčů a budou zadány poptávky a nabídky s těmito parametry. Za neaktivní hráče tak budou na všech trzích zadány nabídky a poptávky vygenerované na základě hry aktivních hráčů.

Tyto vygenerované nabídky a poptávky budou do databáze vloženy pod subjektem s nově založeným druhem `automat_spotrebitel_podnikatel_vyrobce`. V úvahu přichází i varianta, kdy jsou nabídky a poptávky generovány přímo jednotlivým nehrajícím hráčům (tzn., že automat by hrál přímo za nehrající hráče). Tato varianta byla ale zavržena z následujících důvodů:

- Hráči za aktivitu ve hře dostávají body k zápočtu. Pokud by automat hrál přímo za neaktivní hráče, bylo by problematické rozlišit, jakou část hry odehrál hráč a jakou část automat,
- Jestliže by automat hrál přímo za konkrétní subjekt, mohl by subjektu způsobit určité potíže (např. stav financí subjektu by se mohl dostat do záporných hodnot, ze kterých by se subjekt později nemohl dostat).

Popis realizace

Obsluha automatického hraní Pro obsluhu automatického hraní spotřebitelů, podnikatelů a výrobců byla vytvořena třída `Automaticke_hrani_subjekty`. Třída reaguje na událost `KONEC_KOLA` vyvolanou třídou `Casovac`. Generování nabídek a poptávek je realizováno postupně pro jednotlivé integrační celky. Jsou načteny nabídky a poptávky aktivních hráčů na trzích v daném integračním celku a na jejich základě generovány nabídky a poptávky na trzích v tomto integračním celku. V konstruktoru přijímá třída ID integračního celku a číslo aktuálního kola. Provádí následující posloupnost kroků:

- Pomocí metody `over_aktivaci_auto_hrani()` ověřuje, že je automatické hraní spotřebitelů, podnikatelů a výrobců aktivované. Aktivaci/deaktivaci automatického hraní těchto subjektů je možné provést definicí hodnoty v nově založeném sloupci `auto_hrani_subjekty` v tabulce `integracni_celky`. Hodnota `1` znamená aktivaci, `0` deaktivaci.

- Pomocí metody `get_auto_subjekt()` načte ID subjektu, pod jehož ID budou generované nabídky a poptávky vkládány do databáze. Pro vytvoření tohoto subjektu byl číselník `druhy_subjektu` rozšířen o záznam *automat_spotrebitel_podnikatel_vyrobce* a následně byl založen subjekt s tímto druhem.

- Zjisťuje počet neaktivních hráčů v daném kole prostřednictvím metody `get_pocet_neaktivnich_hracu_v_kole()`. Ta načítá počet záznamů v tabulce `subjekty`, u nichž je hodnota sloupce `kolo_posledni_aktivity` nižší než číslo aktuálního kola. Počet je uložen do atributu `$pocet_neaktivnich_hracu`.

- Zjišťuje počet všech hráčů (aktivních i neaktivních) v daném kole prostřednictvím metody `get_pocet_vsech_hracu()`. Ta SQL dotazem načítá počet hráčů v daném integračním celku. Počet aktivních hráčů se získá jako rozdíl počtu všech hráčů a počtu neaktivních hráčů a je uložen do atributu třídy `$pocet_aktivnich_hracu`.

- Metodou `nacti_pole_trhu()` načítá pole trhů v daném integračním celku do atributu třídy `$pole_trhu`.

- Pomocí metody `nacti_poptavky_na_vsech_trzich()` načítá pole poptávek na všech trzích. Metoda iteruje `$pole_trhu` a generuje trojrozměrné pole této podoby: `$pole_poptavek[$id_trhu][$id_meny][$mezni_cena] = poptávané množství`.

- Analogickým způsobem generuje metodou `nacti_nabidky_na_vsech_trzich()` pole nabídek této podoby: `$pole_nabidek[$id_trhu][$id_meny][$mezni_cena] = nabízené množství`.

- Metodou `generuj_poptavky()` prochází pole poptávek. Jednotlivá poptávaná množství dělí počtem aktivních hráčů (který získá z atributu `$pocet_aktivnich_hracu`) a následně násobí počtem neaktivních hráčů (získaným z atributu `$pocet_neaktivnich_hracu`). Během průchodu pole poptávek skládá metoda řetězec SQL příkazu pro vložení všech záznamů. Tento příkaz je zavolán po průchodu cyklem - všechny záznamy jsou vloženy najednou.

- Analogickým způsobem generuje nabídky metodou `generuj_nabidky()`.

6.2.2 Automatické hraní vlády

Návrh řešení

Pro automatické hraní vlády byla navržena následující formulace rovnice vycházející z rovnice č. 3.9

$$\text{saldo rozpočtu (deficit rozpočtu)} = \text{příjmy} - \text{výdaje}. \tag{6.2}$$

Rovnici je možné rozepsat následovně:

$$\text{saldo (deficit)} = \text{výběr daní} + \text{emise obligací}^2 - \text{nákup spotř. zboží}. \tag{6.3}$$

Administrátor hry má možnost ovlivňovat, zda bude vláda hospodařit s přebytkem nebo deficitem rozpočtu. Zadává tyto parametry rovnice:

- *Saldo*. Jestliže zadá kladnou hodnotu, bude vláda hospodařit s přebytkem, pokud zápornou hodnotu, bude hospodařit s deficitem. Nula znamená vyrovnaný rozpočet.

- *Nákup spotřebního zboží*. Zadává množství k nákupu. V rovnici je pak množství násobeno cenou spotřebního zboží v minulém kole.

Parametr *výběr daní* je vypočítán. Parametr *emise obligací* je neznámou rovnice. Rovnici je tedy možné přepsat do této podoby:

$$\text{emise obligací} = \text{saldo (deficit)} - \text{výběr daní} + \text{nákup spotř. zboží} \tag{6.4}$$

[2] *„Obligace je certifikát, přinášející úrok. Vláda nebo soukromá firma, která jej vydala, se jím zavazuje splatit k určitému datu dlužnou částku (jistinu) plus úroky.“* [27, str. 743]

Pokud vyjde emise obligací (dluhopisů) kladná, což znamená, že vláda hospodaří se ztrátou, jsou emitovány obligace pro pokrytí této ztráty. Jedná se o tzv. bezkupónové obligace. Cena, od které vláda obligace nabízí, se spočítá podle následujícího vzorce:

$$\text{min. cena obligace} = \frac{\text{hodnota obligace}}{(1 + \text{max. úroková míra})^{\text{počet kol do splatnosti}}} \tag{6.5}$$

Množství nabízených obligací se vypočte jako

$$\text{množství obligací} = \frac{\text{emise obligací}}{\text{min. cena obligace}} \tag{6.6}$$

Parametry *hodnota obligace* a *počet kol do splatnosti* zadává administrátor hry při jejím nastavování. Parametr *maximální úroková míra* říká, jakou maximální úrokovou míru dluhopisů je vláda ochotná akceptovat a administrátor hry jej bude zadávat pro jednotlivá kola, stejně jako parametry rovnice *saldo* a *nákup spotřebního zboží*.

Vláda tedy bude automaticky zadávat poptávku po spotřebním zboží (dle parametru *nákup spotřebního zboží*, zadaného administrátorem hry) a nabídku obligací (dle výsledku rovnice) - tím provádí expanzi[3] respektive restrikci[4].

Popis realizace

Ukládání parametrů automatického hraní vlády K ukládání parametrů automatického hraní vlády byla založena tabulka `auto_hrani_vlada_parametry` s těmito sloupci:

- `cislo_kola`: číslo kola, pro které jsou parametry nastaveny.
- `id_ekonomiky`: ID ekonomiky, pro kterou jsou parametry nastaveny.
- `saldo`: nastavené saldo (nebo deficit) rozpočtu.
- `nakup_spotrebniho_zbozi`: množství spotřebního zboží, které se má nakoupit.
- `max_urokova_mira`: maximální úroková míra dluhopisů, kterou je vláda ochotna akceptovat (v promilích).

[3] *„Expanzivní fiskální politika podporuje agregátní poptávku. Jedná se o snížení čistých daní, zvýšení vládních nákupů apod.“* [14]

[4] *„Restriktivní fiskální politika omezuje nebo snižuje agregátní poptávku Může se jednat o zvýšení čistých daní nebo snížení vládních nákupů.“* [14]

Pro správu parametrů automatického hraní vlády byl vytvořen administrační formulář *auto_hrani_vlada*. Oděděním od třídy `Formular_administratora` byla vytvořena třída `Formular_auto_hrani_vlada`. Tato třída generuje HTML formulář pro vkládání/editaci záznamů v tabulce `auto_hrani_vlada` a zároveň zobrazuje aktuálně uložené hodnoty v této tabulce. Pro zobrazení těchto hodnot byla třída `Informace_o_ekonomikach_xhtml_vystupy` rozšířena o metodu `generuj_tabulku_auto_hrani_vlada()`.

Obsluha automatického hraní vlády Automatické hraní vlády obsluhuje třída `Automaticke_hrani_vlada`, reagující na událost `KONEC_KOLA` vyslanou třídou `Casovac`. Hraní je realizováno postupně pro jednotlivé ekonomiky. Třída v konstruktoru přijímá ID ekonomiky a číslo aktuálního kola. Provádí následující posloupnost kroků:

- Pomocí metody `over_aktivaci_auto_hrani()` ověřuje, že je automatické hraní vlády aktivováno. Aktivaci/deaktivaci automatického hraní vlády je možné provést pro jednotlivé ekonomiky definicí hodnoty v nově založeném sloupci `auto_hrani_vlada` v tabulce `ekonomiky`. Hodnota `1` znamená aktivaci, `0` deaktivaci.

