Univerzita Hradec Králové
Fakulta informatiky a managementu

Bakalářská práce

2021 David Urban

Univerzita Hradec Králové
Fakulta informatiky a managementu
Katedra informatiky a kvant. metod

Užití SQL pro vývoj ekonomických systémů

bakalářská práce

Autor: David Urban
Studijní program: B1802 / Aplikovaná informatika
Studijní obor: 1802R001 / Aplikovaná informatika

Vedoucí práce: doc. RNDr. Petra Poulová, Ph.D.

Hradec Králové, 2021

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval (pod vedením vedoucího bakalářské práce) samostatně a uvedl jsem všechny použité prameny a literaturu.

V Hradci Králové dne 16.08.2021

Poděkování

Děkuji vedoucí mé bakalářské práce doc. RNDr. Petře Poulové, Ph.D. za pomoc při kontrolování mé práce a za pomoc při hledání vhodných zdrojů, ze kterých jsem čerpal a firmě Softbit Software s.r.o. za příležitost.

Anotace

Bakalářská práce je v teoretické části zaměřená na objasnění struktury relačních databází a principů jejich fungování včetně objasnění problematiky transakcí. Popisuje relační datový model, na kterém je relační databáze založena a srovnává je s ostatními typy. Dále objasňuje principy programů systémů řízení báze dat a stručně popisuje a porovnává jejich výhody a nevýhody. Na konci teoretické části se čtenář dozví o SQL jazyku a naučí se základní příkazy spolu s principy přenosu dat mezi dvěma databázemi. V praktické části jsou tyto znalosti realizovány návrhem databáze a vývojem aplikace na efektivní zpracování požadavků od zákazníků. Taková aplikace se obecně označuje jako helpdesk a zprostředkovává komunikaci mezi klientskou a serverovou verzí aplikace.

Annotation

Title: SQL and Design of Economic Systems

The bachelor's thesis is in its theoretical part focused on clarifying the structure of relational databases and the principles of their functionality, including clarifying the issues of transactions. It describes the relational data model on which the relational database is based and compares them with other types. It further clarifies the principles of database management system programs and briefly describes and compares their advantages and disadvantages. At the end of the theoretical part, the reader will learn about the SQL language and learn the basic commands along with the principles of data transfer between two databases. In the practical part of the work, this knowledge is realized by designing a database and developing an application for effective handling of requests from customers. The application is typically called helpdesk and processes communication between the client and server versions of the application.

Obsah

Úvod

V bakalářské práci jsem se v teoretické části zaměřil na analýzu relační databáze a způsobu její manipulace a ovládání pomocí SQL jazyka. Hlavním cílem je vysvětlení využitých technologií a zúročení, co nejvíce poznatků a získání obecného přehledu o databázích, jejich vývoje a dalších souvisejících pojmů včetně hlubší znalosti SQL jazyka pro vyvinutí aplikace za účelem zprostředkování komunikace mezi dvěma databázemi.

Konkrétně bych chtěl v teoretické části shrnout a všeobecně popsat základní pojmy a principy relačních databází a SQL jazyka, který se používá na jejich ovládání. V prvních kapitolách definuji základní pojmy a struktury relačních databází. Dále popisuji populární programy SŘBD sloužící jako prostředek pro ovládání databází přes jazyk SQL. Později se věnuji SQL jazyku včetně popisování procedurálních nadstaveb SQL jazyka pro získání schopnosti vytváření komplikovanějších procedur a jiných databázových struktur. V závěru teoretické části se zabývám objasněním způsobů transferu dat mezi dvěma databázemi.

Získané znalosti jsou realizovány v praktické části této práce vývojem aplikace helpdesku určená pro firmu Softbit Software s.ro. Helpdesk je aplikace vytvořena pro zefektivnění komunikace mezi zákazníkem a poskytovatelem služby a pro hlídání akutních požadavků zadané zákazníkem. Aplikace bude umožňovat vyplnění nového požadavku nebo zprávy se všemi důležitými údaji a poskytovat přehledné zobrazení a filtrování dat pomocí mnoha volitelných podmínek. Dále bude aplikace rozlišovat uživatele s různými typy práv a privilegií s možností tvorby nového uživatele. Nejdůležitější část aplikace bude spočívat ve schopnosti komunikace mezi serverovou aplikací techniků a klientskou aplikací zákazníka včetně upozornění poslané na email technika při vytvoření nového požadavku nebo obdržení nové zprávy.

Teoretická část

1. Základní popis a definice relační databáze

Databáze jsou logicky uspořádaná data uložená na disku počítače. Relační databáze je databáze využívající relačního modelu dat. Pro reprezentaci dat se používají databázové objekty, jakými je například tabulka.

Data, která se ukládají do těchto tabulek musí splňovat určitá pravidla. Nejedná se jenom o zpřehlednění, čím více dat má uživatel na disku, tím více času bude muset strávit nad její údržbou a samotná databáze bude zabírat ještě více místa. Proces úpravy dat podle pravidel se nazývá normalizace a existují tři hlavní normální formy. Pro strukturování a vkládání dat do tabulek je potřeba, aby byla všechna data alespoň ve 3. normální formě. Při přechodu z jedné normy do druhé dochází ke globální úpravě dat podle pravidla v rámci dané normy.

Pro splnění první normální formy je potřeba, aby všechny hodnoty byly atomické. [6, s. 90] To znamená, aby hodnoty nebyly dále dělitelné. Perfektním příkladem je adresa. Celá adresa je složenina údajů jako je město, ulice, PSČ apod. Pro atomizaci je potřeba všechny tyto hodnoty rozdělit do nových sloupců se správnými datovými typy. Může nastat situace, kdy při návrhu tabulky z analýzy nemá autor detailně prostudovanou problematiku a na konci vytvoří další atributy – sloupce dat, které mají ve finále stejnou funkci a význam. Například, že se může jednat o navržení databáze hotelových hostů a pokojů, kde si zákazník přeje, aby v jeho databázi byla evidována všechna čísla pokojů. Hotel je rozdělen do dvou částí a pokoje jedné části se značí numericky a druhá část alfanumericky. Jelikož se jedná o pokoje jednoho hotelu nemá smysl mít dva různé sloupce s odlišnými datovými typy. Oba sloupce se mohou sjednotit do jednoho s datovým typem VARCHAR o větší velikosti.

Pro druhou normální formu se musí každý řádek (záznam) v tabulce označit speciální klíčem. [6, s. 92] Existují dva typy klíčů primární a cizí klíč. Primární klíč musí být jen jeden v tabulce a jednoznačně určuje každý řádek tabulky. Cizích klíčů můžeme mít v tabulce hned několik. Primární klíč je velmi důležitý, jelikož nám zajišťuje unikátnost záznamu a možnost propojit ho prostřednictvím cizího klíče jiné tabulky.

Jakmile je databáze ve druhé normální formě, je pro převod do 3. normální formy zapotřebí, aby v rámci tabulky nebyly údaje závislé na sobě (kromě klíčů). [6, s. 92] Může nastat situace, kdy máme dvě hodnoty, které jsou na sobě závislé – například plat a funkce. Pří pořízení více záznamů může dojít k situaci, kdy v jednom řádku je u zaměstnance vyšší základní plat než u záznamu jiného zaměstnance, i když mohou vykonávat stejnou funkci. V této situaci by se tabulka měla rozdělit na více tabulek použitím cizích klíčů.

Cizí klíč je v podstatě odkaz na určitý záznam jiné tabulky. Hodnota cizího klíče musí přesně souhlasit s hodnotou primárního klíče druhé tabulky. Tímto způsobem se může tabulka skvěle zjednodušit a zbavit se zbytečných redundancí a závislostí. Místo vypisování stejných údajů pro více záznamů v tabulce se může jednoduše všechno nahradit jedním sloupcem, kde nám hodnota bude opět odkazovat na všechny potřebné sloupce s údaji v jiné tabulce. Pro odstranění závislostí a redundancí bychom měli vytvořit ze složitých tabulek dvě a více tabulek, které budou svázány s hlavní tabulkou přes cizí klíč. Tyto pomocné tabulky jsou obvykle nazývány číselníky a mohou se použít pro libovolný počet nových tabulek (například číselník druhů sazeb DPH, číselník měst).

Podle Lacka [6, s. 93] existují další úrovně formalizace jako jsou Boyce – Coddova normální forma (BCNF), čtvrtá forma a pátá normální forma. Ohledně čtvrté normalizační formě Lacko [6, s. 93] píše: „Tabulka je ve čtvrté normální formě tehdy, je-li ve třetí normální formě a popisuje jen jeden fakt nebo souvislost.“ Pro konverzi tabulky do páté normální formě je zapotřebí, aby tabulka byla ve čtvrté formě a nemohla se rozšířit o atribut (sloupec) nebo skupinu atributů, aniž by se neporušila jiná pravidla normální formy a nemusela by se rozdělit na další jednotlivé tabulky.

Pokud by se databáze nepřevedla do čtvrté, páté nebo BCNF formy, tabulka by nemusela být úplně dokonalá, ale pro většinu databází bohatě stačí první tři formy.

1.1. Popis základních pojmů

V této části práce budou definovány vybrané pojmy a termíny seřazené abecedně, které by byly při čtení této práce užitečné vědět, a které nebyly doposud vysvětleny plně vysvětleny.

- Atribut – je nezbytnou součástí všech tabulek a můžeme si je představit v tabulce jako sloupce. Jejich hodnoty závisí na zvoleném datovém typu. Procházka [10, s. 126] ohledně datových typů píše, že „každá konstanta nebo proměnná má svůj datový typ, který určuje formát, ve kterém je hodnota uložena, a specifikuje také omezení této hodnoty“. Různé programy SŘBD mají své vlastní datové typy. Například ve Firebirdu se proměnné s číselnou hodnotou typicky deklarují typem NUMERIC (5,0), kde číslo pět znamená maximální počet číslic a číslo nula je počet desetinných míst. Dále jako příklad datového typu je VARCHAR (30), který se používá pro definování volného textu. Číslo třicet znamená maximální počet znaků, které tento atribut v řádku může maximálně obsahovat. Atributy se mohou rozšířit o další vlastnosti jako je například klauzule default, přes který může uživatel nastavit úvodní hodnotu při založení nového záznamu. Dalším často užívanou klauzulí je klauzule NOT NULL.

Procházka [10, s. 45] uvádí, že toto omezení je u sloupců, kde uživatel chce, aby v každém záznamu tohoto atributu byla uvedena alespoň nějaká platná hodnota. Tato klauzule je velice užitečná například při vytváření atributu pro primární klíč.

- Datový řádek – (nebo také databázový záznam) je jedna konkrétní část dat, která je, ale nutně hned nemusí být součástí tabulky. Je definována počtem atributů, kde každý atribut má svůj vlastní datový typ a představuje sloupec v tabulce. Při vytvoření nové tabulky nemá tabulka žádné záznamy. Ty jsou později pořízeny uživatelem příslušným programem SŘBD nebo příslušnou aplikací přes uživatelské rozhraní. Každý řádek je unikátně označen primárním klíčem.
- Embedded databáze – mohou se charakterizovat jako opak klient-server databází. Databáze není uložena ve vyhrazeném databázovém serveru, ale běží v rámci prostoru samotné aplikace. Z principu je tato architektura vytvořena pro menší databáze, ke kterým nemá smysl nainstalovat speciální program pro její úpravu. [3]
- Klient-server databáze – je jedna z nejčastějších architektur databáze. Je tvořena dvěma částmi, první z nich je část serverová, kde jsou uložena data a nad daty se vykonávají operace. Klientská část slouží pro snadný přístup na server k databázi. Výhodami jsou centralizace, kdy jeden výkonný počítač, na kterém server běží zpracovává všechny dotazy a procesy. Za pomocí uživatelských práv je možné snadno nastavit práva uživatelů a přístupy z více klientský počítačů. Hlavní nevýhodou je, že při výpadku serveru nebo internetu se přeruší kontakt se serverovou aplikací. [12]
- NULL – tento údaj znamená prázdnou hodnotu – v daném záznamu u atributu není žádný údaj vyplněn. Pozor neplést NULL a s nulou! Opakem je NOT NULL. NOT NULL je omezení, které je vždy využito alespoň u atributu (sloupce) s primárním klíčem, jelikož každý záznam v tabulce musí být unikátně touto hodnotou označen.
- Open source licence – jedná se o programy, které jsou dostupné pro všechny uživatele, jsou bezplatné a každý může modifikovat zdrojový kód s použitím specifických nástrojů pro daný jazyk. Ale i zde platí určité podmínky, typicky se nesmí použít pro komerční použití.

1.2. Struktura relační databáze

Samotná databáze se skládá z několika typů objektů. Každý z nich má jinou roli. Jazyk SQL pro vytvoření těchto databázových struktur se nazývá ve zkratce DDL – jazyk pro definici dat. I když by měly být základy pro většinu databází stejné, každý program SŘBD si může konkrétní implementaci těchto datových struktur řešit vlastním způsobem. Aplikace SŘBD mají všechny

databázové objekty přehledně uspořádané na liště. Pro větší přehlednost pro uživatele existují konvence a pravidla na pojmenovávání. Například Lacko [6, s. 67] vysvětluje, že databázové objekty jako například tabulka nebo pohled musí vždy začínat alfanumerickým znakem nebo podtržítkem. Pokud se uživatel rozhodne pro název složený z více slov, musí název objektu ohraničit uvozovkami nebo hranatými závorkami.

1.2.1. Tabulky

Procházka [10, s. 67] uvádí „tabulky jsou naprostým základem každé databáze. Definují rámec pro ukládání dat.“ Svou strukturou definují (sloupce) o jaká data se jedná svým názvem a přes datový typ určují množinu platných hodnot. V zásadě je vždy lepší vytvořit samotnou tabulku na základě dostupné analýzy řešeného problému a pak posléze ji naplnit daty. Tabulka je rozdělena do řádků a sloupců. V rámci sloupců je každý sloupec definován svým datovým typem. Řádky nebo záznamy jsou data samotná a v tabulce jich teoreticky můžeme mít nekonečně mnoho. Každý řádek (záznam) musí být unikátní, a to je dosaženo pomocí jedinečné hodnoty primárního klíče. Datové typy sloupců jsou vždy deklarovány v určitém rozsahu. Pro každý sloupec můžeme dále definovat jeho přesný rozsah pomocích integritních omezení – konkrétně pomocí příkazu podmínky CHECK.

Základní příkaz pro vytvoření tabulky je přes CREATE TABLE. Po zadání příkazu CREATE TABLE definujeme název tabulky a jsme připraveni k deklaraci atributů a jejich datových typů spolu s ostatními klauzulemi jako je NOT NULL, default atd.

V některých programech SŘBD jako je Oracle jsou k dispozici další funkce jako je „dočasná tabulka.“ V určitých případech, kdy si uživatel přeje jen testovat si může vytvořit tabulku pomocí příkazu TEMPORARY, tato tabulka bude přítomná v databázi do té doby, než se uživatel odhlásí a poté ji systém automaticky smaže.

1.2.2. Procedury

Procedury nebo uložené procedury jsou skupiny příkazů SQL jazyka. Procedury jsou volány v rámci aplikací nebo samotného programu SŘBD. Při spuštění mohou zpátky vracet výstupní hodnoty. Procedury se mohou skládat ze tří částí. V první části se deklarují a inicializují vstupní proměnné, které jsou naplněny manuálně uživatelem v prostředí SŘBD nebo podle příkazů v programu přiřazeny v aplikaci. Posléze se deklarují dočasné proměnné, které se vyplní až v další fázi. Ve druhé části se naplní dočasné proměnné a použijí se pro vykonání samotného kódu. V poslední části se hodnoty vypočtené a jiné vkládají do samotných tabulek.

Procedury se vytvářejí přes příkaz CREATE PROCEDURE. Pro spuštění procedury v rámci například v Microsoft SQL Serveru se použije příkaz EXECUTE následovaný názvem procedury a poté vstupními hodnotami pro procedury. [6, s. 259]

1.2.3. Trigger/spouštěč

Triggery/spouště jsou typy procedur, které provedou nějakou činnost po jejich spuštění. Na rozdíl od procedur se triggery spouštějí automaticky podle definované podmínky. Nevracejí zpátky žádné hodnoty, ale po spuštění provádí úpravy na datech uložené v tabulkách. Mohou se například použít pro zautomatizování určitých činností jako je naplnění primárního klíče přes volání sekvence nebo při vytvoření nové věty v tabulce.

Spouště se vytvářejí pomocí základního příkazu CREATE TRIGGER, posléze je třeba definovat jméno pro spouštěč a pomocí příkazu for se kterou tabulkou chceme trigger „spárovat“. Možné podmínky pro spuštění naprogramovaného triggeru jsou před (BEFORE), po (AFTER) nebo místo (INSTEAD OF) úpravy záznamu. Následně je třeba určit jakým typem operace se trigger spustí – INSERT (vložení záznamu), UPDATE (změna) nebo DELETE (smazání). Nakonec následuje samotné tělo triggeru, kde se vykonají požadované příkazy.

Procházka [10, s. 48] vysvětluje, že v jazyce PL/SQL existují dva typy triggerů. Jeden typ se vždy spustí pro každý řádek nebo záznam v tabulce přes příkaz FOR EACH ROW. Druhý typ se podle definované podmínky spustí jenom jednou.

1.2.4. Pohledy

Pohledy jsou příkazy pomocí, kterých uživatel může vyfiltrovat data z libovolného počtu tabulek. Pohledy jsou obzvláštně užitečné u velkých databází, kde je potřeba vyfiltrovat jenom potřebná data z určitého počtu tabulek. Na druhou stranu není neobvyklé mít databáze, kde pro každou tabulku je definován speciální pohled. Pomocí cizích klíčů nebo jiných identifikačních hodnot se mohou spojit data v pohled z více tabulek podle potřeby. Dále se mohou definovat nové sloupce (atributy), jejichž hodnoty jsou pouze vypočítány pomocí hodnot sloupců jiných tabulek. Další výhody je v řízení práv uživatelů, pomocí pohledů se mohou lépe rozlišit, které pohledy a tím pádem která data je daný uživatel oprávněn sledovat. Tato oprávnění se udělují pomocí SQL příkazu GRANT ACCESS.

Pohledy v SQL vytváříme pomocí příkazu CREATE VIEW. Po tomto příkazu se definuje rozsah pohledu, kde deklarujeme proměnné, které budou součástí pohledu. V zájmu přehlednosti by měly tyto atributy mít stejný název nebo označení jako názvy atributů z tabulek, ze kterých

pocházejí. Můžou se vytvořit i nové atributy, ale musí se v rámci pohledu inicializovat. Zrušení pohledu je klasicky příkazem DROP VIEW.

Procházka [10, s. 72] uvádí, že v SŘBD Oracle existují tři typy pohledů – jednoduchý, kombinovaný a materializovaný. Jednoduchý typ pohledu je spojovaný jenom s jednou tabulkou, kde se vyfiltrují data pomocí nějaké podmínky nebo si programátor omezí výběr atribut z tabulky. Kombinovaný pohled je nejčastějším typem, jelikož kombinuje více tabulek najednou za pomocí cizích klíčů nebo jiných identifikátorů přes příkaz WHERE. Posledním type je materializovaný pohled, který Procházka [10, s. 72] vysvětluje takto: „Speciální typ pohledu, který data nejen zobrazuje, ale přímo obsahuje. Je tedy možné s daty dále pracovat a upravovat je. Materializované pohledy se nejčastěji využívají u datových skladů s agregovanými daty.“ Lacko [6, s. 97] dodává, že při vytvoření materializovaného pohledu je potřeba vybrat způsob aktualizace dat z tabulek, nad kterými byl pohled vytvořen. Tento pohled může být součástí stejné databáze s tabulkami nebo může být vložen i do jiné databáze.

1.2.5. Sekvence

Procházka [10, s. 47] píše ve své knize, že: „Definice sekvencí jsou uloženy v systémovém katalogu. Tyto sekvence vytvářejí seznam jedinečných čísel.“ Pro sekvenci je nutné definovat počáteční hodnotu a většinou i určený přírůstek, o který se bude původní hodnota inkrementálně zvyšovat při vkládání nových vět. Sekvence se používají nejčastěji při zautomatizování ukládání hodnot jako jsou primární klíč. Nicméně pro plnou automatizaci se musí sekvence propojit se spouštěčem. Pokud dojde k odstranění řádku s atributem přiřazeným k sekvenci a později znovuvytvoření stejného řádku, nastavená sekvence přiřadí nastavenému atributu další hodnotu, což může být problém, jelikož v tu chvíli dojde k „mezeře“ u hodnot. V rámci primárního klíče to příliš nevadí, jelikož tyto hodnoty jsou nebo by měly být pro běžného uživatele v aplikaci neviditelné.

Lacko [10, s. 106] ve své kníže definuje základní SQL příkaz v Oraclu pro sekvence jako „CREATE SEQUENCE nazev_posloupnosti“. Poté se definují klauzule příkazu. Klauzule START WITH určuje startovací hodnotu sekvence, nastavovaná hodnota parametru je většinou jedna. Pokud uživatel žádnou nenastaví, defaultní hodnota je jedna. Klauzule INCREMENT BY navyšuje hodnotu podle nastaveného parametru vždy při spuštění sekvence, implicitní hodnota je zde také jedna. Klauzule MINVALUE nastavuje minimální možnou hodnotu a MAXVALUE zase maximální. Klauzule CYCLE je důležitá v situaci, kdy hodnota sekvence přesáhne maximální hodnotu, v tomto případě se hodnota sekvence nastaví na minimální hodnotu a pokud je klauzule

nastavena jako NO CYCLE, systém vyhodí chybu, pokud hodnota přesáhne maximum nebo minimum. Defaultní klauzule je nastavena jako NO CYCLE. Nakonec je zde klauzule CACHE, Procházka [6, s. 108] definuje tuto klauzuli jako „Cache size“, kdy po dodání parametru si systém hned vygeneruje hodnoty sekvence a uloží si je do vyrovnávací paměti. Výhoda je v situaci, kdy uživatel ví, kolik hodnot bude potřebovat dopředu. Tímto způsobem zrychlí práci s databází. Defaultní hodnota klauzule je CACHE.

1.2.6. Ostatní

V rámci jednotlivých typů SŘBD se mohou najít další databázové objekty v jejich struktuře. [10, s. 78] Například ve Firebirdu je také možnost definovat vlastní datové typy pod názvem domény. Domény nebo subtypy jsou vytvořeny pomocí standartních datových typů jako je například NUMERIC, ale mají konkrétní rozsah. Jejich hlavní výhoda je, že při určování datových typů u atributů tabulky se nemusí určovat specifický rozsah datového typu, ale celý zápis se může nahradit pouze názvem domény (subtypu).

Ve Firebirdu existuje speciální typ datového objektu nazvaný UDF. UDF je uživatelem definovaná funkce (v překladu user defined function) napsaná v jiném programovacím jazyce, než je SQL, která manipuluje s daty. Tyto funkce jsou kompilovány do společné knihovny, ale musí se nastavit pro každou databázi zvlášť. UDF se používají hlavně v případech, kdy je potřeba provádět určité matematické operace s číselnými proměnnými jako jsou funkce sinus, cosinus apod nebo formátování desetinných hodnot. Nevýhoda je, že přes UDF nelze definovat hodnotu NULL a obtížně se přes ní řídí transakce. Nejlepší způsob, jak je využít, je naprogramovat krátké matematické funkce, které po zavolání uživateli vrátí požadovaný formát číselné hodnoty. [17]

2. Přehled datových modelů databází

Data a jejich vazby mohou mít mezi sebou různé vztahy a mohou být uloženy různými způsoby. Model databáze určuje logickou strukturu databáze a specifikuje jakým způsobem mohou být údaje uloženy, organizovány a manipulovány. Celkově existuje mnoho datových modelů, které se vyvíjely a postupně zdokonalovaly nebo byly nakonec nahrazeny novými datovými modely. V současné době je většina databází založena na relačním modelu dat.

2.1. Hierarchický model dat

Tento datový model vznikl v 60. letech 20. století. Model je souborově orientovaný. Pro představu vzhled diagramu datové struktury je strom nebo případně les – několik stromů složeno dohromady.

Každá položka s hodnotou může být napojena na další položky z obou směrů. Pro každou položku může tedy existovat rodič/potomek. Pro nalezení dat ve struktuře je potřeba navigaci přes potomky nebo zpátky na rodiče. V tomto modelu dat může být složité rušení záznamů. Orientace ve struktuře může být matoucí při nejednoznačném pojmenováním položek. [2]

Samotný strom databáze začíná ve zdrojovém záznamu a postupně se rozšiřuje vytvářením potomků. Nevýhodou hierarchického modelu dat oproti síťovému je, že každý potomek může být propojen jenom s jedním rodičem. Všechna data jsou reprezentována kolekcí záznamů a propojena vztahem rodič-potomek nazvaný vztah. Samotný záznam v grafu se může také označovat jako segment a jeho vztahy hranami. [14, s. 95 a 96]

2.2. Síťový model dat

Síťový model dat se silně podobá hierarchickému. Jeho počátky se datují od 70. let 20. století. Na jednoho potomka struktury může připadnout libovolný počet rodičů na rozdíl od hierarchického modelu. K propojeným záznamům se přistupuje napřímo a bez dodatečného vyhledávání. Každá položka je označena klíčem, přes který se lze dohledat. [2]

Podobně jako v hierarchickém modelu jsou všechna data jsou reprezentována kolekcí záznamů a propojena vztahem rodič-potomek nazvaný vztah. Samotný záznam v grafu se může také označovat jako segment a jeho vztahy hranami. [14, s. 95 a 96]

2.3. Relační model dat

Relační model dat je v současnosti nejběžnějším typem. Jak už název napovídá, základním stavebním kamenem je relace. Lacko [6, s. 47] definuje relace jako: „Relace je podmnožina kartézského součinu nad několika doménami, množina vztahů mezi prvky několika domén, dá se zachytit jako tabulka. Aby se pochopil význam věty je potřeba znát pojem doména, který Lacko [6, s. 47] charakterizuje takto: "Doména je množina hodnot stejného významového typu." Lacko [6, s. 386] dále k relačním databázím uvádí: „Údaje jsou uloženy ve dvojrozměrných tabulkách. Každý řádek v tabulce obsahuje data, která jsou zpravidla obsahem reálného světa, tedy data, která se vztahují k nějakému objektu nebo k jeho části."

