VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

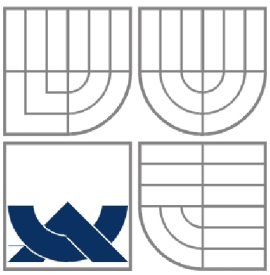
DATOVÁ INTEGRACE MEZI DATABÁZOVÝMI SYSTÉMY

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

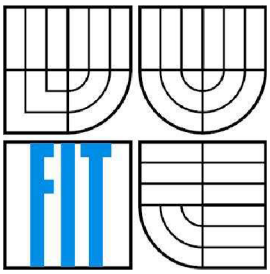
AUTOR PRÁCE
AUTHOR

BC. ZDENĚK PAPEŽ

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DATOVÁ INTEGRACE MEZI DATABÁZOVÝMI SYSTÉMY

DATA INTEGRATION BETWEEN DATABASE SYSTEMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. ZDENĚK PAPEŽ

VEDOUCÍ PRÁCE
SUPERVISOR

ING. JAROSLAV RÁB

BRNO 2010

Abstrakt

Tato diplomová práce pojednává o datové integraci, která je využívána pro přenos dat mezi různými databázovými systémy a to v obou směrech – migrace a replikace dat. Seznamuje nás s technologiemi distribuovaných databází. Konkrétně popisuje systém poskytovatelů lékařské péče a zkoumá jednotlivé entity zainteresované v jeho datové integraci. Pro realizaci projektu vytváří návrh na integraci tohoto registru a na jeho základě dále popisuje následnou implementaci.

Abstract

This master´s thesis deals with data integration that is used for data transfer within various database systems in both directions - data migration and replication. We become familiar with the technologies of distributed databases. In detail the system of health care providers is described and particular tables involved into its data integration are explored. For the project execution the proposal for integration of this system is created and whereupon following implementation is described.

Klíčová slova

Datová integrace, databáze, Oracle, migrace, replikace, advanced queuing, materializované pohledy.

Keywords

Data integration, database, Oracle, migration, replication, advanced queuing, materialized views.

Citace

Papež Zdeněk: Datová integrace mezi databázovými systémy, diplomová práce, Brno, FIT VUT v Brně, 2010

Datová integrace mezi databázovými systémy

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Jaroslava Rába.
Další informace mi poskytli zaměstnanci firmy PIKE Electronic s.r.o. a to především Ing. Jindřich Doležal.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Papež Zdeněk
25. 5. 2010

Obsah

1 Úvod

V dnešní době informačních technologií jsou téměř všechny podnikové i jiné systémy zpracovávány elektronicky v podobě informačních systémů. Některé tyto informační systémy jsou již v provozu několik let a začínají být zastaralé. Po mnohých úpravách se jednoho dne zastaví vývoj stávajícího systému a je zahájen vývoj nového moderního systému, s takovými rozšířeními, které zákazník požaduje. V takových případech přichází na řadu datová integrace. Je vyvinut nový systém, který však neobsahuje žádná data. Datová integrace musí vhodným způsobem vyřešit, jak platná data vyjmout z původního informačního systému a vložit je do aplikace nové. V některých případech je také zapotřebí nově pořízená data promítnout zpět do původního systému. Důvodem může například být dočasný paralelní běh obou systémů z důvodu napojení jiných aplikací na starý systém.

Tato diplomová práce se snaží zprostředkovat náhled do dané problematiky a zrealizovat datovou integraci pro systém registru poskytovatelů lékařské péče. Stávající systém je decentralizován na několik desítek databázových systémů. Systém je již nevyhovující, a proto byl vytvořen nový centrální systém registru. Datová integrace musí zajistit import stávajících dat do nového systému a zároveň umožnit zprostředkování nových dat, pořízených na centrálním systému, všem původním decentralizovaným aplikacím. Všechny databázové systémy jsou řešeny v prostředí Oracle, a proto je návrh datové integrace vytvořen s pomocí prostředků Oracle.

První kapitola popisuje prostředky společnosti Oracle, které mohou sloužit pro propojení vzdálených systémů a tím pomoci k realizaci datové integrace. V textu jsou mimo slovního popisu též uvedeny definice systémových prostředků s příklady.

Druhou kapitolou je detailní analýza obou integrovaných databázových systémů. Především jsou zkoumány jednotlivé databázové entity, které jsou zapojeny do datové integrace. U každé entity je uveden její význam, seznam atributů, seznam integritních omezení a seznam indexů. Informace o atributech jsou klíčové k mapování jednotlivých položek z jedné databáze do druhé. Integritní omezení a indexy pak slouží k zachování původních vazeb mezi stávajícími entitami. Pro korektní datovou integraci je zapotřebí dokonale pochopit logické vazby v obou systémech a k tomu by měla právě přispět tato studie.

Další část práce je zaměřena na technické provedení datové integrace. Jedná se o návrh obousměrného toku dat mezi databázovými systémy. Nahrání – migrace – stávajících dat do nového centralizovaného systému je jednorázovou akcí, kterou začíná zprovoznění tohoto systému. Replikace je pak proces přenosu všech změn z centrálního do původního systému. Tento proces musí fungovat automaticky a bezchybně do té doby, dokud stávající decentralizovaný systém bude muset fungovat.

Poslední blok této práce představuje detailní popis provedené datové integrace nad registrem poskytovatelů péče. Nezbytnou součástí musí být i provedení integračních testů ověřujících bezchybnou funkčnost replikačního aparátu a identifikujících úzká místa celého systému. Závěr práce je samozřejmě věnován zhodnocení výsledků a predikci dalšího vývoje.

2 Prostředky pro distribuované aplikace

Komerční databázové systémy v dnešní době obsahují databázové objekty, procedury a funkce, které umožňují vytvářet distribuované prostředí a komunikovat mezi databázemi, jež jsou fyzicky propojeny počítačovou sítí. Tato kapitola se bude zabývat popisem těchto prostředků. S ohledem na vývoj v prostředí Oracle, budou uvažovány nástroje implementovány v Oracle a uváděna syntaxe tohoto systému.

2.1 Databázový link

Databázový link je klíčovým stavebním prvkem k propojení více databází v jeden celek. Je to ukazatel v lokální databázi, který umožní přistupovat k objektům vzdáleného databázového systému. Při otevření databázového linku vzniká nové sezení na vzdálené databázi. Jeden databázový link vždy slouží k propojení 1:1 databází. Pokud ale je cílová databáze umístěna v Oracle Real Application Cluster (RAC) může link směřovat na více databázových instancí a tím zajistit vyšší propustnost mezi systémy. Oracle poskytuje tři typy databázových linků: Private (soukromý), Public (veřejný) a Global (globální). Nejvíce využíván je soukromý link, který bude dále rozveden [1].

Vytvoření databázového linku

Pro vytvoření linku je zapotřebí znát adresu cílového serveru, port a System ID (SID). Dále je nutné mít přístup do vzdálené databáze. Nejlepší praxí je vytvoření zvláštního uživatele v cílové databázi, který bude využíván pouze pro účely přístupu přes databázový link. Tento uživatel musí mít právo pro připojení (Connect) a dále přidělena práva pro jednotlivé objekty, podle účelu práce nad nimi. Tím je zajištěna jistá míra zabezpečení.

```
CREATE DATABASE LINK dblink_jmeno
CONNECT TO uzivatel IDENTIFIED BY heslo
USING '(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(COMMUNITY=TCP.WORLD)
(PROTOCOL=TCP)(HOST=adresa)(PORT=port))) (CONNECT_DATA=(SID=sid)))';
```

Použití databázového linku

Databázový link se používá při běžných SQL dotazech. Přístup ke vzdáleným objektům je pak reprezentován pomocí zavináče („@“) a názvu databázového linku za jménem objektu.

```
SELECT * FROM schema.tab_jmeno@dblink_jmeno;
```

2.2 Volání vzdálených procedur

Oracle poskytuje vlastní jazyk pro programování uložených procedur, které se mohou sdružovat do balíčků, PL/SQL - Procedural Language/Structured Query Language. Stejně jako v jiných distribuovaných systémech je i v Oracle umožněno vzdálené volní procedur – RPC (*Remote procedure call*). Zvláštností v PL/SQL je způsob předání parametrů. Nedochází k předání hodnotou ani odkazem, ale je využito předání hodnotou s kopií při návratu (call-by-copy/restore). To znamená, že proměnné použité jako parametr se při volání procedury zkopírují do cílového systému a pokud jsou tyto parametry zároveň výstupní, dojde ke zpětnému zkopírování hodnoty do lokální proměnné.

Veškeré volání vzdálených procedur probíhá s dvoufázovým potvrzením (two-phase commit). Tento mechanismus garantuje, že všechny provedené změny v transakcích ve všech databázích se buď provedou (commit), nebo se neprovedou (rollback). Jeden databázový systém se stává koordinátorem, který řídí distribuované transakce. Před potvrzením transakce se koordinátor zeptá všech vzdálených databází, zda jsou připraveny provést commit. Pokud alespoň jedna odpoví záporně, je koordinátorova transakce zrušena a tím odvolány i ostatní distribuované transakce. Pokud se vrátí od všech kladná odpověď, změny v systémech jsou zaneseny do databází [2].

Vzdálená transakce je tedy závislá na lokální transakci, která je zodpovědná za její vyvolání. Pokud je odvolána lokální transakce, jsou zrušeny i změny ve vzdáleném systému. Vzdálená procedura však může obsahovat příkazy commit, rollback či savepoint, ale jejich provedení nebude mít žádný vliv na lokální systém a probíhá jako autonomní transakce. K implicitnímu potvrzení ve vzdálené transakci dochází při uzavření databázového linku [3].

Použití vzdálené procedury

Vzdálená procedura se volá přes databázový link obdobným způsobem, jak se přistupuje ke vzdáleným objektům. Uživatel figurující v definici linku však musí mít práva pro spuštění (execute) konkrétní procedury nebo balíčku.

```
BEGIN
  Balik_jmeno.procedura_jmeno@dblink_jmeno(parametry_seznam);
END;
/
```

2.3 Oracle Advanced Queuing

Advanced Queuing (AQ) tvoří asynchronní komunikační aparát mezi distribuovanými databázovými systémy, čímž umožňuje jejich integraci. Je založen na předávání zpráv, které jsou uchovávány ve frontách a pomocí front lze tyto zprávy přijímat a propagovat do okolních systémů. Zprávy většinou přenáší byznys data ve formátu XML a tím se stává Advanced Queuing součástí nejen databázových systémů, ale také SOA aplikací na webových serverech. Za zmínku stojí Oracle BPEL (Business Process ExucutionLanguage) nebo ESB (Enterprise Service Bus) [4].

Zpráva

Zpráva je nejmenším nositel informace, který lze vložit a opět vyjmout z fronty. Zpráva se skládá ze dvou částí: uživatelská data (payload) a kontrolní informace (metadata). Kontrolní informace jsou využívány k řízení zpracování zpráv AQ. V uživatelských datech jsou uloženy věcné informace, které se předávají mezi systémy. Nejčasněji mají strukturu definovanou uživatelsky definovaným typem (UDT), který obsahuje položku CLOB (Character Large Object), ve kterém je uložen XML dokument a další informace, jejichž obsah je záměrně vytknut z XML a může sloužit pro zpracování na úrovni aplikací.

Zpráva může vždy existovat pouze v jedné frontě a vzniká voláním procedury enqueue ze systémového balíčku DBMS_AQ. Při vzniku zprávy se může nastavit její priorita, zpoždění, doba existence, odesilatel, příjemci a další, které jsou obsaženy v databázovém typu DBMS_AQ. MESSAGE_PROPERTIES_T [5].