- Pomocí metody `get_auto_subjekt()` načte ID subjektu, pod jehož ID budou generované nabídky a poptávky vkládány do databáze. V tomto případě nebyl vytvářen nový subjekt, ale příkazy jsou vkládány přímo pod již existující subjekt *vlada*, který je ovládán administrátorem.

- Počítá sumu peněz vybranou na daních v minulém kole pomocí metody `spocti_vyber_na_danich_v_kole()`. Tato metoda načítá sumu peněz SQL dotazem na tabulku `toky_penez`, který hledá jen takové toky, jejichž ID původu odpovídá nějaké nastavené dani.

- Pomocí metody `nacti_parametry_pro_kolo()` načítá nastavené parametry automatického hraní vlády z tabulky `auto_hrani_vlada_parametry` do příslušných atributů třídy.

- Pomocí metody `zadej_poptavku_po_spotrebnim_zbozi()` zadává poptávku na trhu spotřebního zboží. Poptávané množství je načteno z atributu `$nakup_spotrebniho_zbozi`. Pro načtení ceny spotřebního zboží v minulém kole slouží metoda `nacti_cenu_spotrebniho_zbozi_v_minulem_kole()`.

- Vypočítává emisi obligací pomocí metody `spocti_emisi()`. Pro výpočet emise obligací je použit vzorec uvedený v návrhu řešení v této kapitole.

- Pokud emise obligací vyjde klidná, je volána metoda `emituj_obligace()`. Ta počítá cenu obligace pomocí vzorce, uvedeného v návrhu řešení v této kapitole. Následně zadává poptávku na trhu obligací.

6.2.3 Automatické hraní centrální banky

Návrh řešení

Pro automatické hraní centrální banky bylo navrženou použití Taylorova pravidla (popsáno v kapitole 3.2.3) modifikovaného do následující podoby:

$$i = c * i_{t-1} + a(\pi - \pi^*) + b(y - y^*), \tag{6.7}$$

Rovnovážná úroková sazba je nahrazena vyhlášenou úrokovou sazbou v minulém kole. Parametr c určuje váhu této úrokové sazby. Administrátor hry bude zadávat následující parametry:

- c - váhu vyhlášené úrokové sazby v minulém kole,
- a - váhu inflační mezery,
- π^* inflační cíl,
- b - váhu růstové mezery,
- y^* cíl růstu HDP.

Parametr π (míra inflace) je spočítán podle vzorce:

$$\pi = \frac{\text{cena spotřebního zboží}_{t-1}}{\text{cena spotřebního zboží}_{t-2}} - 1 \tag{6.8}$$

Parametr y (růst reálného HDP) je spočítán podle vzorce:

$$y = \frac{\text{produkt minulé kolo}}{\text{produkt předminulé kolo}} - 1, \tag{6.9}$$

který lze rozepsat takto:

$$y = \frac{\text{suma výroby spotřeb. zboží}_{t-1} + \text{suma výroby kapitál. zboží}_{t-1}}{\text{suma výroby spotřeb. zboží}_{t-2} + \text{suma výroby kapitál. zboží}_{t-2}} - 1, \tag{6.10}$$

kde t představuje číslo aktuálního kola.

Hodnota parametru i_{t-1} (vyhlášená úroková sazba v minulém kole) je načtena. Výsledná úroková sazba (vypočtena z rovnice) je zadána na trhu úvěrů (jako nabídka) a na trhu úspor (jako poptávka). Úroková míra zadávaná na trhu úspor je ponížena o tzv. *úrokový diferenciál*, což je další parametr, který bude administrátor hry zadávat. Tímto způsobem řídí centrální banka úrokovou sazbu podle výsledků Taylorova pravidla.

Popis realizace

Ukládání parametrů automatického hraní centrální banky Pro ukládání parametrů automatického hraní centrální banky byla vytvořena tabulka `auto_hrani_cb_parametry` s těmito sloupci:

- `cislo_kola`: číslo kola, pro které jsou parametry nastaveny.
- `id_ekonomiky`: ID ekonomiky, pro kterou jsou parametry nastaveny.
- `vaha_sazby_minule_kolo`: váha úrokové sazby v minulém kole (v procentech).
- `vaha_inflace`: váha inflace (v procentech).
- `vaha_produktu`: váha produktu (v procentech).
- `cilova_inflace`: cílová inflace (v procentech).
- `cilovy_produkt`: cílový produkt (v procentech).
- `urokovy_diferencial`: úrokový diferenciál (v procentech), určuje rozdíl mezi sazbou zadávanou na trhu úvěrů a úspor.

Pro správu parametrů automatického hraní centrální banky byl vytvořen administrační formulář *auto_hrani_cb*. Oděděním od třídy `Formular_administratora` byla vytvořena třída `Formular_auto_hrani_cb`. Tato třída generuje HTML formulář pro vkládání/editaci záznamů v tabulce `auto_hrani_cb` a zároveň zobrazuje aktuálně uložené hodnoty v této tabulce. Pro zobrazení těchto hodnot byla třída `Informace_o_ekonomikach_xhtml_vystupy` rozšířena o metodu `generuj_tabulku_auto_hrani_cb()`.

Obsluha automatického hraní centrální banky Automatické hraní centrální banky obstarává třída `Automaticke_hrani_centralni_banka`. Reaguje na událost `KONEC_KOLA` vyslanou třídou `Casovac`. Stejně jako automatické hraní vlády je i automatické hraní centrální banky provedeno postupně pro jednotlivé ekonomiky. Třída v konstruktoru přijímá ID ekonomiky a číslo aktuálního kola. Provádí následující posloupnost kroků:

- Pomocí metody `over_aktivaci_auto_hrani()` ověřuje, že je automatické hraní centrální banky aktivováno. Aktivaci/deaktivaci automatického hraní centrální banky je stejně jako u automatického hraní vlády možné provést pro jednotlivé ekonomiky definicí hodnoty v nově založeném sloupci `auto_hrani_cb` v tabulce `ekonomiky`.

- Pomocí metody `get_auto_subjekt()` načte ID subjektu, pod jehož ID budou generované nabídky a poptávky vkládány do databáze. Stejně jako v případě vlády, nebyl ani nyní vytvářen nový subjekt, ale příkazy jsou vkládány pod již existující subjekt *centralni_banka*, který byl dosud ovládán administrátorem.

- Načítá nastavené parametry automatického hraní centrální banky z tabulky `auto_hrani_cb_parametry` do příslušných atributů třídy prostřednictvím metody `nacti_parametry_pro_kolo()` .

- Pomocí metody `nacti_cenu_zbozi()` načítá cenu zboží v minulém a předminulém kole. Tyto ceny metoda ukládá do polí `$cena_spotrebniho_zbozi` a `$cena_kapitaloveho_zbozi`.

- Pomocí metody `nacti_sumu_toku_zbozi()` načítá sumu toku zboží v minulém a předminulém kole. Tyto sumy toků ukládá metoda do polí `$suma_toku_spotrebniho_zbozi` a `$suma_toku_kapitaloveho_zbozi`.

- Pomocí metody `spocti_inflaci()` počítá míru inflace využitím vzorce uvedeného v návrhu řešení v této kapitole. Ceny zboží jsou načteny z polí `$cena_spotrebniho_zbozi` a `$cena_kapitaloveho_zbozi`. Vypočítaná míra inflace je uložena do atributu třídy `$inflace`.

- Pomocí metody `spocti_produkt()` počítá růst produktu využitím vzorce uvedeného v návrhu řešení v této kapitole. Sumy toků zboží jsou načteny z polí `$suma_toku_spotrebniho_zbozi` a `$suma_toku_kapitaloveho_zbozi`. Vypočítaný růst produktu je uložen do atributu třídy `$produkt`.

- Pomocí metody `spocti_urokovou_sazbu()` vypočítává úrokovou sazbu na základě vzorce uvedeného v návrhu řešení v této kapitole.

- Zadává nabídku na trhu úvěrů a poptávku na trhu úspor prostřednictvím metod `zadej_nabidku_na_trhu_uveru()` a `zadej_poptavku_na_trhu_uspor`.

7 Další provedené úpravy

Následující kapitola popisuje úpravy realizované nad rámec zadání práce.

7.1 Rozšíření generovaného XLS souboru s výsledky kola

XLS soubor s výsledky jednotlivých kol zobrazuje mimo jiné tabulku vývoje tržních cen na jednotlivých trzích. Třída `XLS_vyhodnoceni_kola` byla upravena tak, aby generovaná tabulka zobrazovala navíc obchodované množství na trhu při tržní ceně. Díky tomu mají hráči k dispozici více informací o realizovaných obchodech na trzích, což usnadňuje jejich rozhodování ve hře.

	A	B	C	D
1	Číslo kola	Název trhu	Tržní cena	Obchodované množství
2	1	trh práce	150	1433
3	1	trh kapitálového zboží	60	4679
4	1	trh spotřebního zboží	3	11554
5	1	akciový trh	1000	140
6	1	trh úvěrů	0	0
7	1	trh úspor	0	0
8	1	trh obligací	900	0
9	2	trh kapitálového zboží	1	4090
10	2	trh spotřebního zboží	10	17400
11	2	akciový trh	975	6
12	2	trh úvěrů	0	0
13	2	trh úspor	0	0
14	2	trh obligací	900	0
15	2	trh práce	113	1302

Obrázek 7.1: XLS soubor s výsledky kola rozšířený o obchodované množství - zdroj: vlastní zpracování

7.2 Úprava úrokové sazby na trhu úvěrů a úspor

Třída `Tok_penez` byla upravena, aby na trhu úvěrů a úspor zpracovávala zadávanou úrokovou sazbu v promilích místo v procentech jako doposud. Důvodem k tomu je možnost jemnějšího škálování úrokové sazby (systém

umožňuje zadávat jen celá čísla). Dříve používaná procenta byla příliš hrubá (např. změna úrokové sazby z 2 % na 3 % představuje výraznou změnu). Nyní je možné zadat např. hodnotu 21 promile (2,1 %), což představuje menší změnu.