Tabulky mohou mít navíc mezi sebou různé typy vztahů, které jsou rozlišeny jejich typem a kardinalitou. Existuje několik typů vztahů mezi dvěma tabulkami, 1:1, 1: N a M:N. Vztah 1:1

představuje vztah, kdy jeden záznam nebo řádek v první tabulce je propojen jenom s jedním záznamem druhé tabulky. Vztah typu 1: N slouží například pro číselníky, kdy jeden záznam v číselníku může být propojen s libovolným počtem záznamů druhé tabulky. Vztah M: N možno v relačním modelu přímo realizovat a většinou se řeší vazební tabulkou, kdy se vztah rozdělí na dva vztahy 1:N. Každá věta by měla mít speciální hodnotu definující unikátnost věty v rámci tabulky – primární klíč. Propojení dvou a více tabulek je dosaženo pomocí cizích klíčů. Cizí klíč je sloupec se stejným názvem a stejným datovým typem jako primární klíč, který odkazuje na hodnotu primárního klíče jiné tabulky.

2.4. Objektově-relační datový model

První objektově-relační databáze pochází z 90. let. Cílem bylo rozšířit stávající relační model přidáním konceptu objektů a interpretovat každou tabulku jako samostatnou třídu s funkcemi jako je dědičnost a polymorfismus. Dalšími výhodami je rozšíření datového modelu o vlastní datové typy a metody. Na rozdíl od objektových databází, kde jsou všechny informace reprezentovány čistě v podobě objektů, objektově-relační databáze fungují jako prostředník. V objektově-orientovaných databází jsou data uložená opět v relačních tabulkách, ale přistupuje se k nim pomocí dotazů daného dotazovacího jazyka. [2]

2.5. Srovnání výhod a nevýhod

V zásadě by se měli hlavně srovnávat modely objektové, objektově-relační a relační. Ostatní modely jsou v podstatě zastaralé a nejsou vhodné pro potřeby moderních databází.

Relační model je nejpoužívanějším modelem dnešní doby. Podporuje ho nejvíce aplikací SŘBD. Relační model zajišťuje většinu potřeb moderních databází a je nepřehlednější. V určitých situací může být nedostatečný z funkčního pohledu. Objektově-orientované databáze rozšiřují relační model dat přidáním nových abstraktních datových typů jako je seznam apod. Přístup k datům může být ale komplikovanější.

Objektové databáze představují úplně jiný systém strukturování databáze. Výhodou je jednoduché začleňování nových objektů a typů objektů přes zdrojový kód. Nevýhodou je zvýšená složitost, která se může způsobit problémy s výkonem a design nových databází může být časově a finančně náročnější. [11]

3. Programy SŘBD

Program systém řízení báze dat (anglicky Database Management Systém – DMBS) je software spravující velké množství dat uložené v databázi podle určité logiky na bázi její struktury. Nejčastější databáze je databáze založená na relačním modelu dat, proto SŘBD relační databáze se někdy označuje jako RDBMS. Ve zkratce jsou programy SŘBD typem softwaru, který spojuje data a samotnou aplikaci. Jeho cílem je možnost data ukládat, modifikovat, mazat a provádět nad nimi dotazy. Pro zjednodušení nebo rozšíření funkcí může použít SŘBD upravenou verzi jazyka SQL. V současné době je k dispozici na trhu celá řada těchto systémů, v praktické části se využije systém Firebird, ale je užitečné srovnat tento databázový server s ostatními. Většina z nich má základní verzi pro menší firmy a uživatele zdarma a jejich nejpokročilejší verze jsou zpoplatněné.

3.1. MySQL

MySQL je druhý světově nejpopulárnější a nejrozšířenější databázový systém založený na relačním modelu dat. Původně byl vytvořen švédskou firmou MySQL AB v roce 1995. [7, s. 11] Nicméně v roce 2008 byl odkoupen firmou Oracle. MySQL je často používám u webových aplikací, jelikož je jednou z důležitých součástí systému tzv. LAMP. LAMP je akronymem všech programů, které jsou součástí tohoto balíčku pro vývoj webových aplikací (L – operační systém Linux, A – webový server Apache, M - MySQL, P - skriptovací jazyky PHP / Perl / Python) [8, s. 14]. V současné době popularita klesá hlavně kvůli odkoupení firmou Oraclem a jejímu částečnému zpoplatnění.

3.2. Oracle

Dalším systémem řízení báze dat je Oracle. Oracle byl vytvořen společností Oracle Corporation a první komerční verze, která implementovala jazyk SQL byla k dispozici na trhu v roce 1982. V roce 1983 byla vydána další verze a nově vznikla relační databáze, která byla poprvé v historii dostupná na osobní počítače. Skoro každá další verze poskytovala průlomové inovace. [6, s. 12] V současné době se jedná o nejpopulárnější systém navzdory tomu, že většina produktů je firmou zpoplatněna.

3.3. Firebird

Firebird vznikl jako evoluce z databázového systému InterBase 6.0, když v roce 2000 majitel programu firma Borland rozhodla vydat systém jako open source. Navzdory očekávání nabyl tento open source projekt velké popularity a nedlouho potom z úprav komunity byla vydána

první verze pod názvem Firebird. [7, s. 12] Firebird je v současné chvíli vyvíjen firmou Firebird Foundation. Firebird využívá procedurální nadstavbu jazyka SQL známou pod názvem PL/SQL. [10, s. 39]

3.4. Microsoft SQL Server

Microsoft SQL Server je v době psaní této práce 3. nejpopulárnějším SŘBD. Jak už název napovídá, autorem tohoto programu je Microsoft, který vstoupil na trh s první verzí programu v roce 1989. [6, s. 11] Původně byl program plně zpoplatněn. Nyní je na trhu verze zdarma pro uživatele a menší firmy pod názvem SQL Server Express. Pro manipulaci s databázemi Microsoft SQL Server, které mají příponu .bak, byl vytvořen speciální rozšíření SQL jazyka pod názvem T-SQL. Tato modifikace například přidává funkce jako je procedurální programování.

3.5. Stručné srovnání vybraných databázových systémů

Při volbě SŘBD si uživatel musí nejdříve určit pro jaké účely program potřebuje. Potřebuje program na studentský projekt nebo na komerční účely?

MySQL v současnosti podporuje nejvíce operačních systémů v porovnání s ostatními. MySQL byl původně vyvinut pro web a má lepší nástroje a podporu pro cloudové zálohování. Struktura a styl je pro běžného uživatele přehledná a snadno se s ní pracuje. Nevýhodou je, že MySQL používá jiný standart jazyku SQL, který je sice pro začátečníka jednodušší, pro zkušenějšího programátora může být nedostatečný. Další problémem je malá schopnost rozšířit kapacitu databáze v případě potřeby. [1]

Oracle je oproti MySQL zaměřen zpravidla pro větší firmy. Databáze mají velkou kapacitu a dokáže pojmout a zpracovat velké množství dat a jdou v případě potřeby rozšířit bez větších problémů. Navíc firma Oracle Corporation poskytuje silnou zákaznickou podporu a obsáhlý manuál. Nevýhodami jsou vysoká cena produktu, vysoké nároky na systémové zdroje počítače a složitější obsluhu databáze. [1]

Firebird je ve srovnání s ostatními programy trochu pozadu. Celý projekt je plně open source, ale není vyvíjen velkým týmem programátorů a zájemců. Jako hlavní výhodu Firebirdu představuje schopnost práce s databázemi typu klient-server a embedded. Další výhodou je, že Firebird je multiplatformní. [7, s. 45] Procházka [10, s. 39] uvádí další výhody jako je silná podpora od řady vývojových nástrojů jímž je například Delphi, které je založené

na programovacím jazyku Object Pascal, ve kterém bude naprogramována aplikace helpdesk v praktické části.

Microsoft SQL Server nabízí různé typy verzí, které jsou specializované od běžných uživatelů po velké korporace. Základní verze pod názvem SQL Server Express je k dispozici zdarma a neustále se vyvíjí. Podobně jako databáze od Oraclu je k dispozici bohatá dokumentace s podporou zkušených programátorů včetně podpory cloudového řešení. Pro začátečníky je ale práce s databázemi náročnější a placené verze programu patří k těm nejdražším. [1]

4. Databázové transakce

Lacko ve své knize [6, s. 209] vysvětluje transakce touto větou: „Pod pojmem transakce se rozumí zapouzdření několika zpravidla obchodních operací do jednoho logického celku.“ Databázová transakce není v principu o moc rozdílná, dále se může označit jako základní jednotka práce. Jednou z hlavních vlastností je, že musí proběhnout v celém svém rozsahu. Po úspěšném zakončení dochází k přechodu z jednoho konzistentního stavu databáze do druhého. Konzistentní stav je stabilní stav databáze, kdy by data měla odpovídat skutečnosti.

Pomocí databázových transakcí kontrolujeme stav databáze. Pokud transakce neproběhne v celé své úplnosti, insert nového záznamu nebo update záznamu se nevykoná. Aplikace helpdesku v praktické části bude kontrolovat přístupová práva uživatelů přes kontrolování transakce, pokud nebude mít uživatel dostatečná práva pro editaci určitých údajů v záznamu, daná transakce neproběhne.

Pomocí transakcí můžeme databázi měnit. Při každé úpravě databáze přes transakce se může databáze dostat do nekonzistentního stavu. Z toho důvodu mají programy SŘBD nastaveny různá bezpečnostní opatření, aby se databáze mohla obnovit do původního stavu.

Například v programu SŘBD MS SQL Server se transakce řídí příkazy BEGIN TRAN, COMMIT, ROLLBACK a SAVE TRAN. Syntaxe a názvy příkazů se mohou trochu lišit v jiných programech SŘBD. BEGIN TRAN (+ název transakce) zahajuje celý proces transakce, který pokud je úspěšně proveden, je zakončen příkazem COMMIT. Příkaz COMMIT není většinou nutné v kódu uvést programátorem a měl by být implicitně zadaný. Nicméně uživatel může sám potvrdit celou transakci v kódu. Příkazem ROLLBACK TRAN s dodaným jménem transakce zruší uživatel všechny změny, které databázový server doposud vykonal a vrátí databázi do původního konzistentního stavu. V některých situací si uživatel nepřeje, aby se vrátil zpátky až na začátek transakce. Jednotlivé transakce se mohou pojmenovat a uložit přes příkaz SAVE

TRAN, tím se nastaví záchytný bod a opět přes příkaz ROLLBACK TRAN s dodaným jménem transakce se může databázový server vrátit zpátky do tohoto bodu. [6, s. 210 a 211]

Velká výhoda transakcí je, že pokud transakce poruší konzistentnost databáze, program SŘBD v tom případě zruší všechny změny aplikované transakcí a vrátí databázi do původního stavu. Nevýhodou transakcí je, že čím větší je samotná transakce, tím více dat je potřeba ukládat dočasně na disk, a to může zpomalit práci s databází. Lacko [6, s. 212] dává toto doporučení ohledně transakcí: „Doporučuje se, aby jedna transakce obsahovala co nejméně operací. Pokud transakce zahrnuje množství operací, zpomaluje to práci databázového serveru, který tak musí odkládat mnoho údajů, aby je v případě potřeby dokázal obnovit."

4.1. Vlastnosti transakce

Základní vlastnosti, která by transakce měla mít můžeme definovat pomocí zkratky ACID. „A" znamená atomicita, tento pojem znamená, že transakce musí proběhnout celá, nesmí být přerušena uživatelem a teprve až později dokončena. „C" je pro konzistenci (z angličtiny consistency) transakce. Tento pojem značí transformaci z jednoho konzistentního stavu databáze do druhého. „I" znamená nezávislost transakce (z angličtiny Isolation). Transakce musí být nezávislé a nemohou být spojeny nebo svázány s ostatními a její dílčí efekty nejsou viditelné. Samotné změny jsou viditelné pouze jenom v okamžiku, kdy je celá transakce úspěšně zakončena jejím potvrzením – po úspěšném konci operace COMMIT. „D" je zkratka pro trvanlivost transakce (opět z angličtiny durability). Při úspěšném zakončení transakce jsou změny potvrzené a trvale uloženy do databáze. Pro zrušení změn po transakci by byl potřeba nový SQL kód v rámci nové transakce.

5. Popis SQL jazyka a jeho rozdělení

SQL je zkratka pro Structured Query Language (strukturovaný dotazovací jazyk), pomocí SQL se může modifikovat strukturu relačních a objektově-relačních databází nebo ji naplňovat daty, a nakonec se na ně dotazovat a filtrovat data, které chceme získat z databáze. Další velká výhoda je možnost definovat přístupová práva uživatelských účtů přistupující k databázi. Různé systémy SŘBD mají své různé varianty SQL – např. jiné specifické datové typy.

Vytváření a editace databáze probíhá v programu SŘBD. Zde si uživatel může aplikovat svůj SQL příkaz přes konzoli. SŘBD programy mají své vlastní uživatelské rozhraní pro co největší zpřehlednění struktury databáze. Do databáze může měnit, mazat a upravovat data běžný uživatel, který pouze pracuje s programem, který je napojený na databázi přes jeho uživatelské rozhraní.

5.1. Úvod do SQL

Historie jazyka SQL začíná v 70. letech, kdy se můžeme poprvé setkat s jazykem Sequel. Cílem bylo poskytnout vývojářům a uživatelům způsob, jak řídit přístup k datům uloženým v databázovém systému.

První SQL databází pro komerční použití byla Oracle vytvořená stejnojmennou firmou v roce 1980 pro počítače VAX. [4]

V té době měl jazyk SQL hodně konkurentů. Až v roce 1987 byl SQL zvolen americkými instituty ANSI a ISO jako standart pro manipulaci s relačními databázemi.

Od té doby byl další vývoj SQL jazyka v plném proudu a v roce 1989 firma ISO vydá update pod názvem „Integrity Enhancement Feature“, který je znám později jako SQL-89. O tři roky později dochází k další úpravě jazyka změnou ISO standartu a upravený jazyk se nazýval SQL-92. [4]

Tato iterace jazyka byla obrovským rozšířením oproti verzi SQL89. Mezi nové funkce patřily nové datové typy jako např. DATE, TIME, TIMESTAMP a další. Další výhodou je přidání nové funkce check. Tato funkce poskytuje lepší kontrolu nad množinou přípustných hodnot pro daný sloupec v tabulce. Vývoj jazyka SQL pokračuje v určité míře do dnešní doby.

Samotný SQL jazyk většinou není pro potřeby moderních aplikací dostačující, firmy jako je Oracle nebo Microsoft vytvořili svoje vlastní úpravy, které přidávají více příkazů jako je if-then-else.

Lacko [6, s. 241] shrnuje samotný SQL jazyk jako jazyk neprocedurální, kde jsou příkazy vykonávány postupně bez možnosti aplikovat základní programátorské prvky jako jsou cykly, podmínky apod. Aby jazyk SQL splňoval požadavky moderního programátora, byly vytvořeny modifikace známe pod názvem procedurální nadstavby jazyka SQL. Každý moderní systém SŘBD aplikuje nějakou formu této nadstavby.

Nejrozšířenější verze jsou T/SQL vyvíjený firmou Microsoft, Oracle vyvíjený firmou Oracle nebo PSQL pro databáze PostOgre nebo Firebird.

5.2. Primární rozdělení SQL jazyka a příklady příkazů

Jazyk SQL je rozdělen do čtyř typů primárního jazyka: DML, DDL, DCL a TCL. Za pomocí těchto příkazů můžeme například definovat strukturu databáze vytvořením tabulky nebo ji změnit. Dalším příkladem užití těchto příkazů je manipulace s daty v tabulce prostřednictvím aktualizací nebo odstranění. V neposlední řadě můžeme i nastavit speciální přístupová práva pro uživatele nebo skupinu uživatelů a kontrolovat průběh transakcí. [13]

5.2.1. DML (Data Manipulation Language)

DML je zkratkou pro „data manipulation language“ a v překladu do češtiny znamená jazyk pro manipulaci dat. Příkazy DML upravují konkrétní data uložená v tabulkách v databázi. Dalšími příkazy jsou příkazy umožňující selekci dat z určité tabulky nebo tabulek podle nastavených podmínek. Další příkladem DML je vytváření nových vět, jejich editace nebo smazání. Tyto SQL příkazy začínají na SELECT (selekce vět), INSERT (vkládání nových vět), UPDATE (editace vět), DELETE (vymazání věty).

5.2.2. DDL (Data Definition Language)

Tento typ příkazů patří do skupiny se zkratkou DDL – ze zvolného překladu do češtiny: jazyk pro definici dat. Příkazy DDL slouží k vytvoření nového databázového objektu přes příkaz začínající na CREATE – například pro vytvoření nové tabulky CREATE TABLE, k úpravě struktury tabulky atd. přes ALTER TABLE nebo vymazání celé tabulky přes příkaz DELETE TABLE. Určité databázové objekty navazují na jiné například pohled nebo procedura mohou být navázány na tabulku. Při pokusu vymazání tabulky by daná procedura navázaná na danou tabulku neměla význam. V takovém případě nás program SŘBD upozorní na existenci těchto závislostí a měl by nám znemožnit upravit strukturu databáze při ignorování varování.

5.2.3. DCL (Data Control Language)

Dalším typem příkazů je DCL – jazyk pro řízení přístupu k datům. Tato kategorie příkazů SQL jazyka se zaměřuje na nastavování přístupových práv uživatelům. Toto je obzvláště užitečné v případě uložení databáze na serveru, kde k databázi přistupuje a manipuluje s ní mnoho uživatelů. Práva lze udělit konkrétnímu uživateli nebo skupině uživatelů. Práva pro přístup k manipulaci databáze apod. lze uživateli udělit přes příkaz GRANT, odebrat přístup přes příkaz REVOKE.

V SŘBD programu Oracle je několik možností, jak přidat práva novému nebo existujícímu účtu. Pokud programátor vytvoří nového uživatele s heslem příkazem „CREATE USER jmeno_uzivatele IDENTIFIED BY heslo_uzivatele“, dostane uživatelský účet bez jakýkoliv přístupových práv. V této situaci může novému účtu přiřadit jednotlivá práva, jaká jsou například vytvoření tabulky přes příkaz GRANT. Problém v této situaci je, že jednotlivých práv může existovat velmi mnoho. Pro praktičnost je lepší vytvořit jednotlivé role přes příkaz „CREATE ROLE jmeno_role“, k nové roli následně přidat všechna práva a na konci přiřadit roli novému uživateli opět přes příkaz „GRANT jmeno_role TO jmeno_uzivatele“. [10, s. 91 a 92]

5.2.4.TCL (Transaction Control Language)

Poslední typem příkazů je TCL – jazyk pro ovládání transakcí. Tyto příkazy spravují a řídí transakce pro zachování integrity dat. V SŘBD programu Oracle to jsou tyto příkazy SET TRANSACTION, COMMIT, ROLLBACK, ROLLBACK TO, SAVEPOINT. [10, s. 42] přes příkaz SET TRANSACTION danou transakci otevřeme, přes příkaz COMMIT se transakce potvrdí a databáze se přesune z jednoho konzistentního stavu do dalšího. Příkazem SAVEPOINT se nastaví záchytný bod a přes příkaz ROLLBACK nebo ROLLBACK TO se všechny změny databáze ruší a databáze se obnoví do původního stavu před spuštěním transakce nebo se vrátí do stavu definovaný záchytným bodem.

5.3. Procedurální nadstavby jazyka SQL

Procedurální nadstavby jazyka SQL obohacují jazyk o důležité funkce, elementy s jejichž pomocí se příkazy SQL nevykonávají pouze sekvenčně, ale v určitém pořadí definovaném kódem. Mezi hlavními výhodami patří přidání nových prvků, jakými jsou například uložené procedury a funkce, programové balíčky, vlastní datové typy a komentáře.

5.3.1. PL/SQL

Procházka [10, s. 123] uvádí PL/SQL jako procedurální nadstavbu jazyka SQL od společnosti Oracle, která vznikla na základě programovacího jazyka Ada. Základním stavební kamenem PL/SQL je blok. Blok dělíme na tři části. Procházka [10, s. 130] ve své knize popisuje PL/SQL jazyk následujícím způsobem: „PL/SQL je jazyk s blokovou strukturou. To znamená, že základní jednotky (procedury, funkce, nepojmenované bloky), které tvoří program, jsou logickými bloky, které mohou obsahovat libovolný počet vnořených bloků.“ Samotný blok se skládá ze tří částí – z části deklarační, z příkazové části a z části pro zpracování výjimek. V deklarační části se deklarují proměnné pomocí příkazu DECLARE, které se inicializují v následující části nebo mohou být předvyplněny uživatelem nebo automaticky v aplikaci, odkud se funkce může spouštět. V části příkazové dochází k provedení samotných příkazů a nová nebo upravená data jsou vkládány do tabulek. Část příkazová je povinná, bez ní by samotná funkce nebo procedura nic nemohla vykonat. Poslední část slouží pro odchytávání chyb a výjimek přes příkaz EXCEPTION. Příkazová část je obalena příkazy BEGIN a END a mohou se v bloku vyskytovat vícekrát podle potřeby.

Nové programovací elementy jako například podmínky IF-THEN-ELSE nebo cykly FOR, WHILE, LOOP větví SQL kód a příkazy nemusí probíhat sekvenčně, ale podle podmínek.

Jednořádkové nebo víceřádkové komentáře mohou být přímo vnořeny do samotného kódu a programátorovi pomáhají lépe se v příkazech zorientovat.

V rámci PL/SQL se upravují i tradiční objektové struktury SQL jazyka jako je například tabulka. PL/SQL tabulka může dynamicky měnit její velikost, kde každý element je prvek v seznamu a za pomocí cyklů se může naplňovat. Každý element může mít svůj vlastní index a narozdíl od klasických seznamů nemusí být indexy svázány posloupností. [6, s. 134]

Dalším přínosem PL/SQL jsou uživatelsky definované balíky. Programátorský balík (package) je objekt, který sdružuje ostatní objekty, jakými jsou například procedury nebo funkce. Skládá se ze dvou částí, první část je samotná deklarace balíčku spolu s voláním všech procedur a ve druhé části (těle) jsou obsaženy názvy jednotlivých procedur spolu s jejich vnitřním kódem. V těle balíku můžeme mít i jiné příkazy, které nabývají a pozbývají platnosti v rámci balíku. [10, s. 138]

V rámci příkazového bloku PL/SQL je zde část pro odchytávání a zpracování chyb. Existují dva typy výjimek – předdefinované a uživatelsky definované. Předdefinovanými výjimkami se ošetřují chyby, které jsou časté například, když při selekci dat najde systém více řádek (výjimka TOO_MANY_ROWS). Uživatelsky definované výjimky jsou výjimky, které si uživatel vytváří sám podle potřeby. Po ošetření chyb systém zruší dočasné změny vyvolané funkcí a vrátí se zpátky. Způsob odchytávání chyb záleží na typu výjimky.

U předdefinovaných výjimek jsou všechny výjimky deklarované v bloku EXCEPTION. Procházka [10, s. 139] vysvětluje že příkaz, který zde systém hledá, je inicializovaný příkazem „WHEN exc THEN“, kde exc je název výjimky a vykoná ho. Pokud program tento blok nenalezne, bude tento příkaz hledat v nadřazených blocích, odkud byla výjimka zavolána. Pro vypsání chyby při nalezení výjimky se použije funkce RAISE_APPLICATION_ERROR, která přijímá dva vstupní parametry – číslo chyby v určitém intervalu a název chybového hlášení.

U výjimek definovaných uživatelem je potřeba deklarovat proměnnou s datovým typem EXCEPTION. Pokud se proměnná zavolá příkazem RAISE, program se přepne do bloku EXCEPTION, kde podle názvu proměnné vypíše chybu.

Pro ladění programu nebo zobrazování textu do konzole aplikace se může použít procedura PUT_LINE z balíku DBMS_OUTPUT. Nejdříve je nutné aktivovat funkci pro výpisy pomocí příkazu „SET SERVEROUT ON SIZE nmb“, kde nmb je celočíselný parametr maximálního rozsahu textového výpisu. Dále je možné pomocí dalšího příkazu definovat rozsah maximální velikost textového vstupu. Tyto příkazy je nutné vložit při každém startu konzole. [6, s. 248]

5.3.2. T/SQL

T/SQL je procedurální nadstavba jazyka SQL od společnosti Microsoft a Sybase. S touto verzí SQL se můžeme setkat v SŘBD programu Microsoft SQL Server. T/SQL obohacuje jazyk SQL přidáním elementů jako jsou podmínky, cykly apod. Syntaxe příkazů se od PL/SQL nebo ostatních nadstaveb liší.

