VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

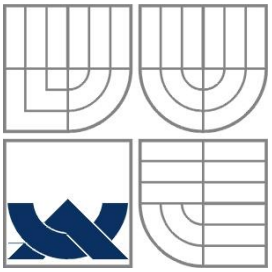
PŘÍSTUPOVÝ SYSTÉM VUT

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

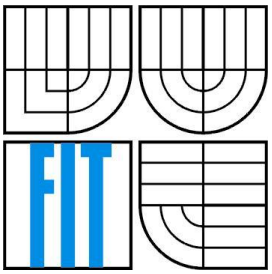
AUTOR PRÁCE
AUTHOR

BC. VÁCLAV BEZDĚK

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

PŘÍSTUPOVÝ SYSTÉM VUT

ACCESS SYSTEM BUT

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — BC. VÁCLAV BEZDĚK
AUTHOR

VEDOUCÍ PRÁCE — ING. JAROMÍR MARUŠINEC, PH.D., MBA
SUPERVISOR

BRNO 2008

Abstrakt

Tato diplomová práce se zabývá návrhem a vytvořením modulu Přístupový systém pro informační systém Apollo na Vysokém Učení Technickém v Brně. Cílem práce je analyzovat technologii Oracle a vybraná datová schémata přístupového systému a následně z výsledků analýzy navrhnout a implementovat aplikaci, jež umožní nastavování přístupů a sledování průchodů na čtečkách identifikačních karet, které jsou základními prvky fyzického rozhraní přístupového systému.

Projekt je vytvářen ve vývojovém prostředí Delphi 7 firmy Borland.

Klíčová slova

Přístupový systém, Oracle, Apollo, SQL, Delphi, Čtečka identifikačních karet, Trasa

Abstract

This master's thesis deals with design and implementation of program unit Access System for BUT Information System Apollo. The goal of this work is to analyze Oracle technology and chosen database schemes of access system. After that use results of analysis to design and to implement of application which provide functionality to creating access to the identification cards readers and support inspection of passing through identification cards readers.

Project is creating in Borland Delphi 7.

Keywords

Access system, Oracle, Apollo, SQL, Delphi, Identification Card Reader, Route

Citace

Bezděk Václav: Přístupový systém VUT. Brno, 2008, diplomová práce, FIT VUT v Brně.

Přístupový systém VUT

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Jaromíra Marušince, Ph.D., MBA.
Další informace mi poskytli Ing. Miloš Trávníček a Ing. Rudolf Musil.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Jméno Příjmení
Datum

Poděkování

Chtěl bych poděkovat svému vedoucímu Ing. Jaromíru Marušincovi, Ph.D., MBA za poskytnutí všech potřebných informací a kvalitní vedení při práci. Dále bych chtěl poděkovat všem pracovníkům Centra výpočetních a informačních služeb, kteří mi poskytli potřebné informace a důležité rady, především Ing. Rudolfu Musilovi a Ing. Miloši Trávníčkovi. A poděkování zaslouží i můj bratr Ing. Michal Bezděk, který mi poradil se stylistickou úpravou textu.

Obsah

1 Úvod

Na zabezpečení majetku, důležitých informací nebo citlivých údajů se ve velkých společnostech vynakládá značné množství finančních prostředků. Tyto společnosti si dokáží vyčíslit, že ztráta z úniku informací několikanásobně převyšuje náklady na pořízení zabezpečovacích zařízení. Součástí těchto zařízení jsou nejrůznější mechanismy, jejichž úkolem je zabránit útočníkovi, aby se dostal k cenným údajům nebo zařízením. Ať už jsou tyto zábrany mechanického nebo programového rázu, tak je spojuje jedna slabina. Tou je lidský faktor. Člověk dokáže svou chybou degradovat i nejkvalitnější bezpečností opatření na tak nízkou úroveň, že útočník ani nezaznamená, že nějaké zabezpečení překonal. Proto je zde snaha omezit účast člověka u bezpečnostních zařízení na minimum pomocí zautomatizování všech postupů, které jdou zautomatizovat. A pokud nějaká činnost nejde jednoduše automatizovat, potom je zde snaha co nejvíce usnadnit uživateli provádění této činnosti tak, aby se co nejvíce snížilo riziko možné chyby.

Přístupový systém je jedním z velmi důležitých zabezpečovacích mechanismů, jelikož může kontrolovat, případně omezit pohyb cizích, čili potencionálně nebezpečných osob, po areálu organizace, která přístupový systém používá.

Vytvořením kvalitního přístupového systému lze uživateli výrazně usnadnit některé činnosti hrající významnou roli v otázce zabezpečení. Jednou z nejdůležitějších činností je odebrání všech přístupových práv osobám, které přestanou pro společnost pracovat. Právě tito lidé totiž mohou být největší hrozbou. Zároveň je odebrání všech přístupových práv osobě, jednou z těch činností, jejichž automatizace je velmi žádaná. Další předností dobrého přístupového systému je jednoduché a přehledné spravování přístupových práv jednotlivých uživatelů. Právě tato jednoduchá a přehledná správa vede k podstatnému snížení rizika vzniku lidské chyby. Je potřeba zmínit, že pokud dojde k nějakému bezpečnostnímu incidentu, tak přístupový systém musí odhalit osobu, která je zodpovědná za narušení bezpečnosti a zároveň musí tuto skutečnost prokazatelně dokázat.

1.1 Současný stav

V současné době existuje na celém Vysokém učení technickém více různých systémů, které spravují čtecí zařízení a řídí přístupy. Dokonce lze nalézt několik odlišných systému přímo na jedné fakultě. Pokud nastoupí na Vysoké učení nový student nebo zaměstnanec, musí být informace o jeho nástupu zavedena do všech systémů. Podobně je tomu při ukončení studia nebo zaměstnaneckého poměru. V takovém případě musí být osoba ze všech systémů vymazána. Tím se zvyšuje pravděpodobnost lidské chyby, protože se může stát, že odpovědný pracovník zapomene odebrat osobu z některého systému, tudíž tato osoba bude mít stále přístup do některé části Vysokého učení technického.

1.2 Cílový stav

Cílem je vytvořit jednotný přístupový systém, který bude komunikovat se všemi existujícími systémy různých dodavatelů. Odstraněním ručního předávání informací mezi přístupovými systémy se zmenší riziko lidské chyby. Tímto zautomatizováním se taktéž odstraní nebezpečí toho, že člověk, který už nebude v evidenci jako aktivní student nebo zaměstnanec, bude mít funkční přístupovou kartu.

Rozhraní systému bude jednoduché, přehledné a uživatelsky příjemné. Pomocí rozhraní budou moci odpovědné osoby spravovat přístupy na jednotlivých fakultách. Díky jednotnému přístupovému systému nebude problém povolit přístup do učeben či prostor fakulty i studentům z ostatních fakult.

Pomocí provázání přístupového systému se systémem identifikačních karet VUT, se budou do centrální databáze VUT ukládat údaje o pohybu osob po areálech VUT. Díky tomu půjde jednoduše zpětně dohledat, kde se kdo nacházel v daném čase.

1.3 Návaznost na předchozí práce

Tato práce navazuje na semestrální projekt, jenž jsem vytvářel v minulém semestru. Obsahem semestrálního projektu bylo provedení analýzy technologie Oracle a vybraných datových schémat přístupového systému. Informace ze semestrální práce jsem použil v kapitole 2 a v kapitole 3.

1.4 Stručný obsah jednotlivých kapitol

Kapitola 1: Úvod

Tato kapitola obsahuje úvod do problematiky přístupového systému, ukazuje současný stav přístupových systémů na VUT v Brně a popisuje cíl diplomové práce. Také je zde uvedena návaznost na předchozí práce.

Kapitola 2: Databázový systém Oracle

Kapitola popisuje technologii databáze Oracle, její architekturu a strukturu. Jsou zde zmíněny databázové objekty a informace o nich. Dále je v této kapitole uveden jazyk SQL a jsou zmíněny jeho vlastnosti.

Kapitola 3: Analýza

V této kapitole je provedena analýza existujícího databázového schéma přístupového systému. Jsou zde popsány jednotlivé tabulky. Dále je zde vysvětlen důvod změny již existujícího databázového schématu.

Kapitola 4: Návrh

V této kapitole je popsán postup navrhování vzhledu a funkčnosti aplikace. Návrh aplikace se taktéž zabýval rozdělením rolí v přístupovém systému a vytvořením diagramu případů použití, který ukazuje dostupné funkce pro jednotlivé role.

Kapitola 5: Implementace

Tato kapitola se zabývá implementací navržené aplikace do informačního systému Apollo. Je zde podrobně uvedeno, jak funguje informační systém Apollo. Poté už následuje popis samotné implementace aplikace a komponent, použitých při implementaci.

Kapitola 6: Nasazení do provozu

Zde je popsáno nasazení funkční aplikace do informačního systému Apollo a způsob předvedení klíčovým uživatelům.

Kapitola 7: Závěr

Tato kapitola obsahuje zhodnocení výsledků diplomové práce. Dále je zde naznačen možný směr budoucího rozvoje aplikace. Nakonec je zmíněn přínos diplomové práce z hlediska mé budoucí kariéry.

Kapitola 8: Literatura

Kapitola Literatura obsahuje seznam použité literatury.

Kapitola 9: Seznam příloh

V této jsou uvedeny všechny přílohy této diplomové práce. Je zde uživatelská příručka, která podrobně popisuje základy práce s modulem Přístupový systém. Dále jsou v této kapitole uvedeny některé významné SQL dotazy, které jsem vytvořil.

2 Databázový systém Oracle

Databáze je sada dat. Oracle umožňuje ukládat data a získávat přístup k nim způsobem odpovídajícím definovanému modelu, který je znám jako relační model. Z tohoto důvodu je Oracle považován za systém zprávy relační databáze, či chcete-li za relační systém řízení báze dat. Více v [1].

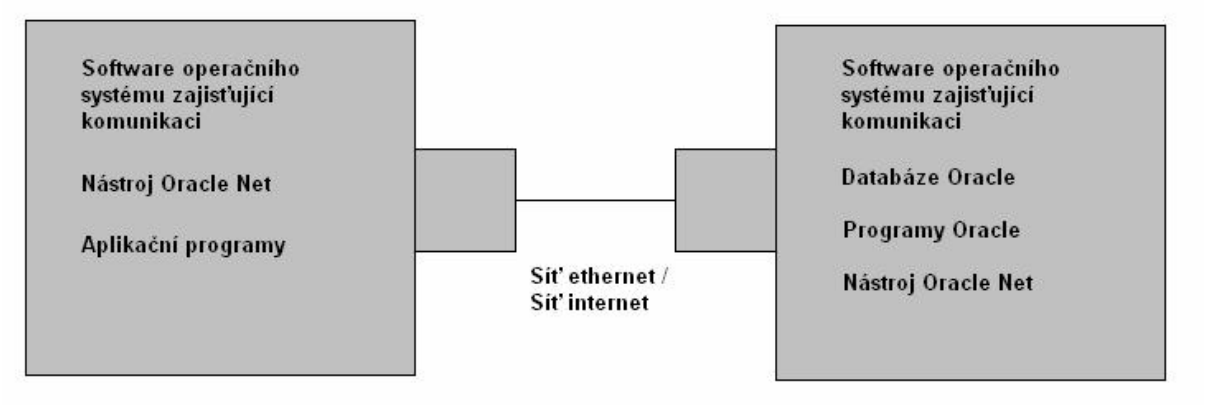
Vysoké učení technické v Brně používá pro správu dat relační databázi od firmy Oracle ve verzi 10. Tato verze obsahuje také různé objektové rozšíření, tudíž se nejedná o čistě relační databázi, ale spíše o objektově-relační databázi. Někdy se takovým databázím říká postrelační. K databázi se přistupuje v rámci architektury klient/server. Pro počáteční seznámení se s obecnou strukturou databáze jsem použil internet, kde jsem našel webovou stránku, viz [3], která mi poskytla potřebné informace. Podrobnější informace přímo o databázovém systému od firmy Oracle jsem následně hledal v knižní podobě, viz publikace [1] a [2].

2.1 Architektura klient/server

Ke vzdálené databázi může mít také přístup počítač, který databázi nezahrnuje. Zpravidla se to děje tak, že aplikační programy v jednom počítači mají přístup k databázi v druhém hostitelském počítači. V této konfiguraci, kterou naleznete na obrázku 1, se počítač, v němž je spuštěna aplikace, nazývá klient, a počítač obsahující databázi je označován jako server.

Z obrázku 1 je zřejmé, že klient musí být prostřednictvím sítě schopen komunikovat se serverem. Aplikační programy jsou spuštěny na straně klienta a databáze je využívána hlavně pro vstupně-výstupní operace.

Zatížení procesoru související se spuštěním aplikace tedy místo serveru přebírá klientský počítač. Podmínkou funkčnosti této konfigurace je, že v klientu musí být spuštěny odpovídající procesy technologie Oracle Net. Pokud aplikační program na straně klienta vyzve uživatele k zadání informací o připojení databáze, měl by uživatel specifikovat název služby. Aplikace potom otevře relaci ve vzdálené databázi. Více viz [1].



Obrázek 1: Architektura klient-server

Jak bylo zmíněno, architektura klient/server obsahuje dvě části:

- Klientská aplikace – klient je aktivní část architektury, vytváří SQL dotazy, které pak zasílá na server.
- Databázový server – server je pasivní část architektury, čeká na dotazy od klienta, přijímá je, zpracovává a následně zasílá klientovi jejich výsledky.

2.2 Struktura relační databáze Oracle

Na začátek je potřeba uvést na pravou míru nadpis této kapitoly. Jak bylo uvedeno v úvodu této kapitoly, databázové systémy firmy Oracle nabízejí již delší dobu rozšíření pro podporu objektového přístupu, tudíž se vlastně jedná o databázi postrelační. Struktura databáze Oracle se dělí na dvě části, na logickou a fyzickou strukturu.

2.2.1 Logická struktura

Logická konfigurace databáze velmi ovlivňuje výkon databázového prostředí a usnadňuje její správu.

Logické objekty je nutné v databázi roztřídit podle způsobu jejich použití a vlivu jejich fyzických struktur na databázi. Tento proces klasifikace zahrnuje oddělení tabulek od jejich indexů a oddělení tabulek s vysokou aktivitou od tabulek s nízkou aktivitou. Přestože stupeň aktivity vůči objektům lze určit pouze během běžného provozu, je možné oddělit hlavní sadu tabulek s vysokým využitím. Mezi další faktory klasifikace lze zařadit data pouze pro čtení, přenosná data a typ požadované zprávy. Více viz [1].

Logická struktura je množina objektů a vztahů mezi nimi. Mezi databázové objekty patří:

- Databázová tabulka
- Pohled
- Materializovaný pohled
- Sekvence

- Index
- Uložená procedura
- Uložená funkce
- Trigger

Databázová tabulka

Tabulky slouží v databázi Oracle jako úložiště dat. Obsahují pevný počet sloupců. Sloupce popisují atributy entity sledované tabulkou. Všechny sloupce mají určitý název a specifické vlastnosti. Sloupci je přiřazen datový typ a délka.

Tabulky jsou vzájemně provázány společnými sloupci. Tyto vzájemné vztahy lze v databázi vynutit pomocí referenční integrity. Používáme-li objektově orientované funkce systému Oracle, můžeme vytvořit vzájemné vazby mezi řádky prostřednictvím odkazů nazývaných identifikátory ID objektů. Referenční integritu můžete na úrovni databáze vynutit prostřednictvím integritních omezení[1]. Viz [1].

Pohled

V knize Mistrovství v Oracle, viz [1], je pohled definován takto:

Pohled je v podstatě tabulka zahrnující sloupce a je dotazováno stejným způsobem jako tabulka. Pohled však neobsahuje žádná data. Pohled lze definovat jako masku překrývající jednu nebo více tabulek tak, aby sloupce v pohledu bylo možné najít v jedné nebo více příslušných tabulkách. Pohled tedy k uložení dat nevyužívají fyzické úložiště. Definice pohledu (zahrnující dotaz, na němž je pohled založen, rozložen sloupců a udělená oprávnění) je uložena v systémovém katalogu.

Jestliže zadáte dotaz na pohled, dotáže se server tabulek, na kterých je pohled založen a vrátí hodnoty ve formátu a pořadí určeném definicí pohledu. Pohled nelze indexovat, protože přímo nesouvisí s žádnými fyzickými daty.

Pohledy jsou často používány k vynucení zabezpečení dat na úrovni řádků tabulky. Které uživatel vlastní, a zároveň neudělit uživatelský přístup ke všem řádkům v tabulce. Stejně tak je možné omezit sloupce, ke kterým bude mít uživatel prostřednictvím příslušného pohledu přístup.

Materializovaný pohled

Na rozdíl od „klasického“ pohledu, který obsahuje jen předpis pro zobrazení dat z jedné, případně více databázových tabulek, materializovaný pohled skutečně požadovaná data obsahuje. Materializované pohledy se často používají v datových skladech (datawarehouse) pro shromažďování

[1] Integritní omezení je funkce databáze, která nedovolí vložit do „omezených“ sloupců tabulky nevhodné údaje. Existují tyto typy integritních omezení: NOT NULL, PRIMARY KEY, UNIQUE KEY, FOREIGN KEY a CHECK

agregovaných dat. Při vytváření materializovaného pohledu určujeme způsob aktualizace pohledu vzhledem ke změnám zdrojových tabulek. Materializovaný pohled může být uložený v té samé databázi jako zdrojové tabulky, nebo v jiné, i vzdálené databázi (například i na mobilním zařízení třídy PocketPC apod.) Materializované pohledy jsou určeny pro sumarizace, předběžné výpočty, distribuci dat apod. Více viz [2].

Sekvence

Jak je napsáno v [1]: *Sekvence usnadňují programování, protože vytvářejí postupný seznam jedinečných čísel. Při prvním volání sekvence je vrácena předdefinovaná hodnota. Všechny následné dotazy na sekvenci vrátí hodnotu zvětšenou o určený přírůstek. Sekvence se mohou opakovat nebo se stále zvětšovat, až dosáhnou zadané maximální hodnoty.*

Index

Index je databázová struktura používaná serverem k rychlému vyhledávání řádku tabulky. Existují tři základní druhy indexů: indexy clusterů, indexy tabulek a bitmapové indexy. Indexy clusterů[2] *uchovávají klíčové hodnoty clusterů. V indexu tabulky jsou uloženy hodnoty řádků tabulky a fyzické umístnění jednotlivých řádků (jejich identifikátory RowID). Bitmapový index je zvláštní druh indexu tabulky určený k podpoře dotazů velkých tabulek se sloupci obsahující několik rozdílných hodnot.*

Vyhledávání dat je v systému Oracle uskutečňováno pomocí identifikátoru RowID, který je přiřazen ke každému řádku ve všech tabulkách. Pomocí tohoto identifikátoru databáze přesně zjistí, kde se daný řádek nachází (umístění je určeno podle souboru, bloku v rámci tohoto souboru a řádku v rámci tohoto bloku).

*Položky indexu jsou v systému Oracle ukládány pomocí mechanismu B*tree, který zajišťuje rychlý přístup ke klíčové hodnotě.*

Indexy zvětšují výkon databáze a (volitelně) zajišťují jedinečnost sloupce.

Index lze vytvářet pro jeden nebo více sloupců tabulky. Další podrobnější informace viz [1].

Uložená procedura

Uložená procedura je blok příkazů PL/SQL. Je uložena v systémovém katalogu a je volána aplikacemi. Pomocí procedur lze uchovávat často používanou aplikační logiku v rámci databáze. Spuštěním procedury spustíte její příkazy jako celek. Procedury nevracejí volajícímu programu žádné hodnoty.

Uložené procedury můžete použít například k vynucení zabezpečení dat. Místo udělení oprávnění k přímému přístupu k tabulkám v rámci aplikace můžete udělit oprávnění ke spuštění

[2] Cluster interní struktura databáze, která určuje, které tabulky budou fyzicky uchovávány společně. Do clusteru je výhodné přidat tabulky, které se často dotazují společně. Tím, že jsou fyzicky uloženy společně, se sníží počet nutných vstupně/výstupních operací, což zvýší výkon databáze.

procedury, která má k těmto tabulkám přístup. Spuštěním procedury se aktivují i oprávnění vlastníka procedury. Uživatelé tedy nebudou moci získat přístup k tabulkám jinak než prostřednictvím dané procedury. Viz [1]. O jazyku SQL a jeho verzi PL/SQL se zmíním později

Uložená funkce

Uložené funkce jsou podobně jako procedury bloky kódu uložené v databázi. Funkce však na rozdíl od procedur mohou volajícímu programu vracet hodnoty. Můžete spouštět funkce, které poskytuje přímo platforma Oracle, nebo vytvářet vlastní funkce a volat je pomocí příkazů SQL. Více v [1].

Trigger

Triggery jsou procedury, které se spouštějí v případě, že dojde ke specifikované události databáze. Můžete je použít ke zvětšení referenční integrity, vynucení dodatečného zabezpečení nebo vylepšení dostupných možností auditování.

Existují dva typy triggerů:

Triggery příkazů – Spustí se jednou při každém odpovídajícím příkazu

Triggery řádků – Spustí se jednou pro každý řádek tabulky ovlivněný daným příkazem

Triggery příkazů jsou vhodné, pokud není kód akce triggeru závislý na ovlivněných datech.

Triggery řádků jsou užitečné v případě, že akce triggeru závisí na datech ovlivněných transakcí. Tyto a další informace se dají nalézt v [1]

2.2.2 Fyzická struktura

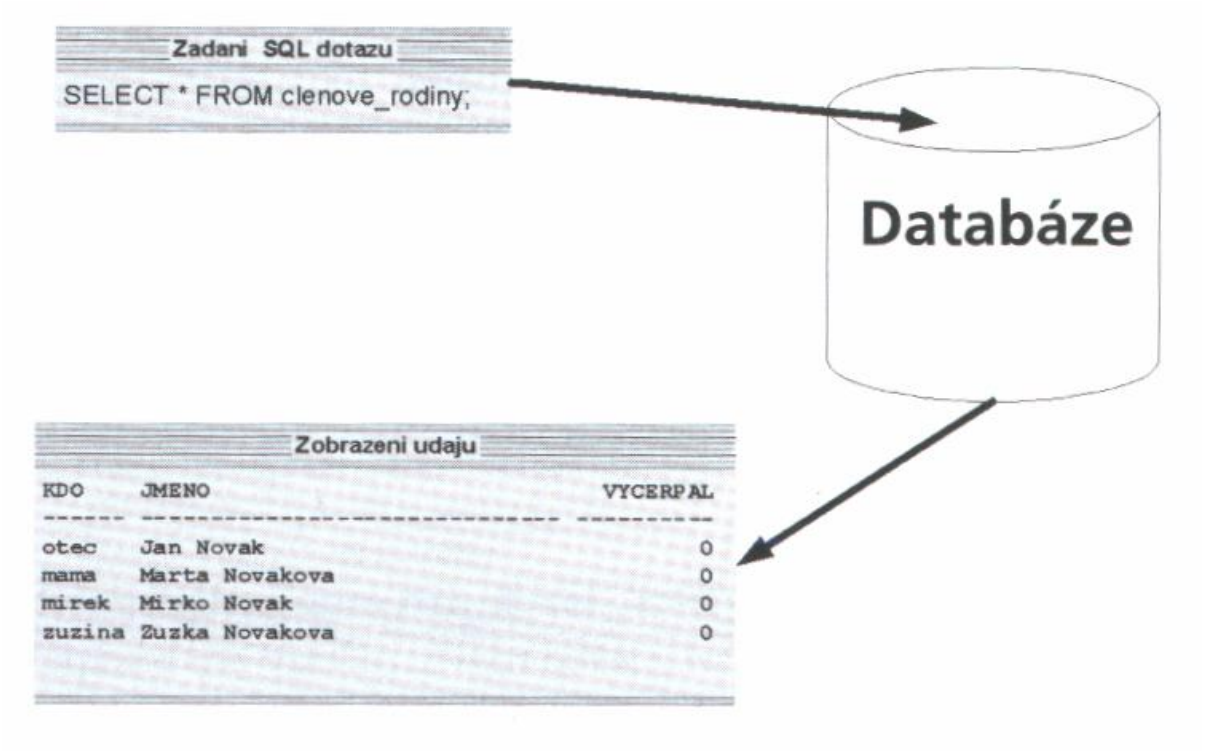
Fyzická struktura databáze zastřešuje soubory operačního systému uložené na disku. Uživateli je skryta, ten vidí pouze logickou strukturu databáze.

Data jsou v databázi ukládána do fyzických souboru (tzv. datových souborů) na disku. Jsou-li tato data používána, ukládají se do paměti. Oracle zvyšuje výkon databáze a řídí sdílení dat mezi uživateli pomocí paměťových oblastí. Viz [1].

2.3 Jazyk SQL

Databázový jazyk SQL (Structured Query Language) vznikl na základě projektu společnosti IBM, tehdy se ovšem ještě jmenoval SEQUEL (Structured English Query Language). Cílem projektu bylo vytvořit jazyk blízký angličtině pro práci s daty v databázích. Později na vývoji databázového jazyka SQL spolupracovali také další firmy, přičemž komerčně jej uvedla do praxe právě společnost Oracle. Postupem času se ujaly vylepšené a upravené standardy tohoto jazyka s označením SQL 86 a SQL 92. Pro verzi SQL 92 se vžilo zkrácené označení SQL 2. V současné době se již několik let pracuje na novém standardu s názvem SQL 3.

SQL je dotazovací jazyk, přičemž základní princip komunikace s relačním databázovým serverem je možné nejjednodušeji vyjádřit obrázkem 2.



Obrázek 2: Princip zpracování příkazu jazyka SQL

Řečeno jednou větou – klient zformuluje a položí svůj dotaz a databázový server na něj odpoví, obvykle tím, že vygeneruje nějakou množinu výstupních dat. Tento princip komunikace s databázovým serverem je velmi jednoduchý a efektivní. Samozřejmě ale jen z pohledu uživatele. Jazyk SQL totiž připomíná klasický přirozený jazyk (anglický), má ale přesně definovaná syntaktická a lexikální pravidla. Z pohledu serveru se příkaz SQL přenese, dekóduje, optimalizuje a provede. Viz [2].