7.3 Úprava omezení cenového rozsahu při zadávání poptávek a nabídek hráči

Pro zamezení výkyvům v cenách bylo upraveno omezení zadávaného cenového rozsahu na trzích. Původně existovalo omezení jen pro maximální zadávanou cenu. Bylo možné zadat maximálně násobek (definovaný v atributu `$maximalni_mezni_cena_nasobek_minule_ceny` třídy `Trzni_prikaz`) tržní ceny z minulého kola. Pokud byla tržní cena v minulém kole nulová, bylo možné zadat cenu ve výši maximálně `$maximalni_mezni_cena`.

Nově přibylo omezení také minimální zadávané ceny - opět násobkem tržní ceny z minulého kola (v atributu `$minimalni_mezni_cena_nasobek_minule_ceny`). Pokud byla tržní cena v minulém kole nulová (neobchodovalo se), načítá se poslední nenulová tržní cena (administrátor hry definuje tržní ceny pro kolo číslo 0, které jsou vždy nenulové).

8 Pilotní projekt

8.1 Optimalizace databáze aplikace

Pro účely optimalizace databáze byla hra spuštěna od 18. října 2016 do 15. prosince 2016 a zúčastnilo se jí 152 hráčů - studentů předmětu *KEM/EKAP (Ekonomická analýza a prognóza)* a *KEM/MIKR2 (Mikroekonomie 2)*.

Období od 18. října 2016 do 24. října 2016 bylo označeno jako testovací, kolo se uzavíralo jednou denně. Účelem těchto cvičných kol bylo odhalit nejvýraznější chyby, studenti za účast ve cvičných kolech nebyli bodováni. V období od 25. října 2016 do 15. prosince 2016 proběhl „ostrý provoz“, kdy byli hráči za účast a předvedené výkony ve hře bodováni.

Od začátku cvičné hry do konce hry byly zaznamenávány SQL dotazy, pokud doba jejich provádění přesáhla hodnotu 0,01 s. Tyto dotazy byly podrobeny analýze a optimalizovány.

V době od 1. prosince 2016 do 7. prosince 2016 (3 kola ve hře) byly zaznamenávány časy provádění všech SQL dotazů před provedením výše popsaných úprav. Poté byly úpravy nasazeny do ostrého provozu a od 8. prosince 2016 do 15. prosince 2016 (3 kola ve hře) byly zaznamenávány časy provádění všech SQL dotazů po provedení úprav.

8.1.1 Problematické uzavírání kol ve hře

V průběhu hry během zimního semestru se objevil problém s voláním služby cron pro uzavření kola. Několikrát se stalo, že služba neproběhla (zřejmě kvůli špatnému nastavení služby na straně CIVu). Bylo tak nutné, uchýlit se k ručnímu uzavírání kol. Uzavírání kol je obstaráno následovně.

Crontab

Soubor `cron_dsge_file.php` je pravidelně spouštěn v časových intervalech nastavených v souboru crontab.

Syntaxe definice crontabu je následující

```
* * * * *  příkaz k vykonání
```

Pět polí na začátku definuje čas:

1. minuta (0-59),

2. hodina (0-24),
3. den v měsíci (1-31),
4. měsíc (1-12),
5. den v týdnu (0-6), 0=neděle, 1=pondělí atd.

Hodnotu každého pole lze definovat následovně:

- konkrétním číslem (15),
- seznamem čísel, oddělených čárkou (10, 15, 20),
- rozsahem, odděleným pomlčkou (10 - 20),
- hvězdičkou (*).

Hvězdička značí, že se zadaný příkaz provede pokaždé (v rozsahu pro dané pole). Např. hvězdička na pozici minut znamená, že se příkaz provede každou minutu.

Po definici času spouštění prostřednictvím pěti polí následuje definice příkazu, který se má vykonat.

Např. následující definice v souboru crontab zajistí spuštění programu `www-dsgegame/cron_dsge_file.php` každý den ve 20:00. [9]

```
0 20 * * *  /www-dsgegame/cron_dsge_file.php
```

Definice časů konců kol

Administrátor hry definuje časy uzavírání kol pro jednotlivé integrační celky v tabulce `casy_koncu_kol` např. takto.

id_integracniho_celku	den_v_tydnu	hodina	minuta
2016ZS	Sat	9	0
2016ZS	Thu	9	0
2016ZS	Tue	9	0

Tabulka 8.1: Tabulka časů uzavírání kol - zdroj: vlastní zpracování

Samotné uzavírání kol pak obstarávají následující tři třídy:

- `Casovy_signal`,
- `Cas_konce_kola`,

- `Casovac`.

V souboru `cron_dsge_file.php`, který je spouštěn prostřednictvím cronu je vyvolána událost `CASOVY_SIGNAL` (princip volání událostí je popsán v kapitole 2.3.2). Tato událost je odchycena ve třídě `Casovac`. Třída `Casovac` následně vyšle časový signál prostřednictvím třídy `Casovy_signal` (parametry časového signálu jsou den v týdnu, hodina a minuta). Ve třídě `Cas_konce_kola` pak dojde k ověření, zda vyslaný časový signál odpovídá záznamu v tabulce `casy_koncu_kol`. Pokud ano, spustí se metoda pro uzavření kola v integračním celku.

Ruční uzavření kola ve hře

Pro ruční uzavření ve hře (pokud služba cron nefunguje), je potřeba, aby byl ze třídy Casovac vyslán časový signál odpovídající záznamu v tabulce `casy_koncu_kol`. Toho je možné docílit dvěma způsoby:

- vložit do tabulky `casy_koncu_kol` záznam s aktuálním dnem v týdnu, hodinou a minutou,
- upravit časový signál vysílaný ze třídy `Casovac` tak, aby nebyl vysílán časový signál s aktuálním časem, ale s časem, uvedeným v tabulce `casy_koncu_kol`.

Znázornění druhé možnosti následuje níže - je zakomentován řádek kódu pro vyslání časového signálu s aktuálním časem a vložen řádek kódu pro vyslání časového signálu odpovídajícího záznamu v tabulce `casy_koncu_kol`.

```
class Casovac extends Trida_generujici_udalosti {
...
  public function vysli_casovy_signal() {
    //$cs = new Casovy_signal(date('D'), date('G'), date('i'));
    $cs = new Casovy_signal('Sun', 20, 0);
  }
}
```

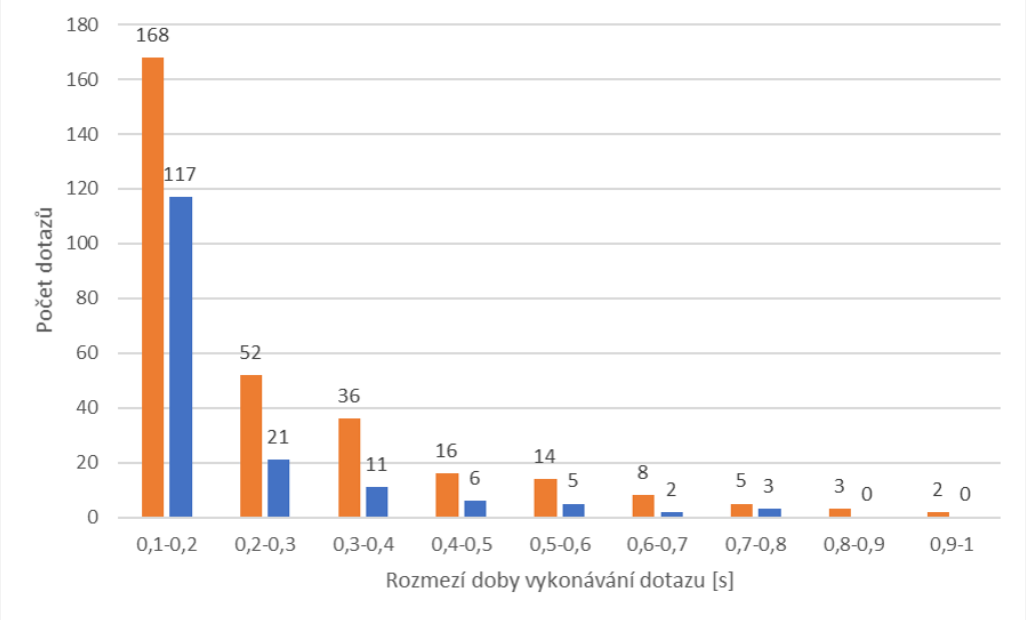
Následně je nutné spustit soubor `cron_dsge_file.php`.

8.1.2 Zhodnocení výsledků optimalizace

Počet příkazů zadaných hráči v období hry před nasazením úprav činil 28 889, v období hry po nasazení úprav pak 32 282. Aktivita hráčů v obou obdobích hry je tedy srovnatelná. Dosažené výsledky:

- Průměrný čas provádění SQL dotazu klesl z 0,637 ms před provedením úprav na 0,438 ms po provedení úprav, tj. pokles na 69%.
- Počet volaných SQL dotazů klesl z 593 363 na 446 279 , tj. pokles na 75%.

Obrázek 8.1 zobrazuje počet volaných dotazů v daných rozmezích doby jejich provádění před (oranžová barva) a po (modrá barva) provedené optimalizaci databáze.



Obrázek 8.1: Počet volaných dotazů v daných rozmezích doby jejich provádění před a po optimalizaci databáze - zdroj: vlastní zpracování

8.2 Akciový trh

Ověření rozšíření DSGEgame o akciový trh v rámci pilotního projektu proběhlo ve dnech 19. března 2017 až 11. května 2017. V této době se hry zúčastnilo 146 hráčů - studentů předmětu *KEM/MAKR2 (Makroekonomie 2)*. Období mezi 19. březnem 2017 a 24. březnem 2017 bylo označeno jako testovací, kolo bylo uzavíráno jednou denně a hráči za hraní v této fázi hry nedostávali žádné body. Účelem těchto cvičných kol bylo odchycení nejzávažnějších chyb. Během cvičných kol byla opravena chyba, kdy byly nekorektně generovány toky akcií - emise akcií nebyla přiřazena fondu, ale kupujícímu subjektu.