Například v porovnání s PL/SQL musí mít názvy atributů před svým názvem zavináč. Deklarace probíhá opět pomocí klíčového slova „DECLARE @nazev_promenne datovy_typ“, ale přiřazení hodnoty je nutné uvést příkazem „SELECT @nazev_promenne = „hodnota“. Komentáře se v T/SQL vytvářejí stejně jako v PL/SQL, kdy uživatel může zakomentovat jeden řádek nebo i více řádků najednou. Pomocí příkazu PRINT může uživatel zobrazit text do konzole. [6, s. 241 a 243]

V T/SQL se rozlišuje pojem dávka. Dávka je blok nebo skupina příkazů, přes kterou se můžou dělit delší a složitější příkazy na libovolné množství menších částí. Dávky musí databázový server zpracovat jako celek, pokud se v průběhu zpracování vyskytne například syntaktická chyba, systém zruší dosud provedené změny z dávky. Dávky se ohraničují příkazem GO. Pokud uživatel deklaruje a inicializuje proměnné uvnitř dávky, musí počítat s tím, že tyto proměnné jsou platné jenom v rámci dávky, ve které je deklaroval. [6, s. 245 a 246]

Způsob ošetření výjimek definuje Lacko [6, s. 246 a 247] následovně: „Programové bloky napsané v jazyce T-SQL, například uložené procedury nebo spouště, hlásí chyb a výjimky pomocí příkazu RAISEERROR.“ Tento příkaz vypisuje pouze chyby do konzole, může vypisovat i hodnoty proměnných. Samotný chod programu si musí uživatel ošetřit sám. Příkaz RAISERROR potřebuje následující parametry: ID_zprávy, úroveň závažnosti a stav. Podle celočíselné hodnoty ID_zprávy systém rozlišuje typ varovné zprávy. Ten může být definovaný uživatelem nebo implicitní. Pokud je ID_zprávy do padesáti tisíc, systém zobrazí implicitní chybnou zprávu jako například výskyt hodnoty NULL u atributu, která musí mít vyplněnou hodnotu. Pro definování vlastní varovné zprávy musí být ID_zprávy větší než padesát tisíc. Chybná zpráva se vytváří přes uloženou proceduru SP_ADDMESSAGE. Parametr úrovně závažnosti může nabývat hodnot od jedné do pětadvaceti. Čím vyšší hodnota, tím je sledovaná chyba závažnější. Nejvyšší hodnoty jsou nastavitelné pouze administrátorem. Hodnoty do patnácti nezapisují chybu do systémového protokolu. Pro úroveň hodnoty šestnáct se zobrazí varovná zpráva a od hodnoty sedmnáct proběhne zápis chyby do systémového protokolu. [6, s. 246 a 247] Parametr stavu může nabývat celočíselných hodnot od nuly do dvě stě padesáti pěti. Parametr stavu se používá

pro snadné dohledání, z jaké části kódu chyba pochází, pokud se stejná chybná zpráva zobrazí vícekrát. [15]

6. Transfer dat mezi databázemi

V určitých situacích uživateli nestačí jenom jedna databáze. Pokud programátor vytváří novou aplikaci, která spolupracuje s databází, lze předpokládat, že aplikace bude pracovat s velkým množství dat. Například se může jednat o software pro zpracování a inventarizaci zboží na skladě, kde zákazník bude chtít úplné a přehledné zobrazení dat přes tiskové sestavy. Samotné tiskové sestavy se všemi potřebnými daty by byly uložené také v databázi. V této situaci je charakter ukládaných dat docela odlišný od zbytku dat. Všechna data mohou být uložena v jedné databázi nebo se mohou rozdělit do více databází pro zlepšení přehlednosti. Programy SŘBD umožňují uživateli vytvářet a připojit se k více databází najednou. Tyto databáze mohou být uložené na lokálním počítači nebo k nim může uživatel přistupovat na serveru přes síť. Pokud chce uživatel přenášet data mezi svými databázemi má k dispozici několik možností podle využívaného programu SŘBD. Mohou se přenášet samotná data z tabulek nebo i celé databázové objekty. Obtížnost přenosu záleží na množství dat a na umístění databází.

Pro zefektivnění práce s databázemi je vhodné využít speciální vývojová prostředí vyvinutá pro zjednodušení práce s databázovým serverem. Jeden z těchto programů je IB Expert navržený pro prací s databázemi od Firebirdu nebo Microsoft SQL Server Management Studio pro MS SQL Server. Velká výhoda těchto programů je, že poskytují přehledné uživatelské rozhraní, kde uživatel vidí všechny databázové objekty a uživatel může upravovat strukturu nebo obsah databáze bez aplikování SQL příkazů.

6.1. Lokální databáze

Přenos dat mezi databázemi je jednodušší, pokud se jedná o lokální databáze běžící na databázovém serveru uživatele. Způsobů exportu dat je několik. Struktura a obsah databáze se může vyexportovat například jako .csv nebo .xml soubor, který se musí většinou musí pomocí speciálního nástroje upravit na SQL příkaz. Nebo se vygeneruje přímo SQL skript, který se naimportuje do druhé databáze.

V Microsoft SQL Server Management Studiu se SQL skript automaticky generuje po kliknutí na databázi přes pravé tlačítko myši a poté přes volbu „tasks“ a „generate scripts“. Uživateli se otevře okno, ve kterém si navolí, jaké všechny databázové objekty s daty nebo bez dat (nebo jen data samotná) chce vyexportovat. Po vygenerovaný SQL skript se může přímo vložit

do druhé databáze a spustit. Jiná vývojová prostředí založená na jiných SŘBD programech jako je IB Expert nebo Oracle SQL Developer mají stejné nebo podobné funkce pro export celé databáze nebo jeho částí. [16]

Dalším způsobem přenosu dat je přímé napojení do cílové databáze nebo databází a přímý zápis dat přes proceduru nebo např. přes SQL INSERT, UPDATE příkazy apod. V aplikaci programu by na frontendu byly klávesové zkratky pro exportu dat z tabulky do cílové databáze. V tímto by se spustila funkce v backendu aplikace, která by určitá data vložila do cílové databáze.

6.2. Přenos dat přes síť

Přenos dat mezi vzdálenými databázemi, které jsou umístěny na serverech v jiných sítích má své vrstvy dalších komplikací. Pro export dat z jedné databáze do druhé se může použít stejná strategie vyexportování SQL skriptu nebo přímý zápis do databáze přes SQL skript, ale uživatel se nejdřív musí spojit se vzdálenou databází přes IP adresu a číslo portu s dostatečnými právy na úpravu této vzdálené databáze.

Další vrstva komplikace představuje bezpečnost dat. V aplikaci na principu klient-server, která komunikuje se vzdáleným serverem se musí počítat s možností odcizení dat třetí strany. Způsob nastavení připojení ze strany klienta by měl být prostý. Veškerá výstupní komunikace jde na určitý server. Nicméně i zde se může vyskytnout komplikace, pokud je server chráněn bezpečnostními protokoly, které filtrují věrohodné IP adresy. Serverová aplikace může komunikovat s libovolným počtem klientů a jejich databázemi. Proto se zde musí ošetřit věrohodnost klientů a jejich IP adres, aby se citlivá data posílala do správné databáze.

Praktická část

7. Popis vyvíjené aplikace a vývojářských prostředí

Praktická část bude zaměřena na vývoj aplikace pracující s databázovým serverem. Primární cíl aplikace bude pro usnadnění práce, řešení požadavků a komunikace mezi zákazníkem a firmou, která se o něho stará – zkráceně helpdesk. Bude se jednat o samostatný program, který bude navržený pro řešení technických požadavků zákazníků firmami poskytující IT služby nebo programy jako je například program pro účetnictví. Program bude obsahovat dvě verze, jedna verze je verze pro klienty. Druhá verze je verze serverová, serverová verze bude nainstalována podle názvu na server firmy poskytující službu. Serverová verze se bude lišit od zákaznické klientské hlavně tím, že bude obsahovat speciální formuláře a tabulky pro rozlišení jednotlivých firem zákazníků. Program ve své prvotní iteraci bude navržen pro firmu Softbit Software s.r.o. Nicméně bude využitelný i pro jiné firmy, ale před jeho zprovozněním bude muset být nakonfigurován na hostitelský server. Aplikace bude pro správné fungování a zprostředkování komunikace potřebovat stálý přístup k internetovému připojení.

Aplikace bude pracovat s databázemi spravovanými a navrženými v SŘBD programu Firebird. Pro usnadnění práce s databázovým serverem se použije nástroj IB Expert. Aplikace bude naprogramována v jazyce Object Pascal ve vývojářském prostředí Delphi 10.1 od společnosti Borland a Embarcadero Technologies.

7.1. Databázový server Firebird a IB Expert

IB Expert je vývojářské prostředí speciálně navržené pro vývoj a správu databází od SŘBD programu Firebird nebo Interbase. IB Expert má zabudovaný grafický interface pro přehlednou správu a práci s databázovými objekty. GUI je rozdělen na jednotlivé obrazovky a uživatel si jednotlivé moduly může přizpůsobit podle jeho potřeb. Dále program obsahuje a nabízí celou řadu funkcí jako je Debugger pro funkce a triggery. Obsah tabulek nebo celých databází může exportovat nebo importovat ve formátech jako je .csv, sql skript apod. Uživatelské komentáře a poznámky v SQL kódu jsou zde samozřejmostí. Do IB Expertu lze vložit .pdf dokumentace jako samostatný soubor. [19] [20]

7.2. Delphi a Object Pascal

Delphi je vývojové prostředí založené na jazyce Object Pascal. Object Pascal je objektově orientované rozšíření pro jazyk Pascal od společnosti Borland a Embarcadero Technologies. Jazyk Object Pascal je dále upravený v Delphi a jeho verze v tomto vývojovém prostředí se někdy

jednoduše označuje jako „the Delphi language“. Object Pascal používá tři základní koncepty objektového programování – zapouzdření, dědičnost a polymorfismus. [18, s. 63]

Zapouzdření je proces, kde jsou v rámci tříd všechny jeho atributy a proměnné deklarovány se soukromým přístupem. Aby se k těmto atributům mohlo přistupovat mimo třídu, ze které pochází, je zapotřebí vytvořit veřejné metody, pomocí kterých je programátor schopen jejich hodnoty měnit nebo k nim přistupovat v konkrétní instanci objektu.

Dědičnost je koncept, při kterém definujeme termíny jako je rodič a potomek. Rodič může mít teoreticky nekonečně mnoho potomků. Nový potomek implicitně dědí všechny atributy a metody svého rodiče. Nové atributy a metody deklarované a inicializované v rámci potomka se zpětně na rodiče nedědí.

V Object Pascalu je možné, aby v rámci více tříd existovaly metody se stejným názvem. Tyto metody mohou mít jiné tělo kódu. Při volání metody z jiné třídy však musíme specifikovat, ze které třídy pochází. Tento jev se nazývá polymorfismus.

8. Návrh databáze a jeho struktura

Aplikace bude pracovat se dvěma či více databáze databázemi. Aplikace helpdesku bude mít verzi, jak pro zákazníka, tak pro klienta. Na základě toho budou používat trochu odlišené databáze. Databáze budou mít dva následující typy.

První typ databáze bude obsahovat všechna data ohledně firmy a veškerou komunikaci se zákazníkem nebo se serverem zpracovatelské firmy. Pro správnou identifikaci databáze pro komunikaci se při spuštění vygeneruje speciální identifikátor, podle kterého se pozná databáze a o jakého zákazníka se jedná, identifikátor bude uložen v proměnné GUID. GUID databáze je universální unikátní identifikátor (přeloženo z angličtiny: global unique identifier) jedná se o speciální typ ID, má pevně nastavenou velikost a je složen z kombinací čísel a jiných znaků. Možných kombinací všech znaků je tak velký, že šance generování stejného ID u více databází je skoro nulová. Tento datový model je obsažen v příloze jako **Obr 17: datový model klientské aplikace.**Obr 16: use case diagram Tento typ databáze pro serverovou aplikaci bude navíc obsahovat extra tabulky obsahující informace o klientských firmách, jejich IP adresy atd. Pro upřesnění je datový model této databáze zde - **Obr 18: datový model serverové aplikace.**

Druhý typ databáze bude hlavně databází pro uživatelské účty. Datový model části pro řízení uživatelů je k dispozici v příloze - **Obr 19: datový model databáze uživatelů**. Jak již bylo řečeno, zákaznická firma může zpracovávat více firem nebo středisek najednou. Každé středisko představuje jednu databázi. Nicméně všechny tyto databáze budou propojené

na společnou databázi, ve které budou definovány všechny informace ohledně uživatelů a informace, do kterých databází mají přístup. Jeden uživatel může mít přístup do více databází.

Dále obsahuje tento typ databáze potřebné tabulky, které jsou použity pro šablony formulářů a jsou nutné pří spuštění programu, autor této databázové struktury je pan Ing. Radim Holý. [20]

8.1. Role a práva uživatelů

Výsledná aplikace bude obsahovat v základu tři různé role pro manipulaci s programem a databází. Tyto role budou mít striktně nastavená pravidla přístupu a manipulace s jednotlivými tabulkami. Velká výhoda rolí je, že všechna práva se nastaví pro tyto role a poté stačí tyto role přiřadit jednotlivým uživatelům. Ve Firebirdu se práva uživatelů nebo práva rolí musí nastavit zvlášť pro každou databázi.

Ve Firebirdu je implicitně definována administrátorská skupina uživatelů pod název RDB$ADMIN. Pokud uživatel patří do této skupiny, má absolutní přístup ke všem tabulkám včetně schopnosti vytvářet nové uživatele. Při nainstalování Firebirdu je automaticky nadefinován takovýto typ účtu pod názvem SYSDBA s defaultním heslem „masterkey“. Toto heslo je univerzální a pro současný běh aplikace je doporučeno ho neměnit. V programu je tento typ uživatele určen pro pozici jednatele společnosti apod. a v programu bude mít unikátní oprávnění vytvářet nové účty.

Prvním typem role je „TECHNIK“. Role technika bude přístupná jenom v serverové aplikaci, kde uživatel s touto rolí bude mít plný přístup k úpravám všech údajů a přístup do všech formulářů a tabulek. Každý tento technik bude také speciálně definován v číselníku techniků, který bude primárně vyplněn na serverové databázi a při jeho změně se číselník na straně klienta doplní o změny. Z této logiky vyplývá, že každá položka z číselníku by měla reprezentovat jeden účet technika. Účet technika bude mít schopnost přidělovat role technika ostatním uživatelům.

Druhým typem role je „KLIENTADMINI“. Tento typ role bude pouze na klientské části aplikace. Tento typ role bude mít možnost skorou veškerou manipulaci s daty včetně přiřazení rolí „KLIENTADMINI“ a „KLIENTUZIVATEL“ pro další uživatele. Na rozdíl od účtu s rolí technik nebude mít tato role přístup k úpravám některých údajů týkající se fakturačních informací – např. cena sazby práce, počet hodin strávených při zpracování požadavku apod.

Poslední typem role je „KLIENTUZIVATEL“. Uživatel s touto přidělenou rolí bude mít minimální počet práv a jeho manipulace s programem se omezí pouze na vyplnění požadavků a komunikace s technikem přes posílání zpráv.

Pouze účet, který přiřadil role druhému účtu, je schopen ho znovu odebrat, proto je vhodné, aby veškerou manipulaci s novými uživateli prováděl právě účet SYSDBA.

8.2. Představení aplikace

Aplikace bude ve své finální verzi nainstalována na serveru poskytovatele IT služeb a serveru zákazníka. Schéma provozu aplikace je zobrazena v příloze - **Obr 16: use case diagram**. Na serveru poskytovatele IT služeb bude nainstalovaná serverová verze a bude přijímat zprávy a požadavky a posílat zprávy zákazníkům. Zpravidla bude mít zákazník svoji vlastní databázi. Někteří zákazníci mohou například provozovat finanční služby nebo jiné pro více firem a budou chtít v programu rozlišit o jakou databázi se jedná. Z toho důvodu bude každá databáze rozeznávána přes speciální GUID identifikátor a program bude sledovat pořadí požadavků v rámci jednotlivých databází zákazníka. Databáze zákazníka budou uloženy lokálně na jeho zařízení nebo na serveru stejně tak jako databáze provozovatele služby helpdesku. Při práci s aplikací se nastaví připojení do vzdálené databáze a po úspěšném připojení dojde k přenosu dat.

Aplikace bude rozdělena do následujících základních modulů/formulářů: přihlašovací obrazovka, připojení do lokální databáze, nastavení firem a uživatelů, formulář helpdesku a jeho číselníky a nápověda. Úvodní přihlášení do programu proběhne pomocí implicitního účtu Firebirdu SYSDBA. Přes tento účet se všemi právy se vytvoří ve formuláři aplikace ostatní účty a dodá se jim přístupy do jednotlivých databází. Nastavení uživatelů a přístupů se může provést přes stejný administrátorský účet na datovém serveru, nicméně pro aplikaci jsou vyvinuté formuláře, který tento proces zjednodušují. Po kliknutí na nápovědu se zobrazí informace o programu a nápověda k programu.

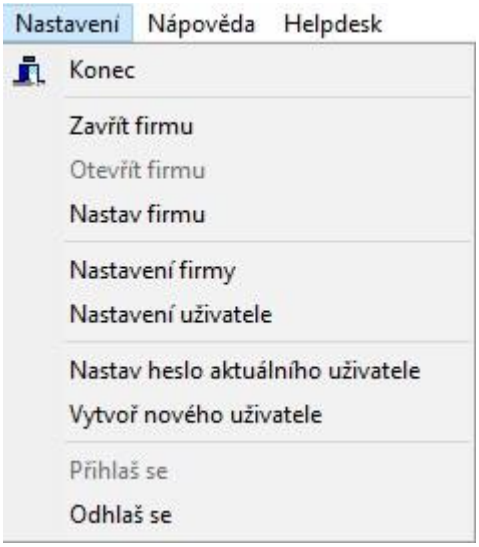
Modul helpdesk bude hlavní částí programu, kde po rozkliknutí nabídky bude mít uživatel v nabídce zobrazeny všechny číselníky k helpdesku a samotný formulář helpdesk. Formulář helpdesk bude rozdělen do více oken, v okně požadavku si zákazník vyplní nový požadavek přes formulář. Po vyplnění požadavku se komunikace s technikem přenese do jiné části formuláře. Jakmile se požadavek zadá a potvrdí, systém vygeneruje automat informační e-mail, který zašle na centrální e-mailovou schránku majitele softwaru, tento e-mail bude hlavně obsahovat informace ohledně datumu a času vytvoření požadavku a o jakou firmu se jedná. Zákazník si může zvolit v hlavičce požadavku, kterého technika si chce vyžádat. Pokud není žádný technik zvolen, tak se implicitně do požadavku vyplní „centrála“. Z centrály se manuálně přidělí pracovník pro požadavek podle toho, jak který pracovník má čas a jestli se o firmu primárně stará. Všechny e-maily se budou brát z číselníků techniků v databázi. Nakonec se technik připojí ke své serverové

aplikaci a začne pracovat na požadavku, po splnění se budou výsledky spolu s počty odpracovaných hodin nebo i úpravy původního požadavku atd. evidovat v tabulce pro komunikaci helpdesku. Pro všechny bude k dispozici statistická tabulka obsahující vypočítané údaje ohledně počtů odpracovaných hodin technikem na požadavku.

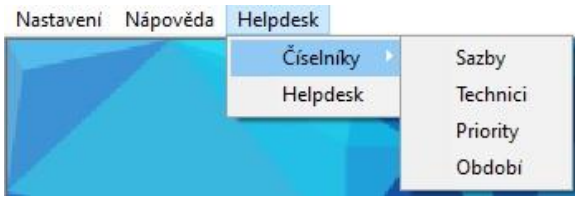
9. Struktura aplikace a formulářů

Aplikace je navržena jako modulární. Základní částí aplikace je formulář. Uživatel může otevřít libovolný počet formulářů a přizpůsobit si jejich velikost nebo pozici na obrazovce podle jeho přání. Přes lištu se dají spouštět i jednoduché příkazy jako je zavření databáze současné firmy nebo obrazovku odhlášení. Po kliknutí na tyto možnosti se zobrazí jednoduchá obrazovka, kde se uživatel rozhodne provést akci, či nikoliv. Formuláře se spouštějí po rozkliknutí lišty a výběru jednotlivých formulářů jako je na Obr. 1,2 a 3.

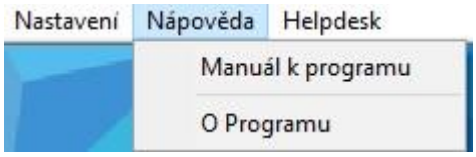
Obr 1: Lišta nastavení



Obr 2: Lišta helpdesku



Obr 3: Lišta nápovědy



Většina formulářů se vytváří přes jednotnou šablonu, která byla původně vytvořena panem Ing. Radimem Holým a tento program využívá její modifikace (k dispozici v příloze - **Obr 38: šablona formuláře**) a sdílí stejný způsob ovládání a základní grafiku. Formulář většinou využívá jeden pohled pro zobrazení dat. Formuláře se skládají z několika částí. [21]

První část formuláře je v příloze barevně označená zeleně. Tato část se v každém formuláři opakuje a obsahuje tlačítka poskytující základní funkce ovládání. Tlačítko „Ok“ potvrdí změny dat a zavře formulář. Tlačítko „Konec“ zavře formulář. Pokud je ve formuláři příliš mnoho dat, využije se tlačítko pro vyhledání konkrétního záznamu v tabulce. Tlačítkem „Filtr“ se zobrazí jen ta data v tabulce, která splňují podmínku, kterou si uživatel musí sám nastavit. Tlačítka v dolním řádku jsou pro manipulaci daty v tabulce a provádí operace jako vložení nového záznamu, odstranění stávajícího záznamu, potvrzení transakce, zrušení transakce, obnovení dat, povolení nebo zrušení možnosti zápisu do tabulky atd…

Druhá část (označena žlutě) je část pro vložení vlastní grafiky na základě napojených dat. Tato část je různá pro každý formulář a poskytuje uživatelsky přívětivý způsob pro vkládání dat do kolonek a update současného záznamu. Dále se může tato část rozšiřovat dělením na více stránek. Každý modul má svoji grafickou část unikátní, při změně modulu dojde k přepnutí na jiný pohled a pro ni vytvořený vzhled. Formuláře helpdesku obsahují kolonky dvou barev. Bílá barva značí, že uživatel je schopen do kolonky zapsat hodnotu, šedivá znamená, že údaj je buď vypočítaný nebo současný uživatel nemá právo do ní nic zapsat.

Třetí část (označena červeně) je prostor pro zobrazení dat v podobě tabulky. Počet sloupců je určený počtem položek v pohledu, který je napojen na modul. Počet řádků je určený počtem záznamů, které jsou obsaženy v pohledu. Každý modul (označen modře) je napojen na svůj vlastní pohled. Implicitně jsou zobrazena všechna data, nicméně jak již bylo nastíněno, uživatel si v této části může nechat data filtrovat.

9.1. Přihlašovací obrazovka

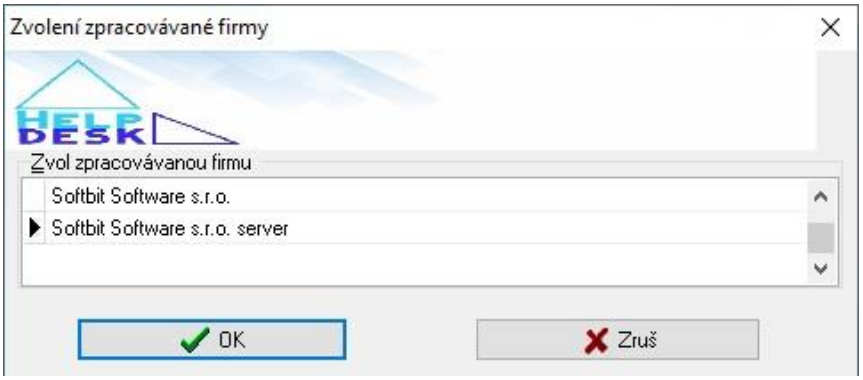
Při jakékoliv práci s aplikací je potřeba, aby byl uživatel pořád přihlášen. Přihlašovací proces je jednoduchý a přihlašovací údaje se musí shodovat s údaji pro přihlášení k databázi přes databázový server Firebird. Každý uživatel musí mít své přihlašovací jméno a heslo. Velikost

a délka hesla je čistě libovolná. Pro zrychlení přihlašovacího procesu si systém pamatuje jméno posledního přihlášeného uživatele, kterého uchovává v .ini souboru. Pohled na přihlašovací obrazovku je k viditelný v příloze - **Obr 25: formulář pro přihlášení.**

9.2. Zvolení firmy

Všichni uživatelé jsou definování v uživatelské databázi, kde jsou přes cizí klíč propojení na tabulku firem. Z tabulky „FIRMYUZIVATELE“ se vezmou všechny firmy, které připadají uživateli a zobrazí se v tomto formuláři. Po vybrání databáze se aktivuje hlavní část programu spolu se všemi funkcionalitami. Formulář výběru firmy je viditelný na Obr 4.

Obr 4: obrazovka zvolení databáze firmy



9.3. Formuláře číselníků

První typem formulářů jsou číselníky. Číselníky jsou speciální typem tabulek, které jsou propojeny v pohledu přes cizí klíč k libovolnému počtu tabulek ve vztahu 1: M. Většina číselníku bude již předvyplněna daty. Pokud bude nutné číselníky rozšířit, stane se tak pouze v serverové databázi, odkud se data zaktualizují do klientské databáze. Podstata těchto speciálních formulářů je usnadnit práci při vyplnění záznamů rozsáhlejších formulářů při opakování určitých dat.

9.3.1. Číselník období

Prvním formulářem číselníků je číselník období. Náhled formuláře období je přístupný v příloze - **Obr 39: číselník období.** Podstata tohoto formuláře je pouze pro funkci výběru období v rámci hlavního formuláře helpdesku.

Formulář obsahuje následující údaje:

- měsíc
- rok

- datum – datum vytvoření záznamu

9.3.2. Číselník techniků

Formulář techniků je zobrazen v příloze - **Obr 22: číselník techniků.** Data se zde budou uživatelsky definovat v databázi serverové aplikace. Při spuštění klientské aplikace se seznam všech techniků zaktualizuje. Proto z bezpečnostních důvodů budou moci jenom uživatelé v serverové aplikaci možnost tuto tabulku měnit. Tabulka číselníku techniků je napojena přes cizí klíč na tabulku požadavků ve formuláři helpdesku.

Formulář obsahuje následující údaje:

- id_c_technik – skrytý údaj – primární klíč tabulky
- jméno technika
- příjmení technika
- pracovní pozice
- email – všechny e-maily musí být unikátní a podle nich se rozeznává konkrétní pracovník
- telefon

9.3.3. Číselník hodinových sazeb

Formulář sazeb je k dispozici v příloze - **Obr 20: číselník hodinových sazeb.** Přístup do tohoto formuláře má každý uživatel, nicméně pouze uživatelé v serverové aplikaci ho můžou upravovat. Data tabulky tohoto formuláře jsou opět synchronizována se serverovou databází.