Jak bylo zmíněno, pro práci s jazykem SQL se používají příkazy. Tyto příkazy se dají podle logiky použití rozdělit do několika skupin:

- Jazyk pro definici dat (DDL, Data Definition Language)
- Jazyk pro manipulaci s daty (DML, Data Manipulation Language)
- Jazyk pro správu dat (DCL, Data Control Language)
- Příkazy pro řízení transakcí (TCC, Transaction Control Language)

2.3.1 Jazyk pro definici dat

V [2] je napsáno: *Pomocí příkazů této skupiny můžeme definovat, vytvářet, měnit a rušit (odstraňovat) různé databázové struktury, jako jsou například tabulky, indexy, triggery, uložené procedury apod. Taktéž můžeme přidělovat a odebírat uživatelská oprávnění jednotlivým uživatelům a skupinám uživatelů. Do této skupiny patří například příkazy:*

CREATE DATABASE *CREATE TABLE* *ALTER TABLE*

DROP TABLE	*DROP VIEW*	*CREATE PROCEDURE*
CREATE INDEX	*DROP INDEX*	*DROP PROCEDURE*
DROP INDEX	*CREATE SEQUENCE*	*CREATE TRIGGER*
CREATE VIEW	*ALTER SEQUENCE*	*DROP TRIGGER*
ALTER VIEW	*DROP SEQUENCE*	

2.3.2 Jazyk pro manipulaci s daty

V [2] se píše: *Do této skupiny patří, jak již vyplývá z jejího názvu, příkazy pro manipulaci s daty, tedy příkazy pro vkládání, aktualizaci, mazání dat a samozřejmě také velmi mocný dotazovací příkaz – SELECT.*

SELECT	*UPDATE*
INSERT	*DELETE*

2.3.3 Jazyk pro správu dat

Jak uvádějí v [2]: *Tato skupina zahrnuje speciální příkazy pro řízení provozu a údržby databáze. Patří sem například:*

GRANT	*ALTER USER*	*GRANT*
REVOKE CREATE USER	*DROP USER*	*REVOKE*

2.3.4 Příkazy pro řízení transakcí

V [2] říkají: Vedle nejznámějších skupin příkazů jazyka SQL můžeme vyčlenit také skupinu příkazů pro řízení transakcí. Do této skupiny patří například příkazy:

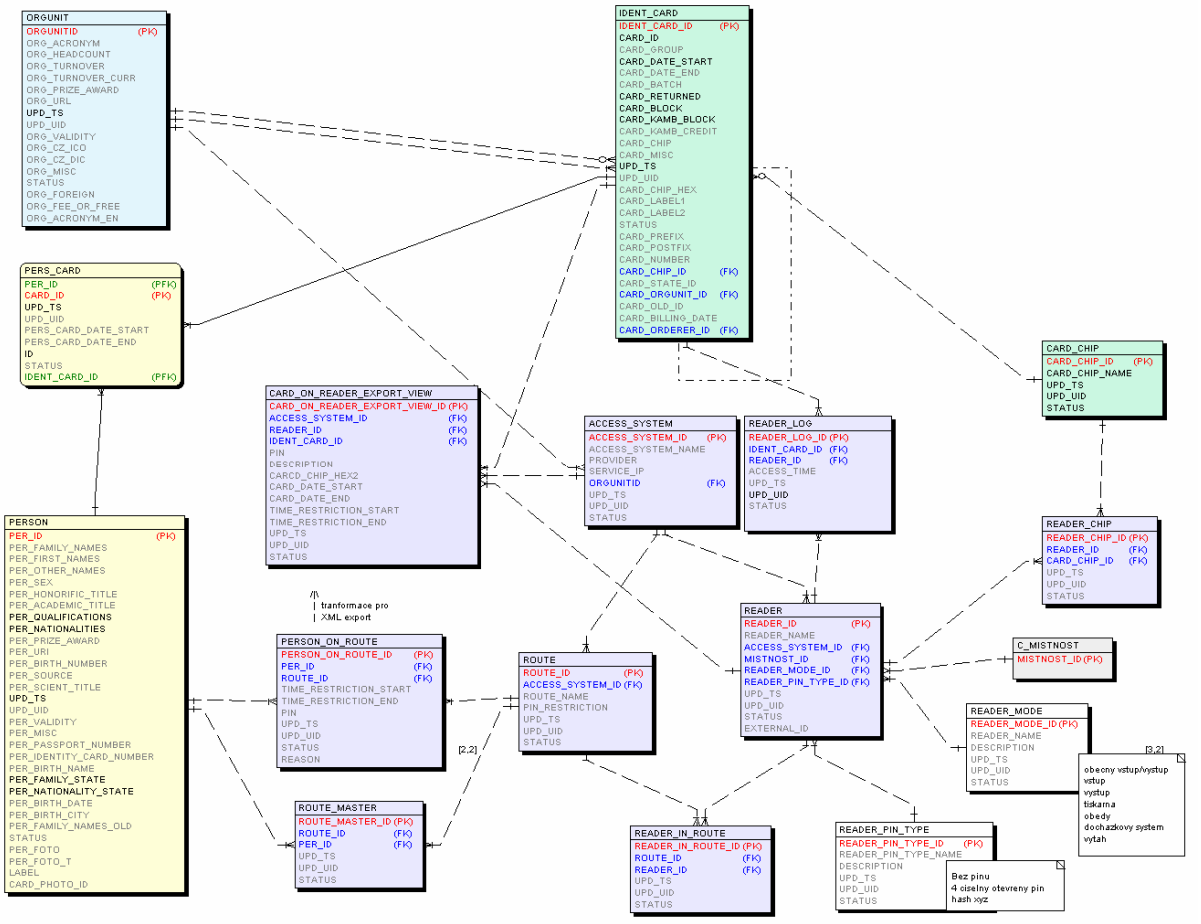
SET TRANSACTION	ROLLBACK
COMMIT	SAVEPOINT

3 Analýza existujícího schéma

V centrální databázi VUT v Brně již existuje návrh databázových tabulek pro přístupový systém. Jeho schéma znázorňuje **obrázek 3.**

3.1 Rozbor existujícího schéma

Hlavní tabulkou přístupového systému je tabulka ACCESS_SYSTEM, ve které jsou uloženy informace o přístupových systémech. Na tuto tabulku jsou navázány tabulky ROUTE a READER. Zmíněné tabulky slouží k definování tras v přístupovém systému (tabulka ROUTE), a k uložení informací o čtečkách (tabulka READER). Tyto dvě tabulky jsou propojeny vazební tabulkou READER_IN_ROUTE, která určuje čtečky patřící na určitou trasu, respektive v jakých trasách je zahrnuta specifická čtečka.



Obrázek 3: původní schéma tabulek přístupového systému

Dále lze ve schéma přístupového systému najít vazební tabulky PERSON_ON_ROUTE a ROUTE_MASTER. Tabulka PERSON_ON_ROUTE je poskytuje informace o osobách, které mají

přístup na určitou trasu, respektive poskytuje údaje o tom, na jaké trasy má sledovaná osoba přístup. Tabulka ROUTE_MASTER poskytuje informace o správcích trasy.

Tímto je uzavřen výčet tabulek souvisejících s trasami, a zbývají tabulky, které souvisejí s tabulkou READER, a poskytují doplňující informace o čtečkách. První z nich je tabulka READER_PIN_TYPE, která ukládá informace o tom, jestli čtečka vyžaduje zadání čísla PIN, aby umožnila osobě průchod. Další tabulka související se čtečkami je tabulka READER_MODE. Tato databázová tabulka určuje účel čtečky. Dále je zde možno vidět tabulku READER_CHIP. Tato vazební tabulka udává typ čipu, který čtečka zvládne přečíst. Samotná tabulka READER obsahuje sloupec, jenž spojuje konkrétní čtečku místností, před kterou se čtečka nachází.
V databázovém schéma přístupového systému zbývají už jen dvě tabulky, o kterých je potřeba se zmínit, protože úzce souvisejí s přístupovým systémem. Tabulka READER_LOG, která uchovává informace o průchodu čtečkou. CARD_ON_READER_EXPORT_VIEW je databázový pohled, který se používá pro exportování údajů o přístupovém systému z tabulek centrální databáze VUT do systémů provozovatelů jednotlivých přístupových systémů.

Z pojmenování tabulek a atributů tabulek lze poznat, že jsou vytvořeny podle konvence zavedené v centrálním datovém skladu CDBX. Názvy tabulek a atributů tabulek mají anglické pojmenování. Jsou výstižné a co nejjednodušší. Primární klíč každé tabulky je označen vždy ve tvaru *název_tabulky* + *„_id“*. Tato vlastnost nedovolí do databáze vnášet nesprávné hodnoty klíčových položek. Cizí klíče mají většinou shodný název jako atributy ve své mateřské tabulce.

Každá tabulka obsahuje vždy tyto tři atributy:

- upd_ts
- upd_uid
- status

Upd_ts je časové razítko poslední změny záznamu. Díky tomuto atributu lze poznat, kdy byla provedena poslední změna záznamu. S tím úzce souvisí atribut **upd_uid**, který udává osobu, která jako poslední provedla změnu záznamu. Když je potřeba, tak s pomocí těchto dvou atributů je možné z databázového logu poznat, kdy která osoba měnila položky jakékoliv tabulky.

Atribut **status** označuje, zda je záznam platný či nikoliv. Status může nabývat hodnot „0“ a „9“ pokud je záznam platný a nebo hodnot „1“, respektive „-1“ pokud je záznam neplatný, respektive neplatný a určený ke smazání.

3.2 Databázové tabulky přístupového systému

V této kapitole jsou shrnuty poznatky o jednotlivých tabulkách, jež byly získány rozborem dostupného schéma přístupového systému.

3.2.1 Tabulka ACCESS_SYSTEM

Tabulka ACCESS_SYSTEM je hlavní tabulka schéma přístupového systému. Tato tabulka uchovává informace o jednotlivých přístupových systémech, jako je například jméno systému, externí provozovatel přístupového systému a organizační jednotka VUT (typicky fakulta), pod kterou tento přístupový systém spadá.

ACCESS_SYSTEM	**tabulka přístupových systémů**
access_system_id	id přístupového systému
access_system_name	název přístupového systému
provider	jméno provozovatele přístupového systému
service_ip	ip adresa serveru provozovatele přístupového systému
orgunitid	organizační jednotka

3.2.2 Tabulka ROUTE

Tabulka ROUTE poskytuje informace o trasách, jako je například jméno trasy nebo přístupový systém, pod který spadá. Důležité je, že každá trasa musí být zařazena pod právě jeden existující přístupový systém.

ROUTE	**tabulka tras**
route_id	id trasy
route_name	název trasy
access_system_id	přístupový systém, pod který trasa patří
pin_restriction	jestli je vyžadován pin na trase. Hodnoty 0 nebo 1

3.2.3 Tabulka PERSON_ON_ROUTE

Tato tabulka je vazební tabulkou mezi tabulkami PERSON a ROUTE. Uchovává informace o tom, které osoby mají přístup na určitou trasu, a naopak na jaké trasy má přístup sledovaná osoba.

Navíc tato vazební tabulka udržuje informace o časových omezeních přístupu na trasu, pokud nějaké existují.

PERSON_ON_ROUTE	**vazební tabulka osob na trase**
person_on_route_id	id záznamu
per_id	id osoby
route_id	id trasy
time_restriction_start	čas, od kdy má osoba povolený přístup na trasu
time_restriction_end	čas, do kdy má osoba povolený přístup na trasu
pin	PIN osoby na trase

3.2.4 Tabulka ROUTE_MASTER

Další vazební tabulka mezi tabulkami PERSON a ROUTE. Tato databázová tabulka, na rozdíl od tabulky PERSON_ON_ROUTE, určuje osoby odpovědné za správu trasy.

ROUTE_MASTER	**tabulka správců tras**
route_master_id	id záznamu
per_id	id osoby
route_id	id trasy

3.2.5 Tabulka READER

Tabulka READER ukládá informace o čtečkách, jako jsou například jméno čtečky, přístupový systém, pod který čtečka spadá. Dále tato tabulka obsahuje položky, které odkazují na místnost, kterou čtečka zabezpečuje, a na údaje o typu čtečky. Místnost nemusí být definovaná. Dále je v této tabulce uvedeno identifikační číslo čtečky v systému poskytovatele přístupového systému. Toto číslo se využívá při exportu dat do databází externích provozovatelů jednotlivých přístupových systémů.

Důležitou informací je, že každá čtečka musí být zařazena pod právě jeden přístupový systém.

READER	**tabulka čteček**
reader_id	id čtečky
reader_name	název čtečky
access_system_id	přístupový systém, pod který čtečka patří
mistnost_id	místnost, ve které se čtečka nachází
reader_mode_id	označení, o jaký druh čtečky se jedná
reader_pin_type_id	označení, jaký typ PINu čtečka používá
external_id	id číslo čtečky v externím systému providera

3.2.6 Tabulka READER_IN_ROUTE

Tabulka READER_IN_ROUTE je vazební tabulkou mezi tabulkami READER a ROUTE. Pomocí této tabulky se určuje, které čtečky se nacházejí na určité trase a naopak pod jaké trasy spadá daná čtečka.

Tato tabulka je velice důležitá, protože přístupová práva se přiřazují ke trase a ne přímo k jednotlivým čtečkám. Bez této tabulky by se nedalo zajistit fungování přístupového systému.

READER_IN_ROUTE	vazební tabulka čteček na trasách
reader_in_route_id	id záznamu
route_id	id trasy
reader_id	id čtečky

3.2.7 Tabulka READER_PIN_TYPE

V této databázové tabulce jsou uloženy informace o tom, s jakým typem PINu čtečka pracuje a jak ho zabezpečuje. Například může čtečka používat hashovaný PIN nebo čtyřciferný otevřený PIN. Ale taky nemusí podporovat vkládání žádného typu PINu, například z důvodů chybějící klávesnice.

READER_PIN_TYPE	tabulka typů PINu
reader_pin_type_id	id typu PINu
reader_pin_type_name	název typu PINu
description	popis

3.2.8 Tabulka READER_MODE

Tabulka READER_MODE určuje účel čtečky. Je to informace o tom, jakým způsobem se čtečka používá. Například mohou být čtečky pro vstup a výstup, čtečky docházkového systému, čtečky v jídelně nebo jednosměrné čtečky u závory v garážích.

READER_MODE	tabulka druhů čteček
reader_mode_id	id druhu čtečky
reader_mode_name	název druhu čtečky
description	Popis

3.2.9 Tabulka READER_CHIP

Tato vazební tabulka poskytuje vazbu mezi tabulkami READER a CARD_CHIP. Pomocí této tabulky se určují typy identifikačních karet, respektive jejich čipů, které dokáže čtečka přečíst. Na VUT v Brně se zatím používají dva typy čipů karet a těmi jsou čipy *mifare* a *h4000*.

reader_chip	vazební tabulka na typ čipu, který zvládne čtečka přečíst
reader_chip_id	id záznamu
reader_id	id čtečky
card_chip_id	id typu čipu

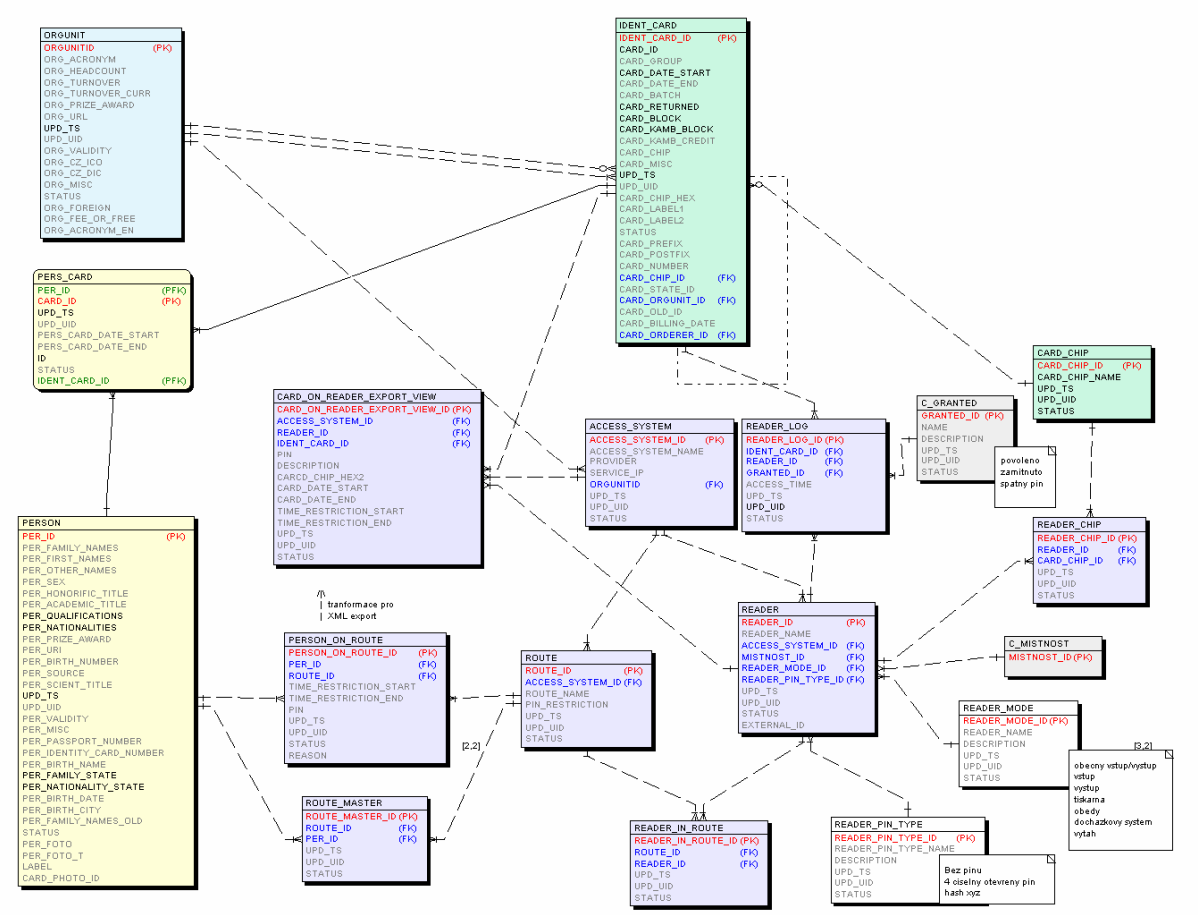
3.2.10 Tabulka READER_LOG

Tabulka READER_LOG ukládá údaje o činnosti čteček. To znamená, že v této tabulce jsou uchovávána data o průchodu čtečkami. Uchovává se identifikační číslo ID karty uživatele, který prošel skrz čtečku a taky čas, kdy tento průchod provedl. Podle těchto údajů lze zpětně zjistit, kdo a kdy „prošel čtečkou“.

READER_LOG	**tabulka záznamů o událostech na čtečce (typicky průchod čtečkou)**
reader_log_id	id záznamu
ident_card_id	id karty, která spustila událost
reader_id	id čtečky
access_time	čas události

3.3 Úprava databázového schéma

Přestože bylo databázové schéma navrženo promyšleně a nepotřebovalo žádné výraznější změny, tak se při analýze objevila jedna chyba, jejíž oprava si vyžádala mírnou úpravu databázového schéma. Tabulka READER_LOG, která uchovává údaje o průchodu čtečkou, sice ukládala údaje o ID kartě osoby a o čase průchodu, jenže neumožňovala uložit informaci o tom, jestli byl průchod čtečkou úspěšný, či nikoliv, případně z jakého důvodu byl neúspěšný. Z toho důvodu byla navržena nová tabulka C_GRANTED, která by měla zajišťovat možnost zjištění informace o úspěšnosti průchodu čtečkou. V případě neúspěšného průchodu by mělo být možné dohledat důvod neúspěchu. Tato tabulka byla navázána na tabulku READER_LOG. Kvůli tomu se změnilo databázové schéma, které je ve své konečné podobě zobrazeno na **obrázku 4**.



Obrázek 4: Konečné databázové schéma přístupového systému

3.3.1 Tabulka C_GRANTED

Tato tabulka obsahuje předem definované položky, jež udávají podrobnosti o průchodu. Může zde být uvedeno, jestli byl průchod úspěšný nebo neúspěšný. V případě neúspěšném průchodu může být v této tabulce uloženo několik možností přesněji definujících důvod neúspěšnosti. Například může být průchod čtečkou neúspěšný z důvodu špatně zadaného čísla PIN.

C_GRANTED	**tabulka s informacemi o úspěšnosti události**
granted_id	id záznamu
name	vystižný název události, popisující úspěšnost či neúspěšnost průchodu
description	popis

3.4 Zbylé tabulky schéma přístupového systému

Doposud nezmíněné tabulky databázového schéma přístupového systému nesouvisí přímo s vlastním návrhem přístupového systému, ale jsou to tabulky, jež poskytují informace důležité pro fungování přístupového systému. Těmito tabulkami jsou:

- Tabulka PERSON
- Tabulka IDENT_CARD
- Tabulka PERS_CARD
- Tabulka CARD_CHIP
- Tabulka ORGUNIT

3.4.1 Tabulka PERSON

Tabulka PERSON je jedna z nejdůležitějších tabulek databázi VUT v Brně. Jsou v ní uloženy informace o všech osobách na VUT v Brně. Například obsahuje jméno, příjmení, titul, pohlaví, datum narození a další údaje.

3.4.2 Tabulka IDENT_CARD

V této tabulce jsou uloženy údaje o všech identifikačních kartách vydaných pro jakékoliv účely na VUT v Brně. Z této tabulky se dá zjistit, jestli je karta platná, jak dlouho ještě bude platná, jestli je blokovaná, jaký používá čip a další informace.

3.4.3 Tabulka PERS_CARD

Toto je vazební tabulka mezi tabulkami PERSON a IDENT_CARD. Jedna osoba tak může mít více identifikačních karet. V této tabulce se taky uchovávají údaje o tom, kdy byla karta uživateli vydána.

3.4.4 Tabulka CARD_CHIP

Tabulka CARD_CHIP obsahuje data o všech druzích čipů, které se používají v identifikačních kartách na VUT. V současné době se používají jen dva typy, a to čip *mifare* a čip *h4000*.

3.4.5 Tabulka ORGUNIT

V tabulce ORGUNIT jsou uloženy všechny organizační jednotky na VUT a informace o nich. Organizační jednotkou jsou fakulty, ústavy a jiné součásti VUT.

3.5 Výstup analýzy

Na základě rozboru existujícího schéma přístupového systému a po schůzce s pracovníky odpovědnými za vznik tohoto schéma, byly zjištěny tyto informace a požadavky na aplikaci:

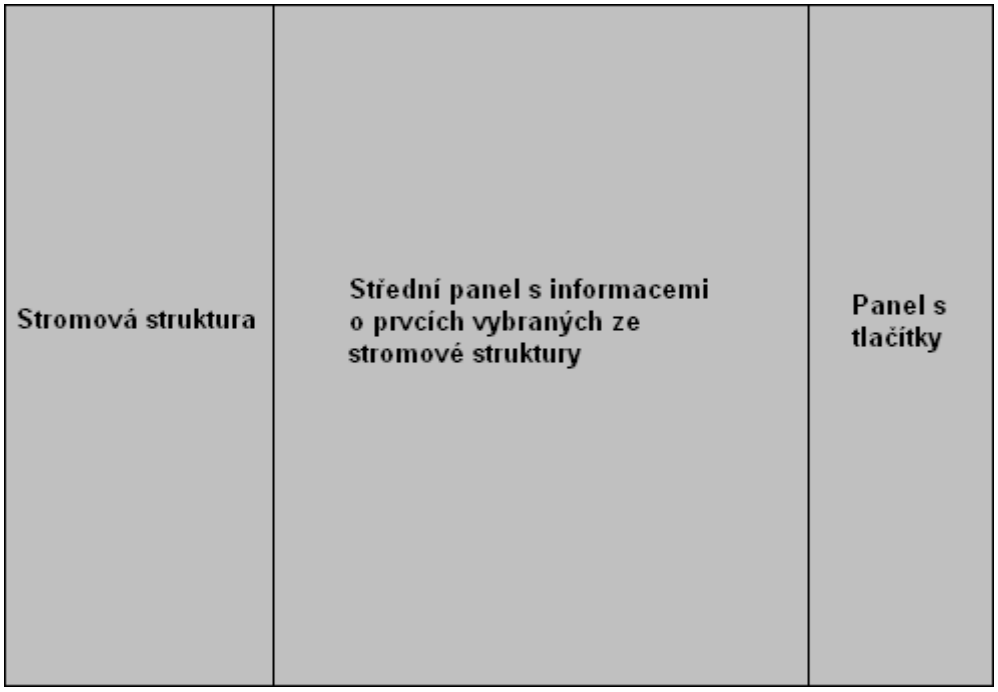
- Možnost spravovat více různých přístupových systémů.
- Spravované přístupové systémy jsou na sobě nezávislé, to znamená, že nesdílejí žádnou trasu ani čtečku.
- Povolení přístupu k nějaké části areálu či budovy se přiřazuje na osobu, nikoliv přímo na identifikační kartu.
- Povolení vstupu se přiřazuje na trasu, nikoliv na konkrétní čtečku.
- Trasa je skupina čteček, přes které musí uživatel projít, aby se dostal do určité části areálu nebo budovy.
- Každý přístupový systém může mít více tras, ale jedna trasa nemůže být ve více přístupových systémech.
- Každá trasa může mít přiděleny osoby, které provádějí správu této trasy.
- Činnost správce trasy spočívá v tom, že přiděluje a odebírá práva k přístupu na spravovanou trasu.
- Čtečky jsou závislé na přístupovém systému, to znamená., že jedna čtečka nemůže být ve více přístupových systémech.
- Čtečka není závislá na trase, to znamená., že jedna čtečka se může nacházet na více trasách.
- Čtečka může zvládnout zpracovat informace od několika různých typů čipu
- Čtečce nemusí být přidělena místnost
- Existují různé druhy čteček

4 Návrh aplikace

Z výstupu analýzy existujícího databázového schéma přístupového systému, bylo zřejmé, že aplikace bude pracovat se třemi základními prvky – s přístupovými systémy, s trasami a se čtečkami. Z analýzy bylo patrné, že tyto prvky tvoří stromovou hierarchii. Na nejvyšší úrovni hierarchie je přístupový systém. Přístupový systém obsahuje alespoň jednu trasu, jelikož přístupový systém bez trasy by nebyl schopný udělovat osobám povolení pro vstup. Každá trasa obsahuje alespoň jednu čtečku, jinak by neexistoval důvod pro udržování takové trasy. Jelikož se stromová hierarchie dobře zobrazuje, bylo rozhodnuto, že základní ovládací komponentou aplikace bude stromová struktura. Podle zvyklostí byla tato struktura umístněna na levou stranu okna aplikace.

Tím bylo zvoleno základní rozmístnění prvků přístupového systému. Zbývalo vyřešit otázku zobrazování, přidávání, editování a mazání těchto prvků. Na základě zkušeností bylo rozhodnuto, že do pravé části bude umístěn panel s funkčními tlačítky, a že se ve střední části okna aplikace budou zobrazovat údaje o prvku zvoleném ve stromové struktuře. Tento návrh struktury vzhledu aplikace ukazuje **obrázek 5**. Zbývalo určit, jaké údaje se zde mají zobrazovat. Pokud by se zde měly zobrazovat podrobné informace o zvoleném prvku, byla by vyřešena otázka editace údajů o prvku a také otázka smazání prvku. Stále by však zůstával otevřený problém vytvoření nového prvku, jelikož by nebylo úplně zřejmé, kam tento prvek přiřadit v hierarchii přístupového systému. Z toho důvodu bylo určeno, že se ve střední části bude zobrazovat seznam přímo podřízených prvků. Tím se vyřešila otázka nejasného přidávání nově vytvořeného prvku do hierarchie přístupového systému, jenž se díky tomuto řešení po svém vytvoření přidá do seznamu podřízených prvků. Otázka smazání prvku se tímto taktéž vyřešila, protože se smaže prvek označený v seznamu podřízených prvků. Zbývalo dořešit otázku editace prvku. Kvůli editaci byl do dolní poloviny střední části přidán panel s detaily prvku označeného v seznamu podřízených prvků, v němž bylo možno prvek editovat. **Obrázek 6** zobrazuje takto upravenou strukturu vzhledu aplikace.

Toto řešení bylo elegantní, ale na druhou stranu se sebou přineslo problém, jakým způsobem se bude provádět vytváření, editace a mazaní přístupových systémů, které byly doposud na nejvyšší úrovni hierarchie. Z tohoto důvodu se do hierarchie zavedl nový kořenový prvek, *Přístupové systémy*, který byl předkem všech přístupových systémů a zastřešoval je jednotným pojmenováním.