V období mezi 25. březnem 2017 a 11. květnem 2017 proběhla hra v „ostrém provozu“. Byla spuštěna varianta s akciovým fondem. 14 nejúspěšnějších hráčů ze zimního semestru vystupovalo ve hře jako výrobci, zbylí hráči vystupovali jako spotřebitelé. V tomto období byla prokázána správná funkčnost rozšíření. Správnou funkčnost rozšíření demonstruje následující ukázka toků akcií a peněz v kole číslo 9.

8.2.1 Generování toků akcií na akciovém trhu

Tabulka 8.2 zobrazuje toky akcií v kole č. 9.

id_toku	cislo_kola	id_subjektu	id_emitenta	id_puvodu	hodnota_toku
106	9	160	160	1232	30
107	9	160	160	1231	-30
108	9	227	160	1231	-1
109	9	168	160	1230	1
110	9	175	160	1230	1
111	9	178	160	1230	3
112	9	180	160	1230	5
113	9	189	160	1230	1
114	9	192	160	1230	1
115	9	193	160	1230	1
116	9	199	160	1230	2
117	9	213	160	1230	3
118	9	236	160	1230	3
119	9	238	160	1230	1
120	9	239	160	1230	3
121	9	249	160	1230	1
122	9	254	160	1230	2
123	9	287	160	1230	3

Tabulka 8.2: Realizované toky akcií v kole číslo 9 - zdroj: vlastní zpracování

Pokud procházíme řádky toků akcií shora, je patrné, že v tomto kole akciový fond (ID subjektu 160) emitoval 30 kusů akcií (ID původu emise akcií je 1232), které následně prodal hráčům (ID původu prodeje akcií je 1231). Kromě fondu prodával v tomto kole akcie také subjekt s ID 227 (1 kus akcie). Zbylé řádky představují nákupy akcií hráči. Suma hodnot ve sloupci `hodnota_toku` činí 30, což odpovídá 30 emitovaným akciím fondu.

8.2.2 Generování toků peněz na akciovém trhu

Tabulka 8.3 zobrazuje toky peněz realizované na akciovém trhu v kole číslo 9.

cislo_kola	id_meny	id_subjektu	id_puvodu	hodnota_toku
9	1	160	1231	30000
9	1	227	1231	1000
9	1	168	1230	-1000
9	1	175	1230	-1000
9	1	178	1230	-3000
9	1	180	1230	-5000
9	1	189	1230	-1000
9	1	192	1230	-1000
9	1	193	1230	-1000
9	1	199	1230	-2000
9	1	213	1230	-3000
9	1	236	1230	-3000
9	1	238	1230	-1000
9	1	239	1230	-3000
9	1	249	1230	-1000
9	1	254	1230	-2000
9	1	287	1230	-3000

Tabulka 8.3: Realizované toky peněz na akciovém trhu v kole číslo 9 - zdroj: vlastní zpracování

Tržní cena akcie v kole číslo 9 činila 1000 Kč. Pokud procházíme řádky toků peněz shora, vidíme, že fondu (ID subjektu 160) bylo přičteno 30 000 Kč za prodej 30 kusů akcií a subjektu s ID 227 přičteno 1 000 Kč za prodej 1 kusu akcie. Následující řádky zobrazují odečítané toky peněz jednotlivým subjektům za nákup akcií. ID subjektů v tabulce `toky_penez` odpovídají ID subjektů v tabulce `toky_akcii`, toky peněz jsou tedy generované odpovídajícím subjektům. Hodnoty toků peněz odpovídají počtu akcií vynásobeným tržní cenou akcie v daném kole, jsou tedy odečítána správné částky.

8.2.3 Výpočet a vyplacení dividend

Tabulka 8.4 zobrazuje zisk výrobců v kole číslo 9.

cislo_kola	id_subjektu	zisk
9	289	-757
9	290	-13778
9	291	-276
9	292	3828
9	293	-221
9	294	845
9	295	-193
9	296	1219
9	297	-433
9	298	471
9	299	-443
9	301	500
9	302	-7259

Tabulka 8.4: Zisky výrobců v kole číslo 9 - zdroj: vlastní zpracování

Vidíme, že zisk v tomto kole vygenerovaly subjekty s ID 292, 294, 296, 298 a 301. Jen těmto výrobcům budou odečítány dividendy. Vážený průměr hlasů, kolik procent ze zisku výrobců připadne na dividendy v tomto kole činil 4,975 %. Tabulka 8.5 zobrazuje toky peněz, které byly jednotlivým výrobcům odečteny na výplatu dividend.

cislo_kola	id_meny	id_subjektu	id_puvodu	hodnota_toku
9	1	292	1233	-190,4287164180
9	1	294	1233	-42,0355970149
9	1	296	1233	-60,6407014925
9	1	298	1233	-23,4304925373
9	1	301	1233	-24,8731343284

Tabulka 8.5: Odečty výrobcům na výplatu dividend v kole číslo 9 - zdroj: vlastní zpracování

Suma hodnot odečtených jednotlivým výrobcům (suma hodnot ve sloupci `hodnota_toku`) činí -341,409 Kč. Tolik peněz bylo odebráno výrobcům. V této částce je zahrnutá marže fondu, která v tomto kole byla nastavena na 1 %. Fond tedy vybral od výrobců o 1 % vyšší částku, než kterou si akcionáři odhlasovali, a rozdíl částek si ponechal. Marže fondu činila 3,38 Kč. Mezi akcionáře bylo rozděleno 338,029 Kč. Tabulka 8.6 zobrazuje toky peněz, které byly jednotlivým akcionářům připsány (ukazuje jen část, výpis těchto toků pro kolo číslo 9 obsahuje 57 řádků).

cislo_kola	id_meny	id_subjektu	id_puvodu	hodnota_toku
9	1	166	1234	1,6817331254
9	1	166	1234	11,7721318779
9	1	166	1234	3,3634662508
9	1	166	1234	1,6817331254
9	1	166	1234	3,3634662508
...	...	...	...	...
				SUM()
				338,0283582090

Tabulka 8.6: Odečty výrobcům na výplatu dividend v kole číslo 9 - zdroj: vlastní zpracování

Suma hodnot toků vyplacených akcionářům povýšená o marži fondu odpovídá tomu, kolik peněz bylo odečteno výrobcům.

8.3 Automatické hraní subjektů

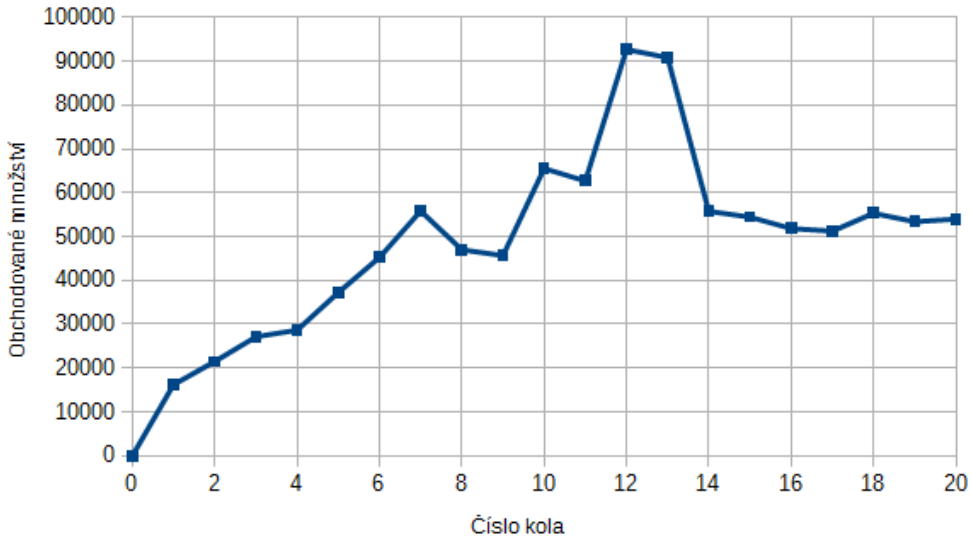
Ověření rozšíření DSGEgame o automatické hraní subjektů v rámci pilotního projektu proběhlo ve dnech 11. dubna 2017 až 11. května 2017. Do již běžící

hry (pro ověření rozšíření o akciový trh) bylo implementováno automatické hraní subjektů. Nejprve bylo nastaveno, aby třídy obsluhující automatické hraní generovaly kontrolní výpis hodnot a zasílaly jej na nastavené e-mailové adresy (autora a vedoucího práce), ale prozatím tyto hodnoty nezadávaly do databáze. Po sérii ověřování těchto výpisů a opravě chyb ve výpočtech bylo automatické hraní 20. dubna 2017 nasazeno do provozu. Byla prokázána správná funkčnost rozšíření. Správně byly generovány jak poptávky a nabídky za nehrající hráče, tak poptávky a nabídky generované vládou a centrální bankou.