Formulář obsahuje následující údaje:

- id_c_hod_sazby – skrytý údaj – primární klíč tabulky
- popisPrace – vysvětlení popisu práce
- sazba – cena sazby v korunách

9.3.4. Číselník priorit

Priorita požadavku v helpdesku označuje urgentnost splnění. Zákazník zpravidla bude tuto hodnotu vybírat z číselníku. Priorita pomáhá centrále a technikovi zařídit se podle naléhavosti požadavků. Příklad formuláře v aplikaci je zde - **Obr 21: číselník priorit**.

Formulář obsahuje následující údaje:

- id_priorita – primární klíč
- priorita – v základu je hodnota A – do 24 hodin a příplatek, B – do 3 dní, C – dle domluvy.
- popis_priority – vysvětlení označení priority

9.3.5. Číselník firem

Formulář číselníku firem (**Obr 36: číselník firem pro serverovou aplikaci)** se vyskytuje pouze v serverové aplikaci. Žádný z uživatelských typů nemá právo přidávat nové záznamy nebo je editovat. Při ustanovení připojení se serverovou aplikací se údaje z nastavení firmy klientské databáze ověří podle IP adresy a GUID databáze, pokud jsou údaje databáze jiné nebo připojení k databázi úplně chybí, údaje se do tohoto číselníku zapíší a použijí se při odpovědi technika.

Formulář obsahuje následující údaje:

- Id_c_firmy – primární klíč
- GUID databáze – unikátní identifikátor na rozeznání databáze. Je přidělen automaticky při prvotním přihlášení do klientské databáze.
- Název firmy
- Název firmy rozšířený
- Ulice
- Město/obec
- PSČ
- IČ
- DIČ
- Telefon
- Č. popisné
- Č. orientační
- IP adresa – spolu s GUID se použije pro rozeznání databáze a jeho klienta
- Název databáze – název databáze může být úplně stejný ve více databázích zákazníků, nicméně je klíčový při nastavení spojení. Pomocí GUID a IP adresy se ověří správnost databáze při zprovozňování komunikace.

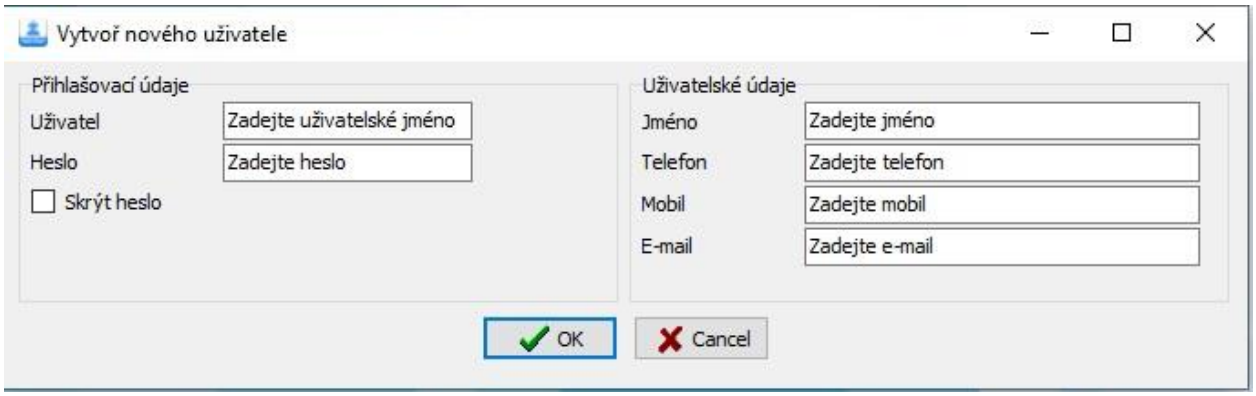
9.4. Hlavní formuláře

9.4.1. Správa uživatelů a jejich práv

Jak již bylo naznačeno, každý uživatel zaregistrovaný v aplikaci je rovněž pod stejným jménem a heslem zaregistrovaný na databázovém serveru. Jeho údaje jsou rovněž uloženy v tabulce uživatelů, kam má přístup každý uživatel kromě uživatelů s rolí „KlientUzivatel“, formulář přehledu uživatelů je k dispozici zde - **Obr 26: přehled uživatelů**. V tabulce uživatelů jsou definovány další volitelné údaje jako je jméno uživatele, jeho mobil, email, atd…

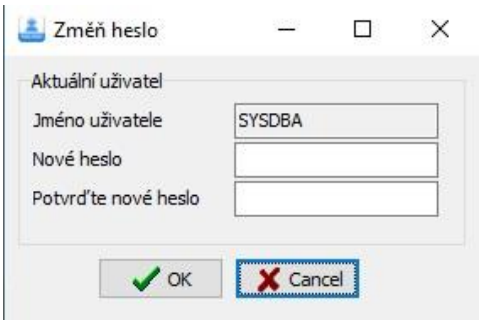
Nový uživatel se může vytvořit přes databázový server. Nicméně se automaticky nedoplní do tabulky uživatelů. Proto se vytvořil formulář v programu (viz Obr 5), který obsahuje procedury, po kterých se uživatel doplní do všech potřebných tabulek. Povinné údaje jsou uživatel, jméno a heslo, které může mít libovolnou délku. Zbytek údajů slouží pro lepší identifikaci a personalizaci uživatele.

Obr 5: *Formulář na vytvoření nového uživatele*



Formulář na změnu hesla (viz Obr 6) funguje pouze pro přihlášeného uživatele. Pro změnu hesla je vyžadováno pouze vyplnění a potvrzení nového hesla. Vyplnění původního hesla není potřeba, jelikož pro zpřístupnění formuláře je nutné se do aplikace nejdříve přihlásit.

Obr 6: *formulář na změnu hesla*



Čistě nový uživatel nemá přístup do žádných firem a žádná práva. Jakmile se vytvoří nový uživatel, může mu účet SYSDBA (globální administrátorský účet) nebo účty s rolí „KlientAdmini“ a vyšší možnost udělit přístup do databáze s určitou rolí. Role se liší na základě typu aplikace. Formulář na úpravu přístupu do databází je v této příloze - **Obr 27: nastavení přístupu uživatelů do firem.** V tomto formuláři je možné přidělit role do jednotlivých firem.

9.4.2. Správa firem a jejich databází

Pro přidělení firem jednotlivým uživatelům je nejdříve potřeba zaregistrovat databázi firmy do programu. Potřebné údaje se mohou přidat přímo do tabulky „Firmy“ na databázovém serveru (viz **Obr 19: datový model databáze uživatelů)** nebo přidat v programu ve formuláři: **Obr 24: nastavení databází firem a jejich cesty.** Tabulka firem se skládá pouze ze tří údajů.

- idfirmy – primární klíč
- jméno – název nebo označení firmy
- cesta – umístění databáze v počítači. Každá databáze je označena aliasem (zkrácený název/označení) spolu s cestou k databázi v konfiguračním souboru Firebirdu. Program helpdesku umí přijímat i aliasy databází, pokud jsou ovšem definovány v konfiguračním souboru Firebirdu.

Informace o firmě jsou definovány v tabulce „Nastavit“. Po spuštění programu a zvolení firmy se informace o aktuální firmě zobrazí ve formuláři **Obr 23: formulář nastavení firmy.**

Formulář obsahuje následující údaje:

- Idnastavit – primární klíč tabulky
- GUID databáze – unikátní identifikátor na rozeznání databáze. Je přidělen automaticky při prvotním přihlášení do klientské databáze
- Název firmy
- Název firmy rozšířený
- Ulice
- Město/obec
- PSČ
- IČ
- DIČ
- Telefon
- Email – použije se jako výchozí email pro kontaktování
- Logo firmy
- Počet nových zpráv – skrytý údaj, je použit pro funkci aktualizace příchozích zpráv

9.4.3. Formulář helpdesk

Formulář helpdesk je hlavní a nejdůležitější formulář celé aplikace. Většina ostatních formulářů a tabulek mají podpůrný smysl. Tento formulář se skládá ze tří modulů, kde každý modul je napojen na svůj pohled a obsahuje mnoho funkcí a procedur. Formulář helpdesku se liší

v klientské a serverové verzi. Na serveru je formulář rozšířen o tabulku firem, podle které lze sledovat z jaké firmy přišel požadavek včetně jiných funkcí. Veškerá komunikace z klienta jde na server firmy. Po spuštění formuláře na straně klienta proběhne aktualizace číselníků. Poté uživatele vyzve obrazovka k vybrání období (viz Obr 7), které vyfiltruje požadavky podle data vytvoření. Po výběru období se spustí hlavní formulář helpdesku s tímto nastaveným filtrováním dat. Tato funkce je obzvláště užitečná u serverové aplikace, kde v tabulce požadavků může být obrovský počet záznamů a uživatel si může vyfiltrovat hned na začátku, které jsou pro něho relevantní. Hned po spuštění obdrží uživatel informace ohledně nových požadavků nebo zpráv, které se zatím uložily do databáze.

Obr 7: Výběr období

9.4.3.1 Modul požadavků

Prvním modulem ve formuláři je modul požadavků. Ten je napojen na tabulku, na kterou jsou napojeny tabulky pro komunikaci a seznam prací podle techniků. Typ vazby je 1:M jako u číselníků. K jednomu požadavku je tedy mít libovolné množství zpráv včetně libovolného počtu techniků. Při vytvoření nového záznamu požadavku se vytvoří první záznam v komunikaci včetně všech řádků pro seznam prací všech techniků, které jsou definované z číselníku. Modul se skládá z více stránek a náhled je dostupný v příloze této práce - **Obr 28: helpdesk požadavek**, **Obr 29: helpdesk popis požadavku**, **Obr 30: helpdesk požadavek příloha** a **Obr 37: helpdesk** . Serverová aplikace je obohacena o další stránku, která obsahuje informace o firmě, která poslala požadavek. Pouze klient má schopnost vytvářet nové požadavky, technici pouze komunikují v modulu komunikace.

Modul obsahuje následující údaje:

- Pořadové číslo požadavku – toto číslo je celkové pořadí požadavku v databázi. Tato hodnota se doplňuje automaticky přes proceduru nebo spouštěč, aby se čísla požadavku přepočítala v případě smazání řádku.
- autor požadavku – do této položky se automaticky doplňuje jméno aktuálního přihlášeného uživatele
- Datum a čas uložení na požadavku– datum se automaticky doplní přes funkci, kde si vezme aktuální čas a uloží ho do této proměnné. Přes hodnotu této proměnné se pomocí funkce filtruje seznam požadavků.
- Datum a čas odeslání požadavku – po úspěšném odeslání požadavku na server se do požadavku zapíše aktuální datum odeslání.
- údaj odeslat A/N – po odeslání požadavku se tento údaj definitivně nastaví na A. Veškerá další komunikace probíhá v tabulce zpráv.
- datum vyřešení požadavku – po vyřešení požadavku má technik povinnost zapsat datum vyřešení požadavku. Záznam se graficky v tabulce změní na zelenou barvu.
- vyřešeno – A/N – Údaj kontroly vyřešení požadavku (implicitní hodnota je N), po vyřešení se změní na A.

Priority urgentnosti jsou A, B, C, priorita A znamená do 24 hodin a příplatek, B – do 3 dní, C – dle domluvy. Přes číselník se bude moci definovat volitelná priorita.

- id priorit – cizí klíč pro napojení na číselník priorit
- hodnota priority – A, B, C, … z číselníku
- text priority – popis priority
- id technik – cizí klíč pro napojení na číselník techniků
- jméno technika – převzatý údaj z číselníku techniků
- email technika – převzatý údaj z číselníku techniků
- příjmení technika – převzatý údaj z číselníku techniků
- pracovní pozice technika – převzatý údaj z číselníku techniků
- telefon technika – převzatý údaj z číselníku techniků

Statistické informace – následující hodnoty proměnných se budou dopočítávat ze seznamu prací jednotlivých techniků. Jedná se o součet všech hodin a celková cena za vyřízení požadavku, který je zákazník povinen uhradit. Všechny hodnoty jsou dopočítány přes spouštěče.

- celkem hodiny požadavků fakturačních
- celkem hodiny potvrzených zákazníkem
- celkem hodin reklamačních za vybrané období
- celkový počet hodin za všechny položky.

- celkem částka k vyúčtování
- zkrácený popis požadavku – zkrácený popis
- popis požadavku – rozsáhlejší popis a vysvětlení požadavku
- Obrazová příloha – zde uživatel vloží ofocenou obrazovku nebo jiný obrázek, který shrnuje požadavek.

Záložka firmy je implementována pouze v serverové aplikaci a podle ní technik ví, od které firmy pochází požadavek. Přes cizí klíč „Id_c_firmy" je spojena na číselník firem odkud si přebírá všechna data viz **Obr 36: číselník firem pro serverovou aplikaci.**

9.4.3.2 Modul komunikace

Všechny záznamy v komunikaci jsou přes cizí klíč provázány s požadavkem. Po přepnutí na jiný požadavek ve formuláři se zobrazí jen provázané zprávy. Zprávy mohou pocházet od klienta nebo technika a mohou mít různé účely. Může se jednat o rozšíření požadavku zákazníkem, fakturační informace vyplněné technikem nebo pouze o vnitřní komunikaci ve firmě. Modul se opět skládá z více záložek/stránek a je obsažen v příloze - **Obr 31: helpdesk komunikace - zpráva**, Obr **32: helpdesk – komunikace, popis zprávy**, **Obr 33: helpdesk – komunikace, příloha**, **Obr 34: helpdesk komunikace – fakturační informace o požadavku.**

Modul obsahuje následující údaje:

- id položky komunikace – primární klíč
- id požadavku – napojení cizího klíče na tabulku požadavků – každý požadavek může mít libovolný počet záznamů komunikace.
- Jméno autora zprávy – určeno podle toho, kdo je přihlášen (technik nebo uživatel) více techniků nebo uživatelů u klientské aplikace může zpracovávat/komunikovat.
- Pořadí záznamu – pořadí zprávy bude vyplněno přes spouštěč.
- datum a čas uložení zprávy – doplněno spouštěčem po uložení záznamu
- datum a čas odeslání zprávy – doplněno spouštěčem po odeslání záznamu
- Odesláno A/N – zpráva se nemusí odeslat, pokud se neodešle, bude viditelná pouze v lokální databázi. Po odeslání na server se tato hodnota automaticky změní na „A".
- zkrácený popis zprávy
- volný text – rozsáhlejší popis zprávy
- Obrazová příloha – zde uživatel vloží ofocenou obrazovku nebo jiný obrázek.

Následující atributy bude schopen editovat pouze přihlášený technik. Tyto vyplněné atributy se přes procedury a spouštěče budou dopočítávat do hlavičky. Technik se bude opět

přebírat z číselníku techniků. Pokud technik splní požadavek, doplní do zprávy délku, sazbu práce a ostatní A/N parametry, podle kterých se přes spouštěče dopočítá celková cena.

- id technik – cizí klíč pro napojení na číselník techniků
- jméno technika – převzatý údaj z číselníku techniků
- email technika – převzatý údaj z číselníku techniků
- příjmení technika – převzatý údaj z číselníku techniků
- pracovní pozice technika – převzatý údaj z číselníku techniků
- telefon technika – převzatý údaj z číselníku techniků
- reklamace A/N
- placená služba A/N
- id sazby – cizí klíč napojený na číselník sazeb
- typ sazby
- hodnota sazby – cena sazby za hodinu práce
- předpokládaný čas práce
- datum a čas začátku řešení požadavku
- datum a čas ukončení požadavku
- zákazník souhlasí s cenou a datum a čas A/N

9.4.3.3 Modul přehledu prací podle techniků

Tento modul slouží pro zpřehlednění seznamu prací od jednotlivých techniků. Přes cizí klíč je tabulka napojena na tabulku požadavků. Vztah je opět 1: M. K jednomu požadavku je přiřazen typicky hlavní technik, ale pokud je požadavek komplexnější, může si vyžádat součinnou spolupráci více techniků. Každý záznam představuje jednoho technika z číselníku. Náhled formuláře je na obrázku: **Obr 35: helpdesk – přehled prací.**

Modul obsahuje následující údaje:

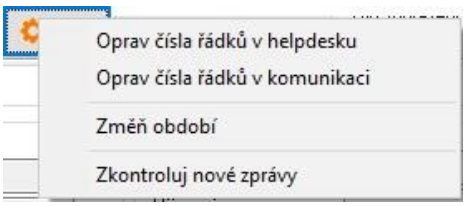
- id technik – cizí klíč pro napojení na číselník techniků
- jméno technika – převzatý údaj z číselníku techniků
- email technika – převzatý údaj z číselníku techniků
- příjmení technika – převzatý údaj z číselníku techniků
- pracovní pozice technika – převzatý údaj z číselníku techniků
- telefon technika – převzatý údaj z číselníku techniků
- celkem hodiny požadavků fakturačních
- celkem hodiny potvrzených zákazníkem

- celkem hodin reklamačních za vybrané období
- celkový počet hodin za všechny položky.
- celkem částka k vyúčtování

9.4.3.4 Implementace systému komunikace a jiné funkce

Formulář helpdesku obsahuje mnoho vypočítaných údajů, které uživatel není schopen měnit v rámci urychlení prací s programem. Nicméně systém není dokonalý a při manipulaci s programem mohou vzniknout logické nekonzistentnosti nebo nepřehlednosti. Z toho důvodu formulář helpdesk obsahuje tlačítko „Akce“, po kliknutí na toto tlačítko se uživateli zobrazí seznam několika funkcí (Obr 8). Funkce na opravu čísel řádku spustí proceduru, která zkontroluje všechny záznamy a očísluje je vzestupně od čísla jedna. Při spuštění formuláře helpdesku má uživatel k dispozici výběr období, podle kterého se data vyfiltrují podle datumu uložení záznamu. Pokud není uživatel se svým výběrem spokojen, může si funkci znovu spustit z nabídky akcí. Data se mohou dále efektivně a rychle filtrovat podle splněných nebo nesplněných požadavků nebo standartním způsobem, který je přístupný ve všech formulářích. Další funkcí je funkce zkontrolovaní nových zpráv. Pokud uživatel obdrží novou zprávu, je uloží se do databáze, ale hned se uživatelovi nezobrazí, po kliknutí na tuto funkci uživatel zjistí, kolik nových zpráv obdržel a poté se formulář zaktualizuje. Stejná funkce se spustí při spuštění samotného formuláře.

Obr 8: Přehled funkcí

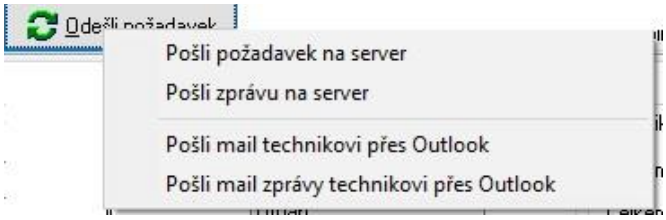


Samotná aplikace helpdesku by byla k ničemu, pokud by neměla naprogramovaný efektivní způsob komunikace. Aby uživatel poslal zprávu nebo požadavek, který chce, musí se v tabulce formuláře kliknout na odpovídající záznam, který po označení bude mít žluté pozadí. Po stisknutí tlačítka „Odešli požadavek“ se otevře seznam (Obr. 9) s funkcemi pro poslání zpráv a požadavků. Základním principem je, že zpráva, která se odeslala, se nemůže poslat znovu. Pokud dojde k výrazné změně požadavku, musí se popsat v modulu zpráv, které se pak odešlou na cílový server. Změněné statistické údaje jako je například celková cena, jsou poté dopočítány z modulu zpráv přes spouštěče. Před tím, než se vůbec mohou posílat jakákoliv data, je potřeba ustanovit spojení mezi klientskou a serverovou databází. Posílání dat na server firmy majitele helpdesku je

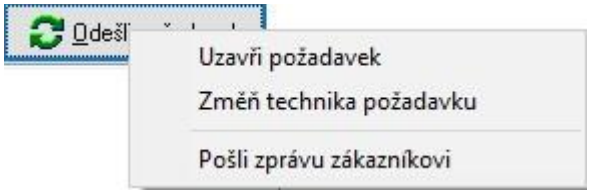
stálé. Obtížnější je komunikace ve směru server – klient. Poté, co technik obdrží požadavek, uloží se do číselníku všechny informace o databázi zákazníka. Jakmile technik odpovídá svojí zprávou zákazníkovi, systém si vezme IP adresu zákazníka a GUID databáze včetně názvu databáze z číselníku firem. Podle IP adres a GUID hodnot při testu komunikace se vzdálenou databází se ověří, zda se jedná o správnou databázi zákazníka. Poté se ustanoví spojení a technik může poslat zprávu. Princip propojení zprávy se záznamem požadavku je následující. Nejdříve se ve vzdálené databázi vyhledá primární klíč požadavku přes autora a datum uložení požadavku. Jakmile se primární klíč požadavku získá, jednoduše se nová odeslaná zpráva propojí s touto hodnotou. V klientské aplikaci může uživatel posílat celý požadavek včetně všech jeho zpráv nebo poslat jenom zprávu samotnou.

V serverové aplikaci má technik přístup k dalším funkcím (viz Obr 10). Uzavření požadavku slouží k nenávratnému splnění a uzamčení záznamu. Do tabulky serveru a klienta se doplní datum a okamžiku uzavření. Tento požadavek se označí zeleným textem a není možné ho měnit nebo v rámci požadavku posílat nové zprávy. Funkce „Změn technika požadavku“ je vhodná ve chvíli, kdy se změní hlavní technik, který požadavek mění. Email se opět změní, jak na straně serverové databáze, tak na straně klientské aplikace.

Obr 9: Menu pro komunikaci na straně klienta



Obr 10 Menu pro komunikaci na straně serveru



9.4.3.5 Implementace způsobu posílání emailů

Poslední funkcí je funkce posílání emailů (opět obr. 9). Pokaždé když zákazník úspěšně odešle požadavek nebo zprávu, program pošle požadavek do aplikace MS Outlook a vygeneruje nový email podle zadaných údajů z formuláře, který pak automaticky pošle email technikovi (pokud je uveden), pokud není, pošle ho na centrálu provozovatele helpdesku. Tímto způsobem bude firma

a technik vždy informována o nových požadavcích. Tělo emailu bude obsahovat základní informace o požadavku nebo zprávě včetně krátkého popisu a o jakou firmu se jedná.

Pokud firma nereaguje a nezpracovává požadavek, zákazník může firmu urgovat mailem znovu pomocí funkce „Pošli mail technikovi přes Outlook“ nebo mail se zprávou přes funkci „Pošli mail zprávy technikovi přes Outlook“. Tato funkce je také obzvláště užitečná, pokud je emailová schránka plná a MS Outlook odmítl poslat email. Po kliknutí na některé z těchto funkcí systém vygeneruje email, který obsahuje data podle záznamu, který má uživatel vybraný. Uživatelovi se zobrazí náhled emailu, který může ještě upravit podle přání před odesláním. Program email vygeneruje pouze v případě, že požadavek nebo zpráva byla již odeslána.

10. Testovaní aplikace

Testovací program je trochu odlišný od finální prezentační verze. Ve finální verzi bude databáze klienta komunikovat s databází na serveru. Z logistických důvodu je testovací verze programu, která je uvedena v příloze, naprogramována na komunikaci mezi dvěma lokálními databázemi. Princip komunikace je stejný jako u finální verze, zde se ale nepoužívá IP adresa nastavení připojení do druhé databáze. Pro spuštění aplikace musí mít uživatel zprovozněný databázový server Firebird a nainstalovaný program MS Outlook pro emailování.

10.1. Obsah přílohy programu

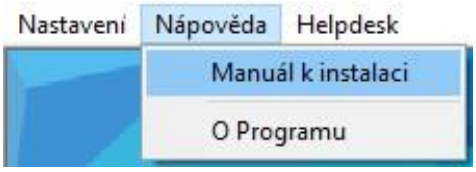
10.1.1. Databáze a skripty SQL

V této složce jsou dvě podsložky – databáze, skripty a firebird. Složka s databázemi obsahuje již nakonfigurované databáze se všemi datovými objekty a vyplněnými číselníky. Složka s SQL skripty obsahuje většinu SQL skriptů. Jediné skripty, které neobsahuje, jsou skripty nutné pro správný chod šablonových formulářů, které vytvořil pan Ing. Radim Holý. [21] Ve složce firebird se nachází konfigurační soubor, který se použije při instalaci databázového serveru.

10.1.2. Manuál na instalaci

Další zdroje manuálu jsou ve složkách s programem nebo se tento manuál spustí z menu lišty programu serverové nebo klientské verze progranu viz Obr 11.

Obr 11 Menu helpdesku – přístup k manuálu



10.1.3. Program

Samotný program má dvě verze – serverovou a klientskou, obě verze jsou uloženy ve svých podsložkách spolu se svými soubory obrázků a manuálů.