Obrázek 5: Počáteční návrh struktury aplikace

Obrázek 6: Konečný návrh struktury aplikace

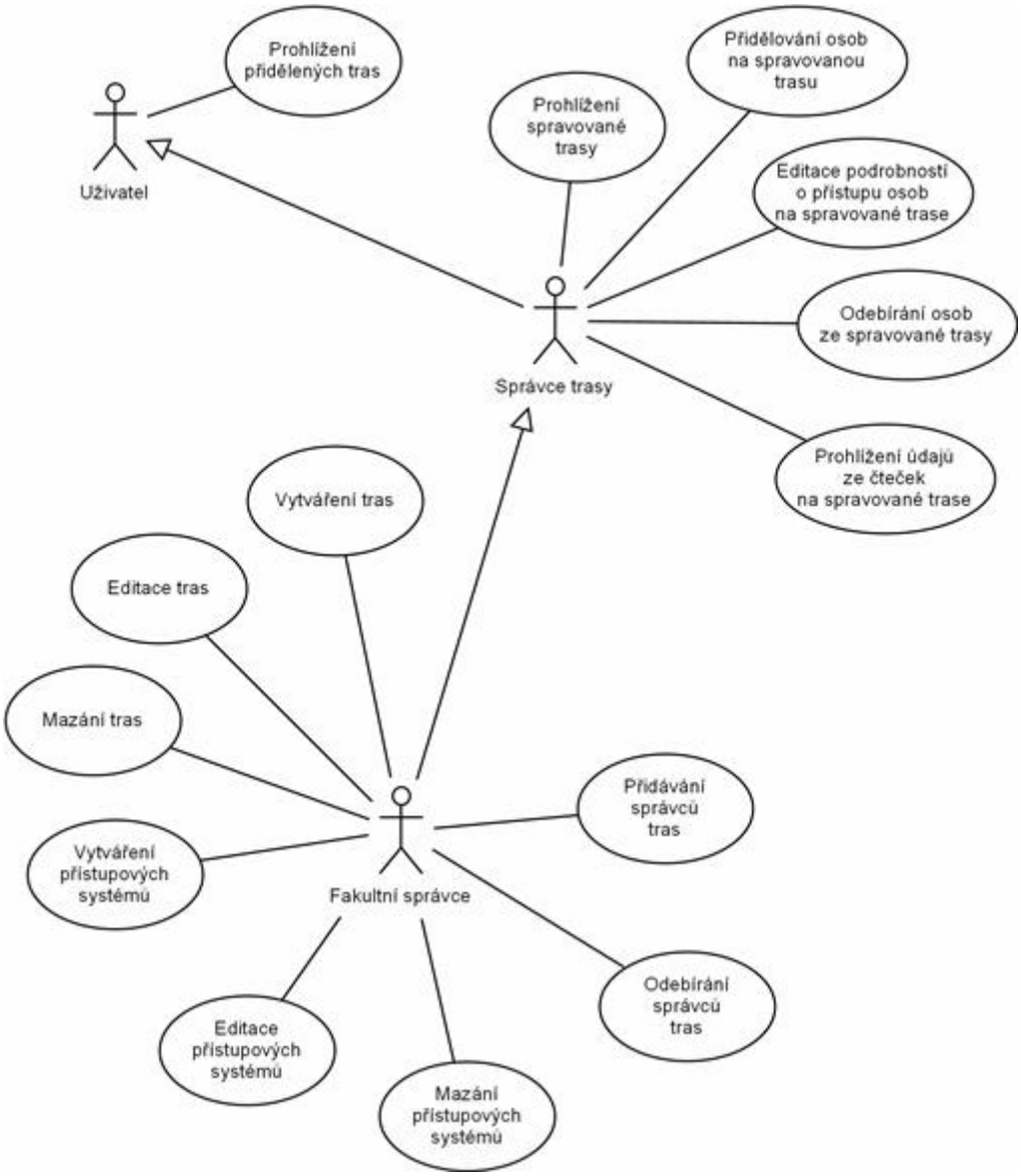
Struktura aplikace byla navržena. Pro dokončení komplexního návrhu aplikace zbývalo určit role, pod kterými budou uživatelé s aplikací pracovat, a rozhodnout jaká práva budou mít v rámci těchto rolí.

Bylo jasné, že zde musí existovat role hlavního správce, který bude mít největší pravomoc, a kterému bude umožněno pracovat se všemi funkcemi, které aplikace nabídne. Dále bylo potřeba navrhnout uživatele s nejnižšími právy, roli obyčejného uživatele. Z analýzy vyplynulo, že tento

uživatel bude mít právo podívat se pouze na trasy, na které má přístup. V žádném případě nebude mít oprávnění cokoliv vytvářet, měnit nebo mazat.

Díky informacím získaným z analýzy databázového schéma bylo zřejmé, že zde musí být role správce trasy. Správce trasy bude mít stejná práva jako obyčejný uživatel, k nimž mu přibude právo na přidávání a na odebírání přístupů osobám na trasu, již bude spravovat. Nebude mít práva nějak zasahovat do již vytvořeného přístupového systému, ani do existujících tras v přístupovém systému. Správce trasy bude moct zároveň spravovat více tras.

Jednotlivé role a možnosti jejich práce se systémem shrnuje diagram případů použití, viz **obrázek 7**. Lze z něj vyčíst, jaká oprávnění bude mít konkrétní role.



Obrázek 7: Diagram případů použití (use-case diagram)

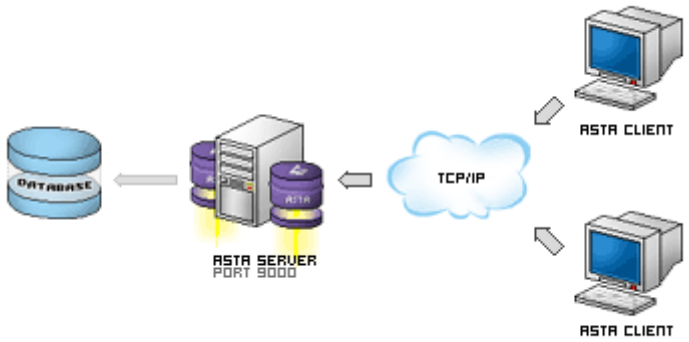
5 Implementace

Po fázi návrhu struktury aplikace následovala implementace navržené aplikace do celoškolského informačního systému Apollo. Nejprve bylo nutné seznámit se s tímto informačním systémem a se základy jeho fungování.

5.1 Informační systém Apollo

Informační systém Apollo je postaven na třívrstvé architektuře od firmy Asta Technology Group. **Obrázek 8** ukazuje schéma typické třívrstvé architektury firmy Asta Technology Group. V této architektuře se k datům v databázi nepřistupuje přímo od klienta, ale přes aplikační server. Tím, že přesuneme aplikační a datovou logiku na aplikační server snížíme požadavky na klientskou aplikaci, které se může zabývat jen prezentací výsledků. Navíc vložením střední vrstvy mezi klienta a databázi dosáhneme vetší bezpečnosti systému. Zabezpečit jeden nebo více aplikačních serverů je nepoměrně jednodušší, než zabezpečit několik desítek či stovek klientských aplikací.

Z **obrázku 8** by se mohlo zdát, že je možné použít pouze jeden aplikační server. Použití jen jednoho aplikačního serveru není podmínkou. Aplikačních serverů může být i několik v závislosti na tom, kolik uživatelů využívá systém založený na této třívrstvé architektuře. Potřeby uživatelů informačního systému Apollo by jeden server nedokázal pokrýt, proto informační systém Apollo používá šest serverů.



Obrázek 8: Třívrstvá architektura

5.2 Databáze IS Apollo

IS Apollo získává informace z centrálního datového skladu VUT, který je postaven na databázovém systému VUT a provozován na víceprocesorovém stroji. Existuje několik instancí databáze. CDBX, CISB, CISC a CISD. CDBX oficiální datový sklad VUT v Brně, ke kterému se připojuje nejenom

celoškolský informační systém Apollo, ale i Portál VUT v Brně a další. CISB má identické nastavení jako CDBX a slouží pro výzkum a vývoj. Je využívám úzkou skupinou vývojových pracovníků a studentů. CISC a CISD jsou kopie CDBX, jejich data jsou nepravidelně obnovována a slouží hlavně jako testovací databáze pro uživatele.

5.3 Aplikační server Akira

Akira je tedy server střední vrstvy postavený na technologii Asta, který přistupuje k centrálnímu datovému skladu se zohledněním pokročilých přístupových práv. Tento server se po přihlášení uživatele stará o zpracování SQL dotazu a předává odpovídající výsledky zpět klientovi. Komunikace mezi Akirou a Apollem probíhá přes zabezpečené připojení pomocí SSL (*Secure Socket Layer*) protokolu. Apollo neposílá Akiře přímo SQL dotaz, který chce vykonat, ale říká ji, který z uložených dotazů má vykonat (má-li na to dostatečné oprávnění). Díky tomu nelze data z databáze získat neoprávněným způsobem nebo je dokonce modifikovat. Data posílaná mezi Apollem a Akirou jsou komprimovaná, Akira navíc dokáže posílaná data tzv. *fetchovat* – posílat po částech na požádání. Veškeré operace, které uživatel provede, se zaznamenávají do textového souboru přímo na serveru (tedy nezávisle na databázi). Zaznamenávají se všechny databázové dotazy (včetně výběru dat – SELECT) se všemi jejich parametry, takže je možné zpětně dohledat, jakou činnost uživatel v daný okamžik prováděl.

5.4 Klient Apollo

Apollo je takzvaný tenký klient, protože už nemusí řešit zabezpečení a další související věci, ale soustředí se pouze na prezentaci údajů.

Celý informační systém je modulární, takže uživatelé si stahují do klienta jen části (moduly), které používají. Klient zajišťuje automatické aktualizace modulu, připojení k aplikačnímu serveru Akira, zamykání po delší době nečinnosti, ukládání nastavení, tisk a spouštění jednotlivých modulů. Moduly jsou realizovány DLL knihovnami s definovaným rozhraním. Z tohoto důvodu je možné aktualizovat jednotlivé části systému samostatně, aktualizace je tedy jednodušší a rychlejší. Díky modularitě je systém méně náročný na provoz, protože se do paměti načítají pouze aktuálně používané moduly. Taky je díky modularitě systému usnadněn vývoj nových částí, protože každý vývojář může pracovat na jednom modulu nezávisle na ostatních.

5.4.1 Spuštění Apolla

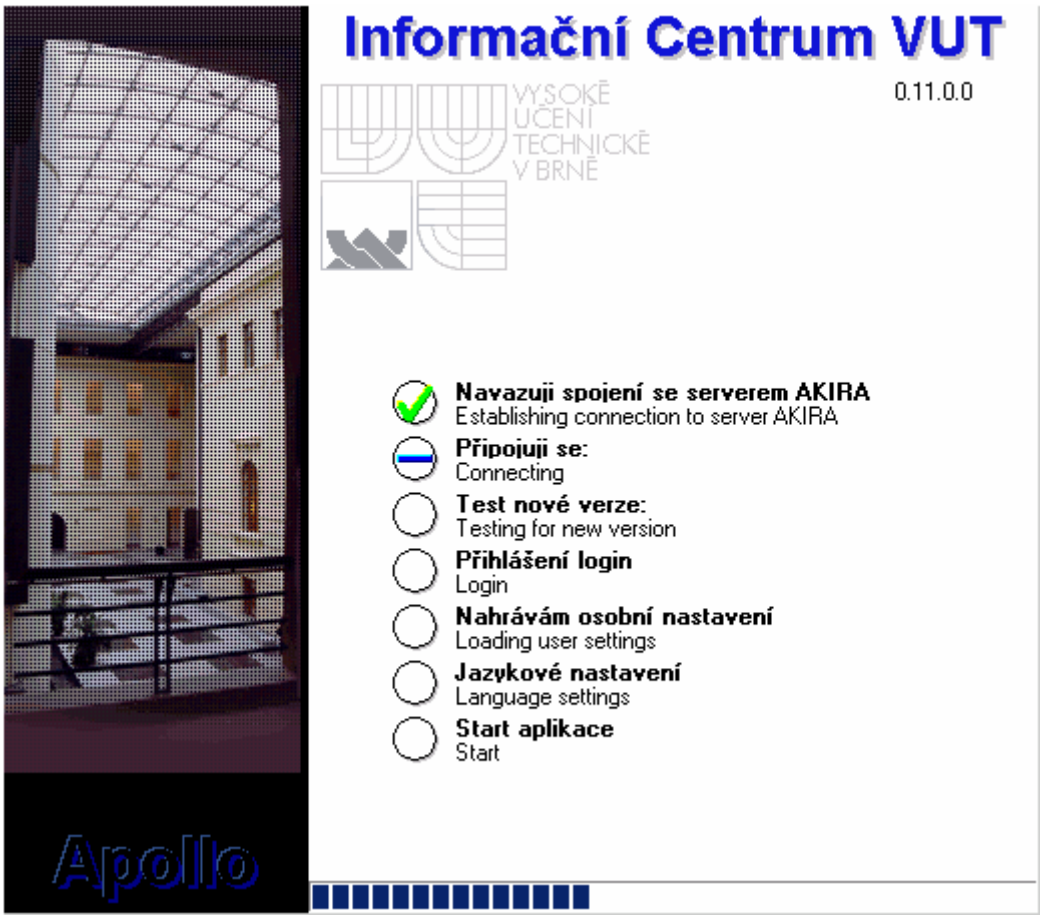
Po spuštění se Apollo spojí se všemi aplikačními servery a oznámí jim, že má v úmyslu se na některý z nich připojit. Každý server odpoví zprávou, ve které Apollu vrátí počet právě připojených uživatelů.

Na základě těchto čísel se Apollo připojí na nejméně vytížený server (server s nejmenším počtem uživatelů). Pokud žádný aplikační server neodpoví, Apollo tuto chybu nahlásí a ukončí se. **Obrázek 9** ukazuje úvodní obrazovku Apolla ve fázi připojování se k aplikačnímu serveru.

Po připojení na nejméně vytížený aplikační server se Apollo dotáže na aktuální verzi, která je k dispozici. Je-li k dispozici nová aktualizace, automaticky se stáhne z aktualizačního serveru, a sama aktualizuje soubory Apolla a spustí novou verzi.

Po ověření nové verze je uživatel vyzván k zadání uživatelského jména (*VUTloginu*) a hesla (*VUThesla*). Oba tyto údaje se zašlou v zašifrovaném tvaru aplikačnímu serveru, který údaje ověří a v případě jejich správnosti umožní Apollu připojení.

Apollo následovně načte z databáze uživatelovo osobní nastavení a nastaví zvolenou jazykovou verzi. Po spuštění se automaticky spustí modul *Desktop*, ve kterém si muže uživatel zjistit svá oprávnění, své přístupy do systému a novinky v sekci *Co je nového*.



Obrázek 9: Spouštění IS Apolla

5.4.2 Komunikace mezi Apollem a spuštěným modulem

Při požadavku na spuštění modulu (vybráním z menu) se Apollo nejprve dotáže aplikačního serveru, zda není k dispozici novější verze modulu, než je ta, kterou má právě k dispozici. Poté se modul

zavede do paměti, zobrazí se do speciálního panelu určeného k otevření modulu, zavolá se událost *AfterCreate* a dále už řídí běh sám modul.

Poté probíhá komunikace mezi Apollem a spuštěným modulem pouze pomocí událostí a zpráv. Odeslání zprávy z modulu se provádí *funkcí FceCMD(Fce, Params: string)*, kde první parametr je typ zprávy, druhý jsou její parametry.

Apollo zasílá zprávy spuštěnému modulu pomocí události *OnMessage*, který má parametr *Id*, což je ordinální hodnota typu zprávy a parametr *Text*, což jsou parametry zprávy. Krom této události na posílání zpráv má objekt třídy *TApFrame* pro nejčastější typy zpráv vyhrazené zvláštní události:

- OnDBRefresh – obnovení dat (F5)
- OnDBFirst – pokyn na přesunutí na první záznam v tabulce (F7)
- OnDBNext – pokyn na přesunutí na další záznam v tabulce (F9)
- OnDBPrev – pokyn na přesunutí na předchozí záznam v tabulce (F8)
- OnDBLast – pokyn na přesunutí na poslední záznam v tabulce (F10)
- OnDBExport – požadavek uživatele na export dat
- OnDBImport – požadavek uživatele na import dat
- OnDBFind – požadavek uživatele na hledání dat (CTRL + F)
- OnDBNew – požadavek na vytvoření nového záznamu
- OnDBDelete – požadavek na smazání vybraných záznamů
- OnDBPrintBefore – inicializace před tiskem
- OnDBPrint – tisk dokumentu
- OnDBPrintAfter – ukončení tisku
- OnDBSave – požadavek uživatele na uložení dat (CTRL + S)
- OnMessage – událost pro zasílaní zpráv
- OnMessageData – událost pro posílaní zpráv s ukazatelem na poskytovaná data

Při ukončování modulu je volána událost *OnCanClose*, ve které se obvykle zkontroluje, zda jsou změněná data uložena a v případě, že ne, je možné uzavírání modulu pozastavit. Nakonec se zavolá událost *OnDestroy*, ve které se uvolní všechny datové struktury z paměti.

5.4.3 Přístup k databázi

Jednou z hlavních funkcí Apolla, jakožto klienta, je zprostředkovávat data z databáze modulům. Data z databáze získává aplikační server *Akira* vykonáváním uložených SQL dotazů.

Na aplikační server *Akira* se neposílá přímo SQL dotaz, který se má vykonat, ale posílá se název, pod kterým je SQL dotaz uložen. SQL dotazy jsou totiž uloženy v databázi v odděleném schématu *ApolloAdmin*. Každý dotaz je propojen s modulem, který ho může vykonat a jsou k němu přirazena přístupová práva, která musí uživatel mít, aby mohl dotaz spustit.

Pro snadné volání SQL dotazu je v jednotce *uAkira.pas* několik funkcí. Nejvíce využívanými funkcemi při implementaci modulu Přístupový systém byly funkce *FastGridOpen, FastQueryExec* a *FastQueryOpen*.

Funkce *FastGridOpen* je funkcí pro rychlé a pohodlné plnění komponenty *TSMDBGrid,* jež je podrobněji popsána podkapitole Komponenty. Funkce *FastGridOpen* má tři základní parametry. Prvním parametrem je jméno komponenty *TSMDBGrid*, která se má naplnit údaji. Druhým parametrem je název dotazu ve schématu *ApolloAdmin* a třetím jsou parametry SQL dotazu. Parametry SQL dotazu se předávají prostřednictvím pole záznamů, vždycky dvojice – název parametru a jeho hodnota. Funkce *FastGridOpen* automaticky vyvolá SQL dotaz typu *select* a výsledky tohoto dotazu naplní komponentu *TSMDBGrid*.

Funkce *FastQueryExec* má taktéž tři základní parametry. Prvním je název SQL dotazu ve schématu *ApolloAdmin*. Druhým parametrem je typ SQL dotazu. Tento parametr může nabývat těchto hodnot:

- *qaCreate* – vytváření hodnot v databázi
- *qaModify* – úprava hodnot v databázi
- *qaDelete* – odstraňování hodnot z databáze
- *qaView* – zobrazení dat z databáze
- *qaViewDetail* – další možnost pro zobrazení dat z databáze

Třetím parametrem funkce *FastQueryExec* je stejně jako u předešlé funkce seznam parametrů SQL dotazu. Tato funkce provede SQL dotaz určený prvním parametrem a vrací počet řádků, které jsou ovlivněny tímto dotazem. Výsledky dotazu u této funkce nejsou důležité

Funkce *FastQueryOpen* má stejné jako předchozí funkce tři základní parametry. Prvním je datový modul typu *TdmApFrame* pro uchování výsledků SQL dotazu. Datový modul je podrobněji popsán v následujícím odstavci. Druhým parametrem je název SQL dotazu a třetím jsou parametry SQL dotazu.

Pro uchování vrácených záznamu z databáze se používá objekt typu *TdmApFrame*. Jedná se o datový modul (*TDataModule*), který obsahuje komponentu firmy *AstaTech*, a to sice *smResultSet*, který je poděděný z třídy *TDataset*. Tomu odpovídá i práce s ním. Je podobná práci s klasickým seznamem – na první záznam se nastavíme zavoláním metody *First*, na poslední položku záznamu přejdeme pomocí metody *Last*, na další záznam se dostaneme zavoláním metody *Next* a konec seznamu indikuje vlastnost *EoF*. Přístup k datům zajišťuje metoda *FieldByName*, která vrací *TField*, přes který se dá přečíst hodnota buď jako typ variant nebo po konverzi jako jakýkoliv jiný datový typ.

5.4.4 Komponenty

Apollo je databázová aplikace a je z větší části tvořena formuláři pro vstup dat a jejich zobrazování. Standardní komponenty, které jsou dodávané s *Delphi*, však plně nedostačují daným požadavkům a

nebo vůbec neexistují. Proto byl vyvinut balíček komponent *Apollo*, který obsahuje přepracované nebo úplně nové komponenty formulářového charakteru používané v Apollu a jeho modulech. Hlavní charakteristikou těchto komponent je integrovaný *label* (label neboli popisek je součástí dané komponenty a kvůli tomu je možné na formulář umístit pouze poloviční množství komponent), podpora barevného označování stavu příslušného pole (takzvaný *recoloring*, což umožňuje jednoduše pohledem zjistit, která pole byla modifikována), podpora označování povinných položek a chybně zadaných údajů apod.

Zřejmě nejdůležitější je již zmiňovaná komponenta *TApFrame*, což je v podstatě okno, které obsahuje jiné komponenty a umísťuje se (tzv. „dokuje“) do jiného okna. Nikdy neexistuje samostatně a slouží jako základní komponenta každého modulu. Za zmínku stojí i komponenta *TSMDBGrid*, což je komponenta tabulkového typu, která se používá pro zobrazení a případnou editaci dat tabulkového charakteru. Mezi jeho vlastnosti patří například propracované filtrování položek, jejich řazení, dále víceúrovňová záhlaví, ukotvené sloupce, přeuspořádání nebo skrývání sloupců a další.

5.5 Programování modulu Apolla

Pro vytváření modulu bylo využito vývojového nástroje Delphi Professional od firmy Borland ve verzi 7.0. Z toho důvodu, že se pro vývoj informačního systému Apollo nepoužívají standardní komponenty, bylo nutné před zahájením implementace rozšířit vývojové prostředí Deplhi o balíčky s potřebnými komponentami.

5.5.1 Vytvoření modulu

Po nainstalování všech nutných balíčků komponent, se mohlo přistoupit k vytváření modulu. Nejdříve se vytvořil *ApFrameMain*, který měl obsahovat jednotlivé komponenty. Přestože modul zatím neobsahoval žádné komponenty, bylo potřeba vytvořit z něj DLL knihovnu a nahrát ji do databáze k ostatním knihovnám s moduly pro informační systém Apollo. Dále bylo potřeba přidat do hlavního menu informačního systému položku, kterou se bude modul Přístupový systém spouštět. Tato položka byla označena *Přístupový systém* a byla přidána do nabídky *VUT*, jelikož se jedná o modul celoškolského významu. Nakonec zbývalo vytvořit práva a role, aby se omezil počet uživatelů, kteří mají přístup k tomuto modulu.

5.5.2 Práva a role modulu

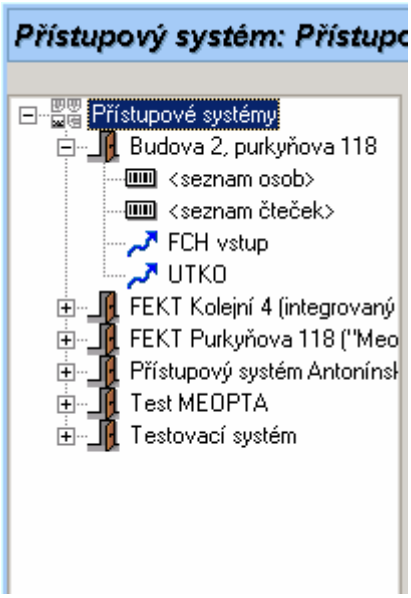
Každý modul v informačním systému Apollo má přiřazeny práva, která umožňují určovat stupeň povolené funkcionality modulu. Pro modul Přístupový systém byla vytvořena tato tři práva:

- apolloPristupovySystemR – toto je základní právo modulu, právo pro čtení. Tohle právo musí mít uživatel, aby vůbec mohl modul Přístupový systém spustit. Díky tomuto právu má uživatel možnost prohlížet všechny části přístupového systému, ale nemá oprávnění přidávat, měnit nebo odebírat žádnou část přístupového systému.
- apolloPristupovySystemW – tohle právo je právo pro zápis. Toto právo rozšiřuje předchozí právo a mělo udělovat správcům trasy, jelikož umožňuje přidávat osoby do seznamu osob, které mají přístup na určitou trasu
- apolloPristupovySystemA – toto právo je určené pro administrátory, uživatel může díky tomuto oprávnění plnohodnotně pracovat s modulem bez jakýchkoliv omezení.

Jelikož se jednotlivé přístupové systémy váží na organizační jednotku (typicky fakultu) a není žádoucí, aby administrátor přístupového systému z jedné fakulty měl taky administrátorská práva na jiné fakultě, zavedla se do systému ještě takzvaná role. Byla pojmenována AP_PRISTUP_SYS. Přes tuto roli je vyřešeno omezení administrátorského práva jen na určitou fakultu nebo součást. Samozřejmě je možné v případě potřeby přiřadit uživateli tuto roli na více fakultách, tudíž může být správcem více přístupových systémů.

5.5.3 Modul Přístupový systém

Jak již bylo uvedeno, základem modulu Přístupový systém je *ApFrameMain*. Tento úvodní rám se zobrazí při spuštění modulu a obsahuje dvě důležité komponenty. Asi nejdůležitější komponentou je *TApTreeView*. Tato komponenta dokáže zobrazit stromovou strukturu, což se v modulu využívá pro zobrazení struktury přístupového systému, viz **obrázek 10**. Tato komponenta se používá ve spolupráci s komponentou *TCoolTabControl* a slouží jako hlavní ovládací prvek celého modulu. V komponentě *TApTreeView* se kliknutím zvolí požadovaný prvek přístupového systému a pomocí *TCoolTabControl* se vybere příslušná záložku, na níž se pak zobrazí podrobnosti o vybraném prvku. Tyto záložky uživatel nevidí, jsou mu skryté. Přímo na těchto záložkách nejsou žádné komponenty, které by zobrazovaly nějaké údaje z databáze, či by nějak ovládaly chování modulu. Každá záložka je pomocí komponenty *TPages* a jí podřízených komponent *TPageSheet* navázána na jednotlivé rámy (komponenty *ApFrame*), které obsahují veškerou ovládací a zobrazovací logiku. Každý tento rám se kvůli přehlednosti kódu nachází ve zvláštním souboru.



Obrázek 10: Stromová struktura

Vytváření stromové struktury

Vytváření stromové struktury je jedna z nejdůležitějších činností modulu Přístupový systém. Plnění komponenty *TApTreeView* vypadá tak, že se nejprve vytvoří kořenový uzel a poté se na tento uzel navazují uzly potomků. Tyto uzly potomků mohou také mít potomky. Komponenta následně prochází tuto struktuře a vykresluje ji na obrazovku.