Tabulka 2.1: Struktura MESSAGE_PROPERTIES_T

Název	Typ	Význam	Implicitní hodnota
priority	BINARY_INTEGER	Priorita, 1 – max	1
delay	BINARY_INTEGER	Zpoždění [s]	NO_delay
expiration	BINARY_INTEGER	Doba, po kterou zpráva čeká na vyzvednutí [s]	NEVER
correlation	VARCHAR2(128)	Dodatečný ID vložený uživatelem	NULL
attempts	BINARY_INTEGER	Počet pokusů pro vyzvednutí zprávy	Nenastavuje se
recipient_list	AQ$_RECIPIENT_ LIST_T	Seznam příjemců	Subscriber fronty
exception_queue	VARCHAR2(51)	Jméno exception fronty	NULL
enqueue_time	DATE	Datum a čas vložení do fronty	SYSDATE
state	BINARY_INTEGER	Stav, ve kterém se zpráva nachází	0 (1 při nastavenem delay)
sender_id	AQ$_AGENT	ID odesilatele	NULL
original_msgid	RAW(16)	Využíváno pro propagaci	NULL

Po celou dobu života se zpráva nachází v určitém stavu, který může nabývat hodnot:
0 – připravena ke zpracování
1 – čeká, je nastaveno zpoždění, které ještě nevypršelo
2 – zpracována, podržena ve frontě
3 – nastala chyba, zpráva vložena do exception fronty.

Fronta a frontová tabulka

Fronta slouží jako úložiště zpráv. Existují dva druhy front. Uživatelské, označované jako normální, a exception fronty. S uživatelskými frontami pracujeme pro předávání a přijímání zpráv. Do exception fronty se ukládají zprávy, které z nějakého důvodu nemohly být zpracovány. Uživatelské fronty se dále dělí na perzistentní a dočasné. Dočasné fronty uchovávají zprávy pouze v paměti a obecně se využívají k zaslání zpráv právě připojeným uživatelům. Perzistentní fronty uchovávají zprávy ve frontových tabulkách, nad kterými lze provádět běžné dotazy stejně jako nad obyčejnou tabulkou.

Frontové tabulky mohou být opět dvojího typu: pro jednoho konzumenta nebo pro více konzumentů. Vytváření, úpravy a další operace nad frontami se provádí pomocí PL/SQL procedur, které jsou obsaženy v balíčku DBMS_AQADM. Detaily front a frontových tabulek lze prohlížet v pohledech DBA_QUEUE a DBA_QUEUE_TABLES [4].

```
-- vytvoření frontové tabulky
DBMS_AQADM.CREATE_QUEUE_TABLE (
   queue_table          IN      VARCHAR2,
   queue_payload_type   IN      VARCHAR2,
   [storage_clause      IN      VARCHAR2        DEFAULT NULL,]
   sort_list            IN      VARCHAR2        DEFAULT NULL,
   multiple_consumers   IN      BOOLEAN         DEFAULT FALSE,
   message_grouping     IN      BINARY_INTEGER  DEFAULT NONE,
   comment              IN      VARCHAR2        DEFAULT NULL,
   auto_commit          IN      BOOLEAN         DEFAULT TRUE,
   primary_instance     IN      BINARY_INTEGER  DEFAULT 0,
   secondary_instance   IN      BINARY_INTEGER  DEFAULT 0,
   compatible           IN      VARCHAR2        DEFAULT NULL,
   secure               IN      BOOLEAN         DEFAULT FALSE);

-- vytvoření fronty
DBMS_AQADM.CREATE_QUEUE (
   queue_name          IN       VARCHAR2,
   queue_table         IN       VARCHAR2,
   queue_type          IN       BINARY_INTEGER DEFAULT NORMAL_QUEUE,
   max_retries         IN       NUMBER         DEFAULT NULL,
   retry_delay         IN       NUMBER         DEFAULT 0,
   retention_time      IN       NUMBER         DEFAULT 0,
   dependency_tracking IN       BOOLEAN        DEFAULT FALSE,
   comment             IN       VARCHAR2       DEFAULT NULL,
   auto_commit         IN       BOOLEAN        DEFAULT TRUE);
```

Enqueue zpráv

Enqueue je systémová procedura, která zajišťuje vytváření a následné vkládání zpráv do fronty. Vstupními parametry procedury jsou název fronty, do které se zpráva vloží, nastavení procesu enqueue ve struktuře DBMS_AQ.ENQUEUE_OPTIONS_T, vlastnosti zprávy ve struktuře DBMS_AQ.MESSAGE_PROPERTIES_T a uživatelská data určená pro přenos. Výstupním parametrem procedury enqueue je jednoznačný identifikátor zprávy MSGID [5].

Tabulka 2.2: Struktura ENQUEUE_OPTIONS_T

Název	Typ	Význam	Implicitní hodnota
visibility	BINARY_INTEGER	Způsob zpracování z pohledu transakce.	ON_COMMIT
relative_msgid	RAW(16)	Identifikátor předchozí zprávy. Je využíváno pouze při sekvenčním zpracování	NULL
sequence_ deviation	BINARY_INTEGER	Nastavení sekvence zpráv.	NULL
transformation	VARCHAR2(60)	Transformace provedená nad uživatelskými daty	NULL

```
-- deklarace procedury ENQUEUE
DBMS_AQ.ENQUEUE (
   queue_name           IN      VARCHAR2,
   enqueue_options      IN      enqueue_options_t,
   message_properties   IN      message_properties_t,
   payload              IN      "<ADT_1>",
   msgid                OUT     RAW);
```

Dequeue zpráv

Procedura dequeue zajišťuje konzumaci zpráv z fronty. Stejně jako u procedury enqueue jsou vstupními parametry název fronty a nastaveni procesu dequeue ve struktuře DBMS_AQ. DEQUEUE_OPTIONS_T. Na výstupu procedury jsou vlastnosti vyzvednuté zprávy, uživatelská data a identifikátor zprávy [5].

Tabulka 2.3: Struktura DEQUEUE_OPTIONS_T

Název	Typ	Význam	Implicitní hodnota
consumer_name	VARCHAR2(30)	Jméno konzumenta. Pouze v případu vícekonzumní fronty	NULL
dequeue_mode	BINARY_INTEGER	Způsob zpracování zpráv	REMOVE
navigation	BINARY_INTEGER	Způsob výběru pořadí zpracovávaných zpráv	NEXT_MESSAGE
visibility	BINARY_INTEGER	Způsob zpracování z pohledu transakce.	ON_COMMIT
wait	BINARY_INTEGER	Doba čekání na další zprávu	FOREVER
msgid	RAW(16)	ID zprávy, která bude konzumována	NULL
correlation	VARCHAR2(128)	ID korelace zpráv, které budou vyzvednuty	NULL
deq_condition	VARCHAR2(4000)	Podmínka složená z vlastností zprávy	NULL
transformation	VARCHAR2(60)	Transformace provedená nad uživatelskými daty	NULL

```
-- deklarace procedury DEQUEUE
DBMS_AQ.DEQUEUE (
   queue_name           IN      VARCHAR2,
   dequeue_options      IN      dequeue_options_t,
   message_properties   OUT     message_properties_t,
   payload              OUT     "<ADT_1>"
   msgid                OUT     RAW);
```

PL/SQL Callback

Oracle Advanced Queuing umožňuje automatické hlídání obsahu fronty. Pokud se do fronty vloží nová zpráva, databázový systém automaticky vyvolá spuštění určité procedury, která je zaregistrována k příslušné frontě. Tato funkcionalita je označována jako callback.

Název procedury může být libovolný, ale musí obsahovat určité parametry, které jsou vyplňovány databázovým systémem. Deklarace procedury pak může vypadat takto:

```
-- deklarace procedury callback
procedure muj_callback(
  context  IN  RAW,
  reginfo  IN  SYS.AQ$_REG_INFO,
  descr    IN  SYS.AQ$_DESCRIPTOR,
  payload  IN  RAW,
  payloadl IN  NUMBER);
```

Procedura se převážně využívá k vyzvednutí zpráv z fronty a vložení do databázové tabulky. Pro tělo procedury je pak vhodné dodržovat jistá pravidla. Procedura by měla zpracovat všechny zprávy, které se ve frontě nachází. Pokud by se zpracovávala každá zpráva zvlášť, mohlo by docházet ke zbytečné režii. Dále je vhodné nastavit ve vlastnostech čekání na NO_WAIT při volání procedury dequeue, aby nedocházelo k zablokování zpracování.

Registrace procedury se provádí pomocí procedury DBMS_AQ.REGESTRACE, která má dva vstupní parametry. Seznam informací, které definují, která procedura callback bude registrována nad jako frontou ve struktuře typu AQ$_REG_INFO, a počet těchto informací [6].

Tabulka 2.4: Struktura AQ$_REG_INFO

Název	Typ	Význam	Implicitní hodnota
name	VARCHAR2(128)	Název fronty ve formátu schéma.fronta:konzument	-
namespace	NUMBER	Jmenný prostor, běžně DBMS_AQ. NAMESPACE_AQ	-
callback	VARCHAR2(4000)	Volaná procedura – plsql://schema.procedura	-
context	RAW(2000)	Kontext předána do procedury	NULL
qosflags	NUMBER	Specifikace QOS (quality of service)	NULL
timeout	NUMBER	Prodlení po registraci	0

```
-- deklarace procedury REGISTER
DBMS_AQ.REGISTER (
   reg_list     IN  SYS.AQ$_REG_INFO_LIST,
   count        IN  NUMBER);
```

Propagace zpráv

Základem datové integrace pomocí Advanced Queuing je propagace zpráv mezi frontami. Oracle poskytuje funkci přenášení zpráv z fronty do fronty, jenž může být i ve vzdálené databázi. Pomocí systémového procesu se automaticky provede dequeue ze zdrojové fronty a enqueue do cílové fronty. Přenos do vzdálené databáze pak probíhá přes HTTP(S) nebo Oracle Net Services.

Existují dva druhy AQ propagací. Queue-to-dblink nebo novější způsob queue-to-queue (Q2Q), která je dostupná od verze 10g R2. Propagace queue-to-queue je zároveň v dnešní době upřednostňována a to z několika důvodů. Q2Q má vlastní job a lze ji tak lépe spravovat. Navíc vyhodnocuje stav databázového linku až při samotné propagaci zpráv a ne při definici propagace. Tím je zabráněno zamrzávání propagací, které vede až k nutnosti propagaci zrušit a znovu vytvořit, což může být komplikované, jelikož při rušení propagace se automaticky smažou všechny zprávy uložené ve zdrojové frontě. Nakonec Q2Q umí využívat Oracle Failover v RAC databázích a tím zajišťovat vysokou dostupnost propagace. Z těchto důvodů bude dále propagací rozuměna propagace Q2Q, jejíž nastavení bude popsáno.

Aby mohla být fronta s frontovou tabulkou použita v propagaci, musí splňovat jistá nastavení. Frontová tabulka musí být vytvořena s kompatibilitou 10.0, která umožňuje propagovat v režimu Q2Q. Tabulka musí také být nastavena pro více konzumentů. To znamená, že jednotlivé zprávy jsou určeny pouze pro některé konzumenty. Aby mohla být zpráva přiřazena některému konzumentovy, musí existovat k dané frontě podpisovatel (subscriber), jenž definuje konzumenta, který bude provádět dequeue z cílové fronty. K přidání podpisovatele slouží systémová procedura DBMS_AQADM. ADD_SUBSCRIBER, která přijímá na vstupu název fronty a konzumenta ve struktuře AQ$_AGENT.