10.1.4. Zdrojové kódy

V této složce jsou všechny zdrojové kódy k formulářům programu. Jsou spustitelné v programu Delphi verze 10.1 Berlin (nicméně při vyvíjení programu byla použita různá grafická rozšíření). K nahlédnutí je stačí otevřít přes libovolný textový editor. Soubory s příponou .dfm obsahují informace ohledně umístění jednotlivých grafických objektů v rámci formuláře. Soubory, které mají příponu .pas, obsahují zdrojový kód se všemi funkcemi, procedurami atd. Většina formulářů je pojmenována podle tabulek, které primárně reprezentují. Hlavní formuláře byly vytvořeny podle šablon od pana Ing. Radima Holého, které jsou uloženy ve složce vzorových formulářů. [21]

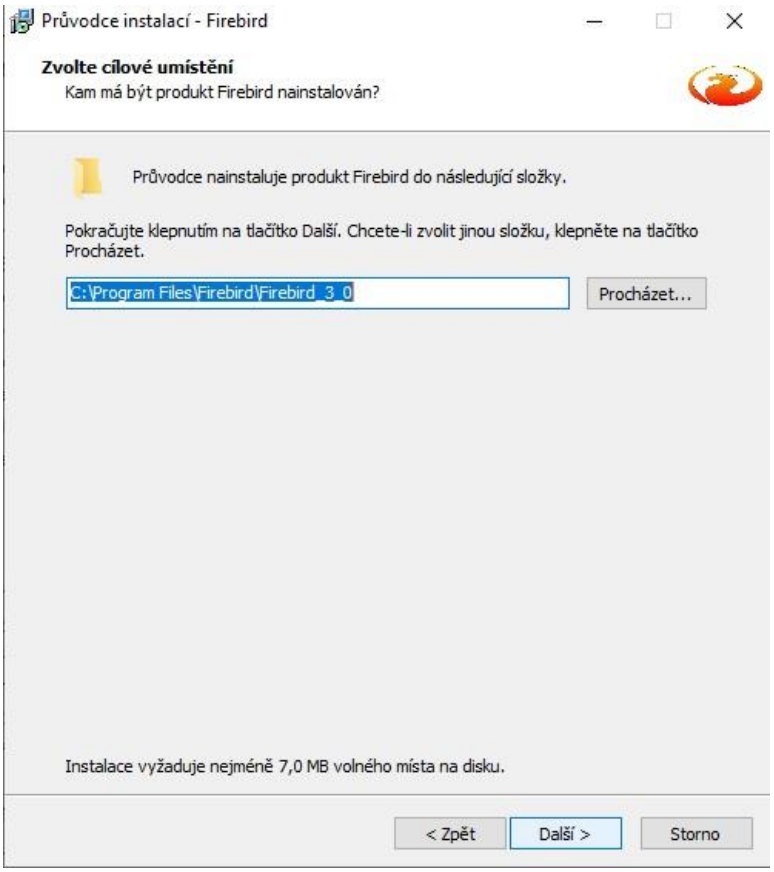
10.2. Instrukce k instalaci

10.2.1. Instalace databázového serveru Firebird 3.0

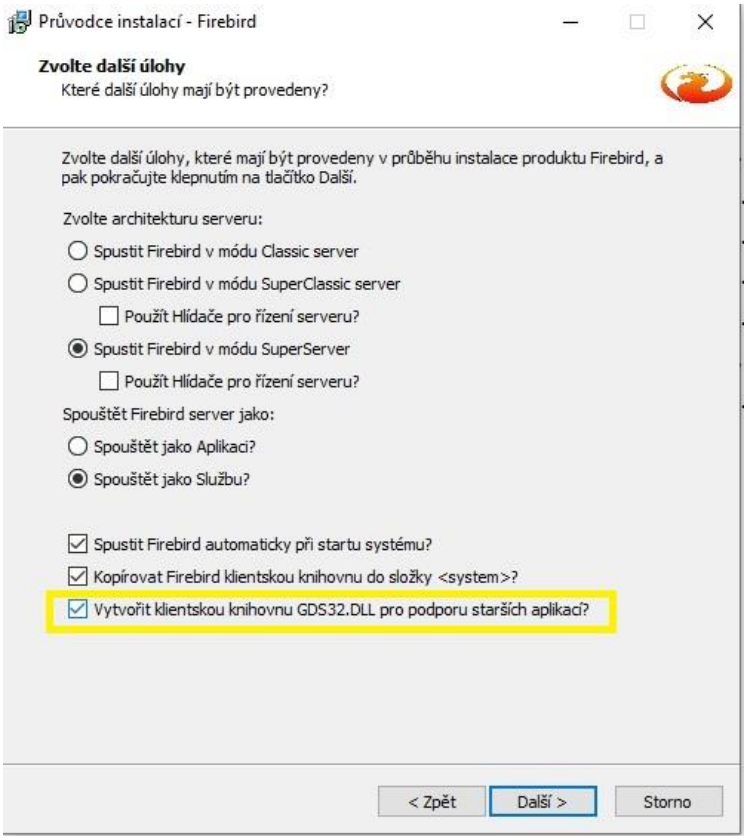
Program Helpdesk běží na databázovém serveru Firebird 3.0, který si uživatel musí nainstalovat před spuštění programu, Firebird 3.0 je ke stažení na těchto stránkách: https://firebirdsql.org/en/firebird-3-0/.

Po spuštění instalátoru naběhne uživateli tato obrazovka jako je na Obr 12. Během instalace stačí proklikat, až se uživatel dostane do místa jako je na Obr 13. Zde si uživatel musí zaškrtnout volbu přesně jako je ve žlutém rámečku.

Obr 12 Úvodní obrazovka instalace



Obr 13 Instalace instrukce



10.2.2. Nastavení cest do databází

Jakmile je databázový server nainstalovaný, je potřeba nastavit cesty k jednotlivým databázím. Databáze jsou uloženy v příloze. Pro nastavení cest k databázím se musí upravit soubor databases.txt ve složce nainstalovaného firebirdu (typicky je cesta z instalátoru C:\Program Files\Firebird\Firebird_3_0). Databáze se musí nacházet na místě, která je specifikována souborem databases.txt z přílohy ve složce „databáze a skripty SQL\firebird\databases.txt“ viz Obr 4. Pokud chce mít uživatel databáze jinde na disku, musí cestu opět určit v souboru „databases.txt“

Obr 14 databases.txt

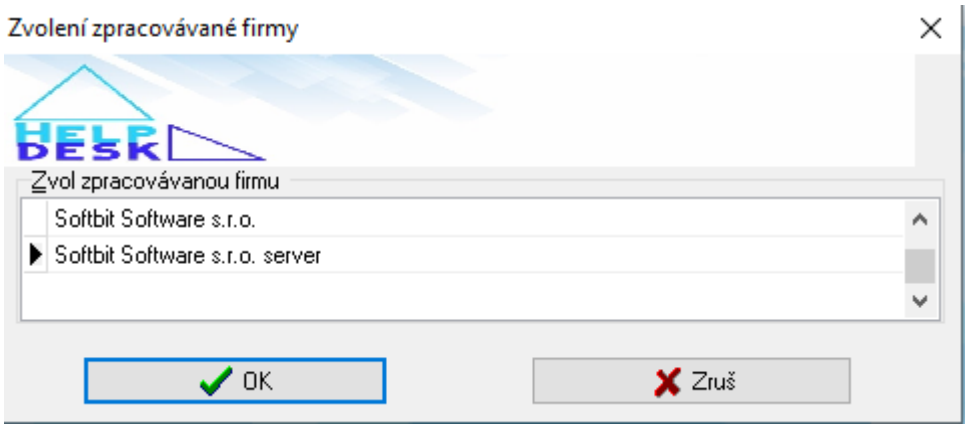
```
# ------------------------------
# List of known databases
# ------------------------------

#
# Makes it possible to specify per-database configuration parameters.
# See the list of them and description on file firebird.conf.
# To place that parameters in this file add them in curly braces
# after "alias = /path/to/database.fdb" line. Example:
#   big = /databases/bigdb.fdb
#   {
#       LockMemSize = 32M       # We know that bigdb needs a lot of locks
#       LockHashSlots = 19927   #   and big enough hash table for them
#   }
#


#
# Example Database:
#

win_ekonomS = C:\helpdesk\win_ekonomS.fdb
win_ekonom = C:\helpdesk\win_ekonom.fdb

databaze = C:\helpdesk\databaze.fdb


#
# Master security database specific setup.
# Do not remove it until you understand well what are you doing!
#
security.db = $(dir_secDb)/security3.fdb
{
    RemoteAccess = false
    DefaultDbCachePages = 50
}

#
# Live Databases:
#
```

10.2.3. Uživatelské účty a spuštění programu

Po nastavení cest jsou aplikace připravené ke spuštění, na databázovém serveru byly předpřipraveny 4 účty – účet SYSDBA, KLIENT, HLAVNIT, KLIENTADMIN. SYSDBA je implicitní administrátorský účet a má přístup do obou aplikací a jejich databází s heslem „masterkey“ (bez uvozovek). Účet KLIENT (heslo „k“) má přiřazenou roli „klientuzivatel“ a přístup do aplikace klienta stejně tak jako účet KLIENTADMIN (heslo „ka“) a jeho role

„klientadmini". Do serverové aplikace se uživatel může přihlásit také účtem HLAVNIT (heslo „h“). Po spuštění programu a přihlášení přes účet SYSDBA se uživateli zobrazí obrazovka jako je na Obr 5. Firma Softbit Software s.r.o. je databáze win_ekonom pro klientskou aplikaci. Firma Softbit Software s.r.o. server je databází win_ekonomS pro serverovou verzi. Jelikož mají databáze jiné struktury a odlišnosti, musí se uživatel připojit do příslušné databáze v aplikaci, jinak nebude aplikace správně fungovat.

Obr 15 Zvolení firmy



Závěr

Tato práce se v teoretické části rozsáhle a obecně věnovala objasněním principů a fungování relačních databází včetně vysvětlení evoluce datových modelů, které postupnou evolucí vedly ke vzniku relačních databází a objektově-relačních databází. Zajímavé je, že relační databáze jsou pořád nejčastějším typem datového modelu vzhledem k jeho staří a přítomnosti objektově-relačního modelu.

Hlavním přínosem práce je vytvořená aplikace helpdesku, která umožňuje rychlou komunikaci mezi zákazníkem a provozovatelem služby a přehlednou evidenci dat. V současném stavu nakonfigurována pro firmu Softbit Software s.r.o. Aplikace byla od začátku koncipována jako nezávislá. Aby mohla fungovat na serverech jiných firem, stačí překonfigurovat IP adresu a cílovou databázi na klientské verzi aplikace. U serverů s větším zabezpečením by fungování aplikace mohlo být komplikovanější a vyžádalo by si při instalaci přítomnost správce serveru, který by nastavil seznam bezpečných a důvěryhodných IP adres. Jelikož byla aplikace vyvinuta v Delphi s databázovým serverem Firebird je zatím naprogramována jako standartní aplikace a hodí spíše pro menší firmy, jelikož jenom jeden technik je schopen se připojit do serverové aplikace na serverové instanci. Pro větší firmy by bylo vhodnější změnit typ aplikace na webovou aplikaci, kam by se mohlo přihlašovat více techniků najednou odkudkoliv.

V závěru lze podotknout, že už samotná teoretická část práce slouží jako dobrý začátečnický manuál k databázím, databázovým serverům a SQL jazyku. Také lze podotknout, že webové aplikace představují výhodu oproti „standartním aplikacím“ v přístupu a dostupnosti aplikace, jelikož se uživatel může skoro odkudkoliv do aplikace připojit a pracovat s ní.

Seznam literatury

[1] Comparing Popular Database Management Systems | AltexSoft. AltexSoft – Technology & Solution Consulting Company [online]. Copyright © Copyright [cit. 23. 11. 2020]. Dostupné z: https://www.altexsoft.com/blog/business/comparing-database-management-systems-mysql-postgresql-mssql-server-mongodb-elasticsearch-and-others/

[2] Databáze. Databáze [online]. Copyright © 2010 Misha [cit. 23. 11. 2020]. Dostupné z: http://www.databaze.chytrak.cz/index.htm

[3] Embedded databáze – Úvod - Root.cz. Root.cz - informace nejen ze světa Linuxu [online]. Copyright © 1998 [cit. 23. 11. 2020]. Dostupné z: https://www.root.cz/clanky/embedded-databaze-uvod/

[4] Historie relačních databází - Root.cz. Root.cz - informace nejen ze světa Linuxu [online]. Copyright © 1998 [cit. 25. 11. 2020]. Dostupné z: https://www.root.cz/clanky/historie-relacnich-databazi/

[5] KOŠÁREK, Lukáš. Výkonnostní srovnání relačních databází. Brno, 2010 [cit. 23. 11. 2020]. Dostupné z: https://is.muni.cz/th/wox38/Vykonnostni_srovnani_relacnich_databazi.pdf. Bakalářská práce. Masarykova univerzita Fakulta informatiky.

[6] LACKO, Ľuboslav. 1001 tipů a triků pro SQL. Brno: Computer Press, 2011. 416 s., ISBN 978-80-251-3010-0.

[7] LANGMAIER, David. Srovnání volně šiřitelných relačních SŘBD z pohledu databázového programátora. Plzeň, 2016 [cit. 23. 11. 2020]. Dostupné z: https://otik.zcu.cz/bitstream/11025/23843/1/BP_David_Langmaier.pdf. Bakalářská práce. Západočeská univerzita v Plzni Fakulta aplikovaných věd Katedra informatiky a výpočetní techniky.

[8] PETŘÍK, Tomáš. Volba databázového systému na základě požadavků projektu IS/ICT. Praha, 2008 [cit. 23. 11. 2020]. Dostupné

z https://vskp.vse.cz/9755_volba_databazoveho_systemu_na_zaklade_pozadavku_projektu_is_ict. Bakalářská práce. Vysoká škola ekonomická v Praze Fakulta informatiky a statistiky Katedra informačních technologií.

[9] Popis normalizace databáze | Microsoft Docs. [online]. Copyright © Microsoft 2020 [cit. 23. 11. 2020]. Dostupné z: https://docs.microsoft.com/cs-cz/office/troubleshoot/access/database-normalization-description

[10] PROCHÁZKA, David. Oracle: průvodce správou, využitím a programováním nad databázovým systémem. Praha: Grada, 2009. 168 s., Průvodce (Grada). ISBN 978-80-247-2762-2.

[11] Object-oriented database: Explanation + Advantages & Disadvantages – IONOS. | IONO [online]. Copyright © 2020 [cit. 23. 11. 2020]. Dostupné z: https://www.ionos.com/digitalguide/hosting/technical-matters/object-oriented-databases/

[12] University information system MENDELU [online]. [cit. 24. 11. 2020]. Dostupné z: https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=39736

[13] Jazyk SQL – Vojtěch Hordějčuk. Vojta Hordějčuk aka voho – Software Engineer and Bedroom Music Producer [online]. Copyright © 2008 [cit. 23. 11. 2020]. Dostupné z: http://voho.eu/wiki/sql/

[14] CONNOLLY, Thomas M. a Carolyn E. BEGG. Database Systems: Practical Approach to Design, Implementation, and Management. 5th ed. Boston: Addison-Wesley, 2010. ISBN 978-0-321-52306-8.

[15] RAISERROR (Transact-SQL) - SQL Server | Microsoft Docs. [online]. Copyright © Microsoft 2021 [cit. 23.04.2021]. Dostupné z: https://docs.microsoft.com/en-us/sql/t-sql/language-elements/raiserror-transact-sql?view=sql-server-ver15

[16] Six different methods to copy tables between databases in SQL Server. SQLShack [online]. [cit. 2021-4-24]. Dostupné z: https://www.sqlshack.com/six-different-methods-to-copy-tables-between-databases-in-sql-server/

[17] Firebird: Writing UDFs for InterBase. Firebird: The true open source database for Windows, Linux, Mac OS X and more [online]. Copyright © 2000 [cit. 24.04.2021]. Dostupné z: https://firebirdsql.org/en/writing-udfs-for-interbase/

[18] MARCO, Cantu. Mastering Delphi 7. 1. Alameda: Sybex, 2003. ISBN 978-0-78-214201-3.

[19] IBExpert Developer Studio. IB Expert [online]. [cit. 2021-4-30]. Dostupné z: https://www.ibexpert.net/ibe/pmwiki.php?n=Main.IBExpert

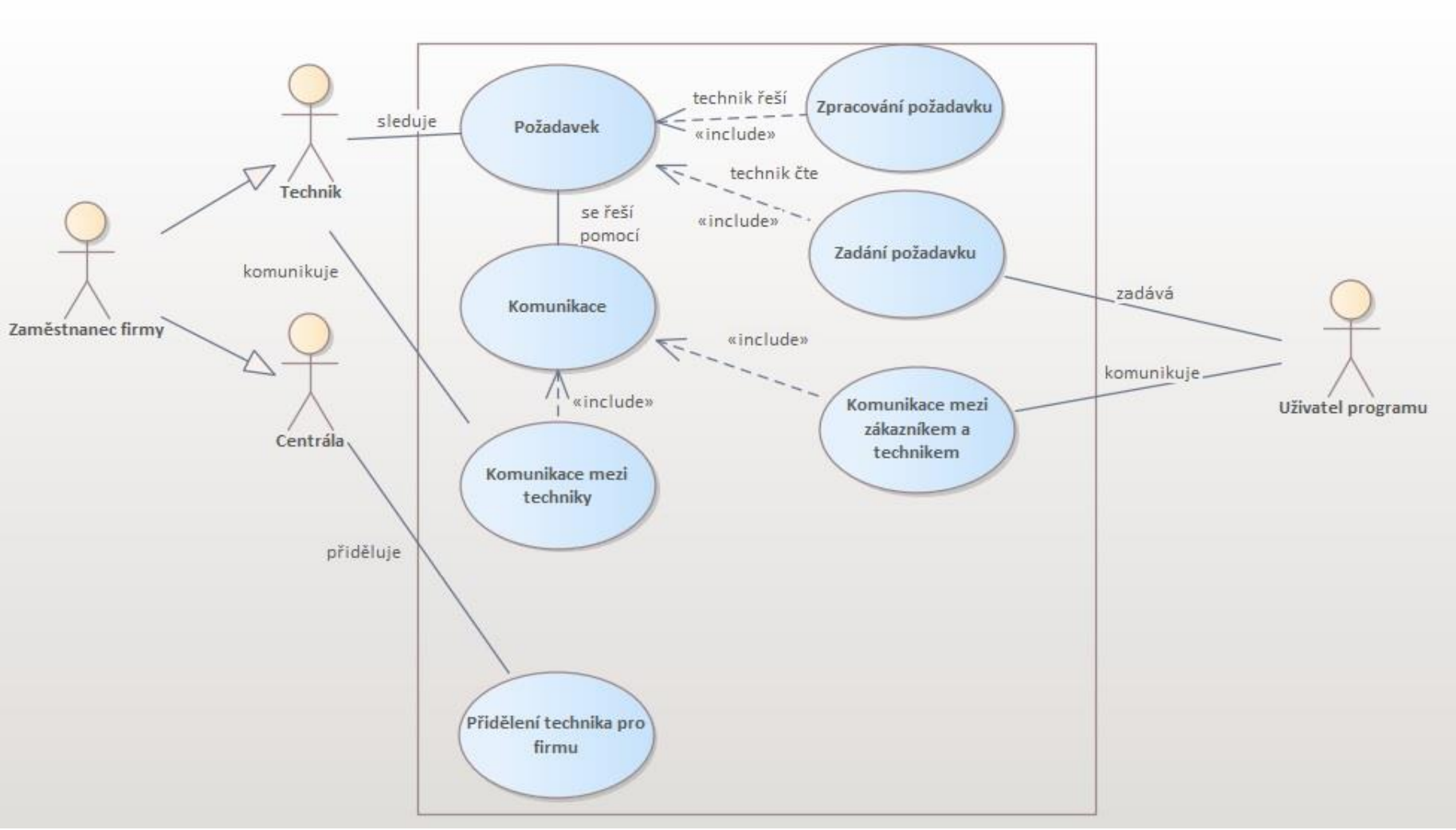
[20] IBExpert product features. IB Expert [online]. [cit. 2021-4-30]. Dostupné z: https://www.ibexpert.net/ibe/pmwiki.php?n=Main.IBExpertFeatures

[21] HOLÝ, Ing. Radim. **SQL Ekonom,** Softbit Software s.r.o., [cit. 2021-8-16].