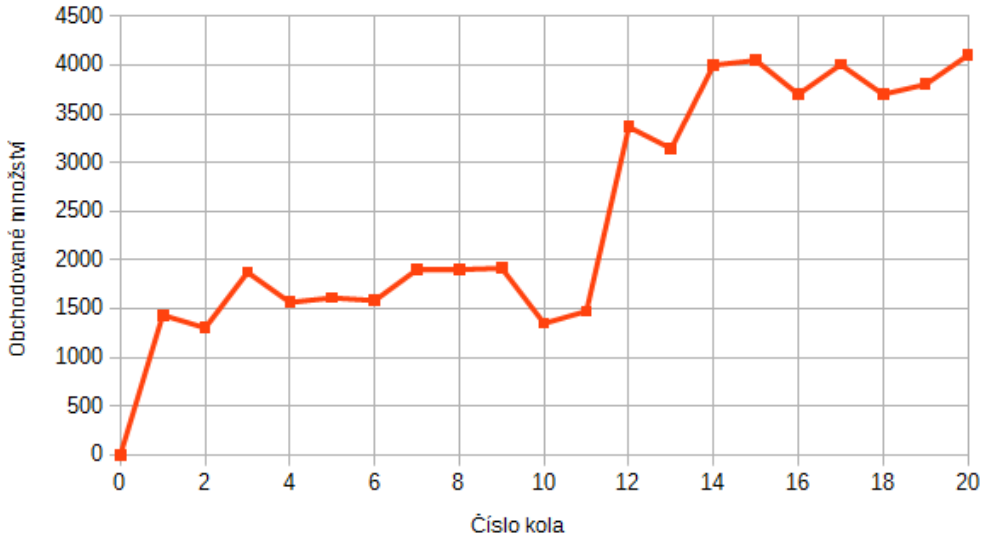
8.3.1 Automatické hraní spotřebitelů, podnikatelů a výrobců

Správnou funkčnost rozšíření demonstrují následující grafy. Automatické hraní bylo nasazeno od 14. kola. Následující grafy zobrazují

- součet obchodovaného množství spotřebního a kapitálového zboží v jednotlivých kolech,
- obchodované množství na trhu práce v jednotlivých kolech.



Obrázek 8.2: Součet obchodovaného množství spotřebního a kapitálového zboží v jednotlivých kolech - zdroj: vlastní zpracování



Obrázek 8.3: Obchodované množství na trhu práce v jednotlivých kolech - zdroj: vlastní zpracování

Od 14. kola je viditelná stabilizace obchodovaného množství na těchto trzích.

8.3.2 Automatické hraní vlády

Správnou funkčnost rozšíření demonstruje následující ukázka generovaných obligací v kole č. 13. Pro toto kolo byly administrátorem hry nastaveny tyto parametry automatického hraní vlády:

Parametr	**Hodnota**
Saldo	-5000 Kč
Nákup spotřebního zboží	10000 ks
Max. úroková míra	9,9 %

Tabulka 8.7: Nastavené parametry automatického hraní vlády pro kolo číslo 13 - zdroj: vlastní zpracování

Pro celou dobu trvání hru byly nastaveny tyto parametry obligací.

Parametr	**Hodnota**
Hodnota obligace	1000 Kč
Počet kol do splatnosti	3

Tabulka 8.8: Nastavené parametry obligací pro celou dobu hry - zdroj: vlastní zpracování

Cena spotřebního zboží v minulém kolo činila 5 Kč, suma peněz vybraných na daních 35 723,70 Kč. Dosazením do rovnice č. 6.4 byla emise obligací vypočtena následovně:

$$-5000 - 35723,7 + 10000 * 5 = 9276,3 K\check{c}.$$

Dosazením do rovnice č. 6.5 byla cena obligace vypočtena následovně:

$$\frac{1000}{(1 + 0.099)^3} = 756,366 K\check{c}.$$

Dosazením do rovnice č. 6.6 bylo množství nabízených obligací vypočteno jako

$$\frac{9276,3}{753,366} = 12,31.$$

Následně byla správně vygenerovaná nabídka na trhu obligací: množství 13 (zaokrouhleno nahoru) od ceny 756 (zaokrouhleno dolu) výše.

8.3.3 Automatické hraní centrální banky

Správnou funkčnost rozšíření demonstruje následující ukázka generované nabídky na trhu úvěrů a poptávky na trhu úspor v kole č. 14. Pro toto kolo byly administrátorem hry nastaveny tyto parametry:

Parametr	**Hodnota**
Váha sazby minulé kolo	90 %
Váha inflace	5 %
Váha produktu	5 %
Cílová inflace	2 %
Cílový produkt	5 %
Úrokový diferenciál	1 %

Tabulka 8.9: Nastavené parametry automatického hraní centrální banky pro kolo číslo 14 - zdroj: vlastní zpracování

Úroková sazba v minulém kole činila 1,1 %, míra inflace 6 % a tempo růstu produktu 2,8 %. Úroková sazba byla dle rovnice č. 6.7 spočtena následovně:

$$0,9 * 1,1 + 0,05 * (6 - 2) + 0,05 * (2,8 - 5) = 1,08\%$$

Následně byla správně vygenerována nabídka na trhu úvěrů a poptávka na trhu úspor, ponížená o jednoprocentní úrokový diferenciál.

9 Závěr

Hlavním cílem práce bylo rozšířit funkcionalitu DSGEgame o nové funkce. Byla navržena a realizována následující rozšíření:

- Model akciového trhu.
- Systém automatického hraní subjektů.

Obě rozšíření byla otestována v rámci pilotního projektu, kterého se zúčastnilo 146 hráčů - studentů ZČU.

Rozšíření DSGEgame o model akciového trhu přináší simulaci dalších reálných tržních situací a umožňuje provádět nové experimenty. Jako vhodné se jeví využití tohoto rozšíření v edukačním procesu. Byly realizovány dvě varianty modelu akciového trhu: varianta s monopolním investičním fondem a varianta s přímým obchodováním, mezi kterými může administrátor hry snadno přepínat.

Rozšíření DSGEgame o systém automatického hraní subjektů je přínosné z několika důvodů. Automatické hraní spotřebitelů a výrobců vyřešilo problém s neaktivitou části hráčů (kolísajícím počtem hráčů v průběhu hry), které působilo vychylování tržních cen a obchodovaného množství. Automatické hraní vlády a centrální banky usnadňuje práci administrátorovi hry. Vláda a centrální banka nemusí být nyní ovládány administrátorem, ale mohou hrát automaticky dle předem nastavených pravidel.

Vedle realizace těchto rozšíření byla věnována pozornost nastavení schématu databáze aplikace a podobě volaných dotazů. V minulosti měl totiž databázový server problémy s výkonem při zapojení velkého počtu hráčů. Byla provedena analýza a následná optimalizace databáze. Srovnání databáze před provedenou optimalizací a po ní prokázalo pokles průměrné doby vykonání dotazu na 69 % a pokles počtu volaných dotazů na 75 %.

Kromě zmíněné realizace rozšíření software o model akciového trhu a systém automatického hraní a optimalizace databáze bylo realizováno několik rozšíření malého rozsahu (např. úprava povoleného cenového rozsahu na trzích). Ta mají za cíl vylepšit simulaci ekonomiky, kterou hra nabízí.

Uživatelská dokumentace z pohledu hráče a z pohledu administrátora hry je součástí příloh práce.

Přehled zkratek

API Application Programing Interface

CDCP Centrální depozitář cenných papírů

CIT Corporate Income Tax

DPH Daň z přidané hodnoty

DPS Dividend Per Share

DSGE Dynamic Stochastic Generic Equilibrum

EPS Earnings Per Share

ERA Entity Relationship Attributes

FIFO First In First Out

GE Government Expenditures

GR Government Revenues

GUI Graphical User Interface

HTML HyperText Markup Language

IPO Initial Public Offering

OLG Overlapping Generations

PHP Hypertext Preprocessor

PIT Personal Income Tax

ROA Return On Assets

ROE Return On Equity

SPO Secondary Public Offering

SQL Structured Query Language

SŘBD Systém řízení báze dat

VAT Value Added Tax

Literatura

[1] Zákon č. 90/2012 Sb., o obchodních korporacích. In: Sbírka zákonů, 25.1.2012. Dostupné z: https://portal.gov.cz/app/zakony/download?idBiblio=74908&nr=90~2F2012~20Sb.&ft=pdf.

[2] *About MySQL* [online]. c2017. [cit. 6.5.2017]. Dostupné z: https://www.mysql.com/about/.

[3] *Akciové analýzy* [online]. c2017. [cit. 23.4.2017]. Dostupné z: https://trhy.mesec.cz/pruvodci/ceske-akciove-trhy/akciove-analyzy/.

[4] BAAR, O. *Historie SQL* [online]. 2008. [cit. 4.4.2017]. Dostupné z: http://pcworld.cz/archiv/historie-sql-17391.

[5] *Bulk Data Loading for InnoDB Tables* [online]. c2017. [cit. 16.4.2017]. Dostupné z: https://dev.mysql.com/doc/refman/5.5/en/optimizing-innodb-bulk-data-loading.html.

[6] *Clustered and Secondary Indexes* [online]. c2017. [cit. 1.5.2017]. Dostupné z: https://dev.mysql.com/doc/refman/5.7/en/innodb-index-types.html.

[7] *Co je rubopis* [online]. c2017. [cit. 23.4.2017]. Dostupné z: http://www.penize.cz/slovnik/rubopis.

[8] *CREATE TABLE Syntax* [online]. c2017. [cit. 10.5.2017]. Dostupné z: https://dev.mysql.com/doc/refman/5.7/en/create-table.html.

[9] *Crontab – Quick Reference* [online]. c2017. [cit. 9.4.2017]. Dostupné z: http://www.adminschoice.com/crontab-quick-reference.

[10] *DSGEgame - info* [online]. c2011. [cit. 5.11.2016]. Dostupné z: http://dsgegame.zcu.cz/index.php?id_stranky=info.

[11] DuBois, P. *MySQL profesionálně: komplexní průvodce použitím, programováním a správou MySQL.* 1.vyd. Mobil Media, 2003. ISBN 80-86593-41-X.

[12] Dědič, J. a. k. *Cenné papíry.* 1. vyd. PROSPEKTRUM Praha, 1994. ISBN 80-85431-98-X.