Tabulka 2.5: Struktura AQ$_AGENT

Název	Typ	Význam	Implicitní hodnota
name	VARCHAR2(30)	Jméno konzumenta	-
address	VARCHAR2(1024)	Cílová fronta – schema.fronta@dblink	-
protocol	NUMBER	Protokol pro interpretaci adresy a propagaci	0

```
-- deklarace procedury ADD_ SUBSCRIBER
DBMS_AQADM.ADD_SUBSCRIBER (
   queue_name      IN    VARCHAR2,
   subscriber      IN    sys.aq$_agent,
   rule            IN    VARCHAR2 DEFAULT NULL,
   transformation  IN    VARCHAR2 DEFAULT NULL
   queue_to_queue  IN    BOOLEAN DEFAULT FALSE,
   delivery_mode   IN    PLS_INTEGER DEFAULT DBMS_AQADM.PERSISTENT);
```

Samotná propagace se vytváří procedurou DBMS_AQADM.SCHEDULE_PROPAGATION. Pro korektní nastavení propagace je potřeba zadat zdrojovou a cílovou frontu, příslušný databázový link a parametry určující dobu a plánování průběhu propagace: čas startu, doba trvání, prodlení a čas mezi jednotlivými spouštěními.

```
-- deklarace procedury SCHEDULE_PROPAGATION
DBMS_AQADM.SCHEDULE_PROPAGATION (
   queue_name          IN    VARCHAR2,
   destination         IN    VARCHAR2 DEFAULT NULL,
   start_time          IN    DATE     DEFAULT SYSDATE,
   duration            IN    NUMBER   DEFAULT NULL,
   next_time           IN    VARCHAR2 DEFAULT NULL,
   latency             IN    NUMBER   DEFAULT 60,
   destination_queue   IN    VARCHAR2 DEFAULT NULL);
```

Sledování průběhu propagace lze dohledat v pohledu DBA_QUEUE_SCHEDULES, ve kterém jsou zaznamenány chyby, časy spouštění a statistiky odchozích zpráv [7].

2.4 Materializované pohledy

Materializované pohledy (MV - Materialized Views, ve starších verzích označovány Snapshots) jsou databázové objekty, které slučují funkce pohledů a tabulek. Obsah materializovaných pohledů je závislý na zdrojových datech master tabulky, ale zároveň vytváří svou vlastní perzistentní kopie dat do databáze, která se může obnovovat. Jelikož zdrojové tabulky mohou být umístěny ve vzdálených databázích, stává se z materializovaných pohledů jedinečný nástroj pro replikaci dat mezi databázovými systémy. Použití materializovaných pohledů přináší několik výhod. Tyto pohledy mohou vytvářet lokální kopie centrální databáze, což naznačuje uplatnění v regionálně rozsáhlých organizacích, kdy s použitím pohledů odpadá značná zátěž síťové komunikace. Materializované pohledy také nabízí off-line použití lokální databáze, například na přenosných počítačích. V definici pohledů se určuje, jaké sloupce ze zdrojové tabulky se budou přenášet a jaká data pohled bude obsahovat. Tato vlastnost může být využita k rozdělení dat do několika skupin podle obsahu či citlivosti. Materializované pohledy se dělí na několik druhů a to podle práce nad daty a zdroje dat.

Rozdělení podle práce s daty

Materializované pohledy primárně slouží pro přenos dat. Tyto data se mohou číst, modifikovat nebo se mohou zapisovat nové záznamy. Při vytváření pohledu se definuje způsob, jakým bude materializovaný pohled s daty pracovat. Implicitním nastavením je materializovaný pohled pro čtení (Read-only). Nad těmito pohledy nelze provádět DML operace.

```
-- vytvoření MV pro čtení
CREATE MATERIALIZED VIEW mv_jmeno AS
  SELECT * FROM schema.zdrojova_tab@dblink;
```

Materializované pohledy, které umožňují zápis dat, se označují jako aktualizovatelné (Updatable). Data zapsaná do těchto pohledů se mohou promítnout do zdrojových tabulek. To může značně snížit zátěž na zdrojové straně, jelikož uživatelé mohou plnohodnotně pracovat nad lokální databází. Takovýto materializovaný pohled se vytvoří s přidáním klauzule FOR UPDATE do DDL příkazu. Aby mohl tento mechanismus fungovat, musí se materializovaný pohled přidat do tzv. MV skupiny (materialized view group), která je vytvořena na straně zdrojové tabulky. MV skupina sdružuje všechny pohledy nad jednou tabulkou a definuje způsob přenosu dat. Práce se skupinami je realizována přes balíček DBMS_REPCAT.

```
-- procedura pro vytvoření MV skupiny
DBMS_REPCAT.CREATE_MVIEW_REPGROUP (
   gname              IN   VARCHAR2,
   master             IN   VARCHAR2,
   comment            IN   VARCHAR2     := '',
   propagation_mode   IN   VARCHAR2     := 'ASYNCHRONOUS',
   fname              IN   VARCHAR2     := NULL
   gowner             IN   VARCHAR2     := 'PUBLIC');
```

```
-- procedura pro přidání MV do MV skupiny
BMS_REPCAT.CREATE_MVIEW_REPOBJECT (
   sname                    IN   VARCHAR2,
   oname                    IN   VARCHAR2,
   type                     IN   VARCHAR2,
   ddl_text                 IN   VARCHAR2  := '',
   comment                  IN   VARCHAR2  := '',
   gname                    IN   VARCHAR2  := '',
   gen_objs_owner           IN   VARCHAR2  := '',
   min_communication        IN   BOOLEAN   := true,
   generate_80_compatible   IN   BOOLEAN   := true,
   gowner                   IN   VARCHAR2  := 'PUBLIC');
```

Posledním typem materializovaných pohledů jsou zapisovatelné pohledy (Writeable). Do těchto pohledů se může zapisovat, ale změny se neprojevují ve zdrojové tabulce. Po obnově dat dojde ke ztrátě provedených změn. Pohled se také vytváří s parametrem FOR UPDATE, ale pohled není přidán do MV skupiny.

Rozdělení podle zdrojových dat

Materializované pohledy mohou být závislé na několika typech zdrojů. Nejčastějším typem materializovaných pohledů jsou pohledy, které pracují s daty nad tabulkou s primárním klíčem (Primary Key Materialized Views). V definici materializovaného pohledu pak může figurovat výběr dat s podmínkou WHERE, která musí obsahovat pravidlo nad primárním klíčem. Kromě jednoduchého dotazu je povoleno použití zanořených dotazů s použitím WHERE EXISTS. Obdobným typem je materializovaný pohled nad tabulkou bez primárního klíče. V takovém případě se pro identifikaci záznamů ve zdrojové tabulce používá ROWID. Třetím zástupcem je materializovaný pohled, který je závislý na objektové tabulce. Objektový materializovaný pohled je složen z řádků daného objektu. Všechny tři typy umožňují inkrementální, rychlou obnovu dat ze zdrojových tabulek (fast refresh). Díky tomu, že jsou jednotlivé záznamy jednoznačně identifikovány, mohou se obměňovat pouze záznamy, které byly oproti datům v materializovaném pohledu změněny. Pro záznam změn nad daty slouží logovací tabulky (MLOG). Tyto logy obsahují identifikátor záznamu (primární klíč, ROWID nebo id objektu), čas provedení změny, vektor změny a typ operace. Nad zdrojovou tabulkou se vytvoří triger a při DML operaci se uloží změna do mlogu. Tam je změna uchována, dokud si ji nevyzvednou všechny závislé materializované pohledy. Název tabulky mlogu je automaticky vygenerován ve formátu MLOG$_název_zdrojové_tabulky.

```
-- vytvoření MLOG
CREATE MATERIALIZED VIEW LOG ON zdrojova_tab WITH PRIMARY KEY [ROWID,
OBJECT ID];
```

Materializované pohledy se však nemusí omezovat na jednu zdrojovou tabulku. Mohou slučovat komplexní data ze vzdálené databáze a ukládat jejich kopii do lokální databáze. Takové pohledy se označují jako komplexní (Complex Materialized View). V definici pohledu při výběru dat se mohou použít množinové operace. Takové pohledy však lze obnovovat pouze jako celek a při obnově se celý obsah materializovaného pohledu smaže a načte znovu ze zdrojových tabulek [8].

Obnovovací skupiny

Materializované pohledy lze seskupovat do skupin (Refresh Group). Tyto skupiny zajišťují konzistenci mezi jednotlivými pohledy. Pokud se replikuje více tabulek z jedné databáze a mezi tabulkami jsou referenční omezení, obnovovací skupina zajistí obnovení všech pohledů najednou v jedné transakci. Pokud dojde k chybě u jednoho pohledu, neobnoví se pohled žádný. Práce se skupinami je zajištěna přes systémový balík DBMS_REFRESH. Skupina se vytváří pomocí procedury MAKE, která zároveň vytvoří databázový job, který automaticky vyvolává obnovu celé skupiny. Procedura ADD slouží k přidání pohledu do skupiny a procedura REFRESH vyvolá ruční obnovení [9].

```
-- definice procedury pro vytvoření skupiny
DBMS_REFRESH.MAKE (
   name                 IN     VARCHAR2
   { list               IN     VARCHAR2,
   | tab                IN     DBMS_UTILITY.UNCL_ARRAY,}
   next_date            IN     DATE,
   interval             IN     VARCHAR2,
   implicit_destroy     IN     BOOLEAN          := false,
   lax                  IN     BOOLEAN          := false,
   job                  IN     BINARY INTEGER   := 0,
   rollback_seg         IN     VARCHAR2         := NULL,
   push_deferred_rpc    IN     BOOLEAN          := true,
   refresh_after_errors IN     BOOLEAN          := false
   purge_option         IN     BINARY_INTEGER   := NULL,
   parallelism          IN     BINARY_INTEGER   := NULL,
   heap_size            IN     BINARY_INTEGER   := NULL);

-- definice procedury pro přidání do skupiny
DBMS_REFRESH.ADD (
   name     IN VARCHAR2,
   { list   IN VARCHAR2,
   | tab    IN DBMS_UTILITY.UNCL_ARRAY, }
   lax      IN BOOLEAN := false);

-- definice procedury pro obnovení
DBMS_REFRESH.REFRESH (
   name   IN    VARCHAR2);
```

3 Popis systémů

Registr poskytovatelů péče je systém, ve kterém jsou uchovány všechny informace týkající se všech lékařských subjektů v České republice. Těmito subjekty jsou myšleny velké fakultní nemocnice, kliniky, ambulance, stomatologická pracoviště, záchranné služby a mnoho dalších typů lékařských zařízení. V systému jsou uloženy všechny důležité informace, které je potřeba znát o jednotlivých poskytovatelích péče. Každé zařízení zaměstnává lékaře a jiný zdravotnický personál, vykazuje provedenou práci a právě pro tyto účely mají zařízení podepsanou smlouvu s pojišťovnou a na základě nasmlouvaných výkonů, odborností aj. dostávají od pojišťovny peníze. Podrobnější databázová struktura systému je popsána v následujících podkapitolách, které se zabývají centralizovaným a decentralizovaným systémem, ve kterých jsou informace uloženy. Detailní výpisy u jednotlivých entit pak napomůžou k rychlé orientaci v problematice datové integraci a to především v samotné implementaci.