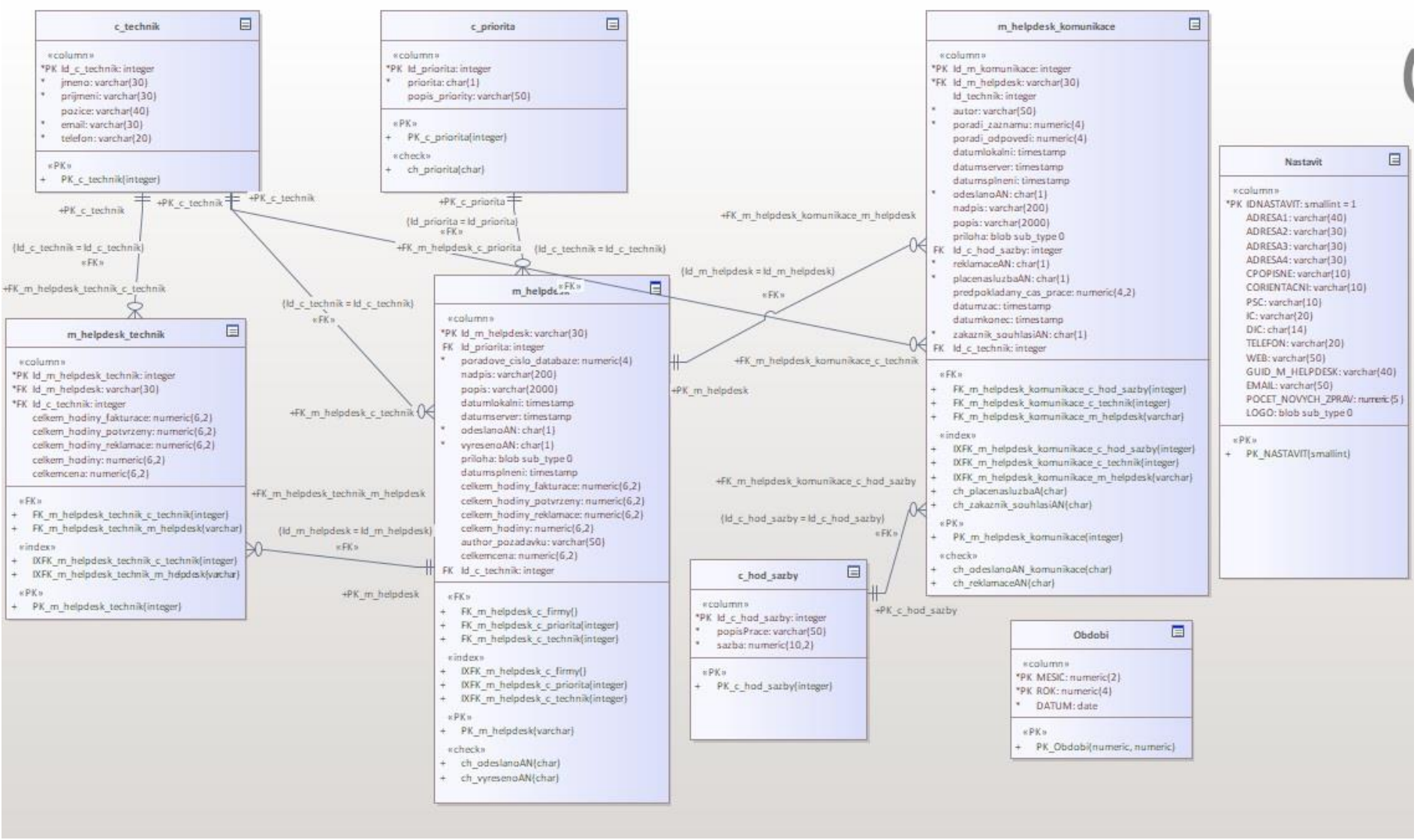
Rejstřík

Seznam příloh

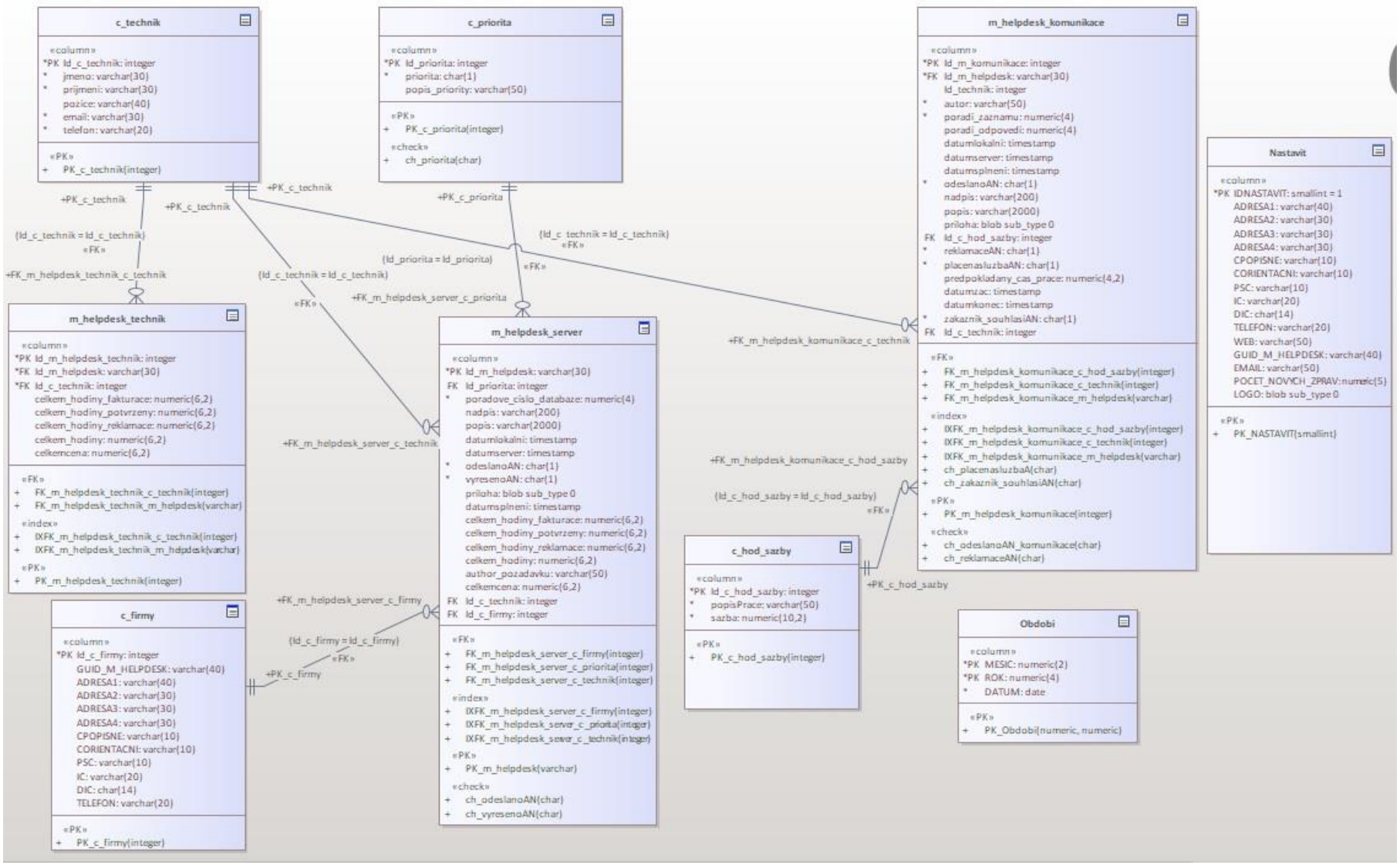
I. Obr 16: use case diagram



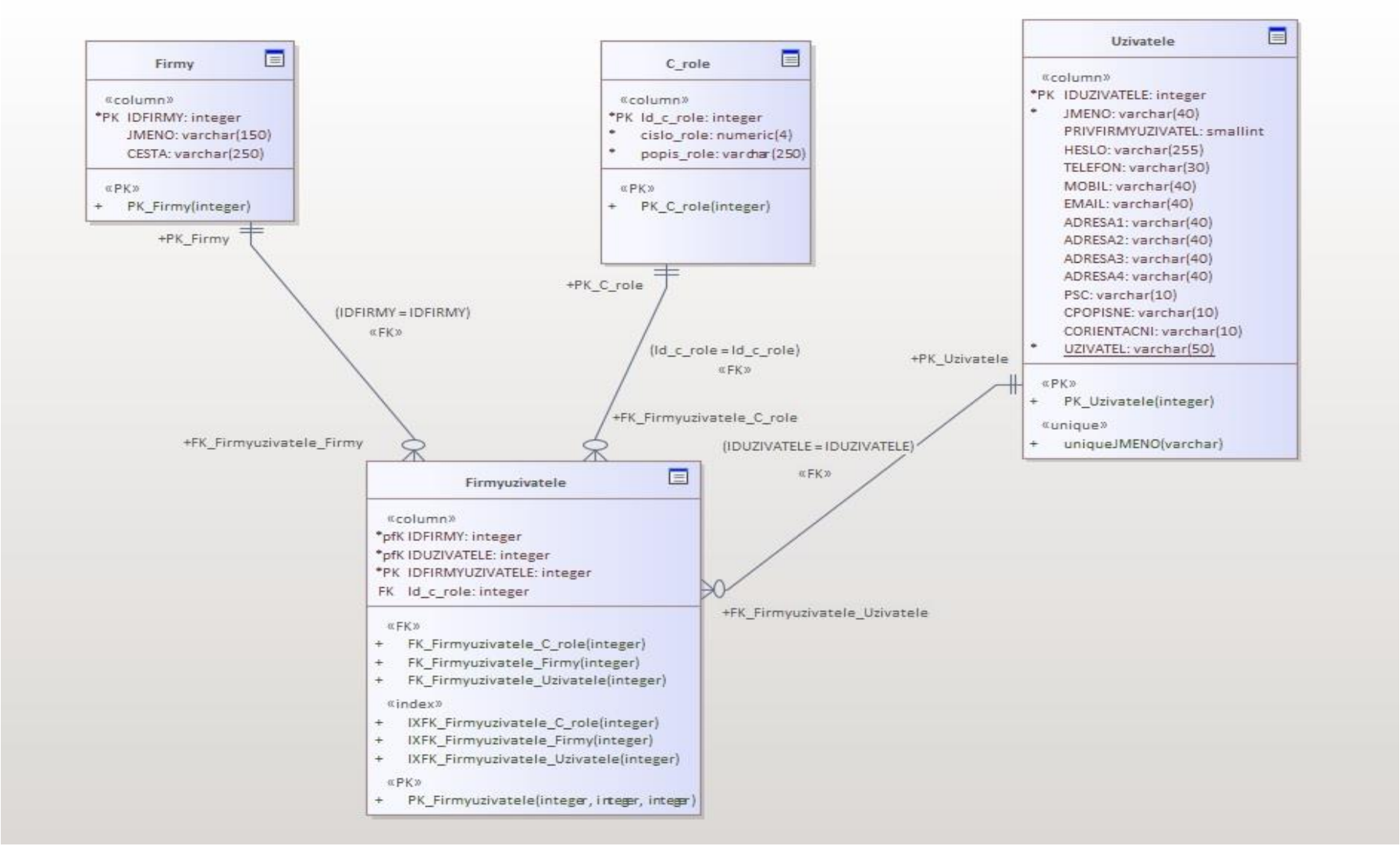
II. Obr 17: datový model klientské aplikace



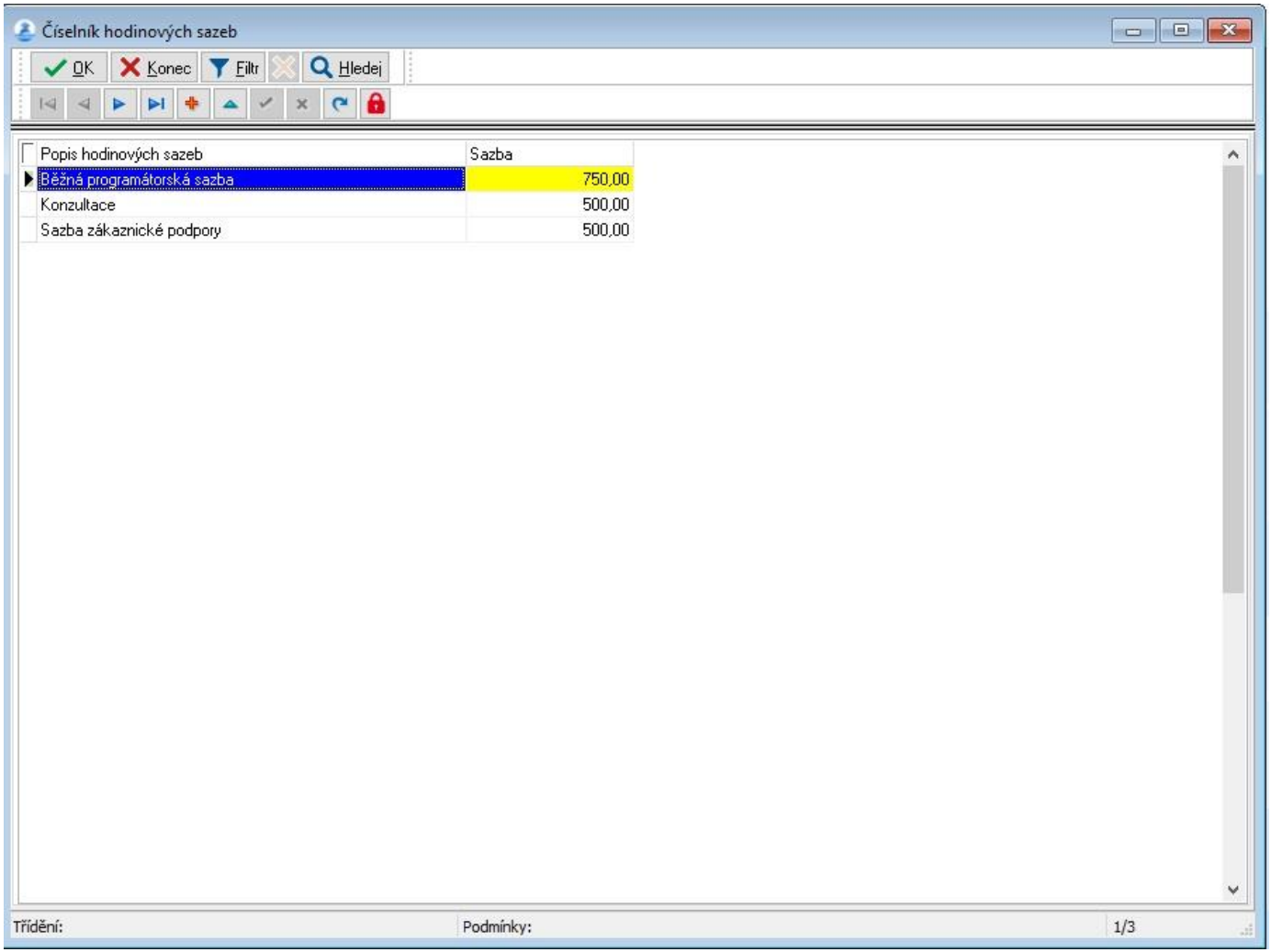
III. Obr 18: datový model serverové aplikace



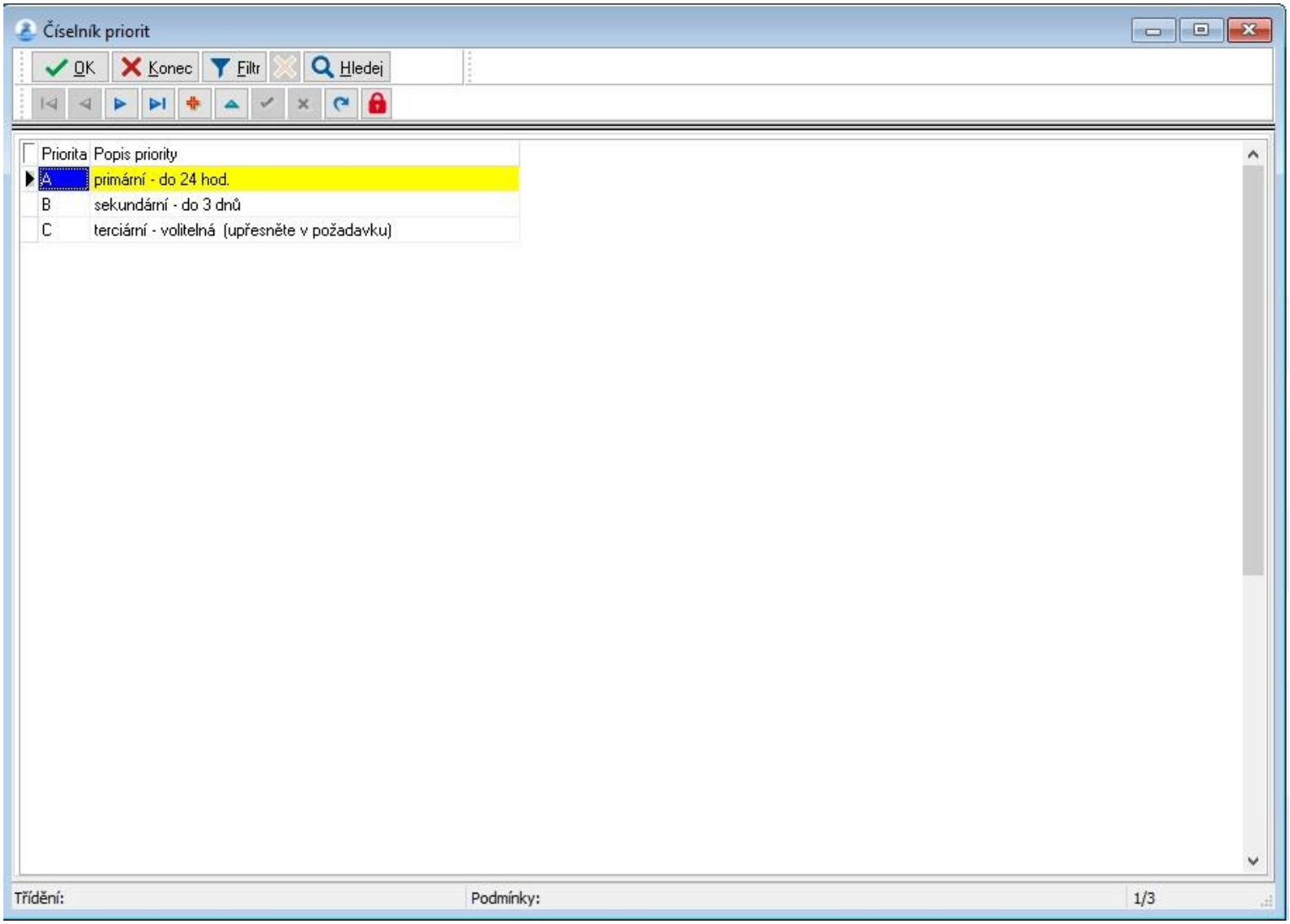
IV. Obr 19: datový model databáze uživatelů