[13] *Emise akcií (IPO, SPO, DPO)* [online]. c2017. [cit. 23.4.2017]. Dostupné z: http://www.penize.cz/15858-emise-akcii-ipo-spo-dpo.

[14] *Fiskální politika* [online]. c2017. [cit. 23.4.2017]. Dostupné z: http://www.finance.cz/makrodata-eu/statni-rozpocet/fiskalni-politika/.

[15] *Floating-Point Types (Approximate Value) - FLOAT, DOUBLE* [online]. c2017. [cit. 16.4.2017]. Dostupné z: https://dev.mysql.com/doc/refman/5.7/en/precision-math-decimal-characteristics.html.

[16] Gangur, M. *Základy analýzy kapitálových trhů* [online]. 2006. [cit. 24.4.2017]. Dostupné z: http://home.zcu.cz/~gangur/ZAKT/Prednasky/prednaska5.pdf.

[17] *Floating-Point Types (Approximate Value) - FLOAT, DOUBLE* [online]. dev.mysql.com, 2017. [cit. 16.4.2017]. dev.mysql.com. Dostupné z: https://dev.mysql.com/doc/refman/5.5/en/floating-point-types.html.

[18] *Integer Types (Exact Value) - INTEGER, INT, SMALLINT, TINYINT, MEDIUMINT, BIGINT* [online]. c2017. [cit. 16.4.2017]. Dostupné z: https://dev.mysql.com/doc/refman/5.7/en/integer-types.html.

[19] Jílek, J. *Finanční trhy.* 1.vyd. Grada, 1997. ISBN 80-7169-453-3.

[20] *Monetární politika* [online]. c2017. [cit. 23.4.2017]. Dostupné z: http://www.finance.cz/makrodata-eu/menove-ukazatele/monetarni-politika/.

[21] *MySQL Community Server* [online]. c2017. [cit. 4.4.2017]. Dostupné z: https://dev.mysql.com/downloads/mysql/.

[22] *O experimentální ekonomii* [online]. c2017. [cit. 23.4.2017]. Dostupné z: http://www.lee-vse.cz/cze/o-lee/experimentalni-ekonomie.

[23] Pešík, J. *Simulační hra pro podporu výuky základů finanční gramotnosti pro střední školy.* Plzeň, 2010. 45 s. Diplomová práce na Fakultě ekonomické Západočeské univerzity v Plzni na katedře statistiky a operačního výzkumu. Vedoucí diplomové práce Mikuláš Gangur.

[24] Pešík, J. – Martinčík, D. *Sbírka příkladů z mikroekonomie.* 1.vyd. Západočeská univerzita v Plzni, 2014. ISBN 978-80-261-0478-0.

[25] Rejnuš, O. *Cenné papíry a burzy.* 2., přeprac. vyd. Akademické nakladatelství CERM, 2013. ISBN 978-80-214-4673-1.

[26] *Rozdělení akcií* [online]. c2013. [cit. 23.4.2017]. Dostupné z: http://www.anonymni-akcie.cz/rozdeleni-akcii/.

[27] Samuelson, P. A. – Nordhaus, W. D. *Ekonomie.* 18.vyd. NS Svoboda, 2008. ISBN 80-205-0590-3.

[28] Schwartz, B. a. k. *MySQL profesionálně: optimalizace pro vysoký výkon.* 1.vyd. Zoner Press, 2009. ISBN 978-80-7413-035-9.

[29] Sheldon, R. – Moes, G. *Beginning MySQL.* 1. vyd. Wiley Publishing, 2005. ISBN 0-7645-7950-9.

[30] *Simulating Events with PHP 5* [online]. c2017. [cit. 1.3.2017]. Dostupné z: http://www.devshed.com/c/a/PHP/Simulating-Events-with-PHP-5/.

[31] *Taylorovo pravidlo* [online]. c2017. [cit. 24.4.2017]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/index.pl?cast=4547.

[32] *Visual Explain Plan* [online]. c2017. [cit. 3.3.2017]. Dostupné z: https://dev.mysql.com/doc/workbench/en/wb-performance-explain.html.

[33] Štork, Z. – Závacká, J. – Vávra, M. *HUBERT: DSGE model České republiky* [online]. 2009. [cit. 7.5.2017]. Dostupné z: http://www.mfcr.cz/cs/o-ministerstvu/odborne-studie-a-vyzkumy/2009/hubert-dsge-model-ceske-republiky-9444.

Přílohy

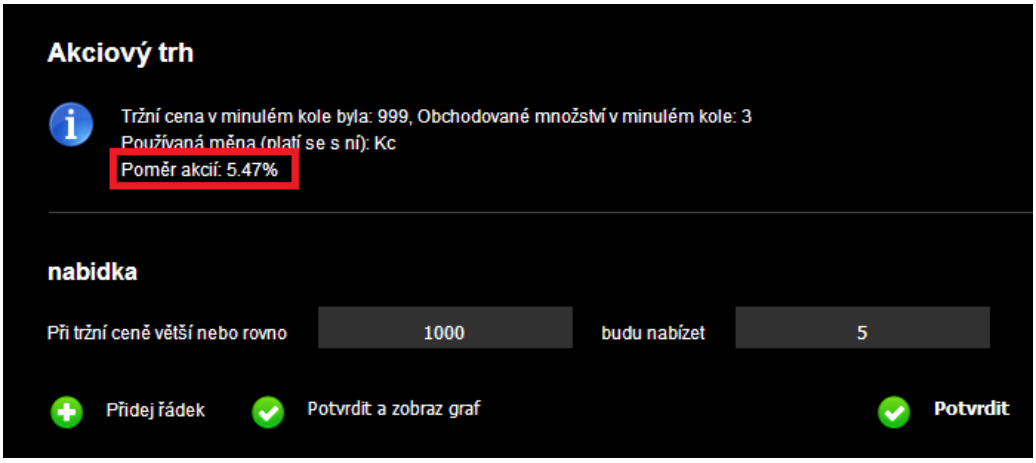
Příloha A: Uživatelská dokumentace

Uživatelská dokumentace z pohledu hráče

DSGEgame byla rozšířena o model akciového trhu. V ekonomice existuje *akciový fond*, který vystupuje jako *stocks packer* - prostředník, jenž *zabalí* akcie všech výrobců do jedné a tím pádem jsou akcie homogenní. Díky tomu existuje ve hře jediný akciový trh.

Akciový trh

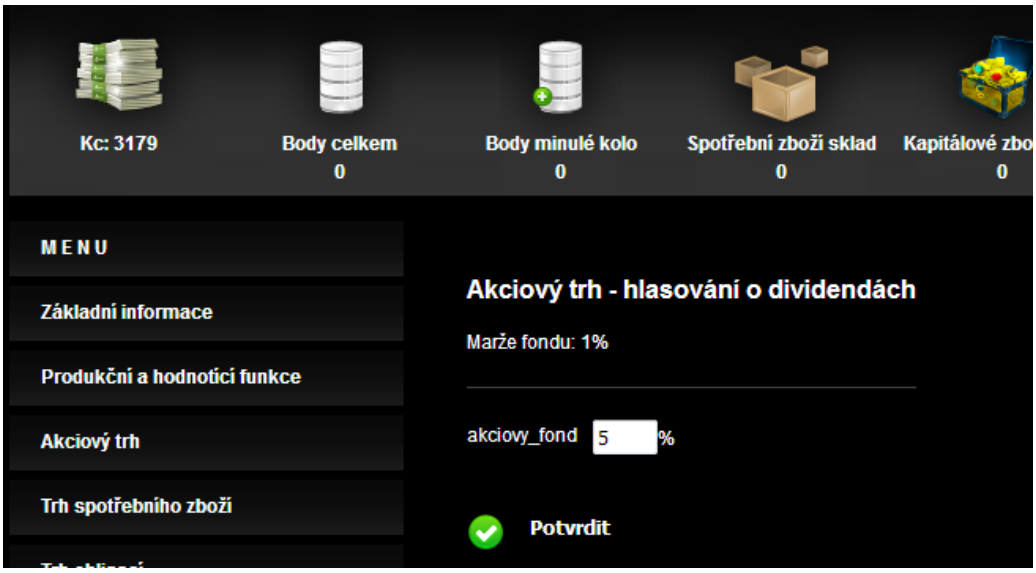
Pro přístup na akciový trh klikněte na položku v levém menu *Akciový trh*. Způsob obchodování na akciovém trhu je shodný s obchodováním na ostatních typech trhů. V horní části stránky *Akciový trh* je zobrazena informace o poměru akcií držených hráčem ku počtu všech akcií.



Kromě této stránky je informace o počtu držených akcií zobrazena také v pravé části horizontálního pruhu.

Hlasování o výši dividend

Ze zisku výrobců jsou vypláceny dividendy akcionářům - spotřebitelům a podnikatelům. Ti mají v každém kole možnost hlasovat, kolik procent zisku společnosti připadne na dividendy. K tomu slouží formulář, nalézající se pod položkou levého menu *Akciový trh - hlasování o dividendách*.