3.1 Decentralizovaný systém

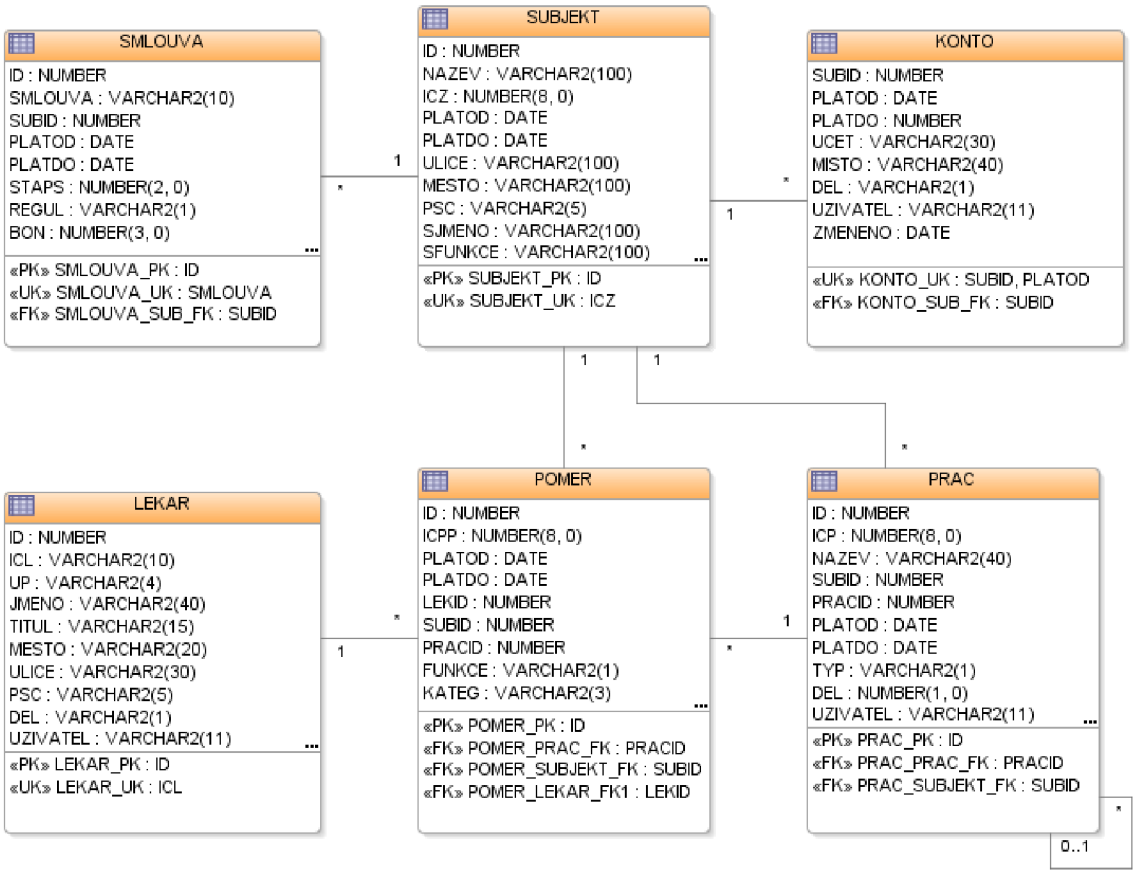
V dnešní době je celý informační systém pojišťovny rozdělen na několik desítek databázových systémů v prostředí Oracle 9i . Toto rozdělení koresponduje s regionálním rozdělením území České republiky podle okresů, což znamená, že existuje 77 okresů a pro každý okres jeden informační systém. Tyto systémy zahrnují veškeré informace a údaje potřebné pro chod pojišťovny. Mimo jiné systém obsahuje výdajovou část, která zastřešuje právě data o poskytovatelích péče.

Výdajová část obsahuje několik desítek databázových tabulek. Spoustu z nich ale plní úkol pomocných tabulek nebo číselníků[1]. Tyto tabulky slouží pro zpřehlednění záznamů, přenos dat a pro datovou integraci jsou nezajímavé, jelikož stejné číselníky jsou i na straně centralizovaného systému a pomocné tabulky neobsahují platná data. Pro přehlednost jsou tabulky s relevantními daty rozděleny do třech logických celků. U každé tabulky je uveden její význam, seznam sloupců s komentáři a seznam integritních omezení a indexů.

3.1.1 Detaily subjektu

Prvním celkem jsou tabulky, které se vztahují k lékařskému zařízení. Tvoří jakýsi fyzický popis poskytovatele péče. Tyto tabulky jsou jádrem celého informačního systému a ostatní tabulky jen rozšiřují a detailně popisují smlouvy a přílohy smluv, které vytváří smluvní podmínky mezi poskytovateli péče a pojišťovnou. Struktura a provázanost jednotlivých entit lze viděl na Obrázek 3.1.

[1] Číselníkem je myšlena tabulka, jejímž obsahem jsou statická data, která jsou typicky ve tvaru: kód, popis a platnost daného záznamu.



Obrázek 3.1: ER-diagram subjektu

Subjekt

Subjekt, neboli zařízení, je základním pilířem v systému a nese informace o samotném poskytovateli péče. Unikátním identifikátorem je identifikační číslo zařízení ICZ. ICZ je osmimístné číslo, které nejen identifikuje subjekt, ale označuje příslušnost ke konkrétnímu okresu a to podle prvních dvou číslic.

Název entity: SUBJEKT

Tabulka 3.1: Seznam atributů SUBJEKT

Název atributu	Komentář
ID	Interní identifikátor přidělený ze sekvence
ICZ	Identifikační číslo subjektu
NAZEV	Název zařízení
PLATOD	Platnost záznamu od
PLATDO	Platnost záznamu do
ULICE	Ulice s číslem popisným
MESTO	Obec
PSC	Poštovní směrovací číslo
SJMENO	Jméno a příjmení statutárního zástupce
SFUNKCE	Funkce statutárního zástupce
DEL	Indikátor zrušeného záznamu (A – zrušeno, N – nezrušeno)
ZMENENO	Datum, poslední změny
UZIVATEL	Osoba, která provedla poslední změnu

Tabulka 3.2: Seznam integritních omezení SUBJEKT

Název omezení	Typ[2]	Seznam atributů	Referenční tabulka
SUBJEKT_PK	P	ID	
SUBJEKT_UK	U	ICZ	

Bankovní spojení

Každý subjekt má uveden své bankovní spojení, přes které dochází k bankovnímu styku. Bankovní spojení je vázáno přímo na entitu SUBJEKT a pro jedno časové období může existovat pouze jeden záznam.
Název entity: KONTO

Tabulka 3.3: Seznam atributů KONTO

Název atributu	Komentář
SUBID	Interní odkaz na entitu SUBJEKT
PLATOD	Platnost záznamu od
PLATDO	Platnost záznamu do
UCET	Učet bankovního spojení
MISTO	Sídlo banky
DEL	Indikátor zrušeného záznamu (A – zrušeno, N – nezrušeno)
ZMENENO	Datum, poslední změny
UZIVATEL	Osoba, která provedla poslední změnu

Tabulka 3.4: Seznam integritních omezení KONTO

Název omezení	Typ	Seznam atributů	Referenční tabulka
KONTO_UK	U	SUBID, PLATOD	
KONTO_SUB_FK	F	SUBID	SUBJEKT

Smlouva

Poskytovatel péče musí mít s pojišťovnou podepsanou smlouvu, ve které jsou definovány právní vztahy mezi subjektem a pojišťovnou. Smlouva je vždy platná v určitém časovém období s unikátním identifikátorem – číslo smlouvy. Hlavička smlouvy je uložena v následující entitě.
Název entity: SMLOUVA

Tabulka 3.5: Seznam atributu SMLOUVA

Název atributu	Komentář
ID	Interní identifikátor přidělený ze sekvence
SMLOUVA	Unikátní číslo smlouvy
SUBID	Interní odkaz na entitu SUBJEKT
PLATOD	Platnost záznamu od
PLATDO	Platnost záznamu do
STAVS	Stav smlouvy
REGUL	Příznak regulační srážky
BON	Bonifikace typovaných dokladů
TYPZAR	Typ zařízení
KATEG	Kategorie činnosti zařízení
SPLATNOST	Splatnost ve dnech
POZNAMKA	Text poznámky
ZMENENO	Datum, poslední změny
UZIVATEL	Osoba, která provedla poslední změnu

[2] Typem se myslí typ integritního omezení. Jednotlivé zkratky znamenají: P – Primární klíč, F – cizí klíč, U – unikátní index.

Tabulka 3.6: Seznam integritních omezení SMLOUVA

Název omezení	Typ	Seznam atributů	Referenční tabulka
SMLOUVA_PK	P	ID	
SMLOUVA_UK	U	SMLOUVA	
SMLOUVA_SUB_FK	F	SUBID	SUBJEKT

Tabulka 3.7: Seznam indexů SMLOUVA

Název indexu	Unikátnost	Seznam atributů
SML1	N	SUBID

Pracoviště

Zdravotnické zařízení je rozděleno na pracoviště. Tyto pracoviště mohou, ale nemusí, být sloučeny ve větší celky – oddělení. Takové pracoviště pak označujeme za podřízené pracoviště. Oddělení i pracoviště jsou uloženy v jedné tabulce a jsou od sebe rozlišeny atributem TYP. Pracoviště tvoří základní jednotku, na kterou se váže příloha smlouvy, která je popsána níže.
Název entity: PRAC

Tabulka 3.8: Seznam atributů PRAC

Název atributu	Komentář
ID	Interní identifikátor přidělený ze sekvence
ICP	Identifikační číslo pracoviště
NAZEV	Název pracoviště
SUBID	Interní odkaz na entitu SUBJEKT
PLATOD	Platnost záznamu od
PLATDO	Platnost záznamu do
TYP	Typ pracoviště (P - pracoviště, O - oddělení)
PRACID	Odkaz na oddělení
DEL	Indikátor zrušeného záznamu (A – zrušeno, N – nezrušeno)
ZMENENO	Datum, poslední změny
UZIVATEL	Osoba, která provedla poslední změnu

Tabulka 3.9: Seznam integritních omezení PRAC

Název omezení	Typ	Seznam atributů	Referenční tabulka
PRAC_PK	P	ID	
PRAC_SUBJEKT_FK	F	SUBID	SUBJEKT
PRAC_PRAC_FK	F	PRAID	PRAC

Tabulka 3.10: Seznam indexů PRAC

Název indexu	Unikátnost	Seznam atributů
PRAC1	N	SUBID
PRAC2	N	ICP, SUBID, DEL
PRAC3	N	PRACID

Lékař

Na pracovištích pracují lékaři a jiný zdravotnický personál. Pojišťovna si vede evidenci těchto pracovníků a na základě jejich počtu a kvalifikace se upravuje suma peněz, které si zařízení účtuje. Zajímavostí je, že lékař je evidován na jedné pobočce jako vlastní, ale může být evidován i na ostatních pobočkách. Pro rozeznání z jaké pobočky lékař pochází, slouží sloupec UP, který je vázán na tabulku UP obsahující všechny pobočky.
Název entity: LEKAR

Tabulka 3.11: Seznam atributů LEKAR

Název atributu	Komentář
ID	Interní identifikátor přidělený ze sekvence
ICL	Identifikační číslo lékaře
UP	Pobočka, na které je lékař evidován
JMENO	Jméno a příjmení lékaře
TITUL	Titul lékaře
MESTO	Obec bydliště
ULICE	Ulice bydliště
PSC	Poštovní směrovací číslo
DEL	Indikátor zrušeného záznamu (A – zrušeno, N – nezrušeno)
ZMENENO	Datum, poslední změny
UZIVATEL	Osoba, která provedla poslední změnu

Tabulka 3.12: Seznam integritních omezení LEKAR

Název omezení	Typ	Seznam atributů	Referenční tabulka
LEKAR_PK	P	ID	
LEKAR_UK	U	ICL, UP	

Pracovní poměr

Vazbou mezi lékařem a pracovištěm je jeho pracovní poměr. Lékař může mít více pracovních poměrů a to i v jednom časovém období. Pracovní poměr také definuje kategorii pracovníka a jeho funkci na pracovišti. Musí platit, že pracoviště má pouze jeden platný pracovní poměr s funkcí vedoucí.
Název entity: POMER