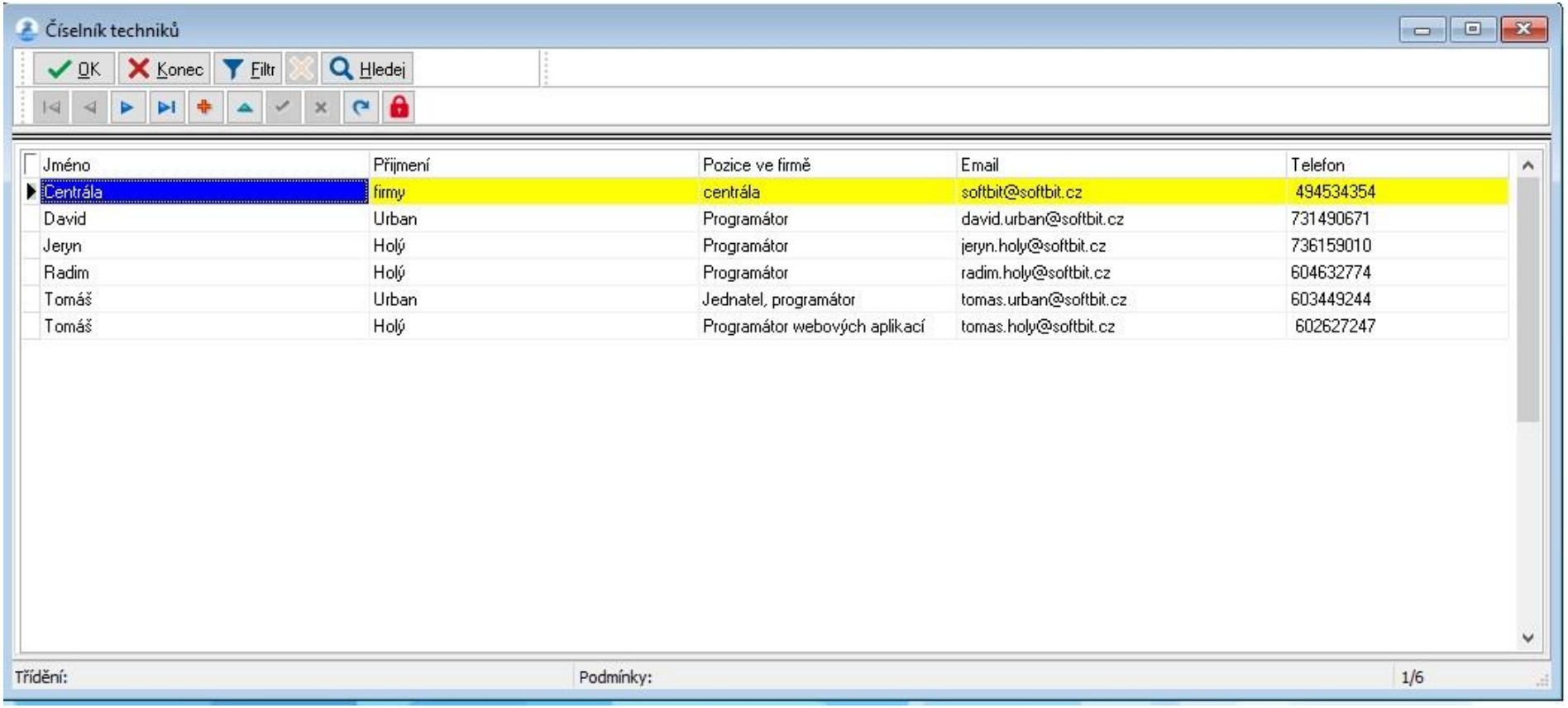
V. Obr 20: číselník hodinových sazeb



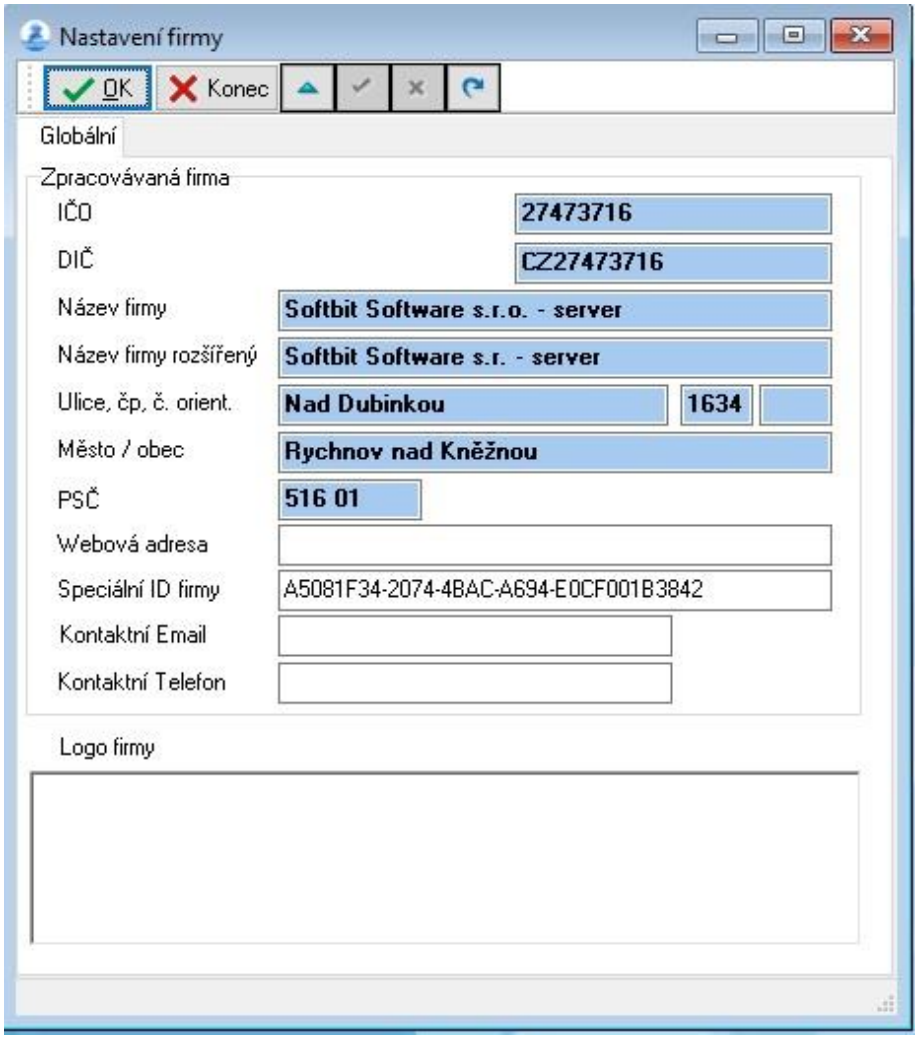
VI. Obr 21: číselník priorit



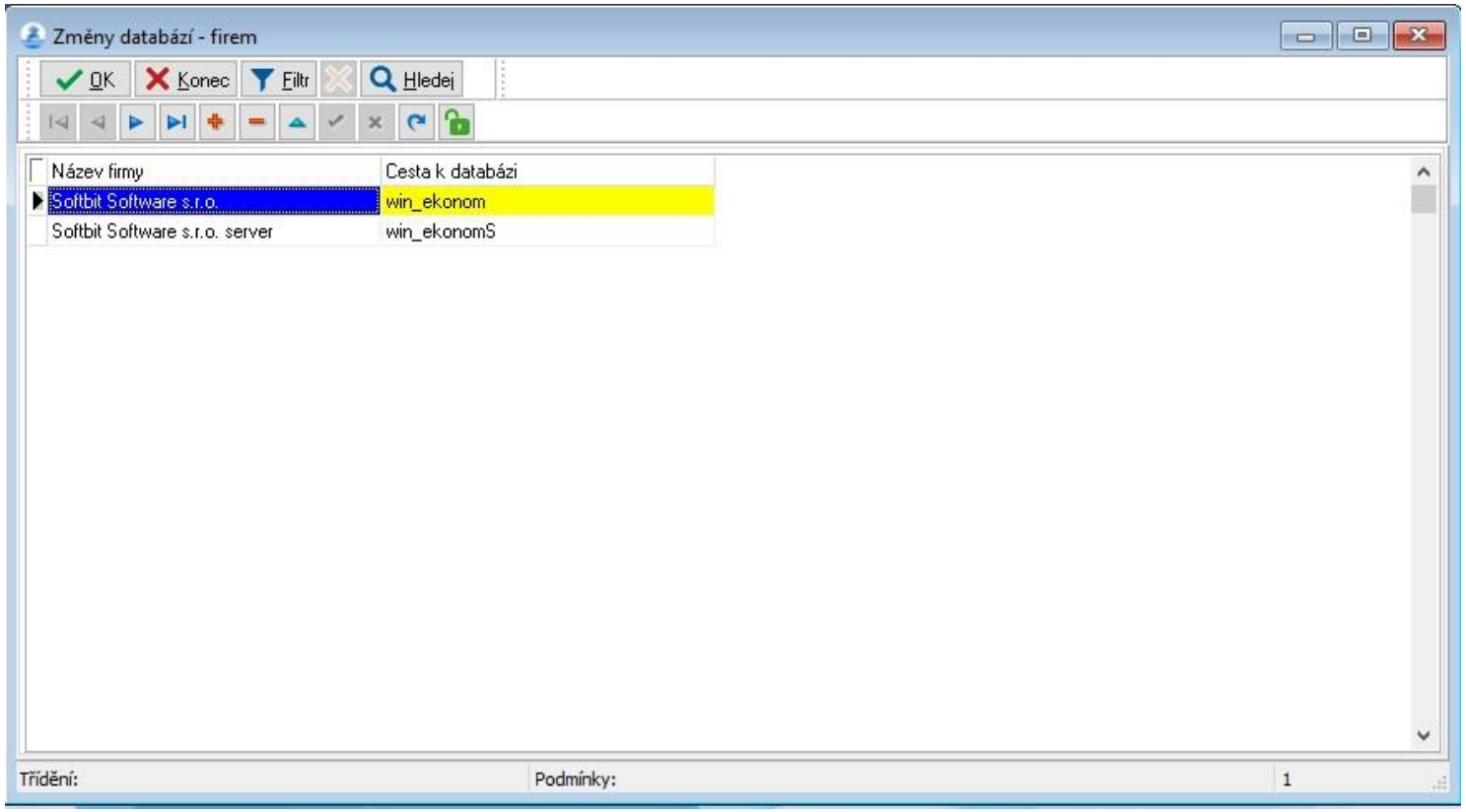
VII. Obr 22: číselník techniků



VIII. Obr 23: formulář nastavení firmy

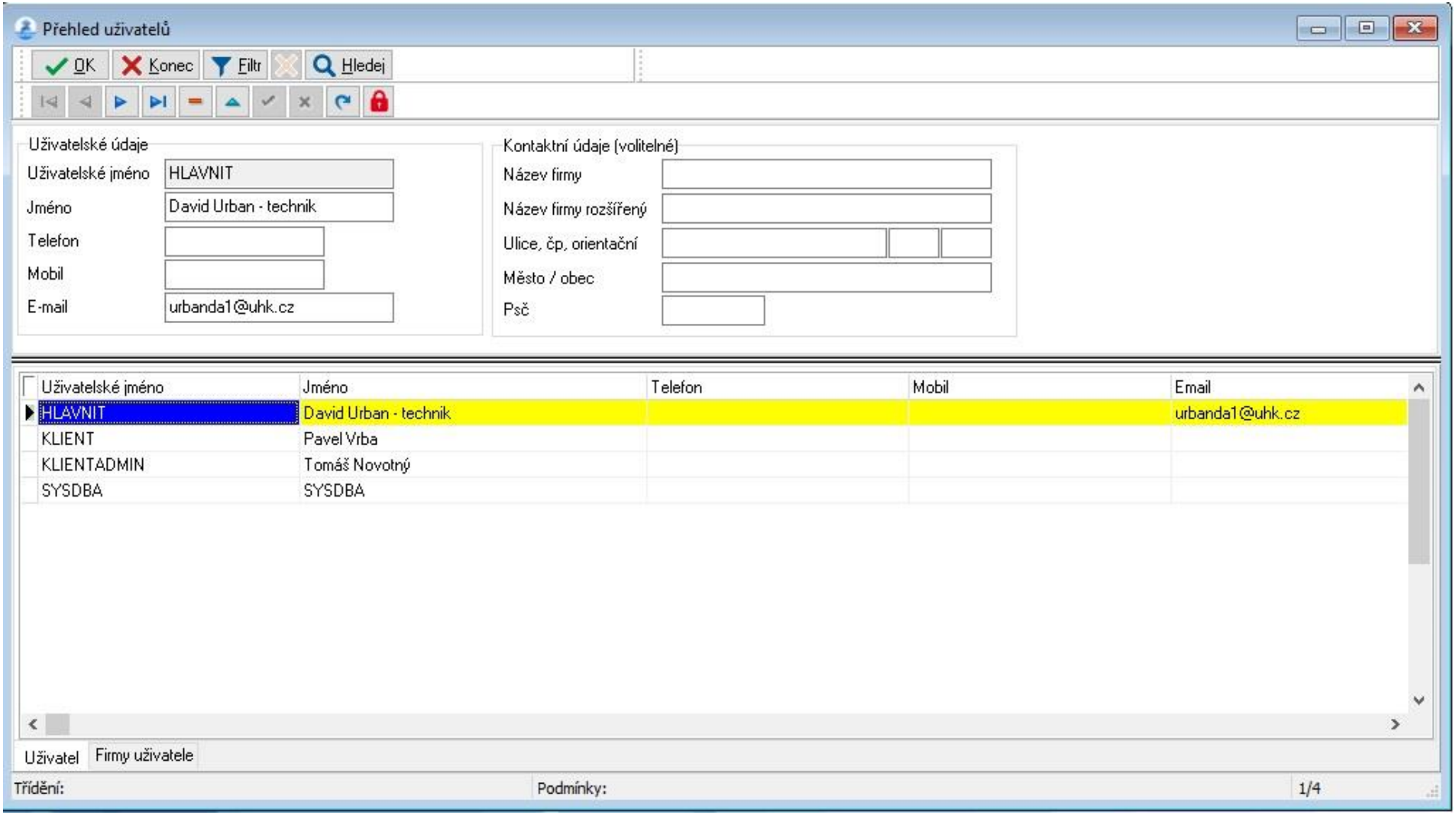


IX. Obr 24: nastavení databází firem a jejich cesty

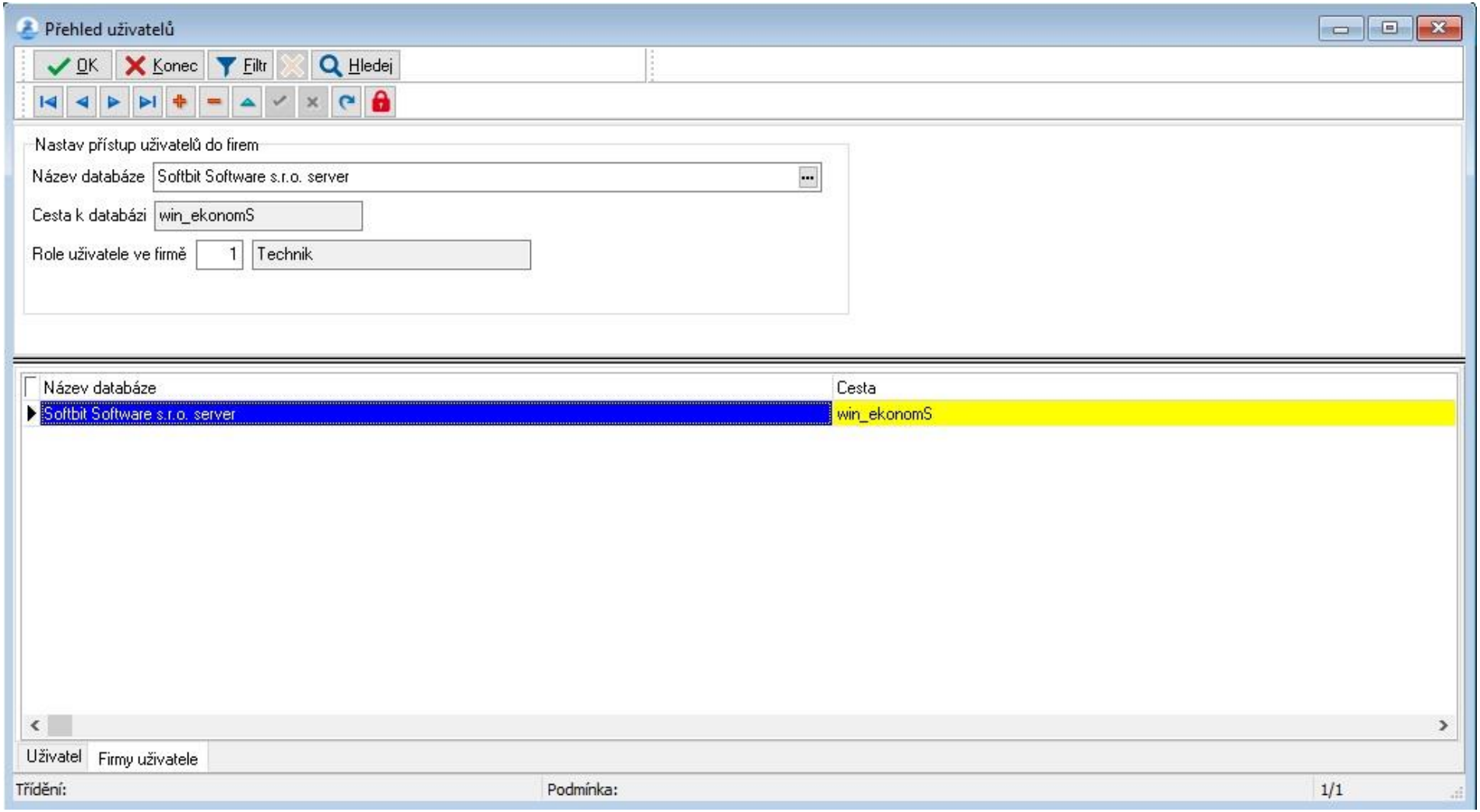


X. Obr 25: formulář pro přihlášení

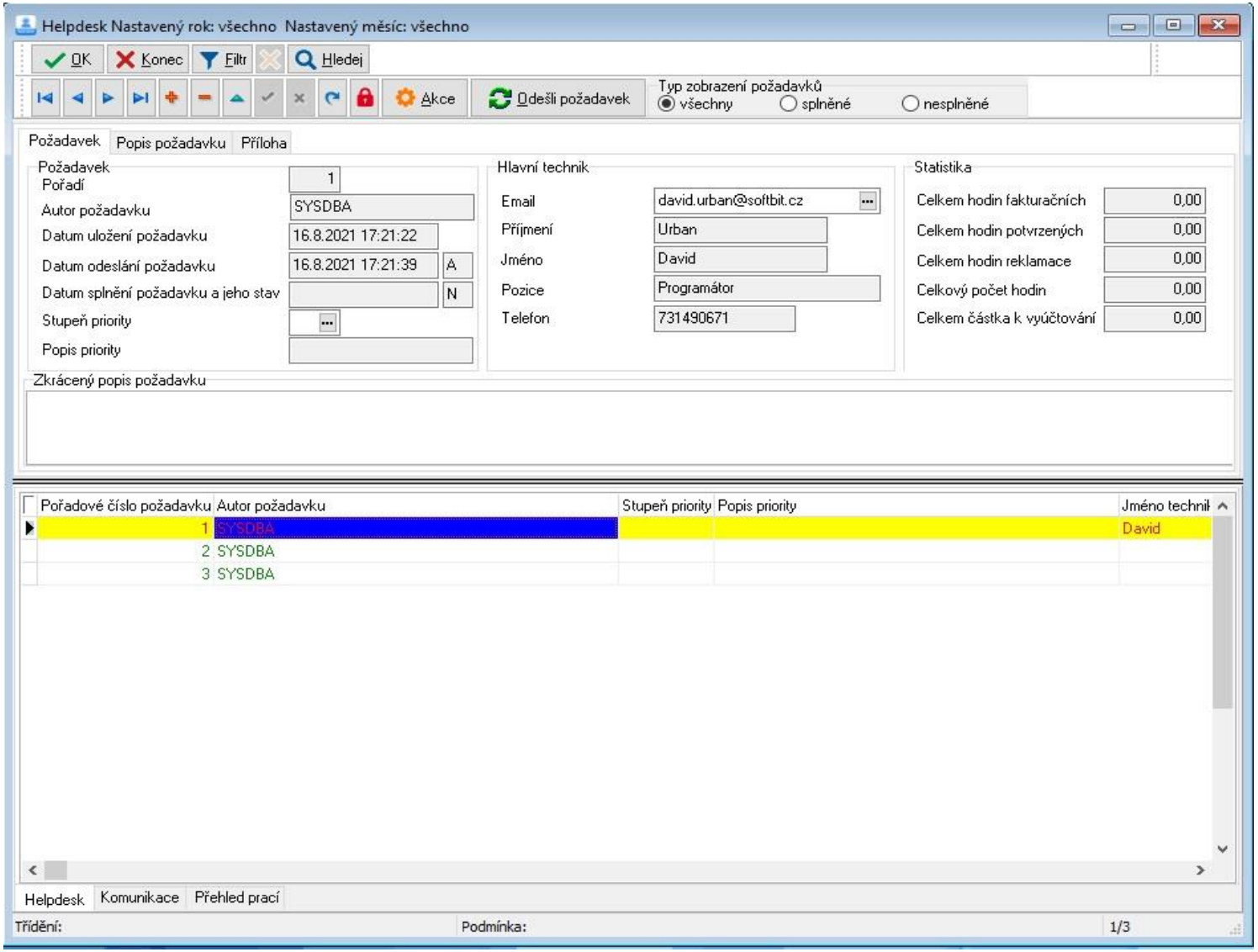
XI. Obr 26: přehled uživatelů



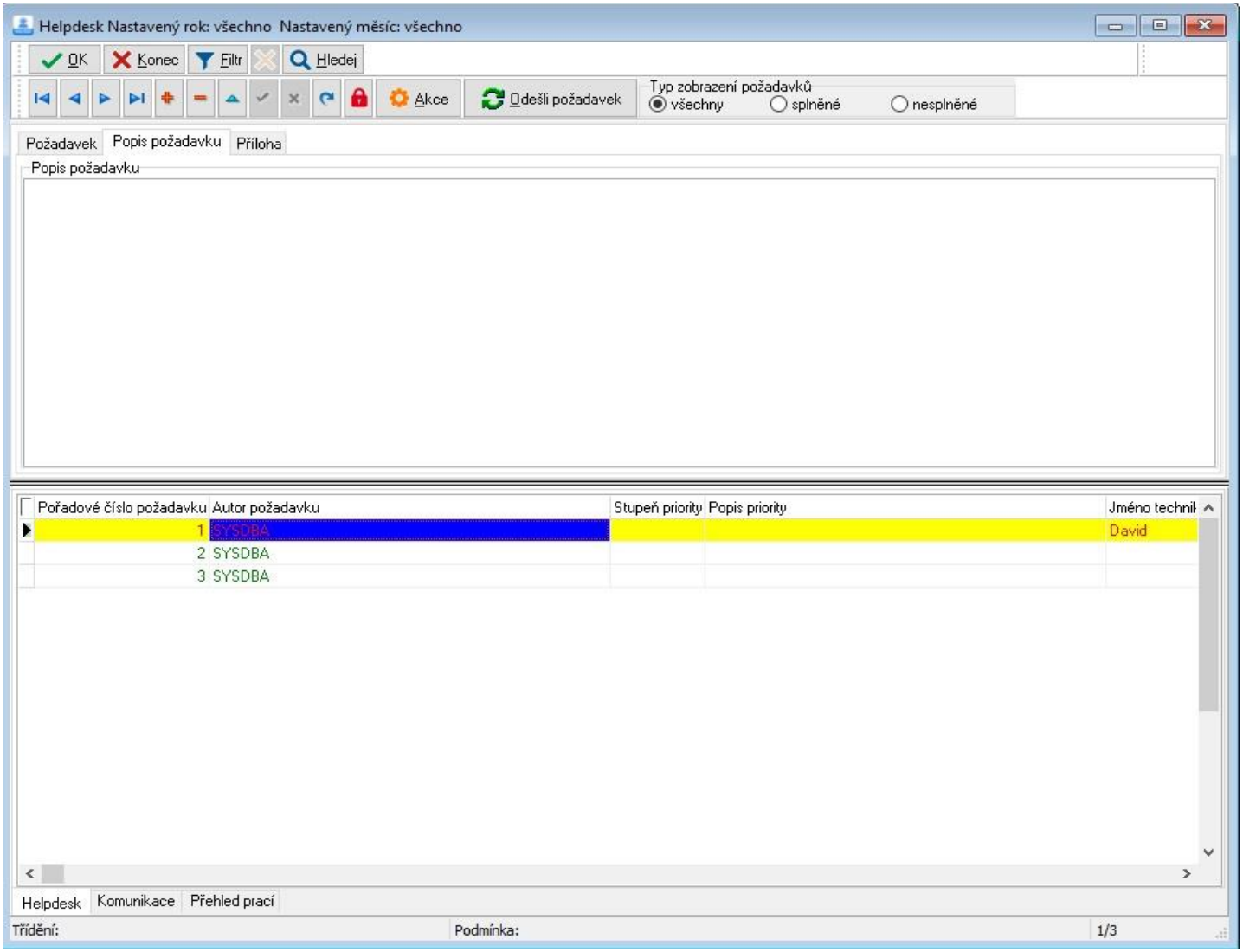
XII. Obr 27: nastavení přístupu uživatelů do firem



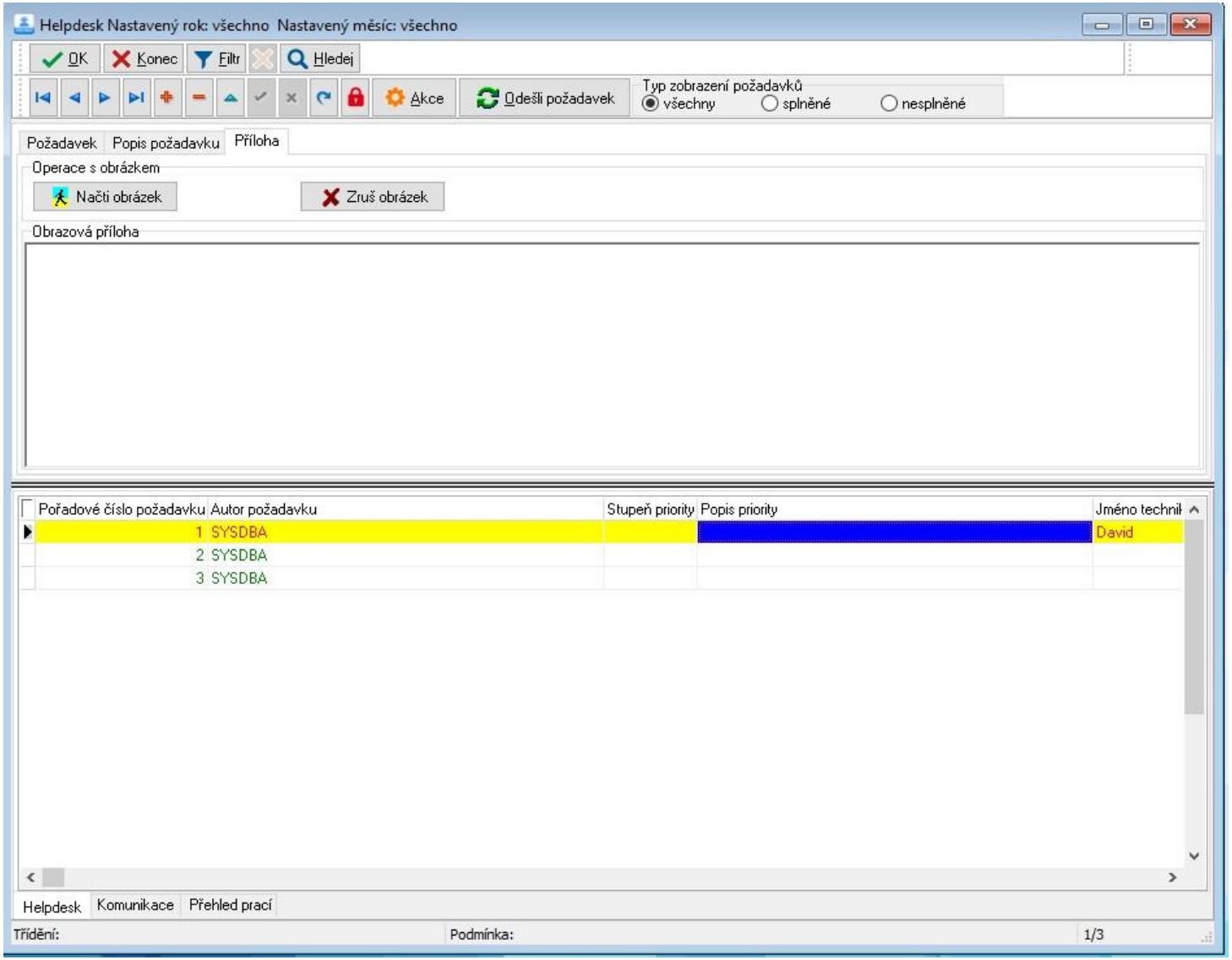
XIII. Obr 28: helpdesk požadavek



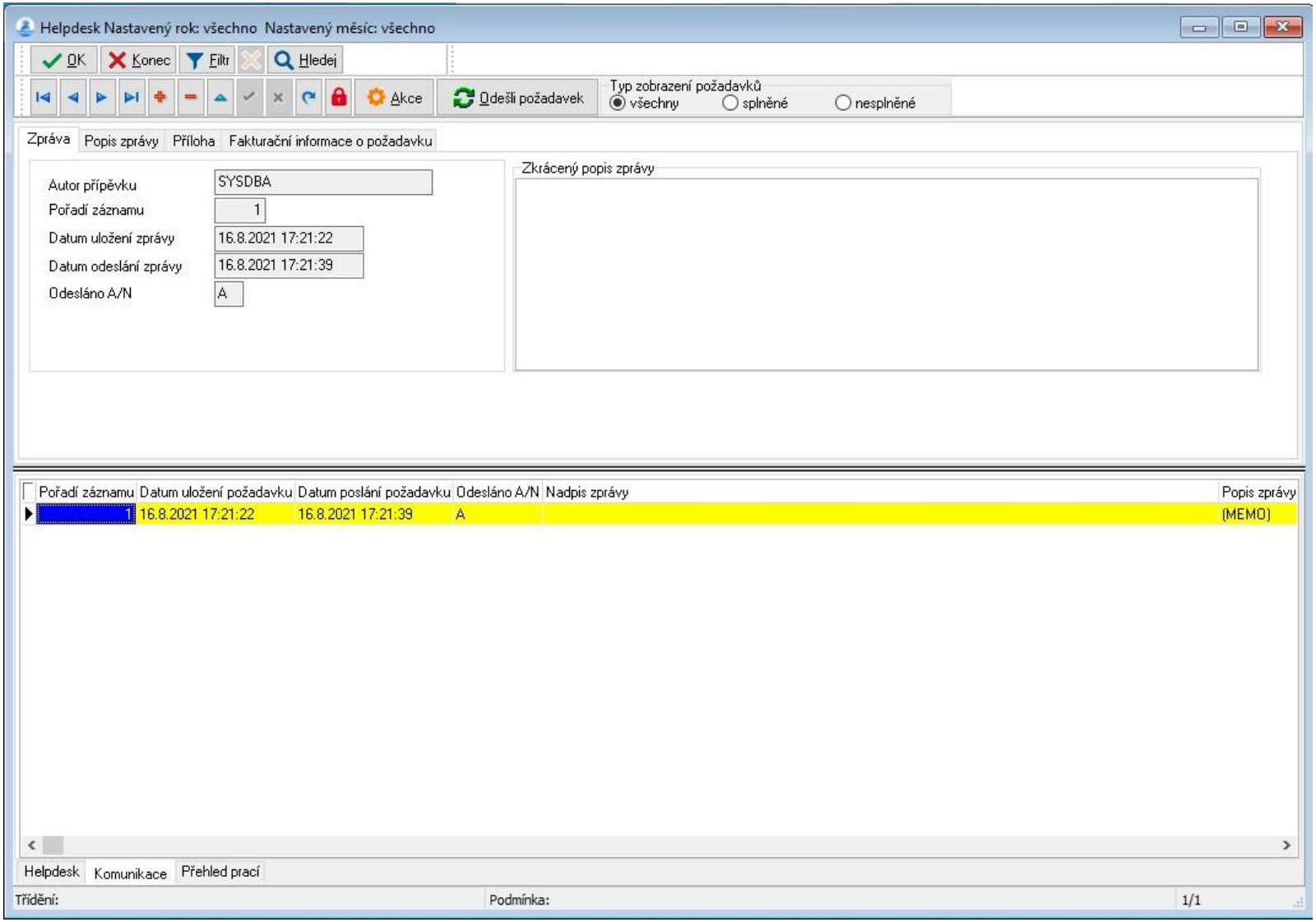
XIV. Obr 29: helpdesk popis požadavku



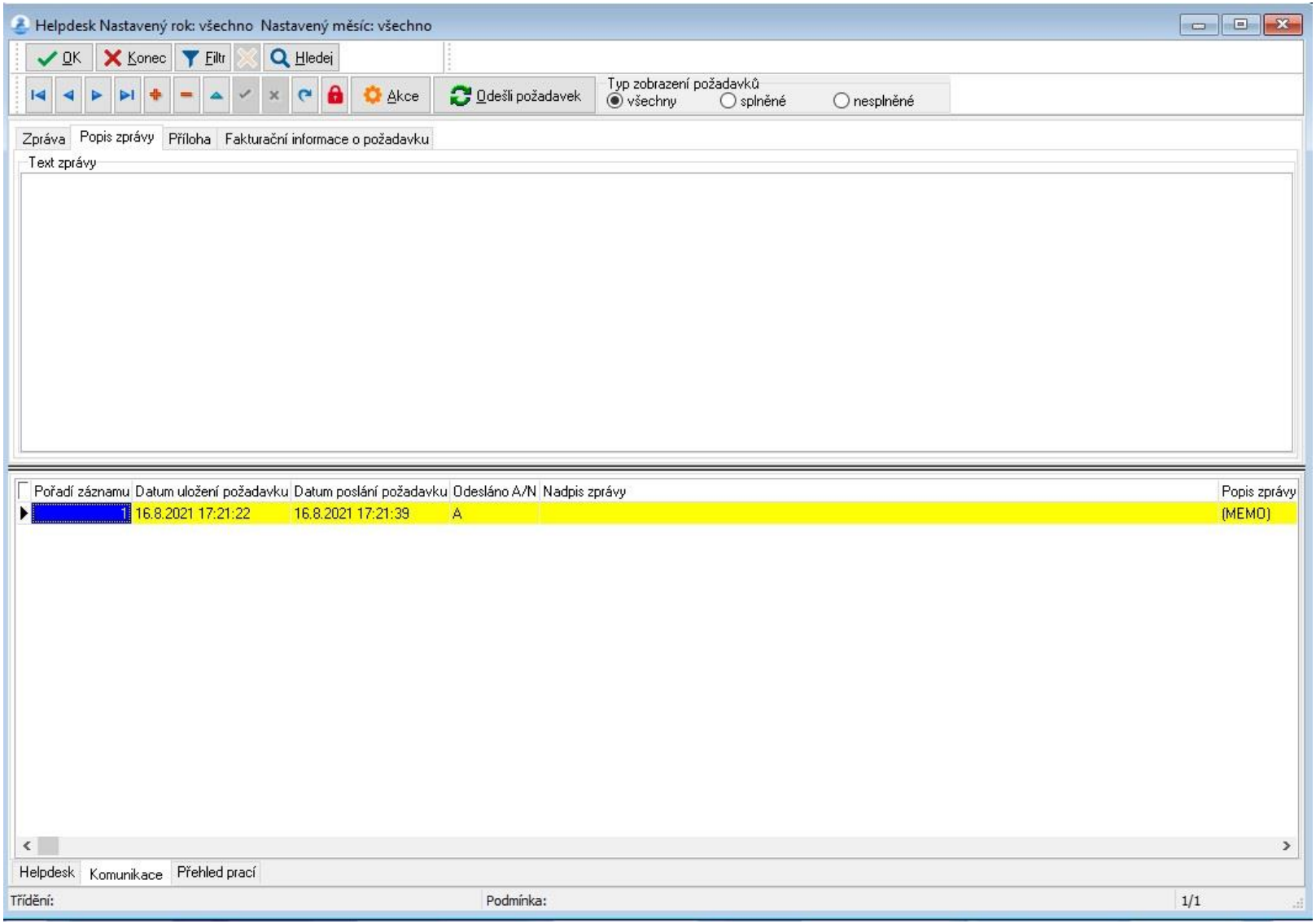
XV. Obr 30: helpdesk požadavek příloha



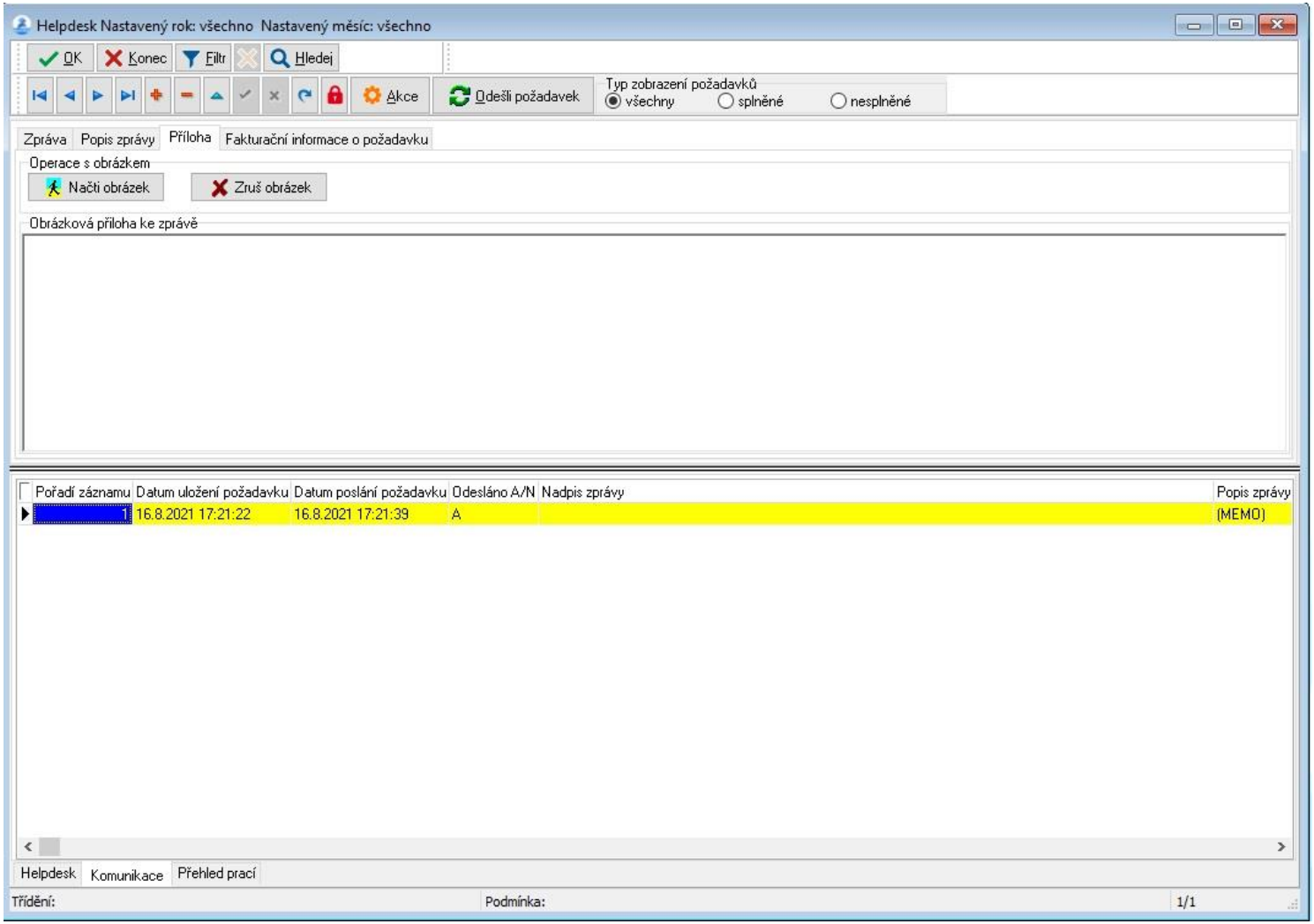
XVI. Obr 31: helpdesk komunikace - zpráva