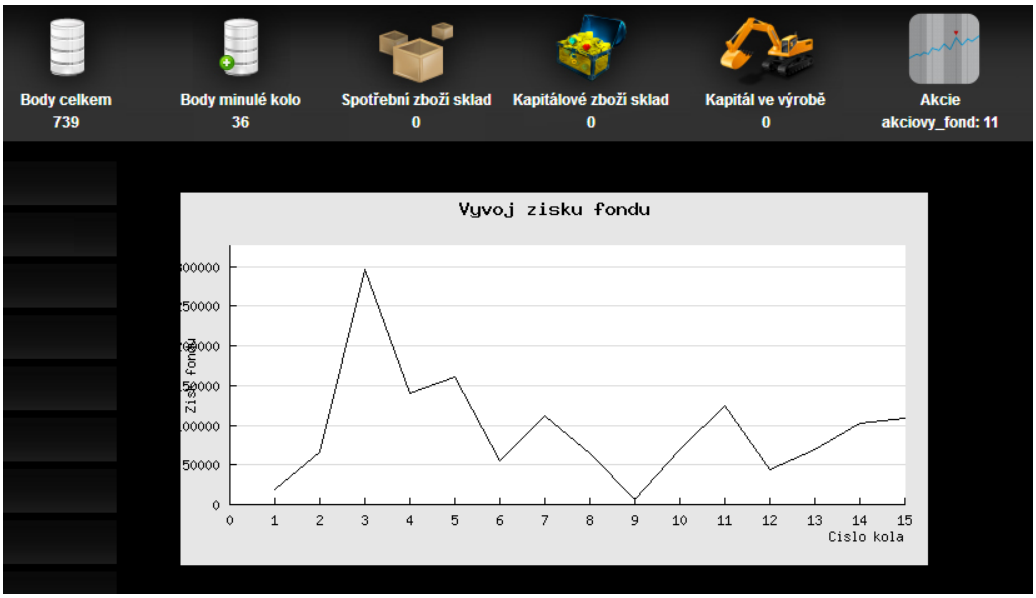


Výsledné procento zisku se vypočte jako vážený průměr zadaných hodnot, kdy váha hlasu je určena poměrem počtu akcií držených akcionářem ku celkovému počtu akcií emitenta.

Akciový fond ze zisků výrobců vybírá marži nastavenou administrátorem hry. Akcionáři odhlasují, kolik procent ze sumy zisku výrobců v daném kole připadne na dividendy, akciový fond tuto částku povýší o marži a povýšenou hodnotu vybere od výrobců. Akcionářům vyplatí tolik, kolik si odhlasovali, a marži si ponechá. Výše marže je zohledněna ve formuláři hlasování o dividendách - zadaná hodnota povýšená o nastavenou marži fondu nesmí překročit 100 %.

Sledování vývoje zisku fondu

Vývoj zisku fondu v jednotlivých kolech je možné sledovat na grafu nacházejícím se na nově vytvořené stránce. Vede na ni odkaz *Akciový trh - graf vývoje zisku* v levém menu. Pro vyplácení dividend není nutné, aby byla suma zisku všech výrobců kladná. Dividendy jsou vypláceny ze zisku jednotlivých výrobců, pokud v kole nějaký vygenerují. Proto graf vývoje zisku fondu zobrazuje sumu kladných zisků výrobců.

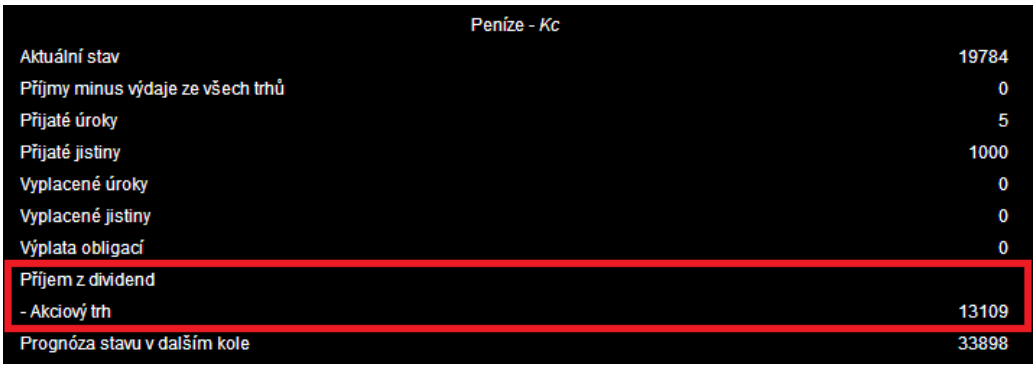


Kromě tohoto grafu je vývoj zisku fondu možné sledovat z generovaného XLS souboru s vyhodnocením kola, ve kterém přibyl list *vyvoj_zisku_fondu*.

	A	B	C
1	Číslo kola	id_fondu	zisk
2	1	160	19284
3	2	160	65673
4	3	160	296958
5	4	160	139809
6	5	160	161254
7	6	160	54490
8	7	160	112700
9	8	160	64146
10	9	160	6863
11	10	160	69037
12	11	160	125334
13	12	160	43942
14	13	160	69402
15	14	160	101725
16	15	160	108925

Prognóza výsledků pro zadané hodnoty

Stránka *Prognóza výsledků pro zadané hodnoty* byla rozšířena o odhadovaný příjem z dividend. Odhad příjmů z dividend se vypočte vynásobením ukazatele *dividenda na akcii* (dividendy/počet akcií) prognózovaným množstvím akcií v držení hráče (akcie aktuálně v držení + akcie poptávané - akcie nabízené).

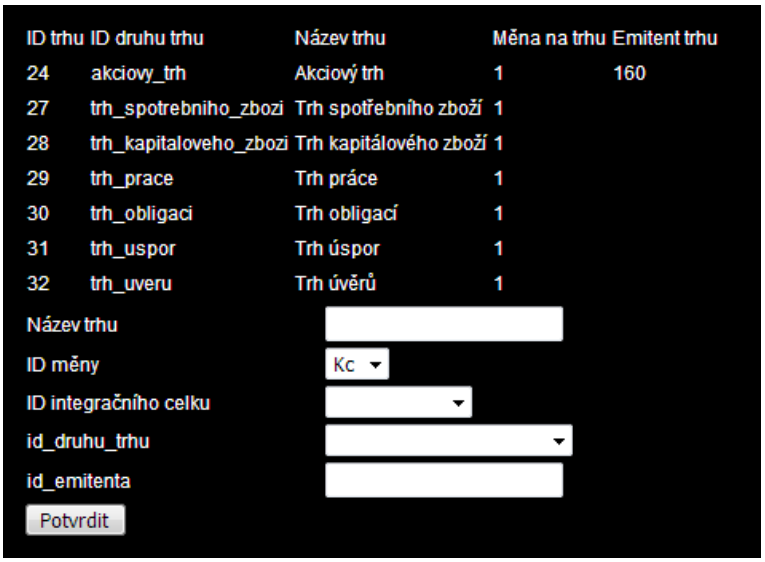


Uživatelská dokumentace z pohledu administrátora

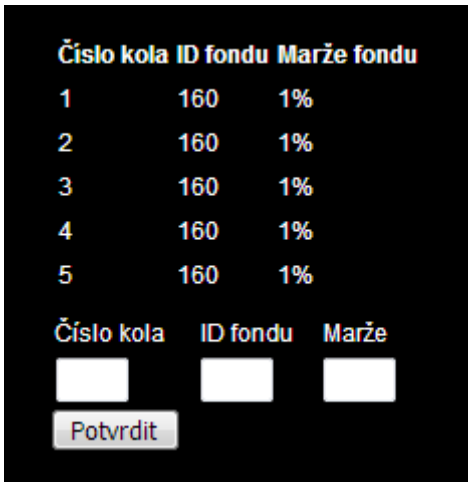
K administraci akciového trhu a automatického hraní slouží administrátorské formuláře v levém menu hry.

Akciový trh

Založení akciového trhu K založení trhu slouží formulář *vytvoreni_trhu*. Způsob založení akciového trhu je obdobný jako způsob založení kteréhokoli jiného typu trhu, jen je třeba navíc definovat ID emitenta na tomto trhu = ID fondu (vytvořeného subjektu druhu fond).

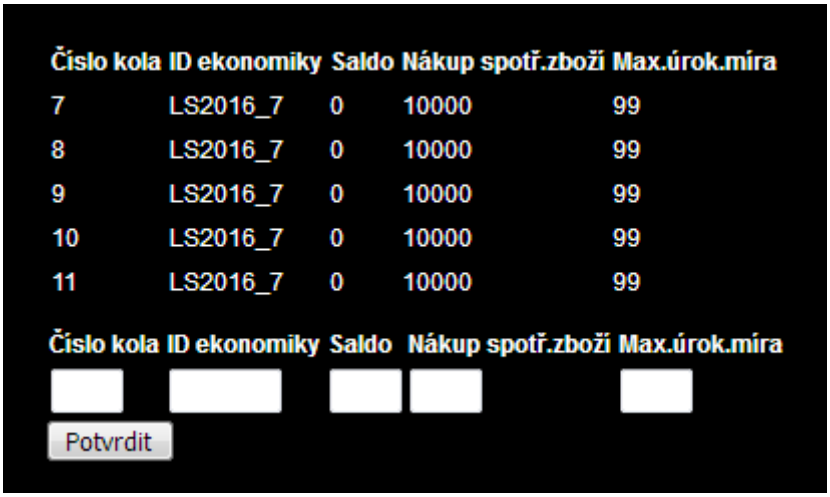


Nastavení marže fondu Pro nastavení výše marže fondu v jednotlivých kolech slouží nově vytvořený formulář *marze_fondu*. Nad formulářem jsou zobrazeny aktuálně nastavené hodnoty marže v jednotlivých kolech. Vyplněním formuláře lze vložit novou hodnotu nebo editovat některou ze stávajících (při zadání stejného čísla kola budou stávající hodnoty přepsány).