Tabulka 3.13: Seznam atributů POMER

Název atributu	Komentář
ID	Interní identifikátor přidělený ze sekvence
ICPP	Identifikační číslo pracovního poměru
PLATOD	Platnost záznamu od
PLATDO	Platnost záznamu do
LEKID	Interní odkaz na entitu LEKAR
SUB	Interní odkaz na entitu SUBJEKT
PRA	Interní odkaz na entitu PRAC
FUNKCE	Funkce na pracovišti
KATEG	Kategorie pracovníka
DEL	Indikátor zrušeného záznamu (A – zrušeno, N – nezrušeno)
ZMENENO	Datum, poslední změny
UZIVATEL	Osoba, která provedla poslední změnu

Tabulka 3.14: Seznam integritních omezení POMER

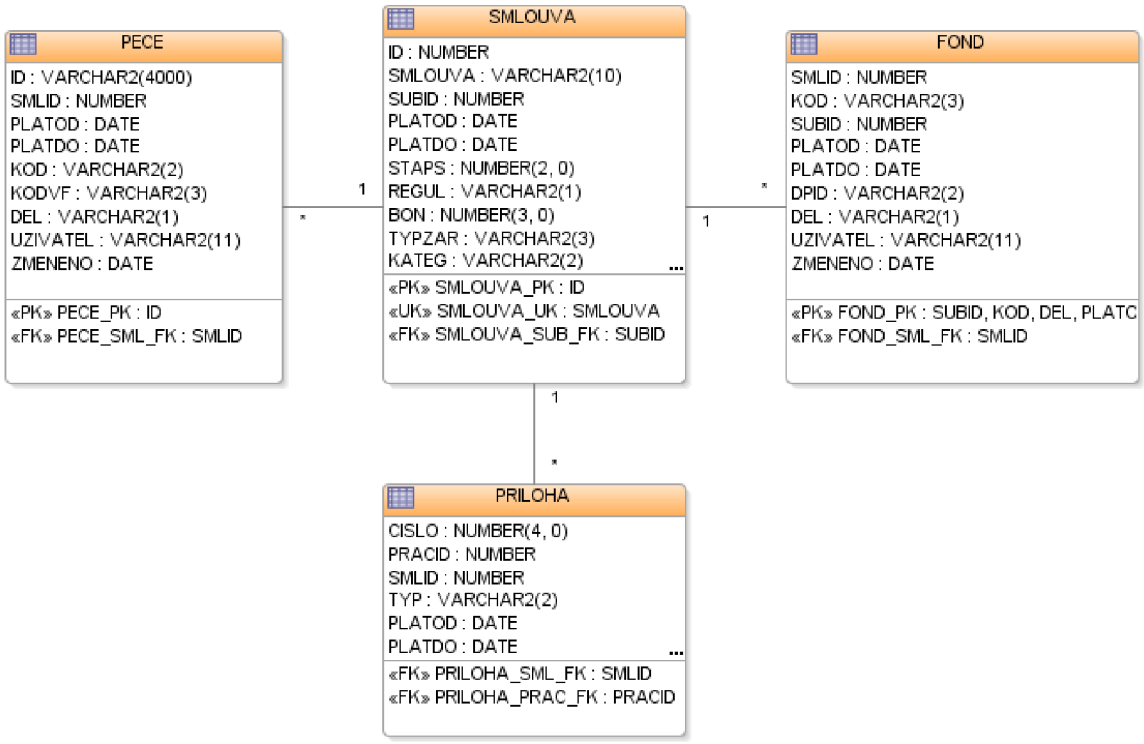
Název omezení	Typ	Seznam atributů	Referenční tabulka
POMER_PK	P	ID	
POMER_SUBJEKT_FK	F	SUBJEKT	SUBID
POMER_PRAC_FK	F	PRAC	PRACID
POMER_LEKAR_FK	F	LEKAR	LEKID

Tabulka 3.15: Seznam indexů POMER

Název indexu	Unikátnost	Seznam atributů
POMER1	N	LEKID
POMER2	N	ICPP
POMER3	N	PRACID

3.1.2 Detaily smlouvy

Druhým logickým celkem je soubor tabulek, které rozšiřují smlouvu o ekonomické detaily – výdajové fondy a druhy péče. Dále je na smlouvu vázána příloha, která definuje vybavení a druh pracoviště nebo oddělení. Na Obrázek 3.2 lze viděl strukturu detailů smlouvy.



Obrázek 3.2: ER-diagram smlouvy

Výdajový fond

Výdajové fondy jsou detailem smlouvy, které můžeme zařadit do ekonomické části smlouvy. Výdajové fondy, jak již vypovídá název, jsou fondy, ze kterých může poskytovatel péče čerpat finance. Záznam výdajového fondu je jedinečně identifikován kódem výdajového fondu, datem platnosti a subjektem, ke kterému smlouva náleží. Mezi smlouvou a fondy platí vztah 1:N, ale musí platit pravidlo, že pro jednu smlouvu existuje pouze jeden kód výdajového fondu pro jedno období.
Název entity: FOND

Tabulka 3.16: Seznam atributů FOND

Název atributu	Komentář
SMLID	Interní odkaz na entitu SMLOUVA
KOD	Kód výdajového fondu
SUBID	Interní odkaz na entitu SUBJEKT
PLATOD	Platnost záznamu od
PLATDO	Platnost záznamu do
DPID	Kód druhu péče
DEL	Indikátor zrušeného záznamu (A – zrušeno, N – nezrušeno)
ZMENENO	Datum, poslední změny
UZIVATEL	Osoba, která provedla poslední změnu

Tabulka 3.17: Seznam integritních omezení FOND

Název omezení	Typ	Seznam atributů	Referenční tabulka
FOND_PK	P	SUBID,KOD,DEL,PLATOD	
FOND_SML_FK	F	SMLID	SMLOUVA

Tabulka 3.18: Seznam indexů FOND

Název indexu	Unikátnost	Seznam atributů
FOND1	N	SUBID,KOD,PLATOD

Druh péče

Druhy péče jsou druhou entitou, která se týká financí. Aby mohlo zařízení čerpat peníze z výdajových fondů, musí poskytovat určitý druh péče. V této tabulce jsou uloženy právě druhy péče, jež zařízení poskytuje. Mezi výdajovým fondem a druhem péče existuje logická vazba, která je realizována atributem DPID v entitě FOND a atributem KODVF v entitě SMLPECE. Vazba mezi smlouvou a druhem péče je stejná, jako tomu je u výdajového fondu.
Název entity: PECE

Tabulka 3.19: Seznam atributů PECE

Název atributu	Komentář
ID	Interní identifikátor přidělený ze sekvence
SMLID	Interní odkaz na entitu SMLOUVA
PLATOD	Platnost záznamu od
PLATDO	Platnost záznamu do
KOD	Kód druhu péče
KODVF	Kód výdajového fondu
DEL	Indikátor zrušeného záznamu (A – zrušeno, N – nezrušeno)
ZMENENO	Datum, poslední změny
UZIVATEL	Osoba, která provedla poslední změnu

Tabulka 3.20: Seznam integritních omezení PECE

Název omezení	Typ	Seznam atributů	Referenční tabulka
PECE_PK	P	Z	
PECE_SML_FK	F	SMLID	SMLOUVA

Tabulka 3.21: Seznam indexů PECE

Název indexu	Unikátnost	Seznam atributů
PECE1	N	SMLID

Příloha

Příloha smlouvy je rozšíření smlouvy, která se vztahuje na jedno konkrétní pracoviště nebo oddělení a popisuje jeho druh a zaměření. Příloh existuje několik typů: A – ambulance, C – primariát, CB – chirurgie a S – stomatologové. Pro všechny typy ale existuje společná tabulka, která je tím pádem dosti nepřehledná, a proto existují databázové pohledy pro každý typ přílohy. Pro přehlednost i zde jsou uvedeny přednostně společné atributy a pak atributy jednotlivých typů příloh.
Název entity: PRILOHA
Název pohledů: PRILOHA <typ přílohy>

Tabulka 3.22: Seznam atributů PRILOHA

Název atributu	Komentář
Obecné atributy	
CISLO	Číslo přílohy. Nemusí být unikátní.
SMLID	Interní odkaz na entitu SMLOUVA
PRACID	Interní odkaz na entitu PRAC
TYP	Typ přílohy
PLATOD	Platnost záznamu od
PLATDO	Platnost záznamu do
DNU	Minimální počet dnů péče za týden
HODIN	Minimální počet hodin péče za týden
OKRES	Příznak územní oblasti – vlastní okres
REGION	Příznak územní oblasti – vyšší region
DEL	Indikátor zrušeného záznamu (A – zrušeno, N – nezrušeno)
ZMENENO	Datum, poslední změny
UZIVATEL	Osoba, která provedla poslední změnu
Příloha C	
LEKARI	Počet lékařů
VS	Počet jiných zaměstnanců s VS
SS	Počet zaměstnanců se SS
ATESTACE1	Počet lékařů s prvním stupněm atestace
ATESTACEVYS	Počet lékařů s vyšším stupněm atestace
LEKINT	Počet lékařů na intenzivní péči
PRIMDNU	Počet dnů přítomnosti primáře v týdnu
PRIMHODIN	Počet hodin přítomnosti primáře v týdnu
Příloha CB	
OPSAL	Příznak – sdílený operační sál
PROVOZ	Příznak – nepřetržitý provoz
SDILENI	Příznak – sdílená dospávající lůžka
LUZKA	Počet dospávajících lůžek
Příloha S	
STOMAG	Příznak nasmlouvání stomatologického výkonu

Tabulka 3.23: Seznam integritních omezení PRILOHA

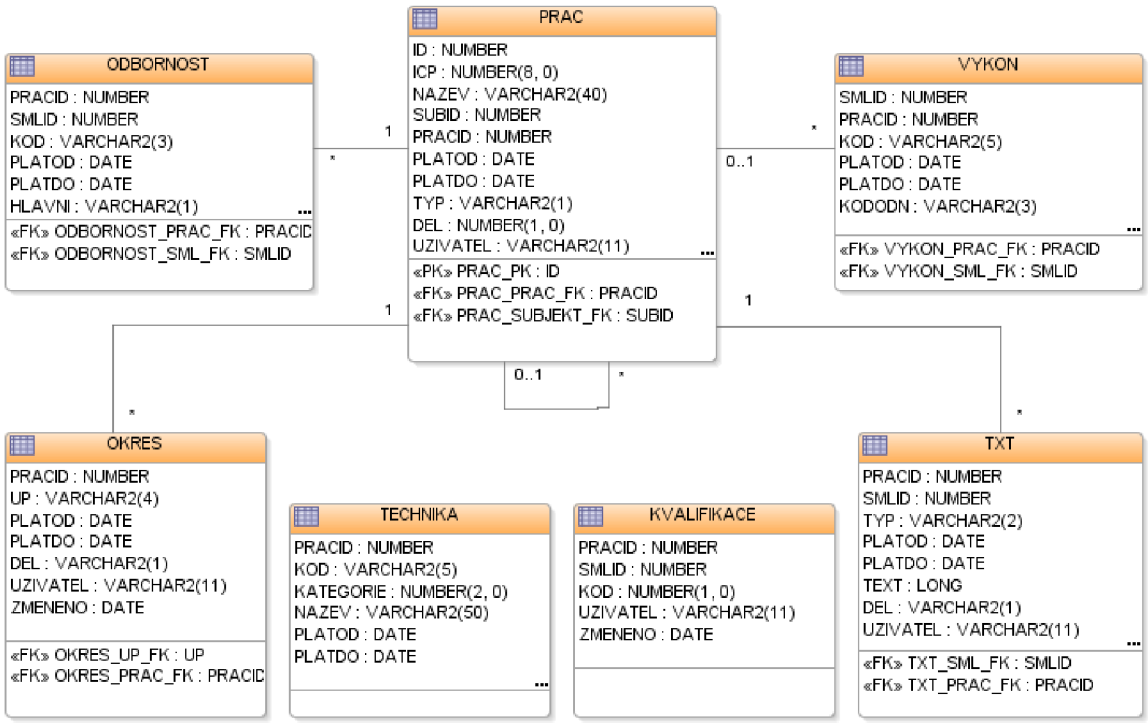
Název omezení	Typ	Seznam atributů	Referenční tabulka
PRILOHA _PRAC_FK	F	PRAID	PRAC
PRILOHA _SML_FK	F	SMLID	SMLOUVA

Tabulka 3.24: Seznam indexů PRILOHA

Název indexu	Unikátnost	Seznam atributů
PRILOHA 1	N	PRACID, TYP, SMLID, PLATDO

3.1.3 Detaily pracoviště

Poslední skupinou entit, které budou navázány na datovou integraci, je skupina tvořená databázovými tabulkami rozšiřující pracoviště. Jedná se o odbornost, výkony, garantované území, zdravotnickou techniku, stomatologickou kvalifikaci a volné texty. Struktura těchto detailů je zobrazena na Obrázek 3.3. Pro všechny detaily platí následující pravidla. Pracoviště je k detailu ve vztahu 1:N. Dále je hlídáno, že žádný detail se stejným kódem se časově nepřekrývá, což ale nebrání tomu, aby bylo více kódu nasmlouváno v jeden časový okamžik.