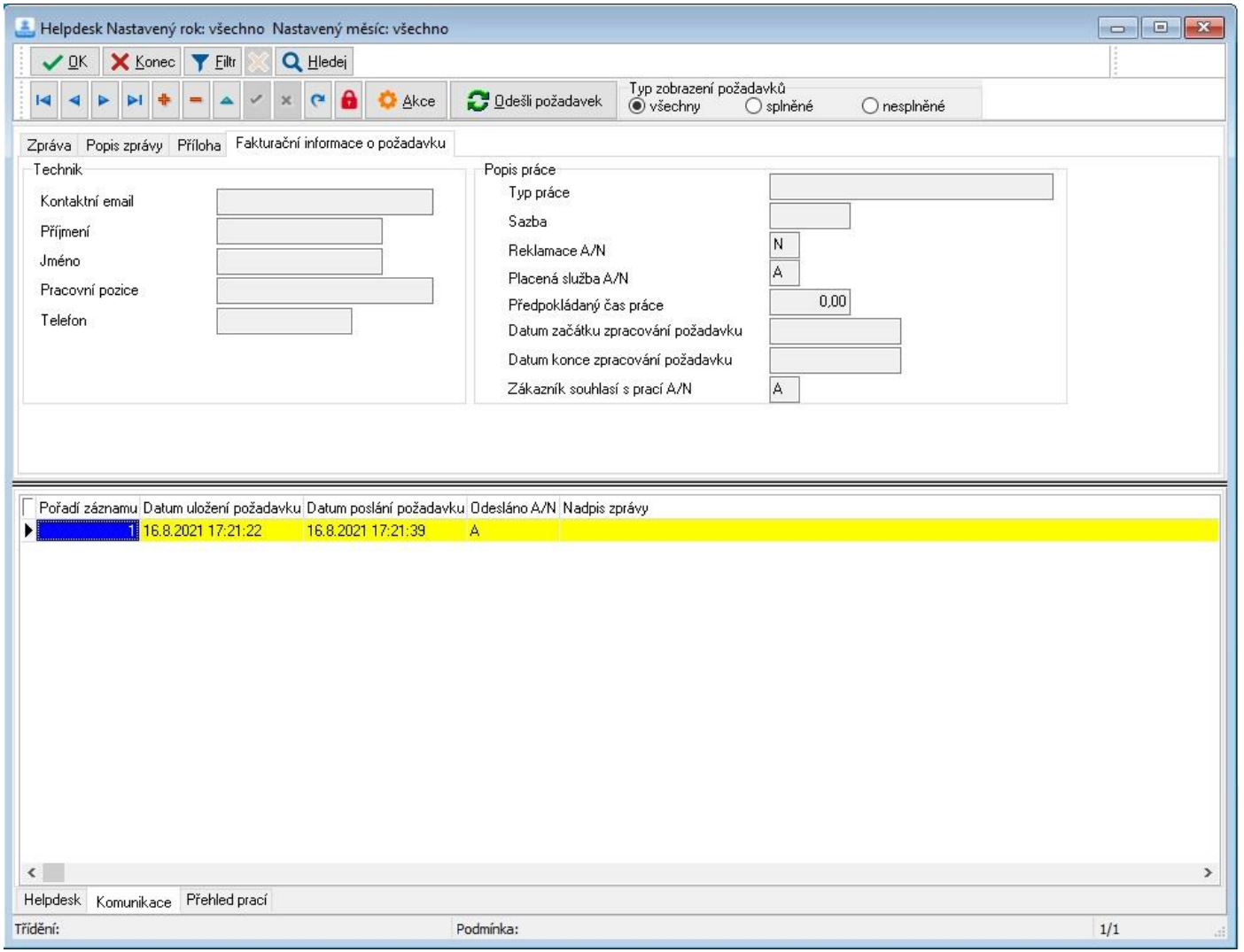
XVII. Obr 32: helpdesk – komunikace, popis zprávy



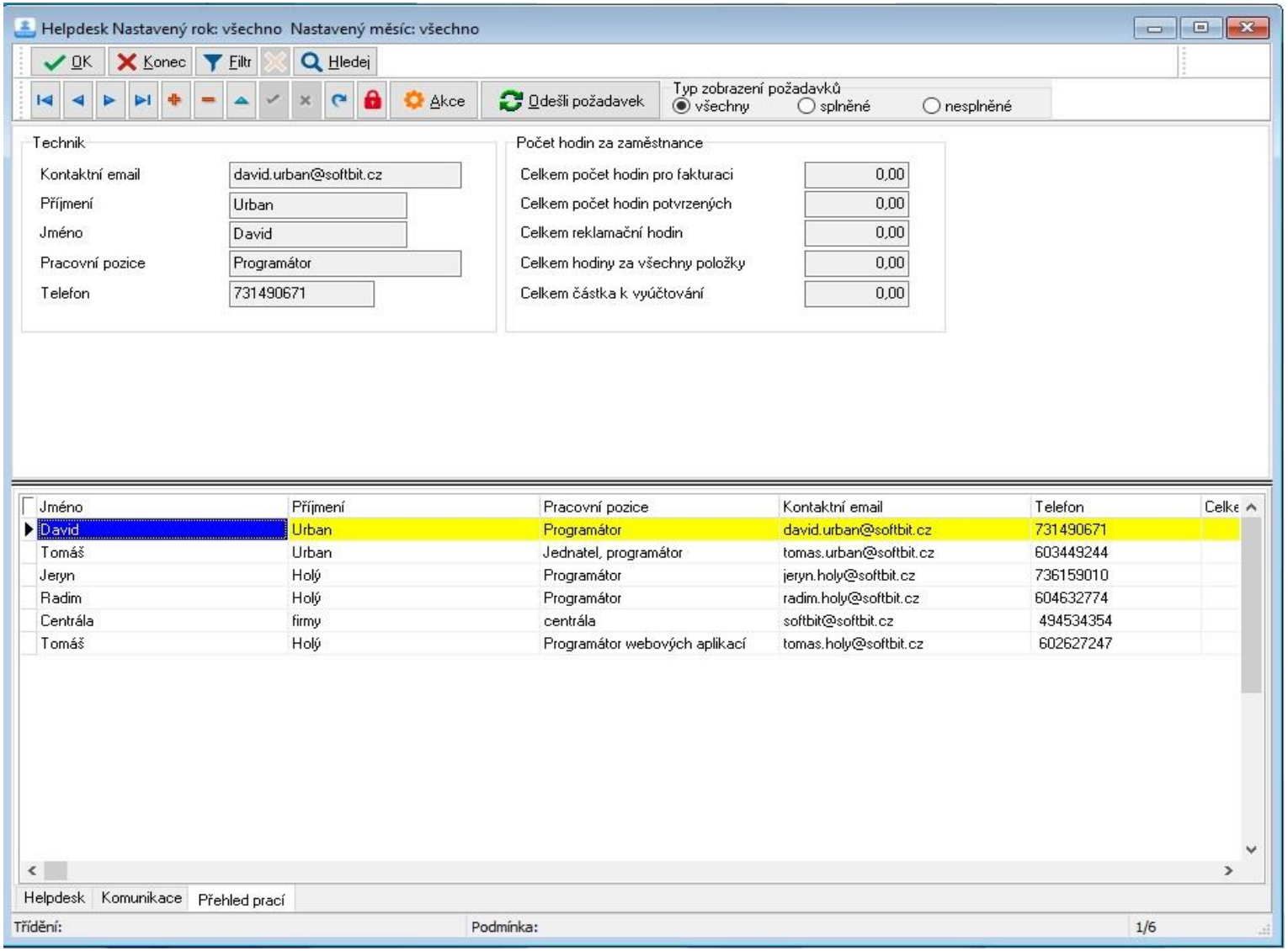
XVIII. Obr 33: helpdesk – komunikace, příloha



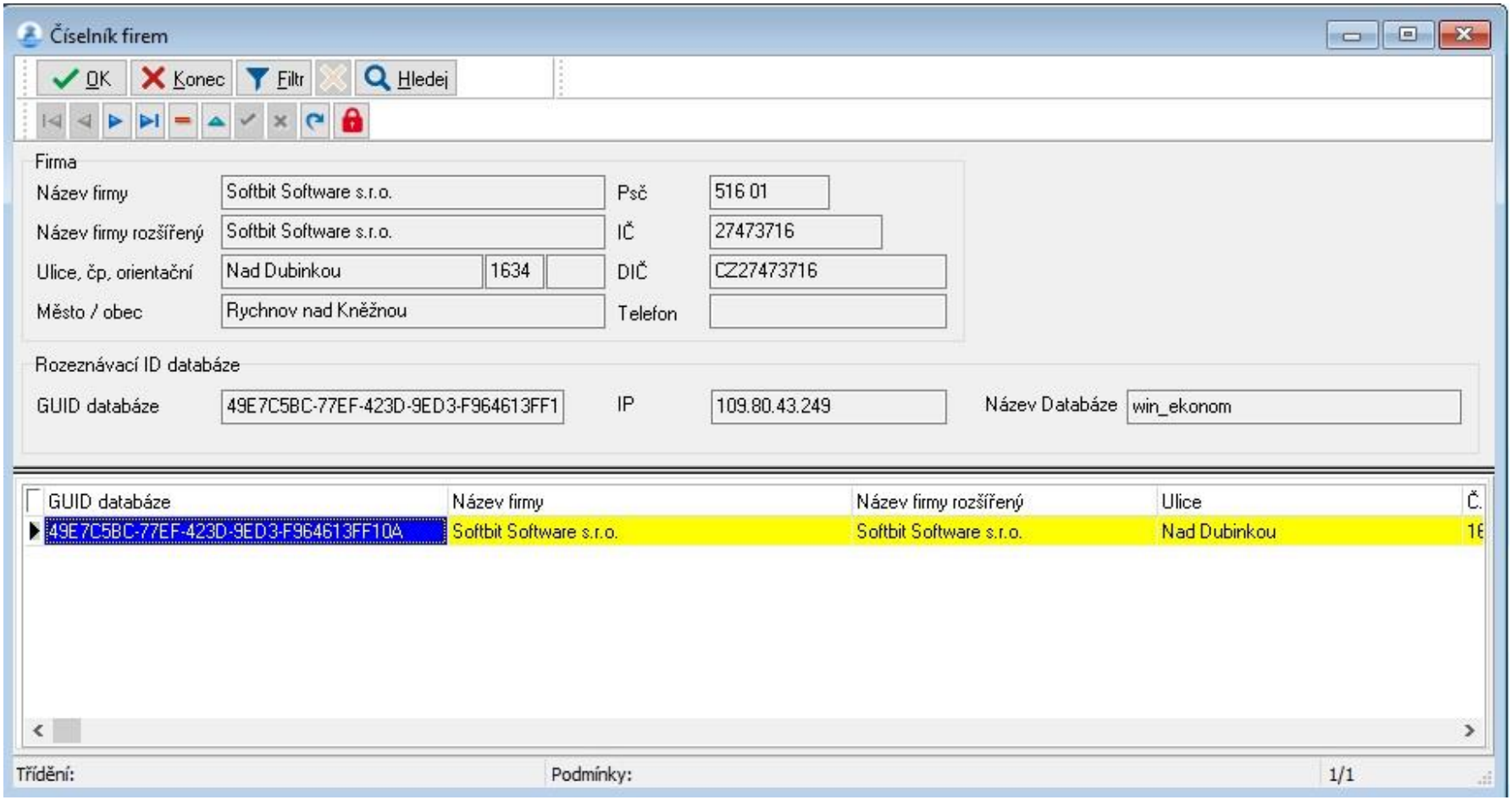
XIX. Obr 34: helpdesk komunikace – fakturační informace o požadavku



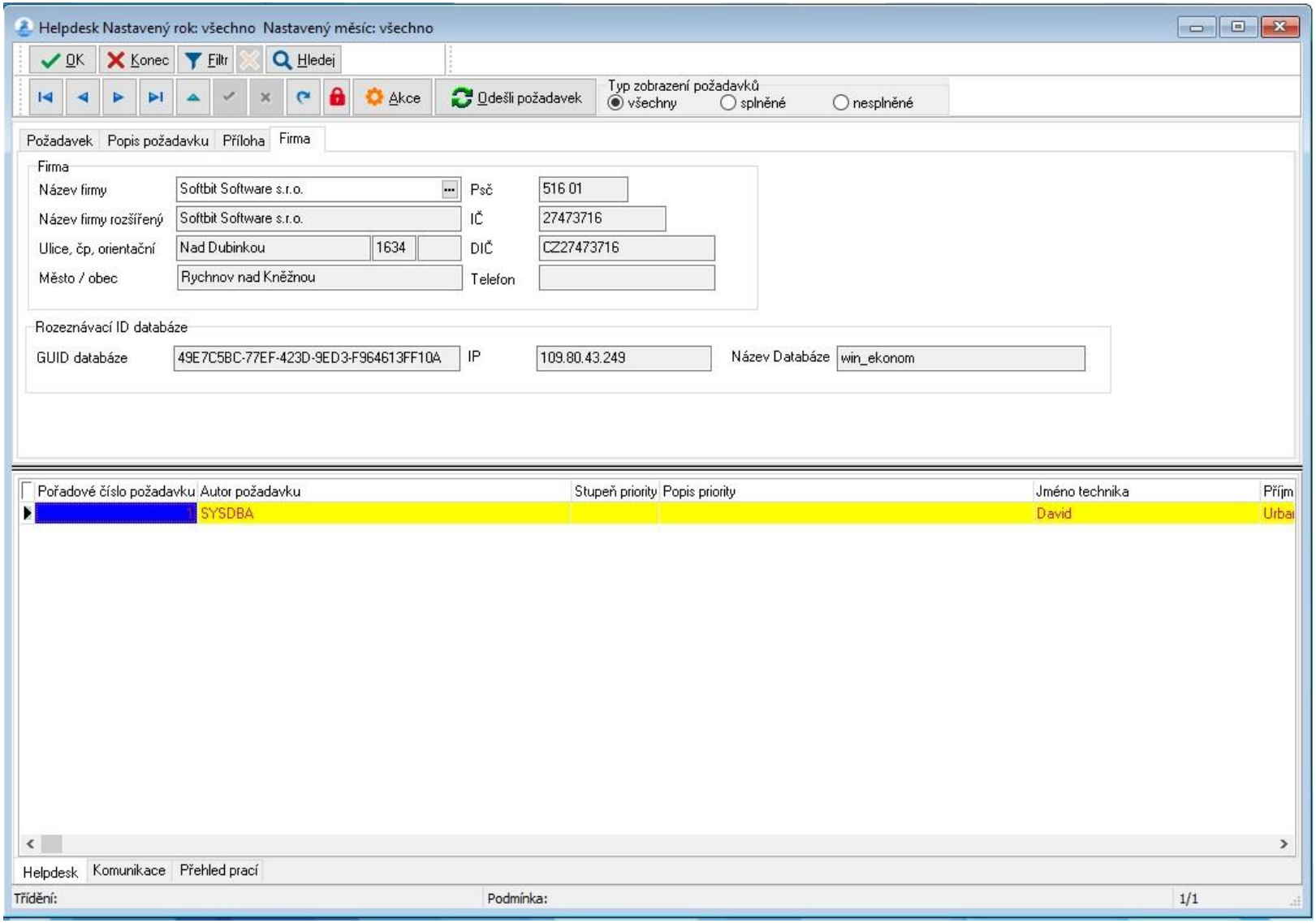
XX. Obr 35: helpdesk – přehled prací



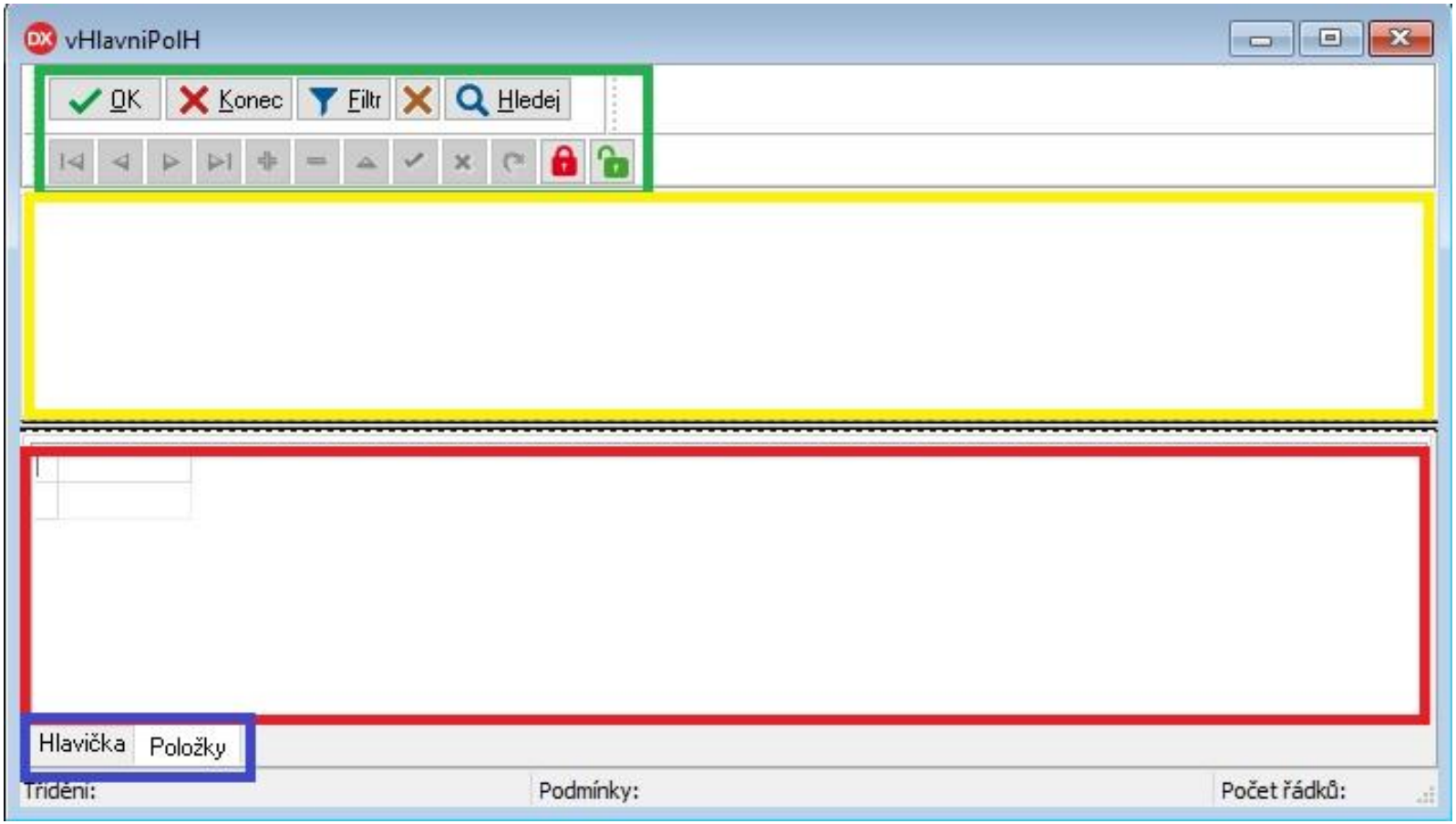
XXI. Obr 36: číselník firem pro serverovou aplikaci



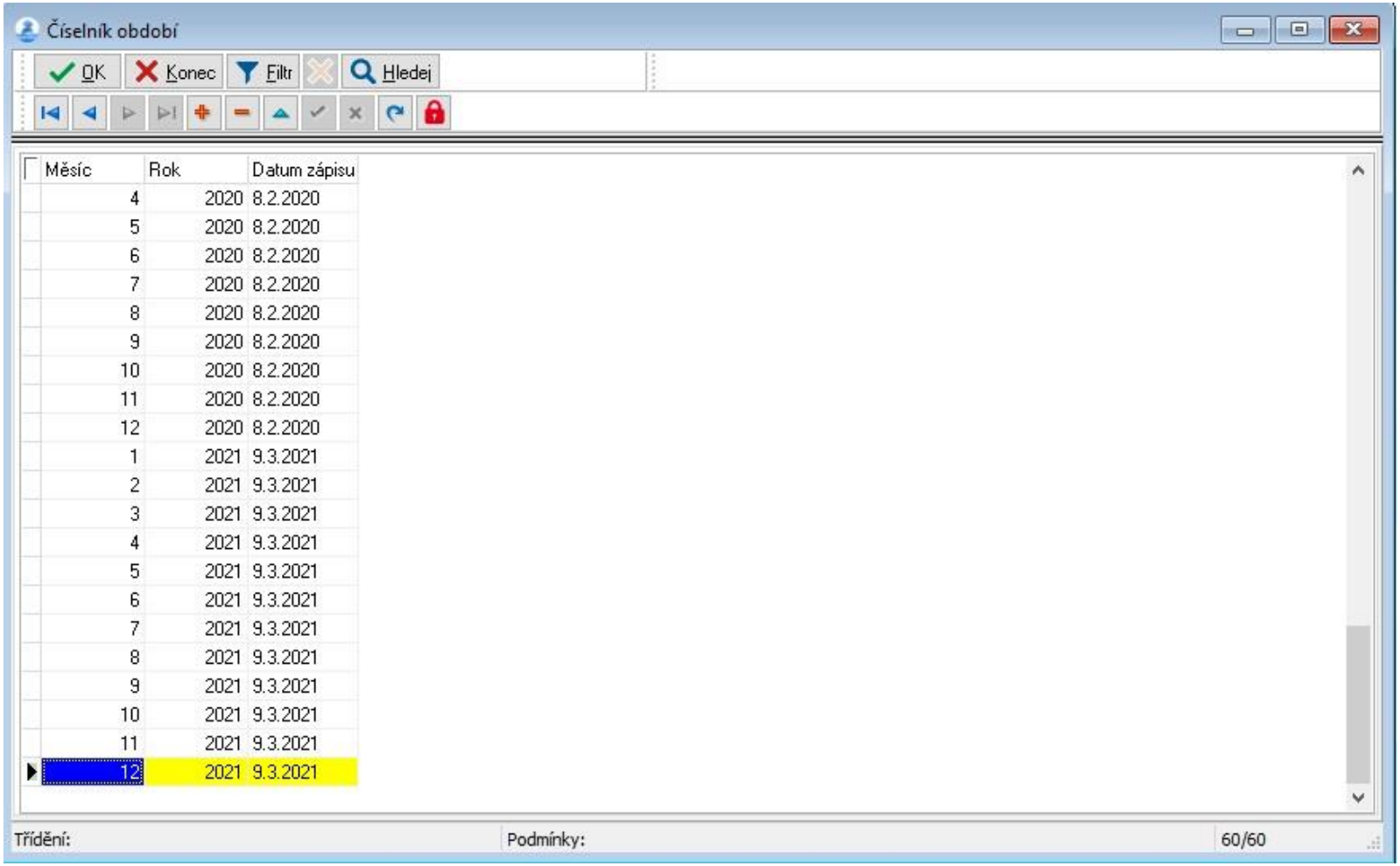
XXII. Obr 37: helpdesk server – firma



XXIII. Obr 38: šablona formuláře



XXIV. Obr 39: číselník období



Zadání práce (kopie)

Zadání bakalářské práce

Autor:	**David Urban**
Studium:	I1700153
Studijní program:	B1802 Aplikovaná informatika
Studijní obor:	Aplikovaná informatika
Název bakalářské práce:	**Užití SQL pro vývoj ekonomických systémů**
Název bakalářské práce AJ:	SQL and Design of Economic Systems

Cíl, metody, literatura, předpoklady:

Cíl práce: Objasnění principů relační databáze a SQL jazyka a naprogramovat aplikaci pro řízení požadavků a komunikace mezi zákazníkem a firmou poskytující IT služby.

Osnova:

1. Úvod
2. Teoretická část
   2.1. Relační databáze a užití SQL jazyka pro jejich manipulaci
   2.2. Principy komunikace mezi databázemi
3. Praktická část
   3.1. Použité nástroje a programovací jazyk
   3.2. Prezentace zpracování samotné aplikace
4. Zhodnocení praktičnosti a využití aplikace
5. Závěr
6. Seznam použitých zdrojů a použité literatury
7. Přílohy

Garantující pracoviště:	Katedra informatiky a kvantitativních metod, Fakulta informatiky a managementu
Vedoucí práce:	doc. RNDr. Petra Poulová, Ph.D.
Oponent:	Ing. Karel Malý, Ph.D.
Datum zadání závěrečné práce:	14.1.2018