Základním problémem bylo vyřešení získávání dat z databáze. Bylo potřeba vytvořit SQL dotaz, který by vrátil data ve vhodném formátu, aby se dal procházet výsledek dotazu po jednotlivých řádcích a podle údajů na řádku určit, kam záznam navázat. Bylo třeba jedním dotazem získat seznam všech přístupových systémů a zároveň seznam všech tras na všech přístupových systémech a zároveň seznam čteček nalézajících se na trasách. Nejprve se řešení zdálo být jednoduché. Vypadalo to, že postačí relativně jednoduchý SQL dotaz využívající vazeb mezi tabulkami. Jenže problém byl v tom, že tento jednoduchý dotaz nezobrazil trasy, které neobsahovali žádnou čtečku. Z toho důvodu bylo nutné použít mnohem složitější dotaz, který využíval několikanásobného sjednocení tabulek, viz podkapitola **9.2** a v ní odstavec **SQL dotaz pro vytvoření stromové struktury**. Výstup SQL dotazu ukazuje **obrázek 11**. Tento výstup byl vhodný pro vytváření stromové struktury. Stačilo procházet strom do hloubky a při zanořování vytvářet nové uzly a napojovat je na již vytvořené rodičovské uzly.

		SYSTEMID	SYSTEMNAME	ROUTEID	ROUTENAME	READERID	READERNAME
▶	1	62	FEKT Kolejní 4	262	role: externí vyučující		
	2	62	FEKT Kolejní 4	82	role: student	307	laboratoř 1
	3	62	FEKT Kolejní 4	82	role: student	305	učebna 1
	4	62	FEKT Kolejní 4	82	role: student	306	učebna 2
	5	62	FEKT Kolejní 4	263	zóna: knihovna FEKT		

Obrázek 11: Výsledek SQL dotazu

Většina údajů ve stromové struktuře je načítána z databáze. Přesto se ve stromu vyskytují položky (uzly), které nemají původ v databázi a jsou do stromu přidány až při jeho vytváření. Jedná

se o kořenový uzel a pak o uzly pojmenované <*seznam osob*> a <*seznam čteček*>. Jejich význam bude vysvětlen později.

5.5.3.1 Záložka přístupové systémy

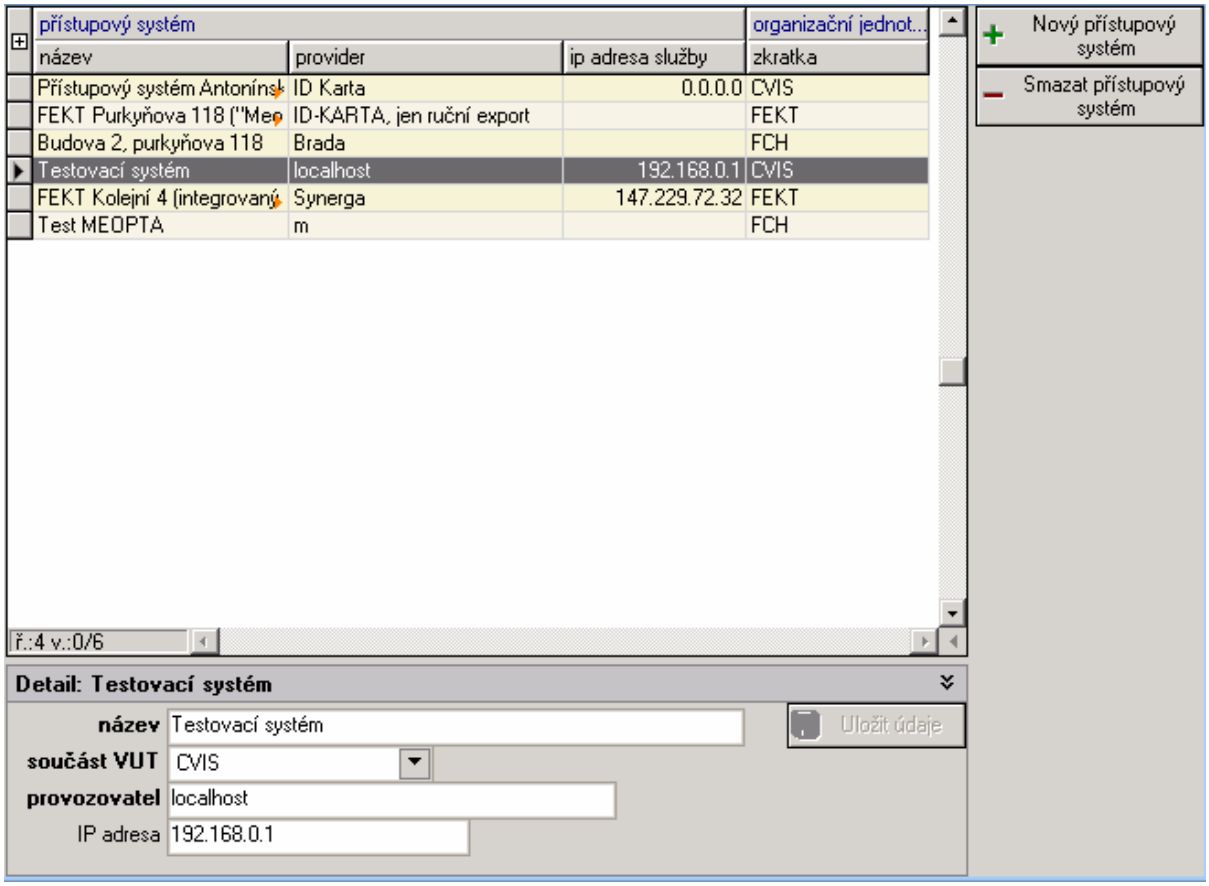
Tato záložka se zobrazí, pokud je ve stromu zvolena položka *Přístupové systémy*. Vedle stromové struktury se zobrazí *ApFrame*, který se stará o zobrazení informací o přístupových systémech. Skládá se ze dvou částí.

Na levé straně je tabulka zobrazující seznam přístupových systémů a informace o nich. S touhle tabulkou souvisí panel s detaily, který je umístěn pod ní.

Tabulka je vytvořena komponentou nazvanou *TSMDBGrid*. Jak už bylo zmíněno dříve, tato komponenta dokáže automaticky řadit záznamy, filtrovat je podle různých kritérií, prohazovat sloupce záznamů, měnit jejich velikosti a případně skrývat nepotřebné sloupce.

Panel s detaily je tvořen komponentou *TApCoolPanel*. Tato komponenta poskytuje záhlaví, do něhož lze zapsat text, a v záhlaví je i dostupné tlačítko, jež slouží ke skrývání tohoto panelu. Na **obrázku 12** lze vidět, že se do panelu s detaily vyplňují údaje o položce vybrané v tabulce. Panel s detaily slouží zejména pro úpravu údajů zobrazených v tabulce. Tento účel plní komponenty *TApEdit* (**název**, **provozovatel** a **IP adresa**) a komponenta *TApComboBox* (**součást VUT**). Komponenta *TApEdit* je textová komponenta, která umožňuje vkládat či upravovat text. Komponenta *TApComboBox* je výběrová komponenta. Uživateli nabídne několik možností a ten si vybere jednu z nich. Zde se *TApComboBox* plní údaji o součástech VUT získaných z databáze. Panel s detaily obsahuje mimo již zmiňovaných komponent ještě funkční tlačítko pojmenované **Uložit údaje**. Toto tlačítko je založeno na komponentě třídy *TapImBtn* a zpřístupňuje se tehdy, když je provedena jakákoliv změna v komponentách v panelu s detaily a jsou zároveň vyplněny všechny povinné údaje a zároveň jsou tyto vyplněné údaje platné (to je hlavně případ IP adresy). Po stisku tohoto tlačítka se provede SQL dotaz, který uloží změněné údaje do databáze. Následně se provede aktualizování údajů v tabulce zobrazující seznam přístupových systémů.

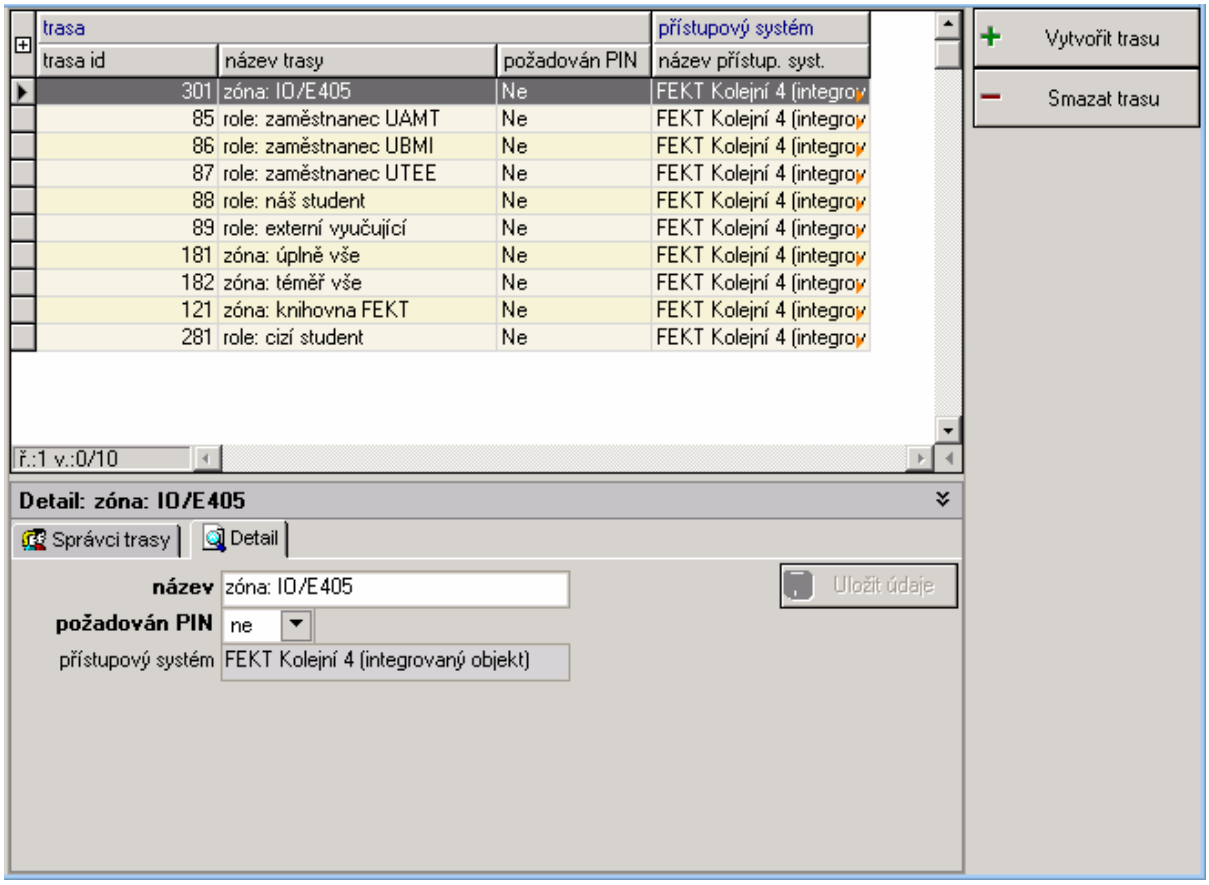
Na pravé straně se nachází panel s funkčními tlačítky, což jsou komponenty třídy *TApImBtn*. Zvláštností oproti obyčejným tlačítkům dostupným v Delphi je možnost přiřadit jim ikonu. Tlačítko **Nový přístupový systém** umožňuje vytvořit další nový přístupový systém. Tlačítko **Smazat přístupový systém** naopak odstraní vybraný přístupový systém. Po jeho stisku se objeví potvrzovací dialog založený na komponentě *TApMsgDlg*. K odstranění přístupového systému dochází až poté, co je operace smazání přístupového systému potvrzena. Následně dojde k aktualizaci dat v tabulce zobrazující seznam přístupových systémů. Pokud uživatel nevlastní administrátorské oprávnění, nejsou mu tyto tlačítka zpřístupněna.



Obrázek 12: Položka Přístupové systémy

5.5.3.2 Záložka Přístupový systém

Zobrazení této záložky nastane, pokud se ve stromové struktuře vybere některý přístupový systém. O zobrazení se stará komponenta *ApFrame*, která je stejně jako v předchozím případě rozdělena na dvě části. Část s funkčními tlačítky, která slouží k vytvoření, respektive k odstranění zvolené trasy, a část tvořená tabulkou tras v systému a s ní souvisejícím panelem s detaily. Tabulka je opět tvořena komponentou *TSMDBGrid* a stejně jako v předešlém případě je provázána s panelem s detaily. Změna aktuálně vybrané položky v tabulce zapříčiní změnu údajů v panelu s detaily. **Obrázek 13** ukazuje finální vzhled záložky Přístupový systém.

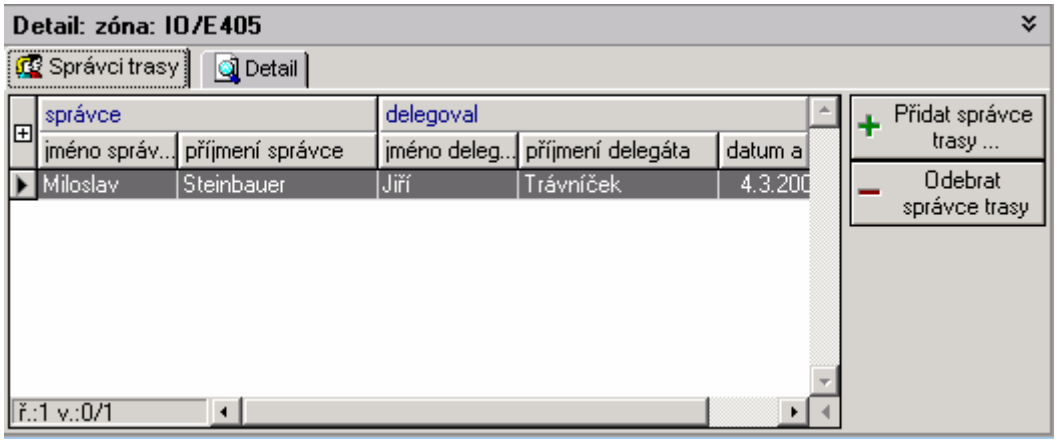


Obrázek 13: Položka přístupový systém

Zajímavější je pohled na panel s detaily. Ten tentokrát obsahuje navíc panel se záložkami. Ten je tvořený komponentou *TCoolTabControl* a jelikož komponent na záložkách není mnoho, obsahuje každá záložka zobrazené komponenty přímo v sobě. Panel záložek má jen dvě záložky.

První záložka se jmenuje **Správci trasy**, viz **obrázek 14,** a obsahuje seznam osob, kteří mají právo dělat správce na zvolené trase. Seznam je opět v tabulkovém tvaru a vytvořen pomocí komponenty *TSMDBGrid*. Dále jsou zde přítomny dvě funkční tlačítka. Tlačítko **Odebrat správce trasy** vybrané osobě ukončí právo na spravování trasy. Tlačítko **Přidat správce trasy…** *vyvolá* dialogové okno se seznamem osob, viz odstavec **Dialog Správci**.

Druhá záložka se jmenuje **Detail** a obsahuje podrobnosti o zvolené položce tabulky. Opět se zde nachází tlačítko pro uložení změněných údajů. Za zmínku stojí textové pole **přístupový systém**, které je zneplatněno, tudíž nejde editovat. Je tomu tak proto, že trasu nelze převést pod jiný přístupový systém.



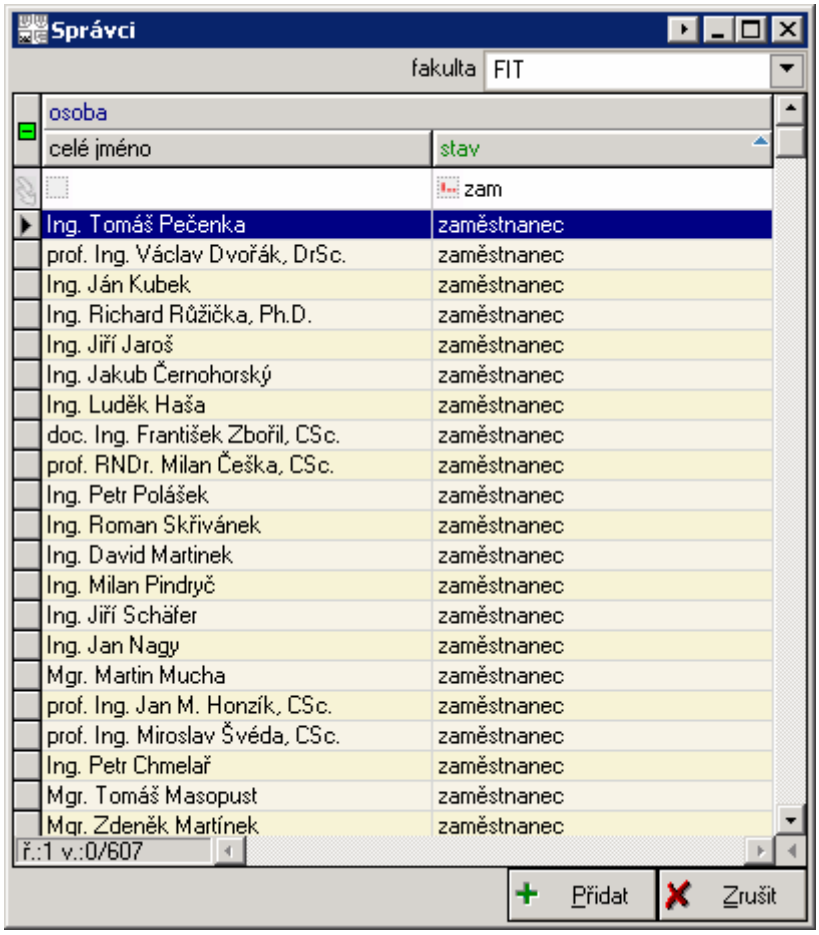
Obrázek 14: Správci trasy

Dialog Správci

O zobrazení tohoto dialogu se stará třída *TForm*. Tento dialog je určený pro výběr osob, které dostanou právo na správu trasy. Základem tohoto dialogu je tabulkový seznam osob vytvořený komponentou *TSMDBGrid*. Z tohoto seznamu je možné vybrat více osob zároveň.

V pravé horní části dialogového okna je umístěna komponenta třídy *TApComboBox*. Tato komponenta slouží k omezení zobrazených osob jen na určitou součást VUT. Hodnoty pro tento *combobox* se získávají z databáze jednoduchým SQL dotazem, ke kterému je přidána hodnota *Nezařazení*, viz podkapitola **9.2** a odstavec **SQL dotaz pro naplnění filtru fakulta**. Když se jakkoliv změní hodnota v comboboxu, tak se do tabulky automaticky načtou nové údaje odpovídající vybrané součásti.

V pravé spodní části dialogového okna jsou tlačítka **Přidat** a **Zrušit**, vytvořené komponentou *TApImBtn*. Tlačítkem **Přidat** se všechny zvolené osoby přidají do seznamu správců trasy, pokud v tomto seznamu dosud neexistují. Dialog **Správci** je ve své konečné verzi zobrazen na **obrázku 15**.



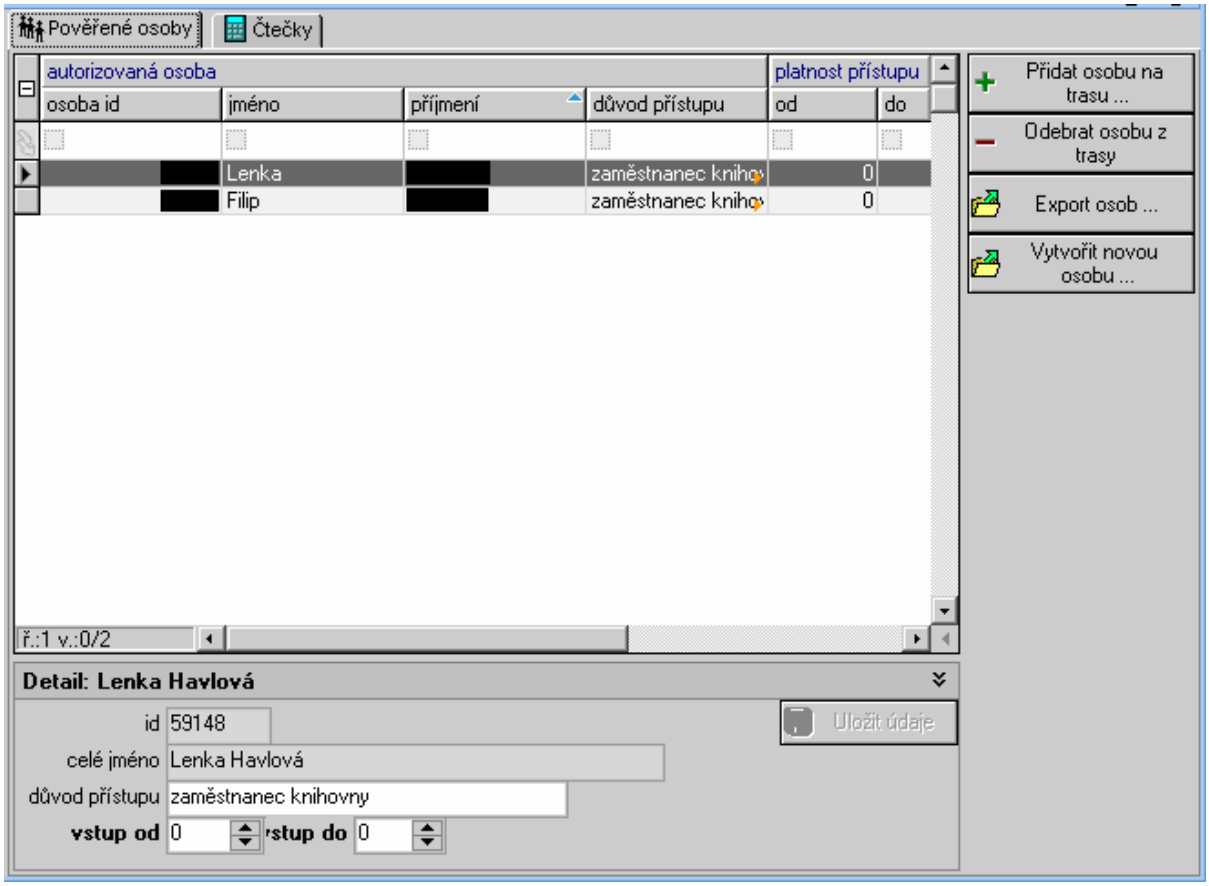
Obrázek 15: Dialog Správci

5.5.3.3 Záložka Trasa

Základem tohoto *ApFrame* je lištu záložek se dvěma záložkami.

První záložka je pojmenovaná **Pověřené osoby**. Na této záložce se nachází seznam osob, které mají právo na průchod zvolenou trasou. Tento seznam je opět tvořený komponentou *TSMDBGrid* a navazuje na něj panel s detaily o vybrané osobě. V detailech osoby jsou dvě položky, které nelze upravovat. Jsou jimi identifikační číslo a jméno osoby, a je nemožné je upravovat kvůli tomu, že jsou tyto údaje globálního charakteru. Navíc od přístupového systému není požadováno, aby umožňoval měnit tyto údaje. Položky **vstup od** a **vstup do** jsou vytvořené pomocí komponenty *TApSpinEdit*, která umožňuje omezit vstupní hodnoty jen na určitý interval. Zde se jedná o časové údaje, přesněji o hodiny, tudíž se využívá interval od 0 do 23. Tyto položky dosud nemají žádný význam a jsou zde uvedeny hlavně z důvodu budoucího rozšiřování přístupového systému, kdy půjde omezit přístup na trasu pro osobu jen na určité časové období. Dále se pravé straně této záložky nachází panel s funkčními tlačítky. Jak je ukazuje **obrázek 16**, nacházejí se zde čtyři tlačítka. Tlačítko **Přidat osobu na trasu…** vyvolá po zmáčknutí dialogové okno, ve němž lze vybrat osoby, které mají mít povolený přístup na trasu, více viz odstavec **Dialog Seznam osob**. Kliknutím na tlačítko **Export osob…** se spouští se dialog, díky jemuž je možno exportovat seznam pověřených osob do externího

souboru, viz odstavec **Dialog Export osob**. Za zmínku stojí ještě tlačítko **Vytvořit novou osobu…**. Toto tlačítko slouží k propojení s dalším modulem informačního systému Apollo, s modulem Osoby. V současné době toto spojení nefunguje, jelikož nejsou dořešeny všechny záležitosti s tímto související[3]. Aby uživatel mohl exportovat údaje nebo vytvářet novou osobu, musí mít právo administrátora přístupového systému.

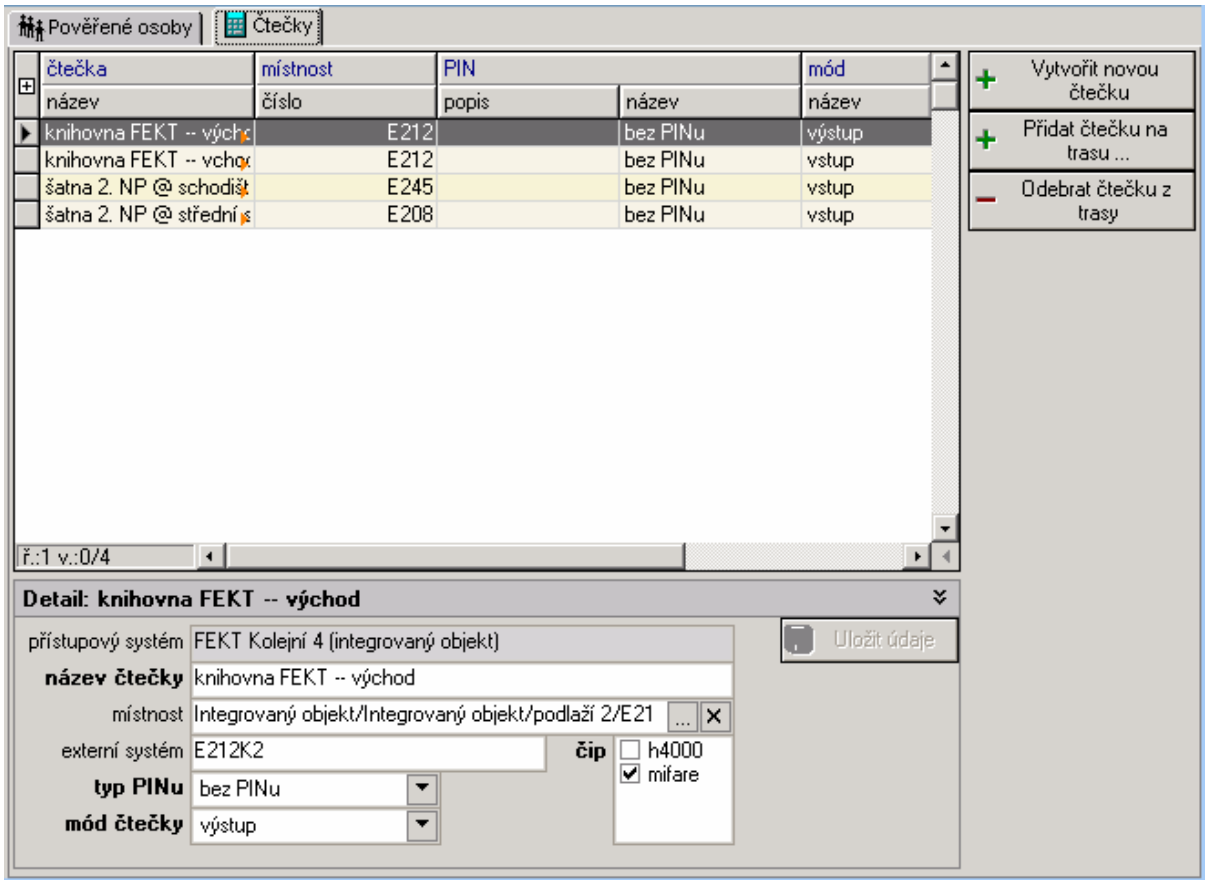


Obrázek 16: Položka Trasa – pověřené osoby

Druhá záložka nese jméno **Čtečky**. Z **obrázku 17** je patrné, že se na této záložce, stejně jako na předchozí, vyskytuje komponenta *TSMDBGrid*, která zde představuje seznam čteček, jež náleží k této trase. Taktéž se zde nachází panel s detaily, který ovšem obsahuje více komponent. Objevují se zde dvě dosud nepoužité komponenty. Těmito komponentami jsou *TApDoubleButtonEdit* (místnost) a *TApCheckListBox* (čip). Komponenta *TapDoubleButtonEdit* obsahuje textové pole a dvě tlačítka. Tlačítkem označeným třemi tečkami (…) se spouští dialogové okno pro přesný výběr místnosti ve všech areálech VUT v Brně. Místnost zvolená v dialogu se poté přiřadí do textového pole této komponenty. Druhé tlačítko, označené křížkem, slouží k vymazání textového pole. Důvodem k nevyplnění místnosti může být například čtečka umístněná na závoře ve vjezdu do areálu. V takové situaci by mohlo být obtížné, ba dokonce nemožné přiřadit správnou místnost. Komponenta *TApCheckListBox* zobrazuje seznam dvojic. Každou dvojici tvoří *checkbox* (zatrhávatelný čtverec) a

[3] Platné k 8. 5. 2008

textový popisek. Tato komponenta je zde využívána pro dynamické zobrazení a následně pro výběr čipů, které čtečka dokáže zpracovat. Data pro tuto komponentu se získávají přímo z databáze, což je velká výhoda z pohledu údržby modulu v případě, že by VUT v Brně zavedlo identifikační karty s novým typem čipu. Záložka **Čtečky** na své pravé straně obsahuje panel s tlačítky. Za zmínku stojí tlačítko **Přidat čtečku na trasu…**, které po stisknutí spustí dialog, jenž umožňuje přidat na trasu některou již existující čtečku, viz odstavec **Dialog Čtečky**.