Obrázek 3.3: ER-diagram pracoviště

Odbornost

Každé pracoviště musí mít doloženou určitou odbornost, podle které je možné určit na co je dané pracoviště specializované. Mimo vazbu na pracoviště je odbornost vázána na smlouvu. Na pracovišti může být více odborností, ale pouze jedna hlavní, která je určena atributem HLAVNI.

Název entity: ODBORNOST

Tabulka 3.25: Seznam atributů ODBORNOST

Název atributu	Komentář
PRACID	Interní odkaz na entitu PRAC
SMLID	Interní odkaz na entitu SMLOUVA
KOD	Kód odbornosti
PLATOD	Platnost záznamu od
PLATDO	Platnost záznamu do
HLAVNI	Příznak hlavní odbornosti
HOSP	Hospitalizační paušál
AMBUL	Ambulantní paušál
DEL	Indikátor zrušeného záznamu (A – zrušeno, N – nezrušeno)
ZMENENO	Datum, poslední změny
UZIVATEL	Osoba, která provedla poslední změnu

Tabulka 3.26: Seznam integritních omezení ODBORNOST

Název omezení	Typ	Seznam atributů	Referenční tabulka
ODBORNOST _PRAC_FK	F	PRAID	PRAC
ODBORNOST _SML_FK	F	SMLID	SMLOUVA

Tabulka 3.27: Seznam indexů ODBORNOST

Název indexu	Unikátnost	Seznam atributů
ODBORNOST 1	N	PRAID, SMLID, KOD, DEL, PLATOD
ODBORNOST 2	N	SMLID, KOD, DEL
ODBORNOST 3	N	PRAID, KOD, DEL

Smluvní výkon

Smluvními výkony chápeme zdravotnické výkony, které může dané pracoviště vykonávat. Jsou úzce spojeny s odborností, jelikož určité výkony mohou být nasmlouvány na pracovišti jen tehdy, pokud má pracoviště nasmlouvánu i odbornost, která je uvedena v atributu KODODN. Jak už nasvědčuje název, tak výkony jsou vázány i na smlouvu.
Název entity: VYKON

Tabulka 3.28: Seznam atributů VYKON

Název atributu	Komentář
PRACID	Interní odkaz na entitu PRA
SMLID	Interní odkaz na entitu SML
KOD	Kód výkonu
PLATOD	Platnost záznamu od
PLATDO	Platnost záznamu do
KODODN	Kód odbornosti
TYP	Typ výkonu (Z – základní, D – další, O – operační)
DEL	Indikátor zrušeného záznamu (A – zrušeno, N – nezrušeno)
ZMENENO	Datum, poslední změny
UZIVATEL	Osoba, která provedla poslední změnu

Tabulka 3.29: Seznam integritních omezení VYKON

Název omezení	Typ	Seznam atributů	Referenční tabulka
VYKON_PRAC_FK	F	PRAID	PRAC
VYKON_SML_FK	F	SMLID	SMLOUVA

Tabulka 3.30: Seznam indexů VYKON

Název indexu	Unikátnost	Seznam atributů
VYKON1	N	SMLID, KOD

Garantované území

Garantované území je spádová oblast, ve které působí dané pracoviště. Čísla okresů korespondují s čísly poboček a tento údaj je vázán cizím klíčem na tabulku UP.
Název entity: OKRES

Tabulka 3.31: Seznam atributů OKRES

Název atributu	Komentář
PRACID	Interní odkaz na entitu PRAC
UP	Číslo garantovaného okresu
PLATOD	Platnost záznamu od

PLATDO	Platnost záznamu do
DEL	Indikátor zrušeného záznamu (A – zrušeno, N – nezrušeno)
ZMENENO	Datum, poslední změny
UZIVATEL	Osoba, která provedla poslední změnu

Tabulka 3.32: Seznam integritních omezení OKRES

Název omezení	Typ	Seznam atributů	Referenční tabulka
OKRES_PRAC_FK	F	PRAID	PRAC
OKRES_UP_FK	F	SMLID	SMLOUVA

Tabulka 3.33: Seznam indexů PRAOKR

Název indexu	Unikátnost	Seznam atributů
OKRES1	N	PRACID, UP, PLATDO

Zdravotnická technika

Zdravotnická technika reprezentuje vybavení pracoviště. Mohou to být drahé diagnostické přístroje i obyčejné lékařské nástroje. Zdravotnická technika je rozdělena do deseti kategorií a právě kategorií a kódem je jednoznačně identifikován typ zařízení.
Název entity: TECHNIKA

Tabulka 3.34: Seznam atributů TECHNIKA

Název atributu	Komentář
PRACID	Interní odkaz na entitu PRAC
KOD	Kód techniky
KATEGORIE	Kategorie techniky
NAZEV	Název techniky
PLATOD	Platnost záznamu od
PLATDO	Platnost záznamu do
CISLO	Evidenční číslo
DEL	Indikátor zrušeného záznamu (A – zrušeno, N – nezrušeno)
ZMENENO	Datum, poslední změny
UZIVATEL	Osoba, která provedla poslední změnu

Stomatologická kvalifikace

Stomatologická kvalifikace je nejvyšší doložená kvalifikace vedoucího na stomatologickém pracovišti. Tato entita se váže pouze k pracovišti, na kterém je nasmlouvána příloha typu S.
Název entity: KVALIFIKACE

Tabulka 3.35: Seznam atributů KVALIFIKACE

Název atributu	Komentář
PRACID	Interní odkaz na entitu PRAC
SMLID	Interní odkaz na entitu SMLOUVA
KOD	Kód kvalifikace
ZMENENO	Datum, poslední změny
UZIVATEL	Osoba, která provedla poslední změnu

Volné texty

Mimo výše uvedené detaily pracoviště ještě existují další údaje, které nelze strukturovaně ukládat do databázových tabulek. K tomuto účelu jsou zavedeny volné texty, které tyto informace ukládají do datového typu LONG, což je dlouhý textový řetězec. Mimo samotného textu je v záznamu uložen také druh volného textu (adresa, lékařská kvalifikace, jiné ujednání a další).
Název entity: TXT

Tabulka 3.36: Seznam atributů TXT

Název atributu	Komentář
PRACID	Interní odkaz na entitu PRAC
SMLID	Interní odkaz na entitu SMLOUVA
TYP	Typ volného textu
PLATOD	Platnost záznamu od
PLATDO	Platnost záznamu do
TEXT	Znění volného textu
DEL	Indikátor zrušeného záznamu (A – zrušeno, N – nezrušeno)
ZMENENO	Datum, poslední změny
UZIVATEL	Osoba, která provedla poslední změnu

Tabulka 3.37: Seznam integritních omezení PRATXT

Název omezení	Typ	Seznam atributů	Referenční tabulka
TXT_PRAC_FK	F	PRACID	PRAC
TXT_SML_FK	F	SMLID	SMLOUVA

Tabulka 3.38: Seznam indexů PRATXT

Název indexu	Unikátnost	Seznam atributů
TXT1	N	PRACID, TYP, PLATDO

3.2 Centralizovaný systém

Původní pobočkové systémy jsou zastaralé a nevyhovující. Pojišťovna proto směřuje další vývoj do centrálních systémů. Jedním takovým centralizovaným systémem je i Centrální registr poskytovatelů péče (CRPP), který bude obsahovat data z výdajové části pobočkové aplikace. Jelikož se jedná o kompletní reengeneering navrhovaný pro databázový systém Oracle 10g, je struktura databáze poněkud odlišná od původního systému. Názvy entit jsou odlišné, ale obsahují ty samé informace, které byly uloženy ve starých tabulkách. Zásadním rozdílem je práce s unikátními identifikátory a vazby mezi jednotlivými entitami, které budou uvedeny v následujících kapitolách. Jsou v nich rozebrány jednotlivé entity a především mapování mezi novými a původními tabulkami. Právě správné mapování mezi jednotlivými sloupci dvou různých databází je klíčové pro vzájemnou datovou integraci. Mimo názvy atributů jsou sledovány rozdíly v datových typech jednotlivých sloupců a primární klíče nových tabulek.

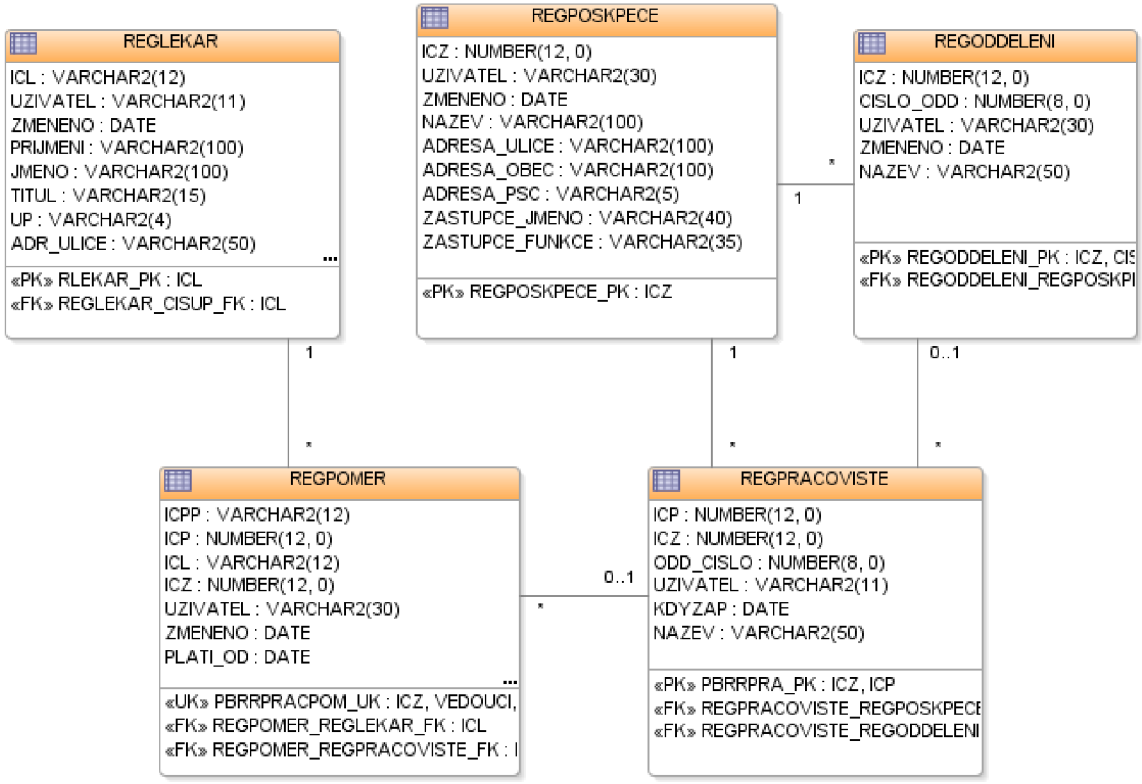
O proti původní databázi, kde rozdělení entit do tří celků bylo spíše intuitivní, v CRPP jsou tabulky rozděleny do šesti samostatných oddílů, které jsou odlišeny prefixem ve jménu databázových objektů: CIS – číselníky, REG – registrační část, VYKS – výkonná oblast smlouvy, VYKP – výkonná oblast přílohy, PRACS – pracovní oblast smlouvy a PRACP – pracovní oblast přílohy. Pro datovou integraci jsou relevantními částmi jen registrační a výkonné oblasti, které budou samostatně popsány. Číselníky jsou pouze statická data a v pracovní oblasti se nachází rozpracovaná data, která nejsou zatím platná, a tudíž se z pohledu datové integrace nijak nezpracovávají.