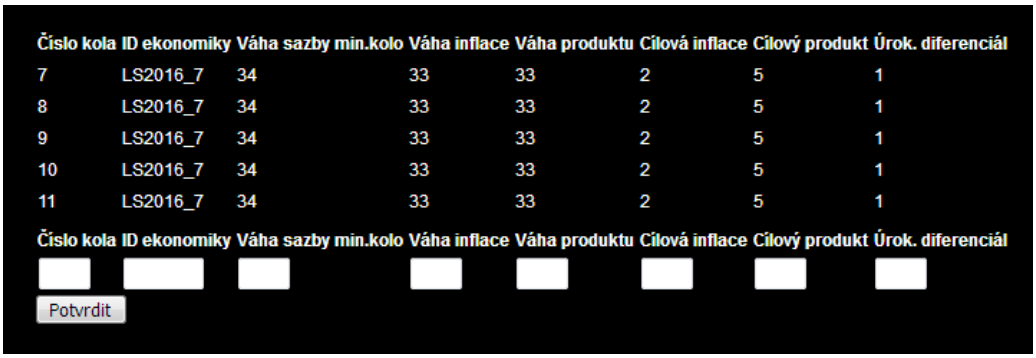
Automatické hraní vlády - nastavení parametrů

Pro nastavení parametrů automatického hraní vlády slouží nově vytvořený formulář *auto_hrani_vlada*. Nad formulářem jsou zobrazeny aktuálně nastavené hodnoty parametrů v jednotlivých kolech. Vyplněním formuláře lze vložit nové parametry nebo editovat stávající (při zadání stejného čísla kola budou stávající hodnoty přepsány).



Automatické hraní centrální banky - nastavení parametrů

Pro nastavení parametrů automatického hraní centrální banky slouží nově vytvořený formulář *auto_hrani_cb*. Nad formulářem jsou zobrazeny aktuálně nastavené hodnoty parametrů v jednotlivých kolech. Vyplněním formuláře lze vložit nové parametry nebo editovat stávající (při zadání stejného čísla kola budou stávající hodnoty přepsány).

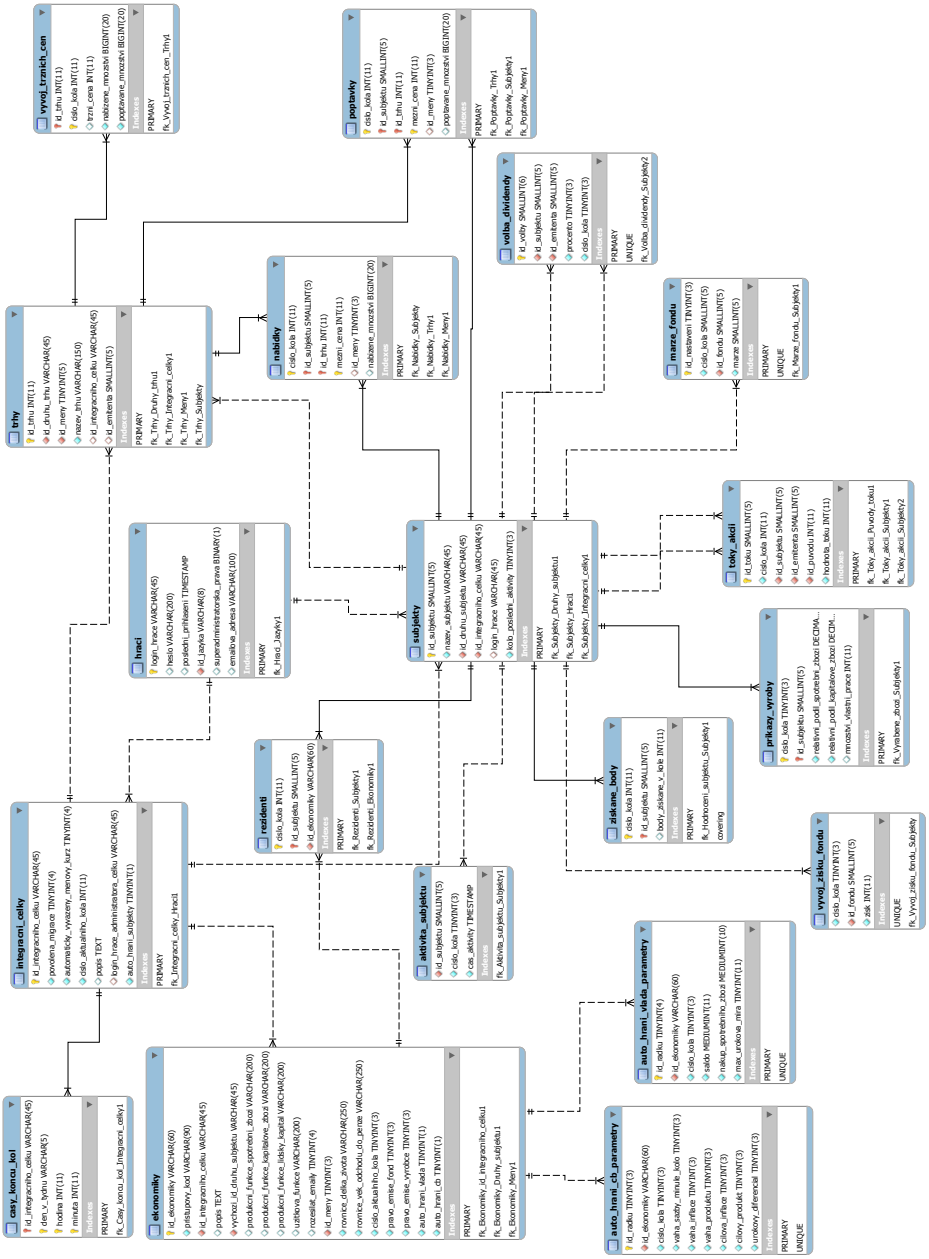


Číslo kola	ID ekonomiky	Váha sazby min.kolo	Váha inflace	Váha produktu	Cílová inflace	Cílový produkt	Úrok. diferenciál
7	LS2016_7	34	33	33	2	5	1
8	LS2016_7	34	33	33	2	5	1
9	LS2016_7	34	33	33	2	5	1
10	LS2016_7	34	33	33	2	5	1
11	LS2016_7	34	33	33	2	5	1

Příloha B: Modely a diagramy

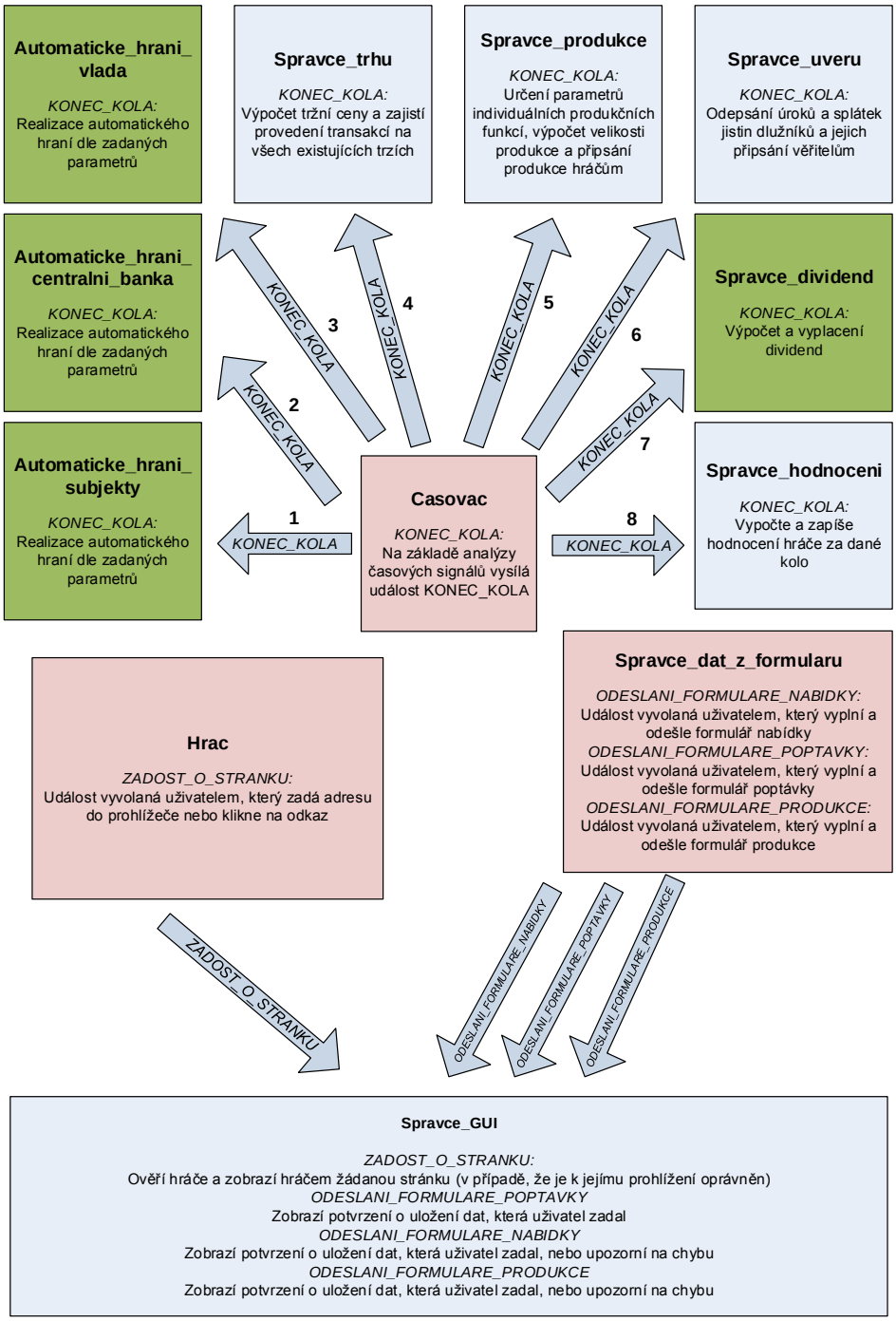
Zjednodušený ERA model

Vzhledem k rozsáhlosti modelu, zobrazuje následující obrázek jen jeho část. Kompletní ERA model je uložen na přiloženém CD.



Zjednodušený objektový model

Nové objekty jsou označeny zelenou barvou.



Zjednodušený UML diagram tříd

Vzhledem k rozsáhlosti modelu, zobrazuje diagram jen jeho část. Kompletní UML diagram tříd je uložen na přiloženém CD. Nové objekty jsou označeny zelenou barvou.