Obrázek 17: Položka Trasa – čtečky

Dialog Seznam osob

Tento dialog je stejně jako ostatní postaven na třídě *TForm*. Jedná se vůbec nejsložitější dialog v modulu Přístupový systém. Lze to odhadnout i z **obrázku 18**, na němž je tento dialog zachycený. Dialog **Seznam osob** se dělí na tři horizontálně oddělené části.

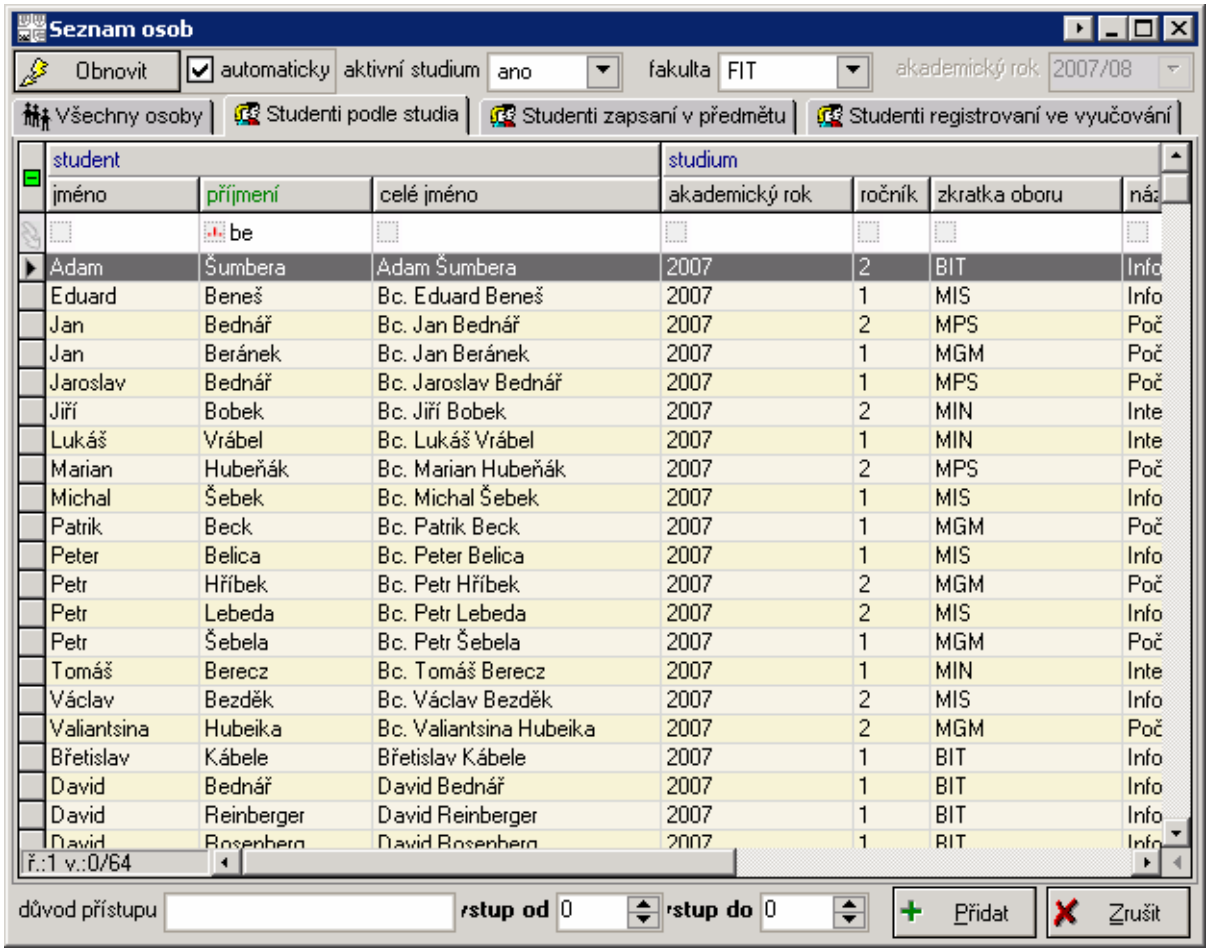
Vrchní část obsahuje jedno tlačítko třídy *TApImBtn*, jednu komponentu třídy *TapCheckBox*, dvě komponenty *TApComboBox* a jednu komponentu *TApYearComboBox*. Tlačítko **Obnovit** po stisknutí načte z databáze údaje o osobách podle kritérií vymezených filtrovacími *comboboxy* a filtry na *gridu* a těmito získanými údaji naplní komponentu *TSMDBGrid* na aktivní záložce. Komponenta *TApCheckBox* vytváří zatrhávací políčko. Toto políčko určuje, zda budou data v seznamu osob aktualizována automaticky (pokud je zatržené) při změně hodnot v *comboboxech* nebo k tomu bude potřeba zmáčknout tlačítko **Obnovit** (tlačítko se rozbliká, pokud budou údaje v tabulce neaktuální).

Třebaže se políčko **automaticky** může zdát zbytečné, plní z hlediska uživatele důležitou funkci. SQL dotazy, které vytvářejí seznamy studentů zapsaných v předmětu a studentů registrovaných ve vyučování, jsou složité a mají vazby přes několik obsáhlých tabulek. Z toho důvodu jsou tyto dotazy časově náročné a mohou trvat dokonce i několik desítek sekund. Pro uživatele je potom pohodlnější, když si nejprve uživatel nastaví veškeré filtry a až poté si nechá načíst data z databáze.

Komponenta *TApYearComboBox*, jež zobrazuje letopočet, umožňuje zvolit si formát, ve kterém se bude letopočet zapisovat. Pro potřeby filtrování studentů je zde použit formát akademického roku. Tato komponenta si sama vytváří seznam zobrazených roků. Stačí pouze nastavit rozmezí let, které chceme mít v možnostech a komponenta je pro nás ve správném formátu vytvoří.

V prostřední části dialogu se nachází lišta se záložkami. Podle vybrané záložky se určuje, které komponenty ve vrchní části budou či nebudou vidět, nebo budou povolené či zakázané. Každá záložka obsahuje seznam osob vytvořený pomocí komponenty *TSMDBGrid*, z níž lze najednou vybrat jednu nebo více osob. Záložky jsou čtyři a liší se v podrobnosti zobrazených údajů o osobách. V záložce **Všechny osoby** je jednoduchý seznam osob. Z tabulky na záložce **Studenti podle studia** lze zjistit program, obor a ročník studenta. Záložka **Studenti zapsaní v předmětu** k těmto údajům ještě přidává informace o zapsaných předmětech studenta. A nakonec záložka **Studenti registrovaní ve vyučování** rozšiřuje všechny tyto informace o studiu o přesný rozvrh hodin studenta a taky o místnost, v níž výuka probíhá.

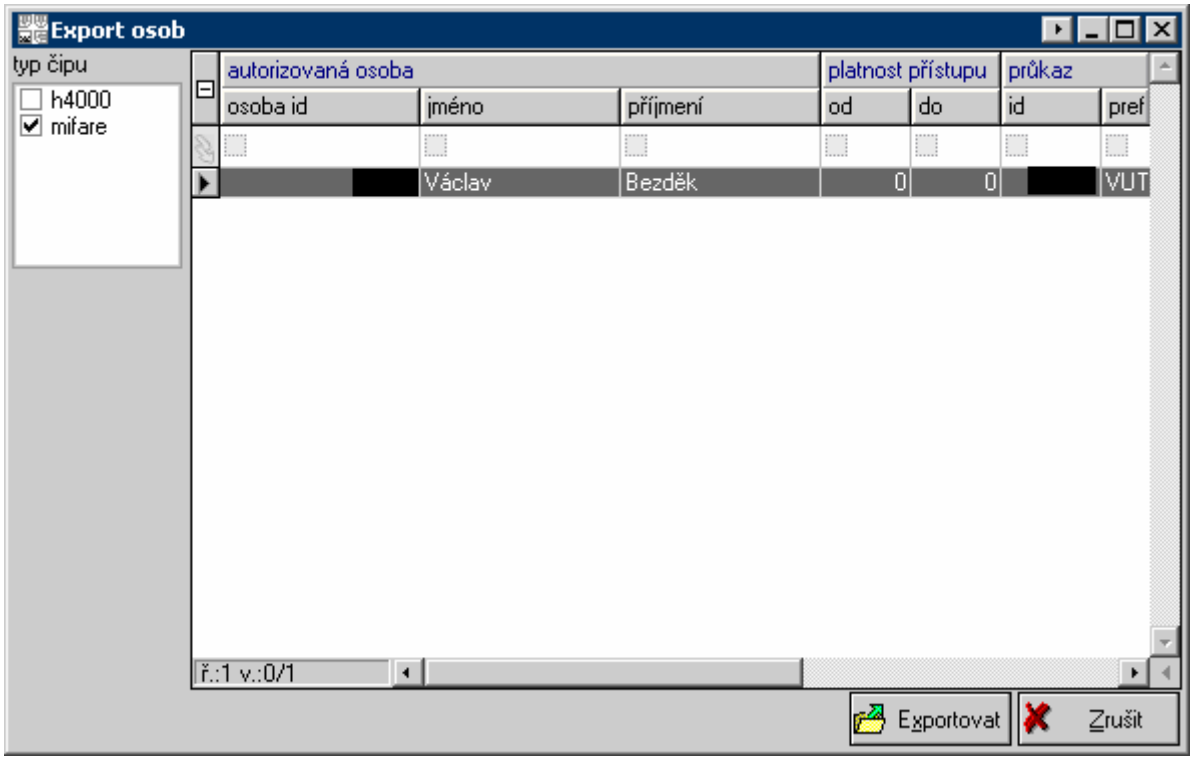
Ve spodní části dialogového okna se nacházejí tlačítka pro potvrzení výběru, respektive pro ukončení dialogu.



Obrázek 18: Dialog Seznam osob

Dialog Export osob

Tento dialog umožňuje uložit do souboru na disku podrobné údaje o identifikačních kartách osob, které mají přístup na trasu. Jeho vznik podnítila dosavadní neexistence automatické synchronizace údajů mezi centrální databází VUT a databázemi několika externích poskytovatelů přístupových systémů. Export zajišťuje komponenta *TApDmgExport*. Tato komponenta umožňuje zvolit sloupce, které se budou exportovat, nastavit oddělovač dat a vybrat název a typ souboru, do něhož se budou data ukládat. Exportují se jen ty řádky v tabulce, které jsou označeny. Vždy se exportuje minimálně jeden, aktuálně označený, řádek. Dialog **Export osob** je zobrazen na **obrázku 19**.



Obrázek 19: Dialog Export osob

Dialog Čtečky

O zobrazení tohoto dialogu se stará *TForm*. Tento jednoduchý dialog je podobný dialogu pro výběr správce trasy. Obsahuje pouze komponentu *TSMDBGrid*, která zobrazuje seznam čteček, a dvě tlačítka vytvořená komponentou *TApImBtn*. V seznamu je možné označit více položek. Tlačítkem **Přidat** se postupně volají SQL dotazy, které vytvářejí záznamy v tabulce READER_IN_ROUTE, čímž připojí čtečku na trasu.

5.5.3.4 Záložka Průchod čtečkou

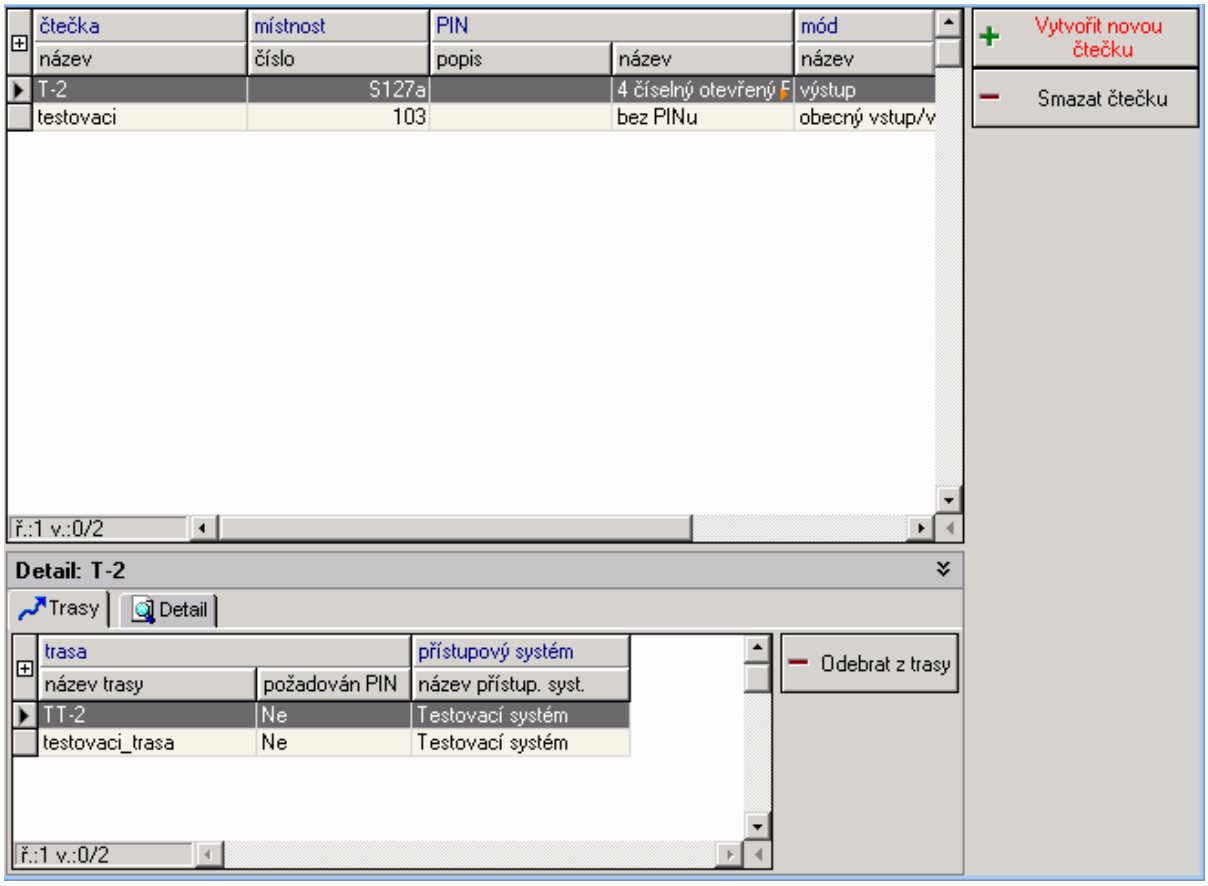
Tento *ApFrame* je velmi jednoduchý. Tvoří ho pouze komponenta *TSMDBGrid*, která zobrazuje uložené informace o průchodu přes zvolenou čtečku. Záznamy o průchodu nelze upravovat, slouží pouze k prohlížení.

5.5.3.5 Záložka Seznam čteček

Myšlenka na vytvoření seznamu čteček vznikla až v průběhu implementace modulu na základě požadavků budoucích uživatelů, kteří si přáli, aby byly všechny čtečky v přístupovém systému zobrazené přehledně na jednom místě. A zároveň byl vznesen požadavek na to, aby zde bylo možné vytvářet nové čtečky, modifikovat existující čtečky a mazat nepotřebné čtečky. *ApFrame* se rozděluje na dvě části – na část obsahující funkční tlačítka a na část slouží k zobrazení čteček a jejich detailů. Jak lze vidět na **obrázku 20**, v panelu s detaily se nachází dvě záložky. Záložka **Trasy** zobrazuje seznam všech tras, na které je čtečka přidělena. Vedle toho seznamu je tlačítko, jež po stisknutí

zavolá SQL dotaz, kterým se provede zneplatnění záznamu o propojení trasy a čtečky v tabulce READER_IN_ROUTE, čímž se čtečka odstraní z trasy. Druhá záložka, pojmenovaná **Detail** obsahuje stejné komponenty a plní stejnou funkci jako panel s detaily čtečky, který je popsán v podkapitole **Záložka Trasa**.

Na pravé straně, v panelu s funkčními tlačítky, se nacházejí dvě tlačítka. Za pozornost stojí tlačítko **Smazat čtečku**. Toto tlačítko zavolá SQL dotaz, který provede zneplatnění záznamu o čtečce v databázi. Pokud je čtečka přidělena na jednu nebo více tras, tak se zároveň zneplatní všechny záznamy v tabulkách READER_IN_ROUTE a READER_CHIP, které souvisejí s rušenou čtečkou.



Obrázek 20: Položka Seznam čteček

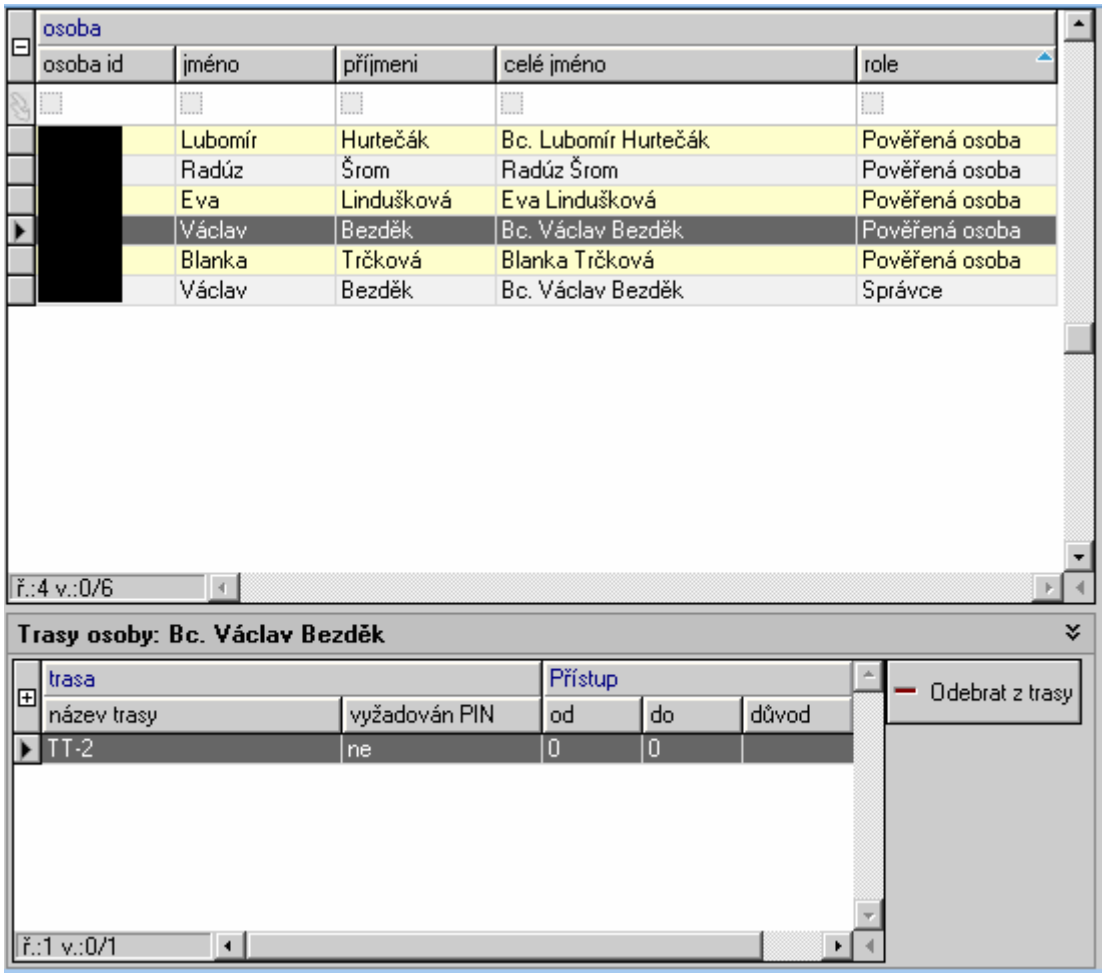
Záložka Seznam osob

ApFrame, který je odpovědný za zobrazení komponent, obsahuje komponentu *TSMDBGrid*, jež zde poskytuje tabulku se seznamem osob na všech trasách v přístupovém systému, a komponentu *TApCoolPanel*, která zajišťuje panel s detaily. Vzhled záložky **Seznam osob** ukazuje **obrázek 21**.

V seznamu osob se mohou některé osoby vyskytovat dvakrát v závislosti na tom, zda jsou zároveň správci a pověřenými osobami.

Panel s detaily obsahuje lištu záložek se dvěma záložkami. Tyto záložky jsou pro uživatele skryté a nezobrazují se. První záložka se jmenuje **Správce** a její obsah se zobrazí pouze tehdy, když má osoba vybraná v seznamu roli *správce*. Na této záložce se nachází komponenta *TSMDBGrid*, která

zobrazuje seznam tras, na nichž je vybraná osoba správcem. Dále se zde nachází tlačítko **Odebrat správcovství**, které po stisknutí zavolá SQL dotaz, kterým se zneplatní záznam o správcovství v tabulce ROUTE_MASTER. V případě, že jsou osobě odebrány všechny trasy, jež spravuje, neexistuje důvod, aby tato osoba s rolí *správce* byla dále zobrazena v seznamu osob, a proto se při obnovení dat v seznamu už nezobrazí. Druhá záložka je pojmenována **Pověřená osoba**, a její zobrazení je podmíněno rolí *pověřená osoba* u osoby vybrané ze seznamu. Stejně jako na první záložce, i zde se nacházejí dvě komponenty, *TSMDBGrid* a *TApImBtn*. Komponenta *TSMDBGrid* zobrazuje seznam tras, na které má osoba právo přístupu. Tlačítkem **Odebrat z trasy** se provede SQL dotaz, který zneplatní záznam o povolení na trasu v tabulce PERSON_ON_ROUTE. Stejně jako v předchozím případě u správce trasy platí, že po odebrání všech trasy, přestává být osoba s rolí *pověřená osoba* zobrazována v seznamu osob.



Obrázek 21: Položka Seznam osob

5.6 Průběh implementace

V průběhu implementace vznášeli budoucí uživatelé další požadavky na funkčnost modulu. Tyto nové požadavky sebou přinášely nutnost rozhodnout se, jak se postupovat při jejich řešení.

Většinou se nabízelo několik variant, jak přidat požadovanou funkcionalitu. Příkladem toho může být implementace seznamu čteček a seznamu osob. Nabízela se možnost vytvořit na záložce **Přístupový systém** lištu se záložkami, které by obsahovali seznam tras, čteček a osob. Další možností bylo vytvoření tlačítek **Seznam osob** a **Seznam čteček** na stejné záložce. Tyto tlačítka by spouštěly dialogová okna, v nichž by byly požadované seznamy zobrazeny. Poslední třetí variantou bylo přidání položek <seznam osob> a <seznam čteček> do stromové struktury. Druhá možnost byla zamítnuta hned v úvodu, jelikož nebyla přijatelná z pohledu dosavadního řešení struktury modulu, zejména kvůli tomu, že nebylo žádoucí mít dialogová okna, z nichž by se spouštěly další dialogy. Na výběr zbyla první a třetí varianta. Obě varianty byly na podobné úrovni z hlediska dosavadního řešení struktury modulu, nicméně bylo rozhodnuto implementovat třetí variantu, která se zdála být vhodnější z pohledu jednoduchosti ovládání a přehlednosti zdrojového kódu.

V jednom případě musel být požadavek uživatele zamítnut, jelikož by jeho realizace vedla ke snížení přehlednosti modulu. Jednalo se o to, že si uživatelé přáli mít v seznamu čteček zobrazené informace o čipu, s nímž dokáže čtečka pracovat. Tento požadavek by nebylo obtížné naprogramovat, ale problém byl v tom, že jedna čtečka může přečíst několik typů čipu. Zavedením sloupce s typem čipu do seznamu čteček by se začaly zobrazovat některé čtečky vícekrát v závislosti na tom, kolik různých typů čipu dokáží přečíst. Po diskuzi, v rámci níž jsem uživateli ukázal tento problém, bylo z jeho strany uznáno, že zavedení takovéto redundance není žádoucí.

6 Nasazení do provozu

Nasazením do provozu se myslí uvolnění modulu pro běžné uživatele, kteří nemají přístup na testovací databázi. Jelikož byl modul Přístupový systém vyvíjen na testovací instanci databáze, bylo potřeba před uvedením do provozu zajistit, aby na hlavní, takzvaně „ostré“, instanci databáze byly všechny součásti modulu. Databázové tabulky byly stejné na všech instancích databáze, ale bylo potřeba doplnit údaje do tabulek C_GRANTED, READER_PIN_TYPE a READER_MODE. Dále bylo nutné převést SQL dotazy, jelikož bez nich by modul nemohl fungovat. Tyto SQL dotazy byly vyexportovány z testovací databáze do souboru typu SQL skript. Vytvořený soubor byl následně spuštěn na „ostré“ databázi pomocí programu PL/SQL Developer, čímž se tyto dotazy nahrály do databáze. Nakonec bylo potřeba modul zkompilovat a výslednou knihovnu *exPristupovySystem.dll* nahrát na aktualizační server, z něhož se každých několik minut nahrávají změny na hlavní server informačního systému Apollo.

V rámci nasazení modulu byla vytvořena uživatelská příručka, viz podkapitola **9.1 Uživatelská příručka**. Cílem uživatelské příručky je předvést budoucím uživatelům práci s modulem a ukázat jim všechny jeho funkce.