V CRPP je veškerá byznys logika napsána v PL/SQL a uložena v balíčcích. Tyto balíčky jsou rozděleny do tří vrstev. První vrstvou je vrstva aplikační, která v sobě neobsahuje žádnou logiku a pouze předává požadavky z formulářů nižší vrstvě. Jedná se o návrhový vzor Facade Pattern, kdy dochází k zakrytí vnitřní struktury a logiky a také k vytvoření nezávislého rozhraní. Prostřední vrstvou je logická vrstva. Zde se nachází veškeré propojení jednotlivých entit, kontroly a validace. Všechny operace probíhají nad datovými typy rekord, které mají stejnou strukturu jako databázové tabulky, ale jsou rozšířeny o jednu položku – THE_ROWID, která je naplněna ROWID[3] konkrétního záznamu a je určena pro urychlení práce nad objekty. Logická vrstva ale stále nepřistupuje k databázi. Pro tyto účely slouží vrstva tabulková. Balíčky této vrstvy obsahují pouze procedury, které vykonávají SELECT (výběr), UPDATE (změnu), INSERT (vkládání), DELETE (mazání) a LOCK (zámek) nad určitou tabulkou.

3.2.1 Registrační část

Registrační část obsahuje entity, které popisují poskytovatele péče, jeho rozdělení na oddělení a pracoviště a lékaře, kteří v něm pracují. Markantní změnou ve struktuře entit je vymizení interních vazeb. Místo nich se používají unikátní identifikační čísla jednotlivých entit a složené primární klíče (v tabulkách vyznačeno tučným písmem). Další novinkou je vymizení údajů o platnosti záznamu. Na následujícím Obrázek 3.4 lze vidět entity registrační části.

[3] ROWID je pseudo sloupec každé tabulky, který jednoznačně identifikuje řádek v rámci celé databáze. Tento identifikátor obsahuje fyzické uložení záznamu: číslo databázového objektu, datový blok, pozici v tomto datovém bloku a číslo souboru v odpovídajícím tabulkovém prostoru. Použití ROWID je nejrychlejším způsobem jak přistoupit k jednotlivým řádkům v databázi a proto je vhodné jej používat v podmínce WHERE u SELECT [10].



Obrázek 3.4: ER-diagram - Registrační část

Poskytovatel péče

V CRPP je subjekt převážně označován jako poskytovatel péče (POSKPECE) a je jednoznačně identifikován pomocí identifikačního čísla zařízení (ICZ), které je ve tvaru osmimístného čísla, kde první dvě čísla označují mateřskou pobočku, pod kterou zařízení územně spadá, další tři čísla pak identifikují samotné zařízení a poslední trojčíslí bývá zpravidla rovno nule.
Název entity: REGPOSKPECE

Tabulka 3.39: Seznam atributů REGPOSKPECE

Atribut v CRPP	Typ	Původní atribut	Typ
ICZ	N(12)[4]	ICZ	N(8)
ZMENENO	D	ZMENENO	D
UZIVATEL	V(11)	UZIVATEL	V(11)
NAZEV	V(100)	NAZEV	V(50)
ADRESA_ULICE	V(100)	ULICE	V(50)
ADRESA_OBEC	V(100)	OBEC	V(50)
ADRESA_PSC	V(5)	PSC	V(5)
ZASTUPCE_JMENO	V(40)	SJMENO	V(40)
ZASTUPCE_FUNKCE	V(35)	SFUNKCE	V(35)

[4] Notace zápisu datových typů: N – NUMBER, V – VARCHAR2, D – DATE, L – LONG. Číslo v závorce reprezentuje počet znaků resp. přesnost u čísla.

Oddělení

Oproti pobočkové databázi jsou oddělení v CRPP uložena v samostatné tabulce a jsou identifikována pomocí čísla oddělení a ICZ zařízení, ke kterému náleží.
Název entity: REGODDELENI

Tabulka 3.40:Seznam atributů REGODDELENI

Atribut v CRPP	Typ	Původní atribut	Typ
ICZ	N(12)	Dohledání ICZ[5] podle SUBID	
ODD_CISLO	N(12)	ICP	N(8)
ZMENENO	D	ZMENENO	D
UZIVATEL	V(11)	UZIVATEL	V(11)
NAZEV	V(50)	NAZEV	V(40)

Pracoviště

Pracoviště může být podřízeno určitému oddělení a k určení oddělení slouží sloupec CISLO a ICZ. Pro identifikaci samotného pracoviště slouží ICZ a identifikační číslo pracoviště ICP.
Název entity: REGPRACOVISTE

Tabulka 3.41: Seznam atributů REGPRACOVISTE

Atribut v CRPP	Typ	Původní atribut	Typ
ICZ	N(12)	Dohledaní ICZ podle SUBID	
ICP	N(12)	ICZ	N(8)
ODD_CISLO	N(12)	Dohledání ICP podle PRACID	
ZMENENO	D	ZMENENO	D
UZIVATEL	V(11)	UZIVATEL	V(11)
NAZEV	V(50)	V	V(40)

Lékař

Lékař je v CRPP jednoznačně identifikován pomocí identifikačního čísla lékaře ICL. V porovnání s původní entitou má lékař rozděleno jméno do dvou sloupců – jméno a příjmení. V pobočkové aplikaci se za ICL vkládalo rodné číslo. CRPP umožňuje za ICL dosazovat i jiná čísla.
Název entity: REGLEKAR

Tabulka 3.42: Seznam atributů RLEKAR

Atribut v CRPP	Typ	Původní atribut	Typ
ICL	N(12)	ICL	N(10)
ZMENENO	D	ZMENENO	D
UZIVATEL	V(11)	UZIVATEL	V(11)
PRIJMENI	V(100)	JMENO (před 1. mezerou)	V(40)
JMENO	V(100)	JMENO (za 1. mezerou)	V(40)
TITUL	V(15)	TITUL	V(15)
UP	V(4)	UP	V(4)
ADR_ULICE	V(50)	ULICE	V(20)

[5] Jedná se o terminologii pobočkové aplikace – ICO = ICZ, ICZ = ICP, ICP = ICPP

ADR_OBEC	V(50)	MESTO	V(30)
ADR_PSC	V(5)	PSC	V(5)

Pracovní poměr

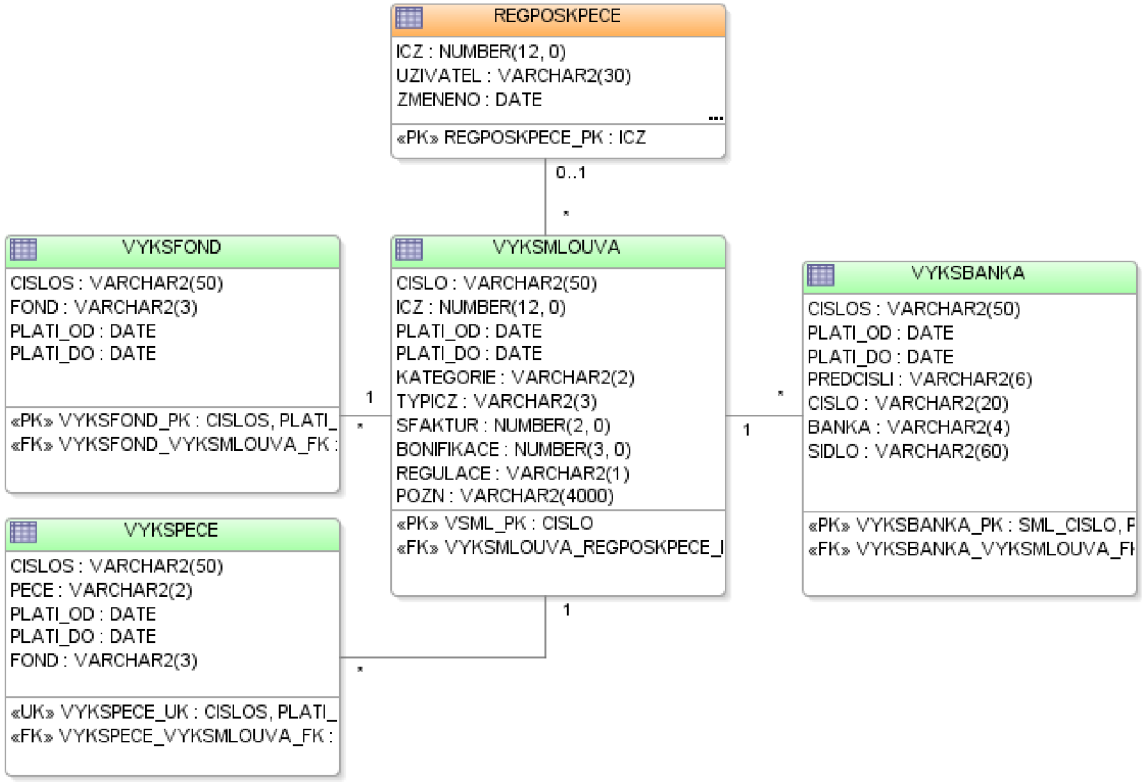
Pracovní poměr je opět vázán na lékaře a určité pracoviště, ale vazba je přímo realizována pomocí identifikátorů ICL, ICP a ICZ. S ohledem na data, která byla pořízena ve stávající aplikaci, se musel u této entity vyloučit primární klíč a byl nahrazen unikátním indexem. Existují totiž pracovní poměry, které nejsou vázány na pracoviště, ale na zařízení. Jde o historická data, která musela být zachována. ICPP znamená identifikátor pracovního poměru.
Název entity: REGPOMER

Tabulka 3.43: Seznam atributů REGPOMER

Atribut v CRPP	**Typ**	**Původní atribut**	**Typ**
ICPP	N(12)	ICPP	N(10)
ICP	N(12)	Dohledání ICP podle PRAID	
ICL	V(12)	Dohledání ICL podle LEKID	
ICZ	N(12)	Dohledání ICZ podle SUBID	
ZMENENO	V(11)	ZMENENO	V(11)
UZIVATEL	D	UZIVATEL	D
PLATI_OD	D	PLATOD	V(20)
PLATI_DO	D	PLATOD	V(30)
FUNKCE	V(1)	FUNKCE	V(1)
KATEGORIE	V(3)	KATEG	V(3)

3.2.2 Výkonná oblast – Smlouvy

Výkonná oblast uchovává data, která můžeme považovat za platné aktuální údaje. Ve výkonné oblasti jsou dva logické celky – smlouvy a přílohy. Na smlouvě jsou opět vázány ekonomické náležitosti a příloha. Změna ale tkví v navázání bankovního spojení na smlouvu, jelikož v původní aplikaci je bankovní spojení přímo na subjektu. Na Obrázek 3.5 jsou znázorněny vazby mezi entitami smluvní části, které jsou vybarveny zeleně, oranžové představují registrační část. Pro detaily smlouvy platí, že pro jedno časové období může platit pouze jeden záznam určitého jednoznačného kódu pro jednu smlouvu. V CRPP také musí platit to, že detaily smlouvy nesmí časově přesahovat samotnou platnost nadřízené smlouvy. Všechny entity, které náleží do této oblasti, mají prefix VYKS – VYK znamená výkonná oblast a S značí podřízenost pod smlouvou. Všechny entity obsahují atribut CISLOS, které představuje číslo smlouvy a odkazuje na nadřízenou smlouvu.