Dalším úkolem byla prezentace modulu klíčovým uživatelům a proškolení budoucích uživatelů. Obě tyto činnosti byly provedeny v rámci komunikace s budoucími uživateli, jež probíhala v průběhu implementace. Klíčovým a zároveň budoucím uživatelům z Fakulty elektrotechniky a komunikačních technologií, panu ing. Pokornému a panu ing. Trávníčkovi byly vysvětleny funkce a bylo jim předvedeno ovládání modulu ihned po vydání první verze modulu. Oba poté zasílali své požadavky na další funkce modulu, případně upozorňovali na některé chyby, jež nalezli při práci s modulem.

7 Závěr

7.1 Hodnocení dosažených výsledků

V rámci diplomové práce jsem se seznámil s problematikou přístupového systému VUT. Zanalyzoval jsem existující schéma přístupového systému a mírně jsem ho upravil. Z výsledků analýzy jsem navrhnul rozvržení hlavních prvků aplikace a vytvořil uživatelské role, které se budou v aplikaci vyskytovat. Vytvořil jsem diagram případů použití, který stručně zobrazuje možné případy použití aplikace v závislosti na roli uživatele.

Po vytvoření návrhu jsem se informoval na fungování centrálního informačního systému Apollo. Zjistil jsem se, že aplikace, kterou je možno nalézt na internetových stránkách VUT, je jen špička ledovce pojmenovaného informační systém Apollo. Dozvěděl jsem se, že za touto aplikací stojí spousta úsilí skupiny lidí, počínaje techniky starajícími se o bezporuchový chod hardwarového vybavení, přes správce databáze, kteří jsou odpovědní za bezproblémový chod databáze, až po programátory, jež neustále vyvíjí nové funkce do informačního systém a zdokonalují funkce již vytvořené. Od programátorů informačního systému Apollo jsem se naučil, jak vytvářet aplikace pro informační systém. Pomocí jejich rad jsem byl schopen vytvořit navrženou aplikaci a přidat ji do informačního systému Apollo. V současné době je modul Přístupový systém v provozu.

K aplikaci jsem vytvořil uživatelskou příručku, jež by měla poskytnout budoucím uživatelům modulu Přístupový systém dostatek informací potřebným k ovládání a používání tohoto modulu.

Věřím, že práce, kterou jsem na modulu Přístupový systém vykonal, byla prospěšná, a že lidem, kteří budou tento modul v budoucnu vyvíjet a rozšiřovat, poskytne kvalitní základ pro jejich práci.

7.2 Další vývoj

Modul přístupový systém má velký potenciál pro budoucí rozšíření. Byla vytvořena základní část, kterou je možno dále rozvíjet. Rozvojem modulu se myslí zejména zautomatizování některých pracných činností, do nichž se může lehce vloudit lidská chyba. Za jednu z těchto činností je možno považovat nastavování přístupů na trasy. Pokud by se přístupový systém provázal se systémem registrace předmětů a vyučování, bylo by možno, aby se studentům po registraci a po zapsání předmětu automaticky povolil přístup na trasy nezbytné k tomu, aby se mohli dostat do místností, v nichž se koná výuka.

S nastavováním přístupů souvisí také jejich odebírání. Systém by mohl automaticky podle rozvrhu vyučování vyhodnotit trasy, na které už student nepotřebuje mít přístup a tento přístup mu

odebral. Zároveň by zde musel být zavedený mechanismus, jímž by se zajistilo, aby osobě nemohla být některá přístupová práva automaticky odebrána.

Dalším možným rozšířením je zprovoznění časového omezení přístupu na trasu. Časové omezení přístupu by mohlo být propojeno se zmíněným automatickým nastavováním přístupů, při kterém nebude problém zjistit čas výuky a pomocí něho zavést omezení přístupu na trasu. Časové omezení přístupu na trasu však se sebou přináší jeden problém, na který je potřeba myslet. Studentovi musí být umožněno opustit trasu, když uplyne vymezený čas přístupu.

Ještě se nabízí jedna zajímavá funkce přístupového systému. Systém by mohl u čteček v garážích počítat počet automobilů, které vjedou dovnitř a odečítat od nich počet vozidel, jež vyjedou ven. Výslednou hodnotu by poté systém porovnal se zadaným s počtem parkovacích míst. Pokud by byla zaplněna kapacita garáže, systém by dovnitř nevpustil další vozidla. Na druhou stranu by tohle řešení bylo poněkud složitější, jelikož v garážích mohou být čtečky použité zvlášť na vjezd a zvlášť na výjezd. Kvůli tomu by se muselo dobře nastavit propojení správných čteček. V případě existence pouze vstupní čtečky a automatického čidla na výjezdu by se situace výrazně komplikovala. V tomto případě by musel přístupový systém zjišťovat údaje z automatického čidla, aby mohl správně vyhodnotit počet vozidel v garáži.

7.3 Vlastní přínosy diplomové práce

Práce na tomto projektu mi přinesla spoustu nových zkušeností. V první řadě jsem si rozšířil povědomí o možnostech jazyka SQL a zároveň jsem si v této oblasti zlepšil svoje schopnosti díky nutnosti vytváření rozsáhlých a složitých SQL dotazů. Dále jsem získal cenné zkušenosti s prací v týmu. Uvědomil jsem si, že základem je nebát se na cokoliv se zeptat, protože většina základních problémů, se kterými jsem se jako nezkušený potýkal, se už někdy objevila a je vyřešena.

Asi nejvíce si cením zkušeností získaných z komunikace s budoucími uživateli. Doposud jsem se při řešení žádného školního projektu nesetkal s tím, že se objevují nové požadavky na aplikaci, na které musím nějak reagovat. Buď je bez připomínek implementovat, a nebo si obhájit jejich úpravu, či dokonce zamítnutí. Myslím, že tyto nabyté zkušenosti se mi budou v budoucím profesním životě velmi hodit.

8 Literatura

[1] Loney, K., Theriault, M.: Mistrovství v Oracle, Praha, Computer Press 2002, ISBN 80-7226-635-7

[2] Lacko, Luboslav: Oracle – Správa, programování a použití databázového systému. Praha, Computer Press 2002. ISBN 80-7226-699-3

[3] Katedra informatiky, VŠB Ostrava: Systém Oracle. Dokument dostupný na URL: http://www.cs.vsb.cz/ticha/oracle/hlavni1.html (květen 2008)

9 Seznam příloh

9.1 Uživatelská příručka

9.1.1 Apollo

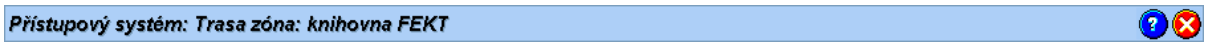
Titulkový pruh

V titulkovém pruhu je vždy uveden název text Informační centrum VUT, za kterým je v závorce napsána verze systému. Čárkou je oddělený název spuštěného modulu, zde je to Přístupový systém. V závorce je uvedena verze modulu.

Panel nástrojů

Umožňuje rychlý přístup k často používaným funkcím. Pokud se příkazy zobrazují šedě, nelze je aktuálně použít.

Informační pruh modulu



V tomto pruhu je vždy uveden název modulu, zde Přístupový systém. Dále je zde uvedena informace o tom, v jaké úrovni hierarchie přístupového systému se nacházíme a její označení.

Tlačítko nápověda : Klepnutím na toto tlačítko se otevře nápověda k modulu, se kterým právě pracujeme.

Tlačítko zavřít : Slouží k zavření aktuálního modulu. Program zůstává nadále spuštěn.

Lokální nabídky

Jedná se o vysouvací menu, které se zobrazí po klepnutí pravým tlačítkem myši na některý prvek a zpřístupní vám příkazy související s tímto prvkem.

Posuvníky

Jestliže se prvky modulu (většinou např. seznam v tabulce) nevejdou na celou obrazovku, zobrazí se po pravé straně (pro vertikální posun) a dole (pro horizontální posun) posuvníky, s jejichž pomocí se můžeme po dokumentu posouvat. Můžeme použít několik způsobů:

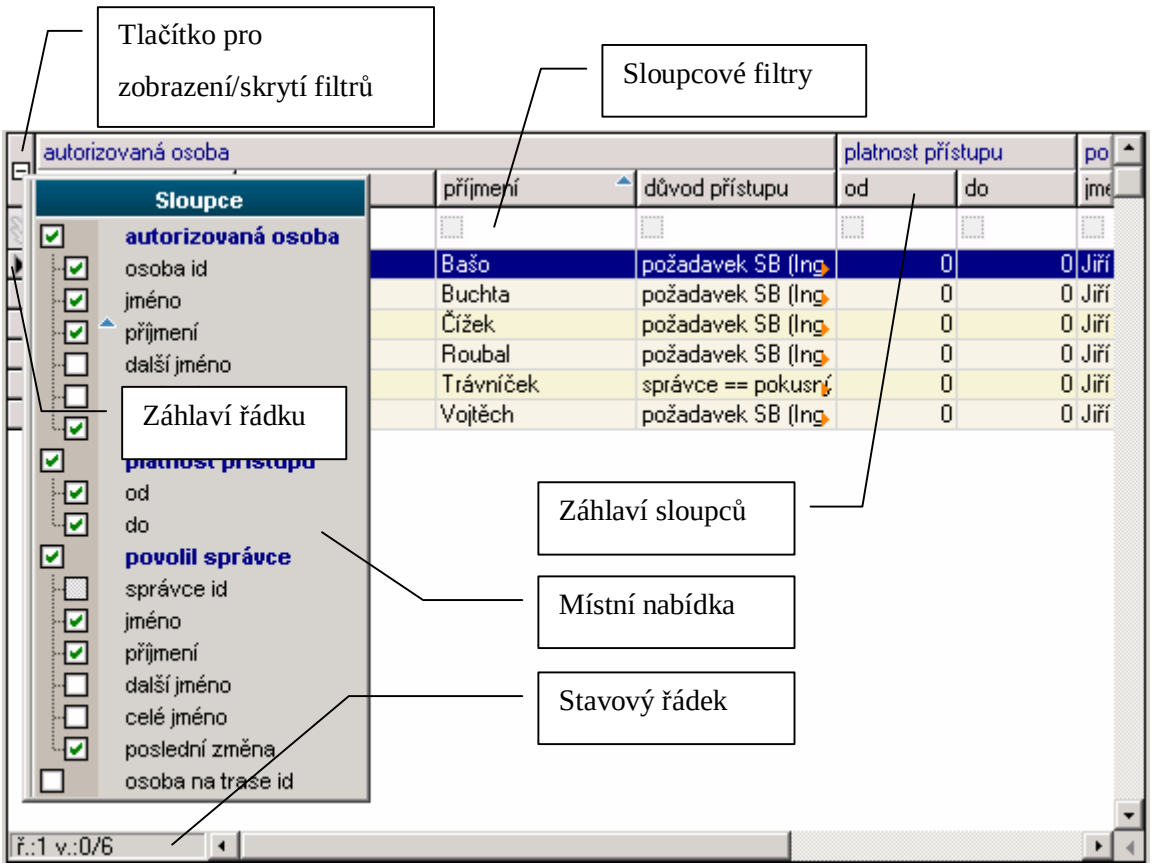
- Kliknutím myší na malé šipky ukazující směr.

- Tažením posuvníku.
- Klepnutím mezi posuvník a šipku.
- U myši s kolečkem se lze posouvat nahoru a dolů otáčením kolečka patřičným směrem.

Řádek s otevřenými moduly

Tento řádek nám poskytuje přehled, o tom jaké moduly máme spuštěné. Kromě toho můžeme ikony (tlačítka) využívat k tomu, abychom se mezi jednotlivými moduly přepínali. Klikneme-li na ikonu, objeví se okno modulu na obrazovce a stane se aktivním. Kliknutím do horního rohu ikony, v němž se nachází na malé tlačítko označené křížkem, můžeme modul zavřít. Aktivní modul poznáme podle toho, že je jeho tlačítko na liště je zmáčknuté.

9.1.2 Práce s tabulkami



Obrázek 22: Ovládání tabulky

Úprava tabulky

Změna šířky sloupce: Pro úpravu šířky sloupce stačí najet myší na pravou stranu čáru oddělující požadované pole v záhlaví sloupců. Po zobrazení obousměrné šipky stačí podržet tlačítko myši a táhnout požadovaným směrem.

Změna výšky řádku: Pro úpravu výšky řádku platí podobná věc, jen se najíždí myší na spodní okraj požadovaného řádku.

Přesun sloupce: Přidržíme tlačítko myši na záhlaví příslušného sloupce a tahem myší přesuneme tento sloupec na námi zvolenou pozici (myší se pohybujeme po záhlaví sloupců).

U některých záznamu se zobrazuje červená šipka v pravém horním rohu (většinou se jedná o zkratky), najedeme-li na ni kurzorem, vyvoláme plovoucí nápovědu obsahující význam zkratky

Výběr záznamu ze seznamu

Záznam, na kterém je umístěn kurzor se v záhlaví řádku označí černou šipkou a v seznamu se zobrazí tmavě modře. Záznam, který je vybrán má v záhlaví řádku černou tečku a v seznamu je zobrazen světle modře.

Výběr všech záznamů:

1. Klávesovou zkratkou CTRL+A: Když klepneme myší na kterýkoliv záznam v seznamu a podržíme klávesu CTRL a zároveň s ní stiskneme klávesu A.
2. Vyvoláním místní nabídky *Sloupce*: Když klepneme pravým tlačítkem myši na kterýkoliv záznam, zobrazí se nám místní nabídka, ve které zvolíme *Označit vše.*

Odznačení všech záznamů: Provedeme vyvoláním místní nabídky *Sloupce* a zvolením příkazu *Odznačit vše* nebo klepnutím myší na libovolný záznam.

Záznamy, které jsou v seznamu setříděné pod sebou, můžeme označit několika způsoby:

- Klepneme myší na první požadovaný záznam, stále držíte tlačítko a přetáhneme kurzorem myši všechny potřebné záznamy.
- Klikneme myší na první požadovaný záznam, podržíte klávesu SHIFT, a následně klikneme myší na poslední záznam a pustíme klávesu SHIFT (v seznamu se pohybujeme pomocí kolečka myši nebo posuvníku na pravé straně).

Záznamy, které na sebe nenavazují:

- Klikneme myší na první požadovaný záznam, stále držíme tlačítko a přetáhnete kurzorem myši všechny potřebné záznamy. Posunete se po seznamu pomocí pravého posuvníku nebo kolečka myši k dalšímu záznamu. Opět na něj klikneme a při stisknutém tlačítku myši přetáhneme kurzorem další potřebné záznamy. A tak pokračujeme dále i u ostatních záznamů. Pokud bychom chtěli ke svému výběru přidat pouze jeden záznam, musíme stisknout klávesu CTRL a teprve potom na něj klepnout myší.

Výběr záznamu pomocí filtru:

- Zvolením příslušného filtru získáme potřebné záznamy, které pak můžeme všechny označit.

Místní nabídky

Nastavení sloupců: Jedná se o roletovou nabídku, která se rozbalí po kliknutí pravým tlačítkem myši na tlačítko v levém horním rohu tabulky (na ikoně ⊟ nebo ⊞). Jsou zde uvedeny názvy všech sloupců, které si můžeme nechat dle potřeby zobrazit v tabulce. Momentálně zobrazené sloupce jsou označeny v zatrhávacím políčku. Pokud chceme některý sloupec zobrazit nebo skrýt, označíme nebo odznačíme jej v příslušném zatrhávacím políčku.

Sloupce: Nabídku vyvoláme klepnutím pravého tlačítka myši na záhlaví sloupce. V nabídce je uveden název sloupce a jednotlivé příkazy. Pokud se příkazy zobrazují šedě, nelze je v aktuální situaci použít. Stručný popis příkazů:

- **Skrýt sloupec:** Skryje sloupec, na který jsme kliknuli. Můžeme si jej nechat znovu zobrazit přes místní nabídku *Nastavení sloupců*, zatržením příslušného políčka.
- **Na začátek (F7) / konec (F10) sloupce:** Slouží k přesunu výběru na první / poslední záznam v seznamu
- **Seřadit vzestupně:** Příkazem se řadí data podle číselných i textových hodnot. Šipky v „Záhlaví sloupců“ směřují nahoru.
- **Seřadit sestupně:** Příkazem se řadí, obdobně jako u příkazu *Seřadit vzestupně*, data podle číselných i textových hodnot. Šipky v „Záhlaví sloupců“ směřují dolů.
- **Zrušit řazení:** Zruší již dříve nastavené seřazení záznamu v seznamu pouze u zvoleného sloupce.
- **Označit / odznačit sloupec:** Pomocí tohoto příkazu můžeme označit libovolný počet sloupců a exportovat jejich data do schránky. Pro export můžete použít klávesové zkratky **CTRL+C**.
- **Ukotvit sloupce:** Ke své práci někdy potřebujeme mít zobrazené velké množství sloupců a tabulka se potom stává nepřehledná. Příkazem *Ukotvit sloupce* ukotvíme požadované sloupce tak, že při posouvání se seznamem do strany jsou požadované sloupce stále viditelné.

Řádky: Nabídku vyvoláte klepnutím pravého tlačítka myši na záhlaví řádku.

- **Zobrazovat číslo řádku:** Zatržením tohoto příkazu (klepnutím myší na příkaz) se nám zobrazí v záhlaví řádku jejich čísla, odtržením čísla skryjeme.
- **Automatická výška řádku:** Příkazem se řádky přizpůsobí tak, aby vyhovovaly nejvyššímu zápisu.

Seznam: Pokud klepneme pravým tlačítkem myši na kterýkoliv záznam v seznamu, vyvoláme místní nabídku:

- **Přizpůsobit šířku sloupců:** Přizpůsobí šířku sloupců tak, aby se všechny současně zobrazily na obrazovce.
- **Označit vše:** Označí všechny záznamy v seznamu.
- **Odznačit vše:** Odznačí všechny záznamy v seznamu.
- **Kopírovat do schránky (jako CSV):** Kopíruje označená data do schránky, tyto data můžete pak pomocí klávesové zkratky CTRL + V vložit do požadovaného programu (WORDU, EXCELU apod.).
- **Kopírovat do schránky (pro EXCEL):** Kopíruje označená data do schránky, jako v předcházejícím příkaze, tyto data jsou vhodná pro práci v EXCELU.
- **Kopírovat obsah buňky do schránky:** Zkopíruje obsah vybrané buňky do schránky.
- **Filtrovat podle hodnoty v buňce:** Hodnotu z buňky vloží do filtru sloupce, ve kterém se buňka nachází.
- **Automaticky přizpůsobovat šířku:** Podobné jako volba *Přizpůsobit šířku sloupců*, ale na rozdíl od něj automaticky přizpůsobuje šířku sloupců po každé změně.
- **Načíst výchozí nastavení:** Zruší všechna naše nastavení vzhledu seznamu, ať už se jedná o ukotvení sloupců, jejich šířku nebo zobrazení. Této funkce je možné využít také v případě, když se v tabulce vyskytuje nějaký problém (např. je v ní všude napsáno červeně ERROR). Ve většině případu se problém touto funkcí vyřeší.
- **Zrušit všechny filtry:** Pokud máme v seznamu nastavené nějaké filtry, budou tímto příkazem zrušeny
- **Rozlišovat vel. písmen:** Tato funkce bývá využívána při filtrování. Standardně je vypnuta, takže nezáleží, zda při vyhledávání napíšete do políčka pro filtry „Novák“ nebo „novák“.

Filtry

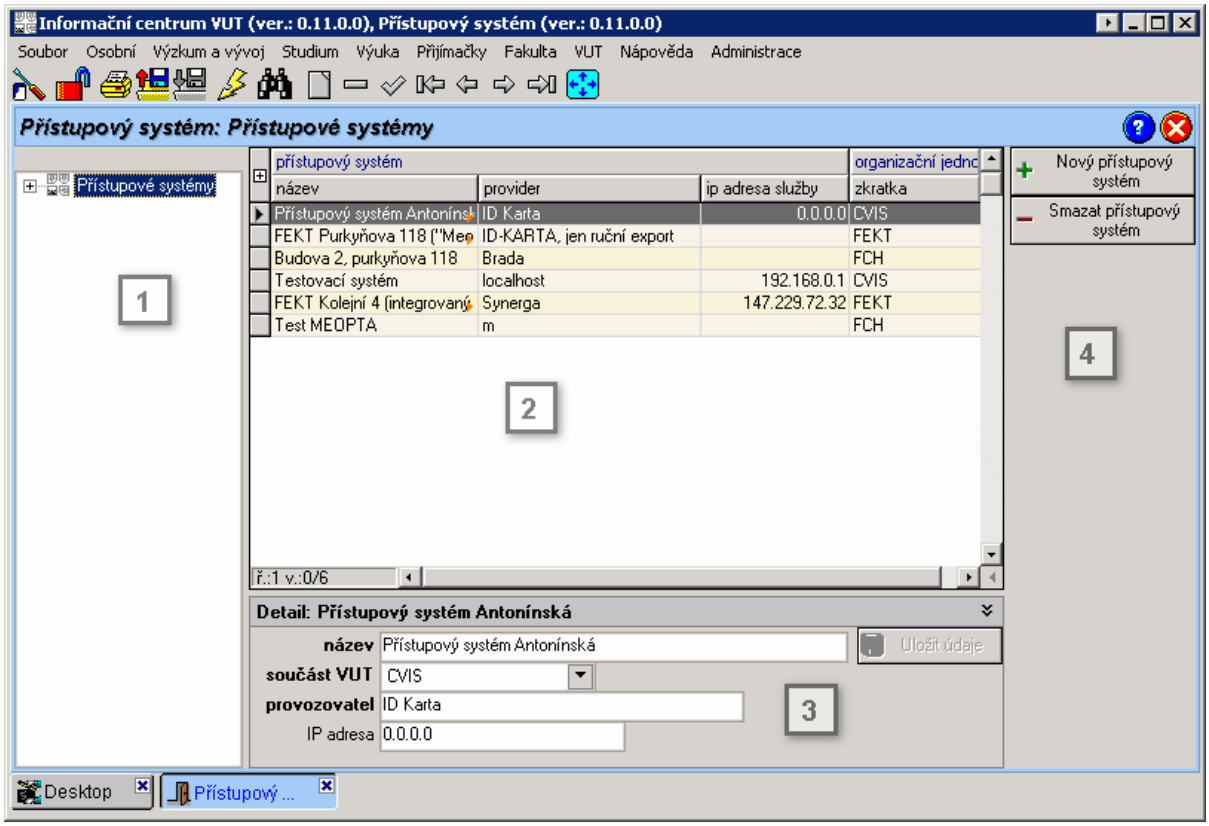
Tlačítko pro zobrazení / skrytí filtru (CTRL+F): Chceme-li si nechat zobrazovat v seznamu řádek s filtry, stiskneme tlačítko pro zobrazování filtru, to se současně změní na tlačítko pro skrytí. Pokud je v seznamu zvolen některý z nabízených filtru, svítí toto tlačítko zeleně, jinak je bílé.

Tabulka obsahuje místní nabídku pro výběr typu filtru. Tuto nabídku vyvoláme pomocí klepnutí pravého tlačítka myši do malého okénka v políčku pro filtr. Vybereme některý z nabízených typů filtru a jeho ikona se zobrazí v okénku (další možností je kliknout do okénka levým tlačítkem myši. Poté se v něm zobrazí první filtr z místní nabídky, pokud klepneme znova tak druhý atd.). Potom zadáme hledanou hodnotu do políčka filtru. V seznamu můžete mít zvolen libovolný počet filtru. Všechny filtry můžete současně vypnout příkazem z místní nabídky *Seznam – Zrušit všechny filtry*.

Tlačítko pro obnovení dat : Toto tlačítko zobrazí v tabulce data v závislosti na zvolených filtrech.

9.1.3 Spuštění modulu Přístupový systém

Modul Přístupový systém se pouští z Hlavního menu -> VUT -> Přístupový systém. Zobrazí se úvodní obrazovka modulu, viz **obrázek 23**. Obrazovka modulu se dělí na čtyři části. V levé části (označené číslem 1) se nachází zobrazovací strom. V tomto stromu se nachází hierarchicky zobrazené prvky přístupových systémů. Ve střední části obrazovky se nachází seznam (2), který zobrazuje přímo podřízené prvky k prvku, který je zvolený v zobrazovacím stromu. Pod tímto seznamem se nachází panel (3), v němž jsou zobrazeny detaily o vybrané položce v seznamu. Tento panel lze skrýt pomocí tlačítka . Když je tento panel skrytý, tak stisknutím tlačítka se zase objeví. A v pravé části okna (4) je panel s funkčními tlačítky, jejich pojmenování a funkčnost se mění v závislosti na prvku, který je zvolený v zobrazovacím stromu.



Obrázek 23: Přístupové systémy – Vytváření, úprava a odstraňování přístupových systémů

9.1.4 Vytvoření, úprava a odstranění přístupového systému

Pokud chceme vytvořit nový přístupový systém, musíme v zobrazovacím stromu vybrat položku **Přístupové systémy**. Následně se nám zobrazí okno, podobné jako na **obrázku 23**. Zde musíme stisknout tlačítko **Nový přístupový systém**. Tím se nám v seznamu vytvoří nová prázdná položka,

kurzor se přesune na položku název v panelu s detaily a všechny položky v tomto panelu budou prázdné. Tučně zvýrazněné položky jsou povinné, tudíž musí být vyplněny. Když vyplníme všechny povinné údaje a zároveň jsou všechny tyto údaje platné (text v položce **IP adresa** musí odpovídat tvaru IP adresy), zpřístupní se tlačítko **Uložit údaje**, jehož stisknutím uložíme vytvořený přístupový systém do databáze.

Úpravy údajů přístupového systému provádíme zvolením příslušného přístupového systému v seznamu. Tím se informace o tomto systému zobrazí v panelu s detaily. Zde je můžeme jednoduše editovat. Uložení změněných údajů provedeme tlačítkem **Uložit údaje**. Toto tlačítko je nepřístupné, pokud nebyly provedeny žádné změny, nejsou vyplněny povinné údaje nebo jsou vyplněné údaje v neplatném tvaru.

Smazání aktuálně vybraného přístupového systému v seznamu provedeme tlačítkem **Smazat přístupový systém**. Po jeho stisknutí se objeví potvrzovací dialog, který se nás ptá na potvrzení rozhodnutí o vymazání zvoleného přístupového systému. Po jeho potvrzení se přístupový systém odstraní. Zároveň se odstraní všech trasy a čtečky tohoto přístupového systému, a s nimi i pověřené osoby a správci tras.

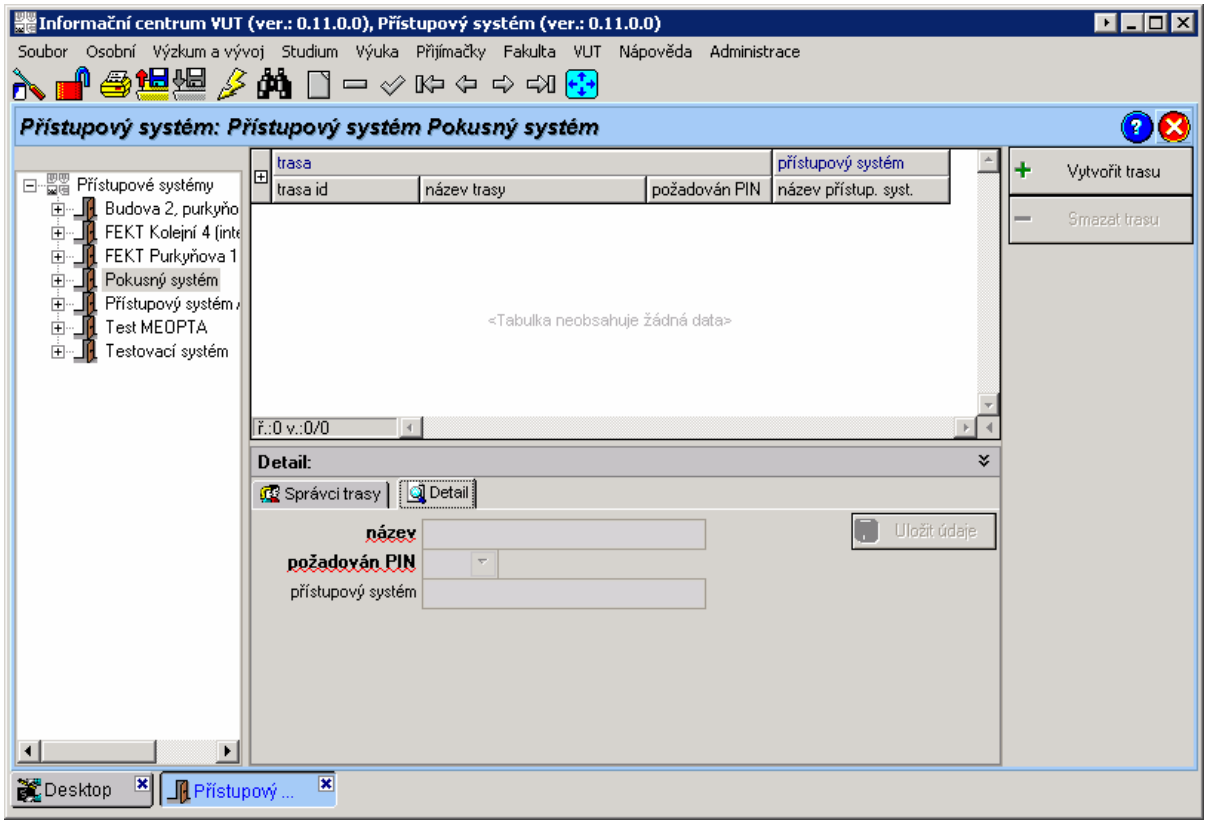
9.1.5 Přidání, úprava a odstranění trasy

Po vytvoření přístupového systému je potřeba do tohoto systému vložit trasy. Toho docílíme tím, že v zobrazovacím stromu vybereme přístupový systém, který jsme vytvořili, viz **obrázek 24**.

Novou trasu vytvoříme tlačítkem **Vytvořit trasu**. Tím se nám přesune kurzor na položku název v panelu s detaily a všechny editovatelné položky se nastaví „defaultní“ hodnoty. Po vyplnění povinných polí se zpřístupní tlačítko **Uložit údaje**. Po jeho stisknutí se nová trasa uloží do databáze.

Editace trasy probíhá stejným způsobem jako editace přístupového systému. Ze seznamu tras si vybereme požadovanou trasu, čímž se podrobnosti o ní zobrazí v panelu s detaily v záložce **Detail**, kde je můžeme upravovat. Uložení úprav provedeme tlačítkem **Uložit údaje**.

Odstranění trasy provedeme zvolením příslušné trasy v seznamu tras a stisknutím tlačítka **Smazat trasu**. Poté se objeví dialog, vyžadující potvrzení našeho rozhodnutí. Když ho potvrdíme, odstraní se trasa ze systému. Zároveň s ní se odstraní pověřené osoby, správci tras a propojení všech čteček s touto trasou.

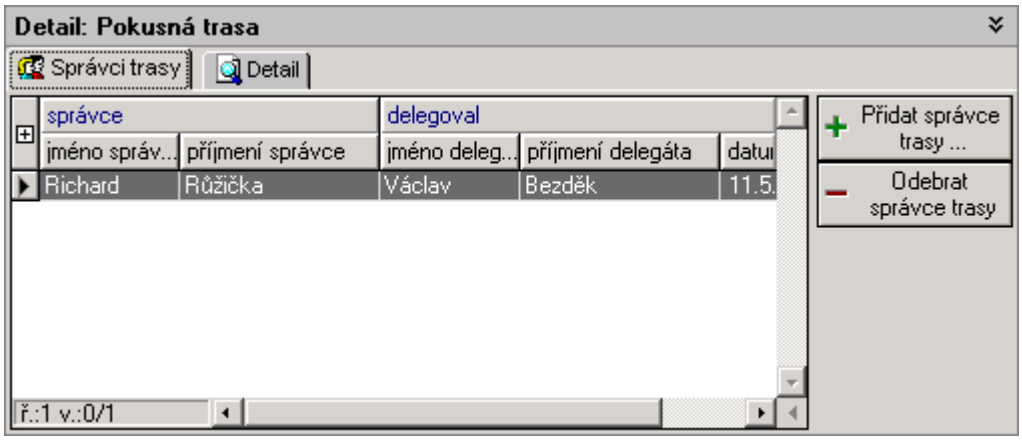


Obrázek 24: Přístupový systém – Přidávání, editace a odstraňování tras

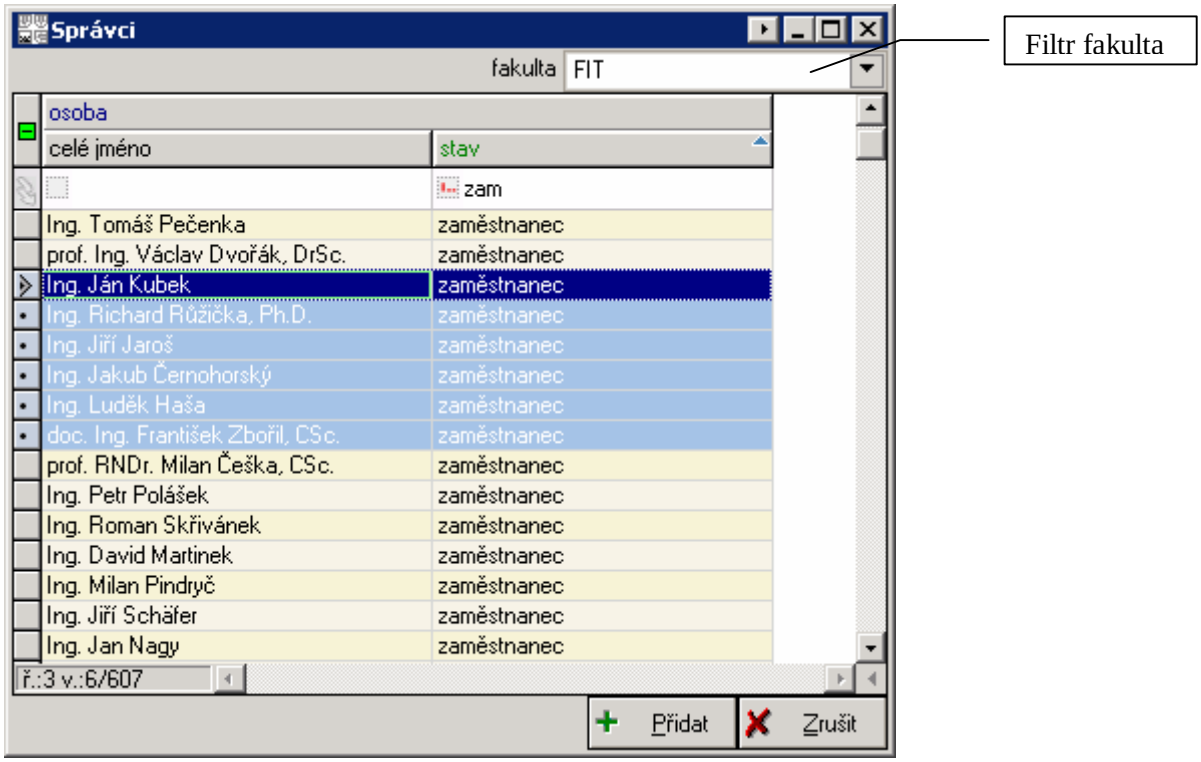
Přidání a odebrání správce trasy

Každá trasa může mít jednoho a více správců. Přiřazovat správce na trasu můžeme v panelu s detaily na záložce **Správci trasy**, viz **obrázek 25**. Zde je zobrazen seznam správců trasy. Vedle tohoto seznamu jsou funkční tlačítka.

Tlačítkem **Přidat správce trasy...** se spustí dialog *Správci*, viz **obrázek 26**, v němž zvolíme osoby, které se mají stát správci trasy. V horní části tohoto dialogu je filtr **fakulta**, s jehož pomocí můžeme vybrat fakultu nebo součást VUT, pod kterou spadá osoba budoucího správce. Po výběru osoby nebo více osob je přidáme na trasu stisknutím tlačítka **Přidat** a potvrzením tohoto rozhodnutí. Tlačítkem **Zrušit** zavřeme dialogové okno bez provedení jakýchkoliv změn.



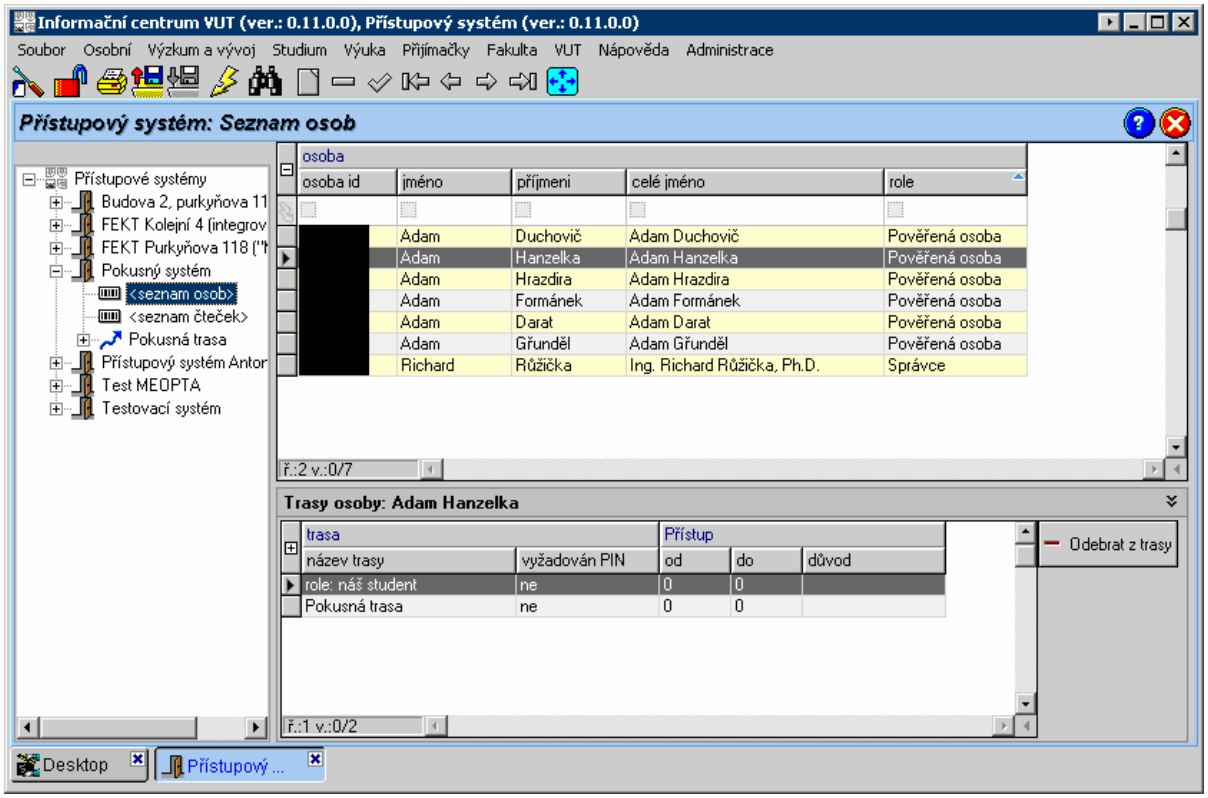
Obrázek 25: Přístupový systém - Trasa - Správci trasy



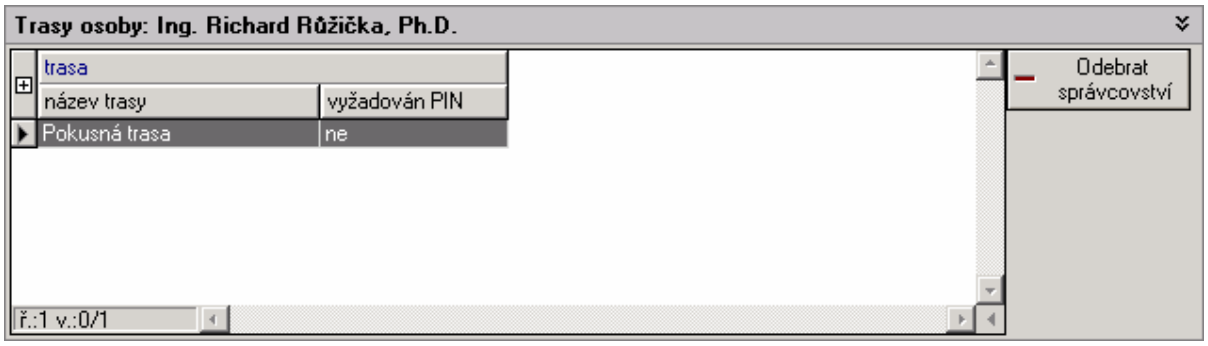
Obrázek 26: Přístupový systém – Trasa – Správci trasy – Dialog Správci

Odebrání správce trasy provedeme vybráním příslušné osoby ze seznamu správců a stiskem tlačítka **Odebrat správce trasy**. Toto rozhodnutí taktéž musíme potvrdit.

Odebrat správce trasy můžeme ještě jedním způsobem. Když v zobrazovacím stromu zvolíme u příslušného přístupového systému položku **<seznam osob>**, viz **obrázek 27**, objeví se seznam osob, které mají nějakou vazbu na alespoň jednu trasu v přístupovém systému. Tato vazba je označena názvem role a může nabývat dvou hodnot. Může se jednat o *Správce* nebo o *Pověřenou osobu*. Zde nás zajímá role *Správce*. Když klikneme na osobu, kterou chceme odebrat z trasy a má roli *Správce*, objeví se v panelu s detaily seznam tras, které tato osoba spravuje, viz **obrázek 28**. Vedle tohoto seznamu je tlačítko **Odebrat správcovství**, po jehož stisknutí a po následném potvrzení potvrzovacího dialogu odebereme osobě práva zvolenou trasu spravovat, tím pádem odebereme tuto trasu i ze seznamu tras, na nichž je označená osoba správcem. Pokud osobu odebere ze všech tras, které spravuje, zmizí i ze seznamu osob.



Obrázek 27: Přístupový systém – <seznam osob>



Obrázek 28: Přístupový systém – <seznam osob> – seznam spravovaných tras

9.1.6 Vytvoření, úprava a odstranění čtečky

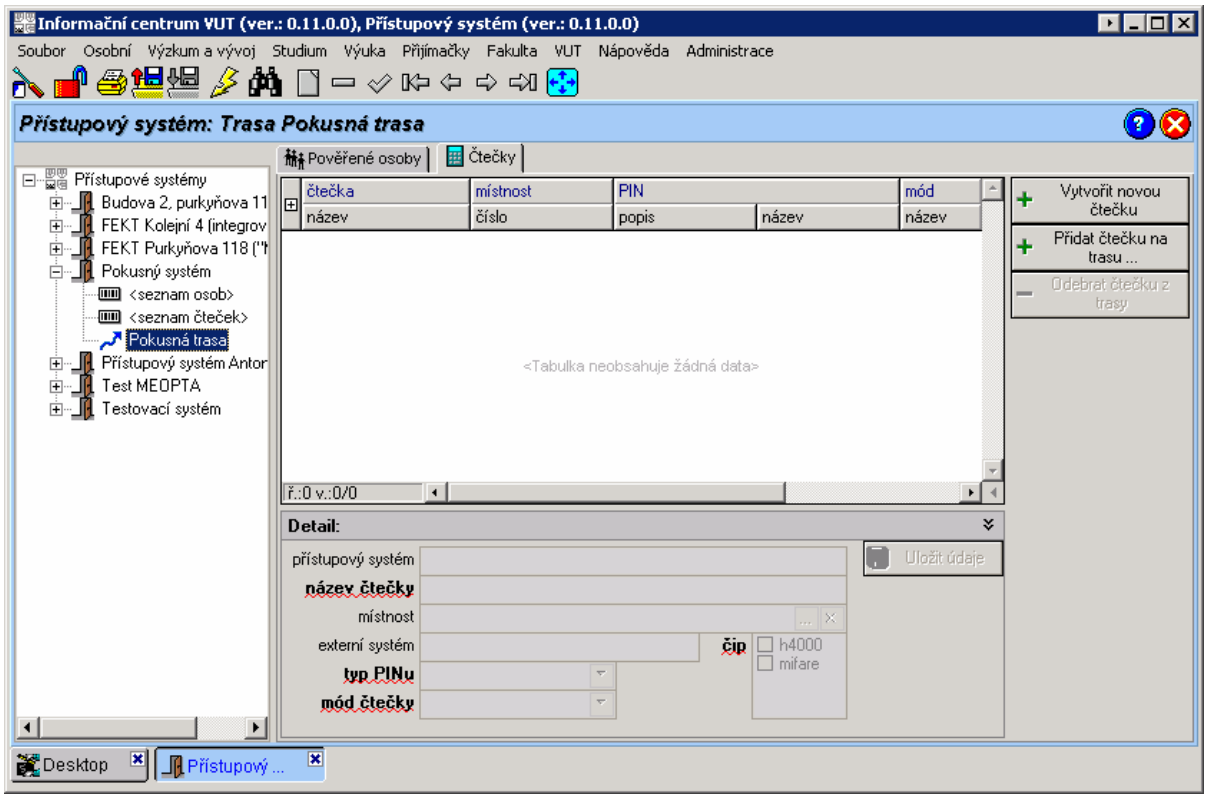
Existují dvě možnosti, jak provést vytvoření nebo úpravu čtečky.

První možnost je vybrání existující trasy v zobrazovacím stromu a kliknutím na záložku **Čtečky**, viz **obrázek 29**. Tímto způsobem se vytvořená čtečka přiřadí rovnou na trasu, kterou máme zvolenou v zobrazovacím stromu.

Druhou možností je kliknutí na položku **<seznam čteček>** v zobrazovacím stromu, viz **obrázek 30**. Zde vytvořená čtečka není přiřazena k žádné trase.

V obou případech při vytváření čtečky postupujeme stejným způsobem a to pomocí tlačítka **Vytvořit novou čtečku**. Po jeho stisknutí se přesune kurzor do panelu s detaily na položku **název čtečky** a všechny ostatní editovatelné položky se nastaví na „defaultní“ hodnoty. Když vyplníme povinná pole, zpřístupní se nám tlačítko **Uložit údaje**, po jehož stisknutí uložíme vytvořenou čtečku.

Editace čtečky taky probíhá v obou případech stejně. V seznamu čteček vybereme požadovanou čtečku a údaje o ní se zobrazí rovnou v panelu s detaily, případně v panelu s detaily v záložce **Detail**. Po dokončení úprav údajů je uložíme stisknutím tlačítka **Uložit údaje**, které je přístupné, pokud jsou vyplněna všechna povinná pole.

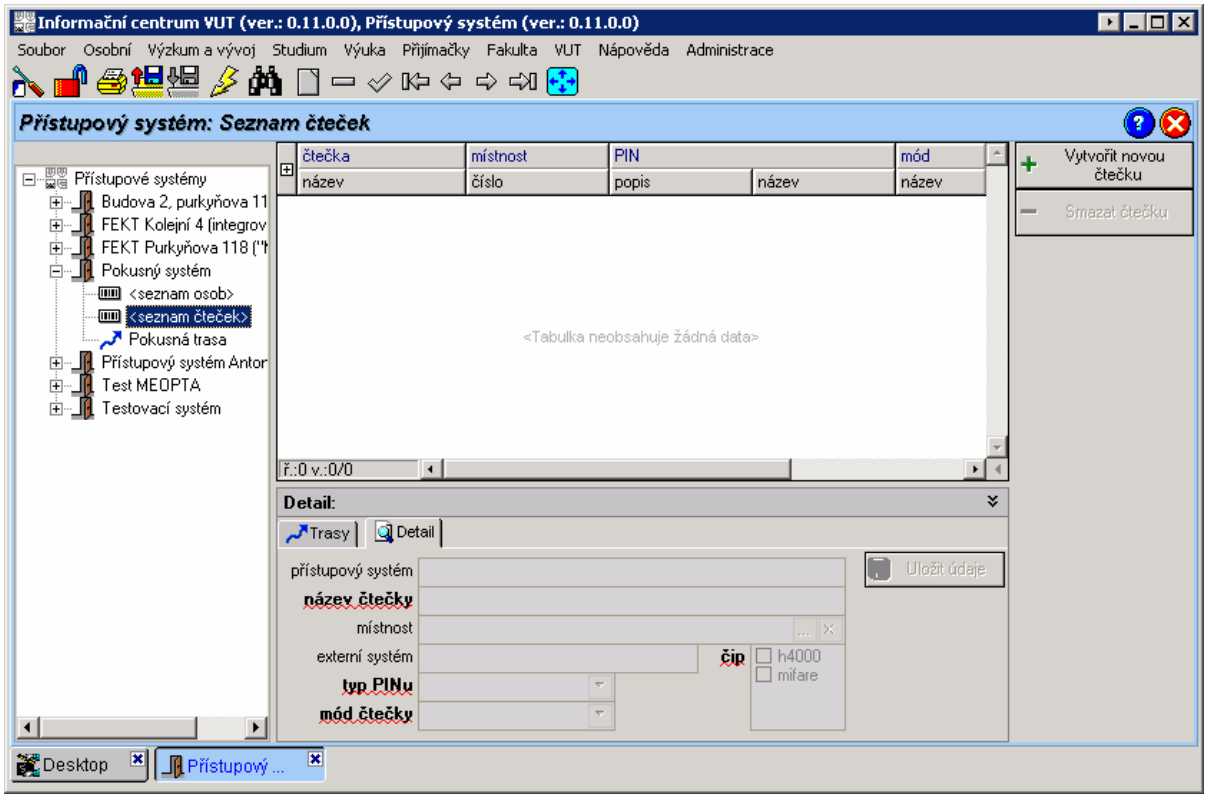


Obrázek 29: Trasa – vytváření, editace a odstranění čtečky

Ještě je potřeba zmínit položku **místnost**, jež nastavuje umístění čtečky, jelikož ta obsahuje mimo jiné i dvě tlačítka.

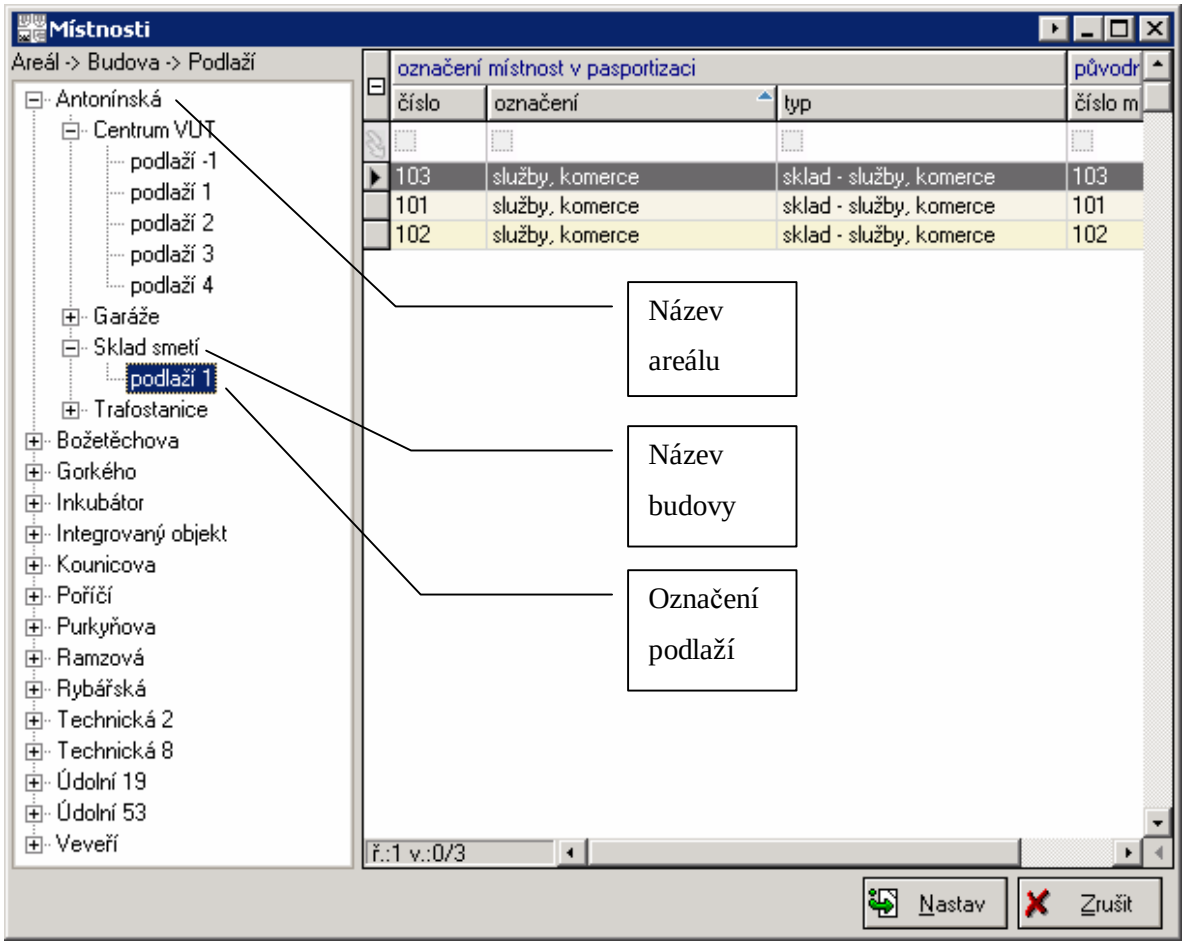
Tlačítkem ... spustíme dialog pro výběr místnosti, viz **obrázek 31**. V tomto dialogu se nachází stromový filtr, ve kterém jsou hierarchicky zobrazeny areály, budovy a podlaží, a dále je zde seznam místností. Když klikneme na požadované podlaží, naplní se seznam místností údaji o všech místnostech ve vybraném patře příslušné budovy a areálu. Po vybrání místnosti ze seznamu a po stisknutí tlačítka **Nastav**, se zavře dialog *Místnosti* a do textového pole u položky místnost se nastaví umístění a jméno místnosti. Tlačítkem **Zrušit** zavřeme dialog Místnosti, ale neprovedou se žádné úpravy.

Tlačítkem ✕ vymažeme obsah textového pole místnost.



Obrázek 30: Přístupový systém – <seznam čteček> – vytváření, editace a odstranění čtečky

Smazat čtečku můžeme jen na obrazovce, která se objeví po vybrání položky **<seznam čteček>** ze zobrazovacího stromu, viz **obrázek 30**. Čtečku smažeme tak, že ji vybereme v seznamu a klikneme na tlačítko **Smazat čtečku**. Následně se objeví dialog, který po nás požaduje potvrzení našeho rozhodnutí. Po potvrzení dojde k vymazání čtečky. Pokud čtečka náleží pod nějakou trasu, případně trasy, je zároveň odstraněna i z této trasy, respektive těchto tras.



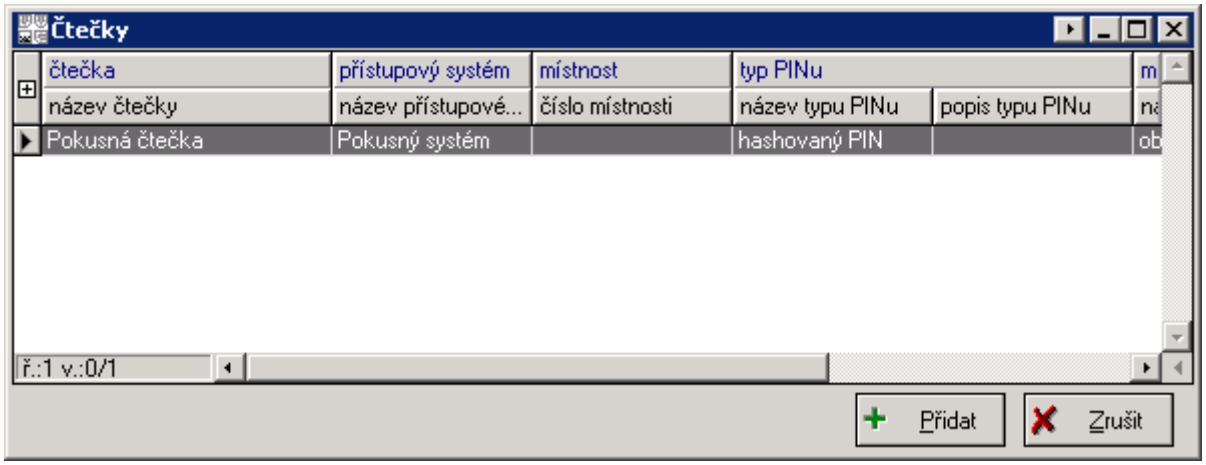
Obrázek 31: Dialog Místnosti

Přidání čtečky na trasu, odebrání čtečky z trasy

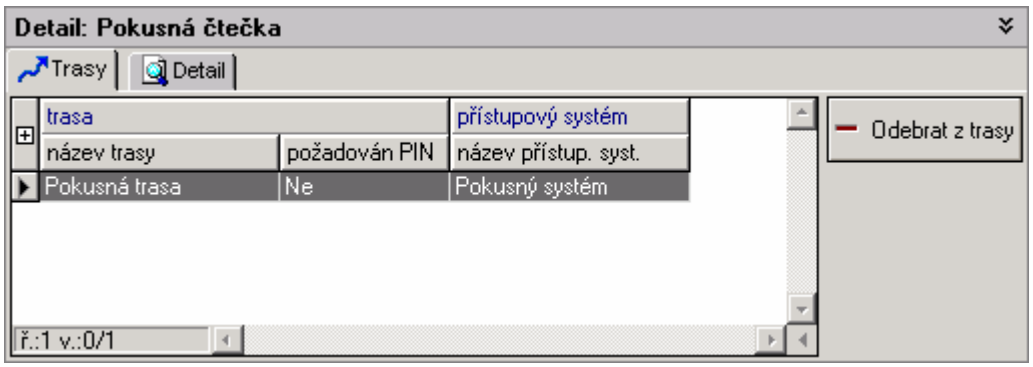
Když chceme přidat čtečku na určitou trasu, tak musíme nejprve v zobrazovacím stromu tuto trasu vybrat. Následně klikneme na záložku **Čtečky** a stiskneme tlačítko **Přidat čtečku na trasu...**. Objeví se dialogové okno, viz **obrázek 32**, které obsahuje seznam čteček a dvě tlačítka. Čtečky vybrané v seznamu přiřadíme na trasu stiskem tlačítka **Přidat**. Zároveň se zavře dialogové okno. Tlačítkem **Zrušit** taktéž zavřeme dialogové okno, ale nepřidají se čtečky na trasu.

Čtečku z trasy odebereme tlačítkem **Odebrat čtečku z trasy**. Čtečka, kterou jsme si označili v seznamu, se po stisknutí tohoto tlačítka a po potvrzení našeho rozhodnutí odebere z aktuální trasy. Můžeme označit více čteček najednou, čímž odebereme všechny označené čtečky.

Ještě existuje jeden způsob odebrání čtečky z trasy. Když vybereme v zobrazovacím stromu položku **<seznam čteček>** a na zobrazené obrazovce záložku **Trasy**, která se nachází v panelu s detaily, tak se zobrazí seznam tras, v nichž je čtečka přiřazena, viz **obrázek 33**. Vedle seznamu je tlačítko **Odebrat z trasy**, po jehož stisknutí a po následném potvrzení potvrzovacího dialogu, odebereme čtečku z označené trasy.



Obrázek 32: Dialog čtečky



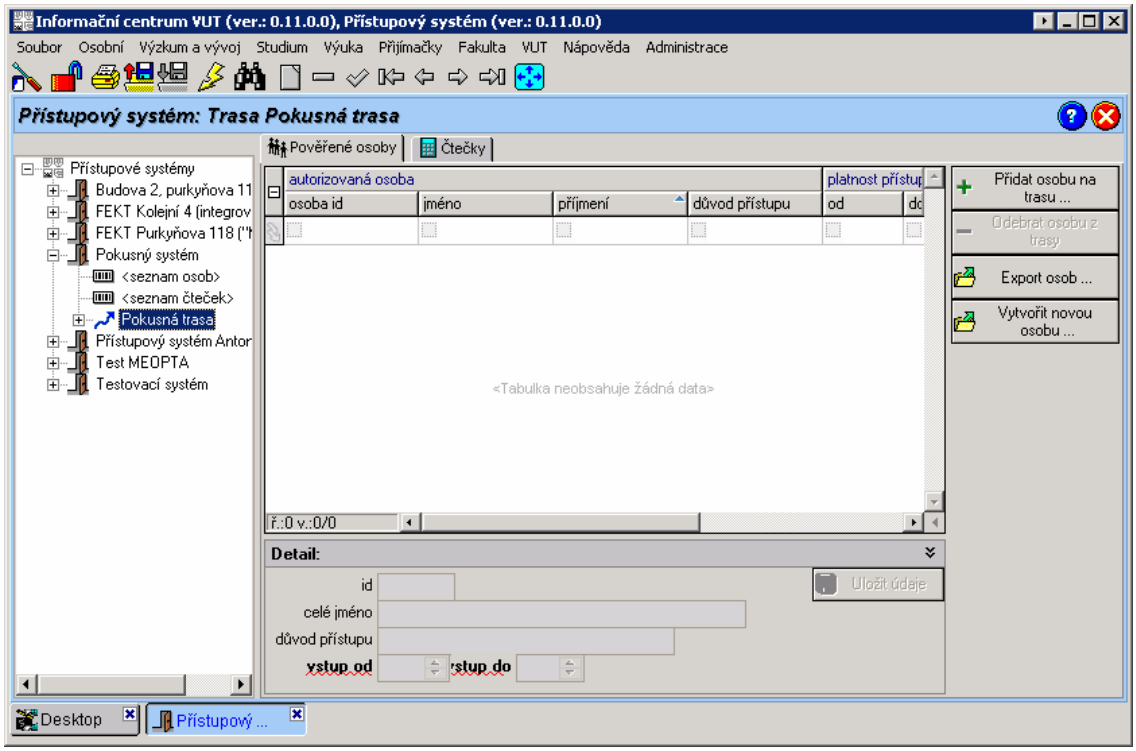
Obrázek 33: Přístupový systém – <seznam čteček> – seznam trasy čtečky

9.1.7 Přidání osoby na trasu, odebrání osoby z trasy

Přidat osobu na trasu můžeme zvolením požadované trasy ze zobrazovacího stromu a následným zvolením záložky **Pověřené osoby**, viz **obrázek 34**. Zde je tlačítko **Přidat osobu na trasu...**, které nám otevře dialog pro výběr osoby, jenž dostane právo pro přístup na trasu, viz **obrázek 35**.

Tento dialog má čtyři záložky, každá záložka zobrazuje seznam, který svými informace odpovídá názvu záložky. Záložka **Všechny osoby** zobrazuje seznam všech osob, jak zaměstnanců, tak studentů. Záložka **Studenti podle studia** zobrazuje seznam studentů v aktuální školním roce a v něm informace o jejich ročníku, a studovaném oboru a programu. Záložka **Studenti zapsaní v předmětu** obsahuje podobný seznam jako předchozí záložka, navíc však k informacím o ročníku, oboru a předmětu přidává informace o zapsaných předmětech studenta. A nakonec záložka **Studenti registrovaní ve vyučování** rozšiřuje předchozí záložku o informace o typu vyučování, o vyučovacích hodinách a o místnosti, ve které výuka probíhá. Údaje v seznamech můžeme filtrovat pomocí filtrů v horní části okna. Pro záložku **Všechny osoby** je povolený pouze filtr **fakulta**, kterým omezujeme zobrazení osob na jednu fakultu nebo součást VUT. Pro záložku **Studenti podle studia** jsou povolené filtry **fakulta** a **aktivní studium**. Filtrem **aktivní studium** určujeme, zda chceme vyhledat studenty, kteří mají aktivní studium. Pro poslední dvě záložky jsou povolené filtry **fakulta** a **akademický rok**. Filtrem **akademický rok** vybíráme studenty se zadaného akademického roku. V horní části okna je

ještě tlačítko **Obnovit**, které nám po zmáčknutí načte údaje do seznamu na aktuální záložce podle omezení určených filtry. Zatrhávacím tlačítkem **automaticky** určíme, zda požadujeme, aby se údaje v seznamu na aktuální záložce obnovovali automaticky při změně jakéhokoliv filtru. Pokud je zatrženo, údaje se obnovují automaticky. Pokud není, tak se při neaktuálnosti dat rozbliká tlačítko **Obnovit**. Ve spodní části dialogu se nacházejí položky **důvod přístupu**, **vstup od** a **vstup do**. **Důvod přístupu** je textové pole, do něhož můžeme slovně zadat, z jakého důvodu dostane osoba povolení k přístupu na trasu. Údaji **vstup od** a **vstup do** určujeme časovou platnost tohoto přístupu. Hodnota v poli **vstup od** musí být menší nebo rovno poli **vstup do**. Časová platnost přístupu na trasu se zatím nekontroluje, ale počítá se s ní v rámci rozšíření modulu. Proto zatím tyto pole nejsou povinná, i když jsou zvýrazněná tučně. Osobu, případně osoby, přidáme na trasu tlačítkem **Přidat**, jehož stisknutím přidáme označené osoby ze seznamu na aktuální záložce na trasu. Tlačítkem **Zrušit** zavřeme dialogové okno bez přidávání osob.

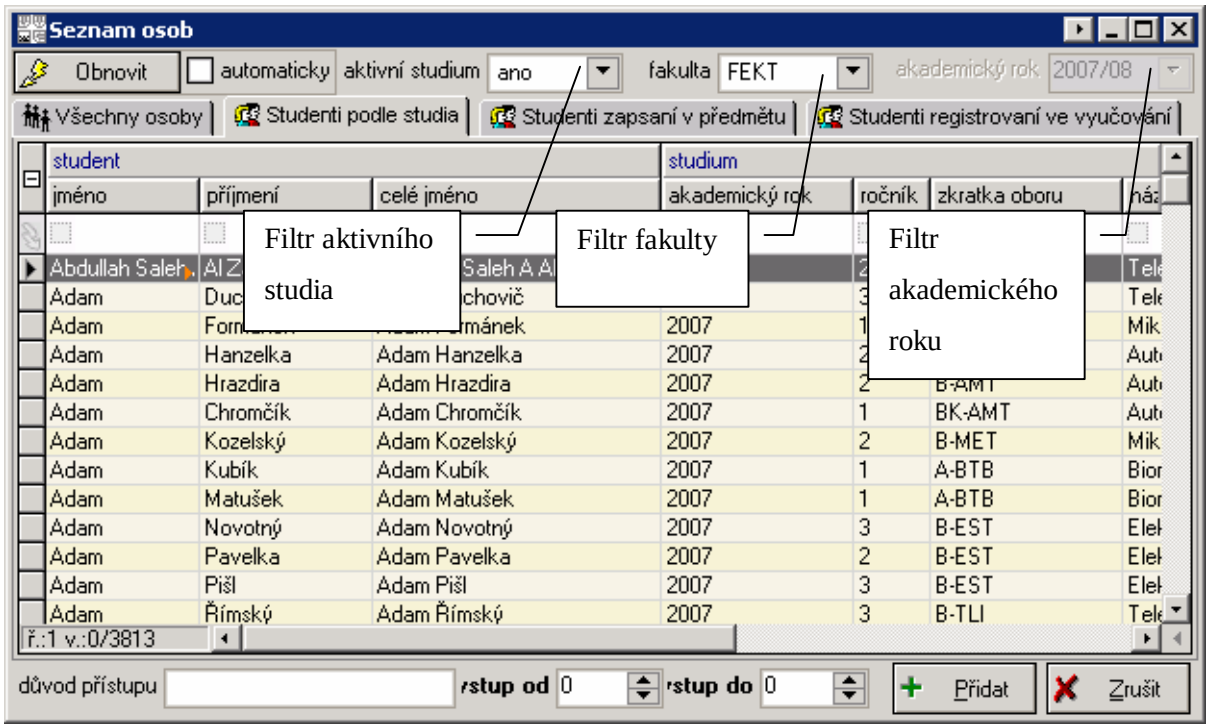


Obrázek 34: Přístupový systém – Trasa – Pověřené osoby

Odebrání osoby z trasy provedeme tlačítkem **Odebrat osobu z trasy**. Toto tlačítko je přístupné, pokud je označena nějaká autorizovaná osoba. Když stiskneme toto tlačítko a potvrdíme potvrzovací dialog, odebereme zvolenou osobu z trasy. Můžeme označit a odebrat víc osob najednou.

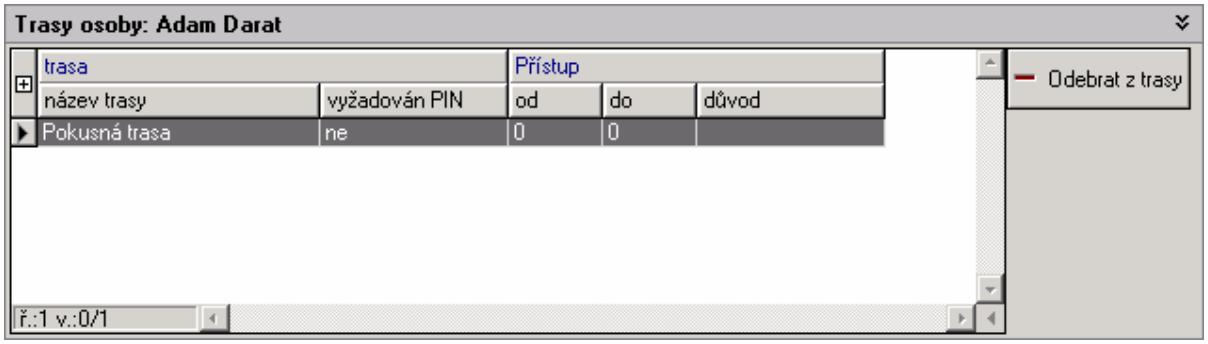
Osoby můžeme odebrat z trasy ještě jedním způsobem. Když v zobrazovacím stromu zvolíme u příslušného přístupového systému položku **<seznam osob>**, viz **obrázek 27**, objeví se nám seznam osob, které mají nějakou vazbu na alespoň jednu trasu v přístupovém systému. Tato vazba je označena názvem role a může nabývat dvou hodnot. Může se jedna o *Správce* nebo o *Pověřenou osobu*. Zde nás zajímá role *Pověřená osoba*. Když klikneme na osobu, kterou chceme odebrat z trasy a má roli *Pověřená osoba*, objeví se v panelu s detaily seznam tras, na které má tato osoba přístup,

viz **obrázek 36**. Vedle tohoto seznamu je tlačítko **Odebrat z trasy**, jehož stisknutím a následným potvrzení potvrzovacího dialogu odebereme osobu ze zvolené trasy, tím pádem odebereme tuto trasu i ze seznamu tras, na které má označená osoba přístup. Pokud osobu odebereme ze všech tras, do kterých má přístup, zmizí i ze seznamu osob.



Obrázek 35: Dialog Seznam osob

Tlačítko **Vytvořit novou osobu**, které lze najít v panelu funkčních tlačítek na **obrázku 34**, zatím není funkční, ale po jeho stisku by se měl otevřít modul, v němž budeme moct vytvořit novou osobu.



Obrázek 36: Přístupový systém – <seznam osob> – trasy pověřené osoby

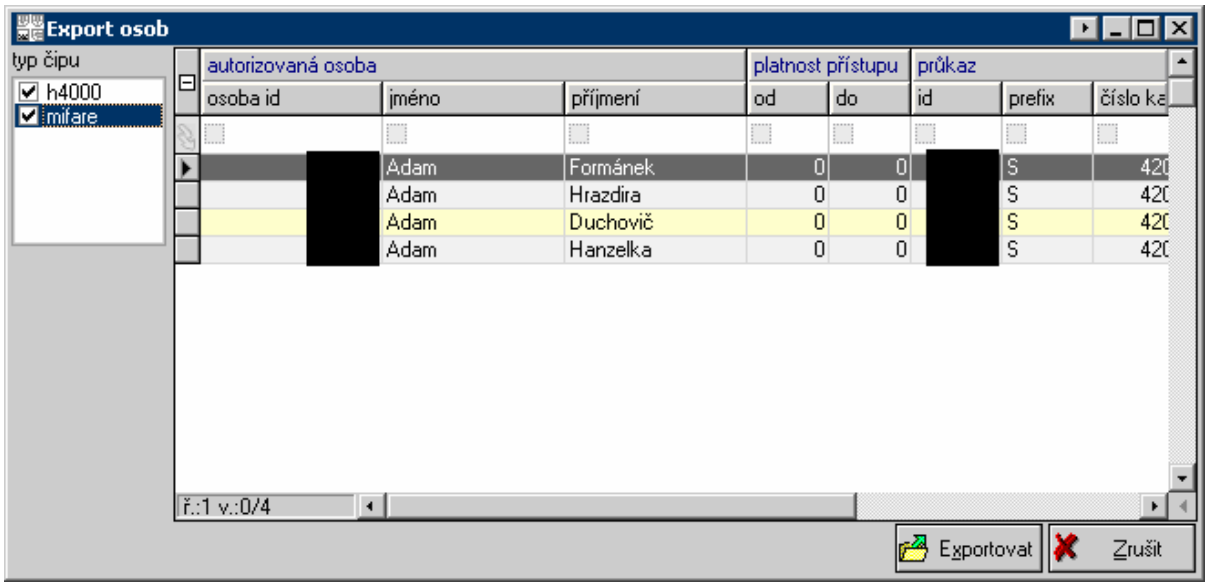
9.1.8 Export pověřených osob do souboru

Modul *Přístupový systém* dokáže do souboru exportovat údaje o osobách a jejich identifikačních kartách ze zvolené trasy. V zobrazovacím stromu vybereme požadovanou trasu a následně zvolíme záložku **Pověřené osoby**. Klikneme na tlačítko **Export osob**. Objeví se dialogové okno, které

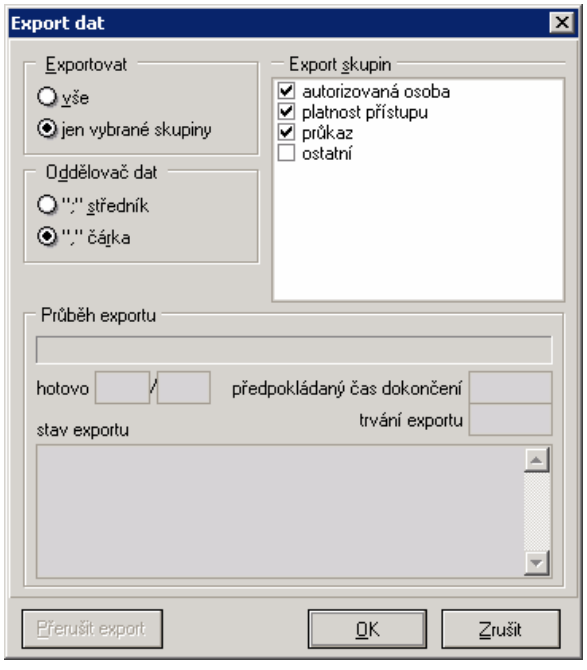
zobrazuje seznam osob a s údaji o jejich platných identifikačních kartách, viz **obrázek 37**. Na levé straně okna je filtr **typ čipu**, kterým můžeme vybrat jen ty karty, které používají vybraný typ čipu.

Exportují se údaje o osobách, které jsou označené v seznamu osob. Export spustíme tlačítkem **Exportovat**, po jehož stisku se otevře dialogové okno **Export dat**, viz **obrázek 38**. V tomto okně si můžeme nastavit další podrobnosti exportu, jako je třeba znak, který bude oddělovat záznamy, nebo které skupiny údajů požadujeme uložit do souboru. Když máme vše nastaveno, tak stiskem tlačítka **OK** se dostaneme do nabídky, kde si zvolíme název a umístění souboru na disku. Samotný export poté spustíme tlačítkem **Uložit**. Pokud se rozmyslíme a nechceme nic exportovat, tlačítkem **Zrušit** se vrátíme do okna **Export osob**.

Okno **Export osob** ukončíme tlačítkem **Zrušit**, nebo křížkem v záhlaví okna.



Obrázek 37: Export osob

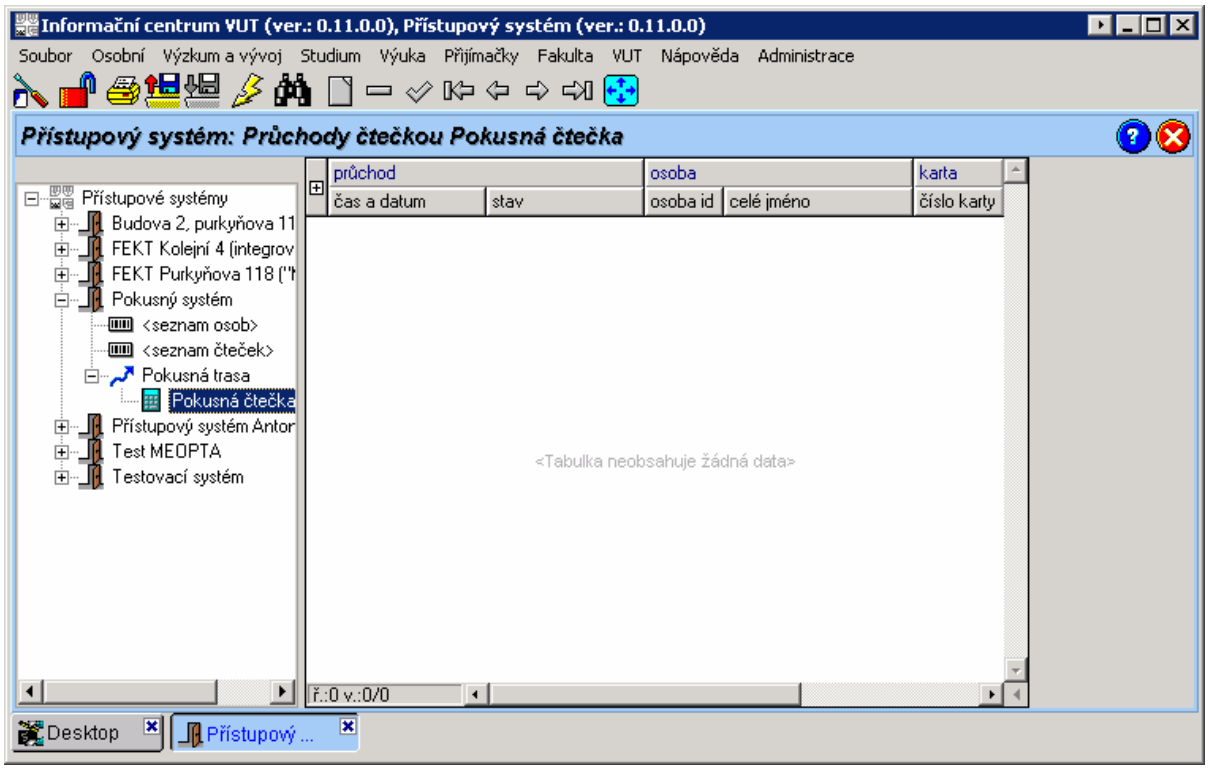


Obrázek 38: Export osob – Export dat

9.1.9 Prohlížení průchodů na čtečce

V zobrazovacím stromu vybereme čtečku, na které chceme sledovat průchody. Zobrazí se nám seznam, viz **obrázek 39**, ve němž budou informace o průchodu (datum, čas, stav), o osobě (identifikační číslo, celé jméno), která čtečku použila a o její identifikační kartě (číslo karty, číslo čipu, typ čipu).

Nejsou zde žádná tlačítka, protože průchody není dovoleno nijak upravovat.



Obrázek 39: Průchody čtečkou

9.2 Vybrané SQL dotazy

SQL dotaz pro vytvoření stromové struktury

```
select
  a.access_system_id as systemID,
  a.access_system_name as systemName,
  posledni.routeID as routeID,
  posledni.routeName as routeName,
  posledni.readerID as readerID,
  posledni.readerName as readerName
from
  brutisadm.access_system a,
  (
```

```
  select
    *
  from
    (select
       systemID, systemName,
       prvni.routeid, prvni.routename,
       druha.readerid, druha.readername
     from
       (select distinct
          r.route_id as routeID,
          r.route_name as routeName,
          ac.access_system_id as systemID,
          ac.access_system_name as systemName
        from
          brutisadm.access_system ac, brutisadm.route r
        where
        ac.access_system_id = r.access_system_id
        and r.status in (0,9)
        and ac.status in (0,9)
        ) prvni,
        (select
          r2.route_id as routeID,
          re2.reader_id as readerID,
          re2.reader_name as readerName
        from
          brutisadm.reader re2, brutisadm.reader_in_route
rir2,brutisadm.route r2
        where
        re2.reader_id = rir2.reader_id
        and rir2.route_id = r2.route_id
        and re2.status in (0,9)
        and r2.status in (0,9)
        and rir2.status in (0,9)
        ) druha
      where prvni.routeid = druha.routeid(+)
      )
  ) posledni
where  a.access_system_id = posledni.systemID(+)
and a.status in (0,9)
and apollo.ma_pravo(982235, :XXXUID, a.orgunitid) = 1
UNION
```

```
select ac.access_system_id, ac.access_system_name, r.route_id,
r.route_name, re.reader_id, re.reader_name
from
    brutisadm.access_system ac,
    brutisadm.route r,
    brutisadm.reader re,
    brutisadm.reader_in_route rir,
    brutisadm.route_master rm
where
     r.route_id = rm.route_id
     and rm.per_id = :XXXUID
     and ac.access_system_id = r.access_system_id
     and ac.status in (0,9)
     and r.status in (0,9)
     and rm.status in (0,9)
     and rir.status in (0,9)
     and re.status in (0,9)
     and re.reader_id = rir.reader_id
     and r.route_id = rir.route_id

order by systemName, routeName, readerName ASC
```

SQL dotaz pro naplnění filtru fakulta

```
select
  fakulta_id,
  zkratka,
  nazev
from
  st01.mv_soucasti_vut t
union
select -2, 'Nezařazení', 'Nezařazení'
from dual

order by nazev
```

9.3 CD s materiály

Na CD přiloženém k této diplomové práci naleznete zdrojové kódy modulu Přístupový systém, a tuto dokumentaci v elektronické formě.