Obrázek 3.5: ER-diagram - Výkonná oblast – Smlouvy

Smlouva

Jednoznačným identifikátorem smlouvy je její číslo, které je unikátní v celém CRPP. Smlouva je vázána k poskytovateli péče a tato vazba je realizována sloupcem ICZ. Pro jedno zařízení může existovat v jednom období pouze jedna platná smlouva.
Název entity: VYKSMLOUVA

Tabulka 3.44: Seznam atributů VYKSMLOUVA

Atribut v CRPP	Typ	Původní atribut	Typ
CISLO	V(50)	SMLOUVA	N(10)
ICZ	N(12)	Dohledání ICZ podle SUBID	
PLATI_OD	D	PLATOD	D
PLATI_DO	D	PLATDO	D
KATEGORIE	V(2)	KATEG	V(2)
TYPICZ	V(3)	TYPZAR	V(3)
SFAKTUR	N(2)	SPLATNOST	N(1)
BONIFIKACE	N(3)	BON	N(3)
REGULACE	V(1)	REGUL	V(1)
POZN	V(4000)	POZNAMKA	V(100)

Bankovní spojení

Bankovní spojení je novým prvkem ekonomické části smluv. Záznam bankovního spojení se identifikováno pomocí čísla smlouvy, platností a celým číslem účtu.
Název entity: VYKSBANKA

Tabulka 3.45: Seznam atributů VYKSBANKA

Atribut v CRPP	Typ	Původní atribut	Typ
CISLOS	V(50)	Dohledání SMLOUVA přes SUBID ve SMLOUVA	
PLATI_OD	D	PLATOD	D
PLATI_DO	D	PLATDO	D
PREDCISLI	V(6)	UCET před ‚-'	
CISLO	V(20)	UCET mezi ‚-' a ‚/'	
BANKA	V(4)	UCET po ‚/'	
SIDLO	V(40)	BANKMISTO	V(40)

Výdajový fond

Výdajové fondy jsou jednoznačně identifikovány pomocí čísla smlouvy, platností a kódem výdajového fondu. Na rozdíl od původní entity FOND, v CRPP není žádná vazba ani na subjekt, ani na druh péče.
Název entity: VYKSFOND

Tabulka 3.46: Seznam atributů VYKSFOND

Atribut v CRPP	Typ	Původní atribut	Typ
CISLOS	V(50)	Dohledání SMLOUVA přes SMLID	
FOND	V(3)	KOD	V(3)
PLATI_OD	D	PLATOD	D
PLATI_DO	D	PLATDO	D

Druh péče

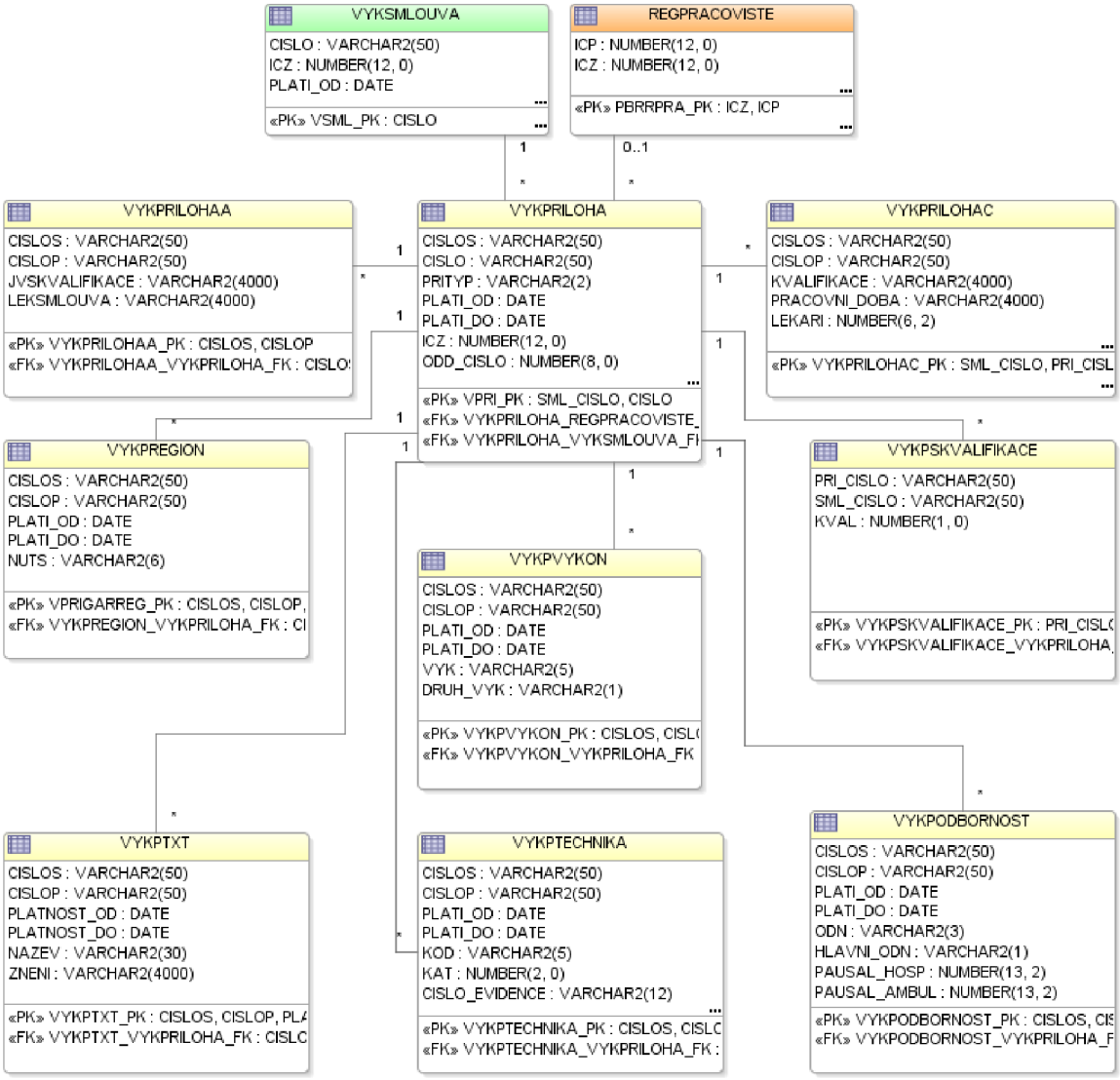
Druh péče nemá v CRPP primární klíč. Je to dáno tím, že odkaz na výdajový fond nemusí být vyplněn. Proto je zaveden unikátní index přes všechny atributy.
Název entity: VYKSPECE

Tabulka 3.47: Seznam atributů VSMLDRUHPECE

Atribut v CRPP	Typ	Původní atribut	Typ
CISLOS	V(50)	Dohledání SMLOUVA přes SMLID	
PECE	V(2)	KOD	V(2)
PLATI_OD	D	PLATOD	D
PLATI_DO	D	PLATDO	D
FOND	V(3)	KODVF	V(3)

3.2.3 Výkonná oblast – Přílohy

Příloha je sice detailem smlouvy, ale na rozdíl od pobočkového systému, kde byla příloha nevýznamným strukturálním prvkem, je v CRPP na příloze vázáno spoustu entit, které byly původně vázány přímo na pracovišti. Tyto entity mají prefix VYKP. Dalším velkým zásahem do struktury je rozdělení přílohy do několika tabulek. V pobočkové databázi byly přílohy uloženy do jedné tabulky, nad kterou existovaly pohledy, v CRPP jsou přílohy rozděleny do generické přílohy, která obsahuje obecnou hlavičku přílohy se společnými prvky, a do detailových tabulek, které představují jednotlivé přílohy. Detaily přílohy jsou vázány přes číslo smlouvy a číslo přílohy. Vazby mezi entitami lze vidět na Obrázek 3.6, kde jsou vyznačeny přílohy žlutě, smlouvy zeleně a registrační část oranžově.



Obrázek 3.6: ER-diagram - Výkonná oblast - Přílohy

Příloha

Příloha je v CRPP jednoznačně identifikována číslem přílohy a číslem smlouvy. Podle typu přílohy je vázána buď na pracoviště, nebo na oddělení. Musí platit, že existuje pouze jedna příloha na jednom pracovišti resp. oddělní pro jedno časové období. Kromě tabulky generické přílohy existují tabulky pro detaily přílohy, které obsahují specifické položky jednotlivých příloh. Jedna pro přílohy typu A a jedna pro přílohy typu C a CB. Příloha S má pouze jeden specifický atribut, proto byl tento atribut přidružen ke generické příloze a neexistuje speciální tabulka pro přílohu typu S. Novinkou u příloh v CRPP je přiřazení aktuálně platných volných textů přímo do entity přílohy.

Název entity: VYKPRILOHA – generická příloha
VYKPRILOHA<typ přílohy> – detaily jednotlivých příloh

Tabulka 3.48: Seznam atributů VYKPRILOHA

Atribut v CRPP	Typ	Původní atribut	Typ
CISLOS	V(50)	Dohledání SMLOUVA přes SMLID	
CISLO	V(50)	CISLO	N(4)
PRITYP	V(2)	TYP	V(2)
PLATI_OD	D	PLATOD	D
PLATI_DO	D	PLATDO	D
ICZ	N(12)	Dohledání ICZ přes SMLID v SMLOUVA	
ODD_CISLO	N(8)	Dohledání ICP přes PRACID v PRAC	
ICP	N(12)	Dohledání ICP přes PRACID v PRAC	
ADR	V(4000)	Přiřazení TEXT z TXT	
UJEDNANI	V(4000)	Přiřazení TEXT z TXT	
KVALIFIKACE	V(4000)	Přiřazení TEXT z TXT	
DNU	N(4)	DNU	N(1)
HODIN	N(4)	HOD	N(4)
VDNU	N(4)	PRIMDNU	N(1)
VHODIN	N(4)	PRIMHODIN	N(4)
OBLAST_OKRES	N(1)	OKRES	N(1)
OBLAST_ REGION	N(1)	REGION	N(1)
SVYKONY	N(1)	STOMAG	N(2)
VYBAVENI	V(4000)	Přiřazení TEXT z TXT	

Tabulka 3.49: Seznam atributů VYKPRILOHAA

Atribut v CRPP	Typ	Původní atribut	Typ
CISLOS	V(50)	Dohledání SMLOUVA přes SMLID	
CISLOP	V(50)	CISLO	N(4)
JVSKVALIFIKACE	V(4000)	Přiřazení TEXT z TXT	
LEKSMLOUVA	V(4000)	Přiřazení TEXT z TXT	

Tabulka 3.50: Seznam atributů VYKPRILOHAC

Atribut v CRPP	Typ	Původní atribut	Typ
CISLOS	V(50)	Dohledání SMLOUVA přes SMLID	
CISLOP	V(50)	CISLO	N(4)
PRACOVNI_DOBA	V(4000)	Přiřazení TEXT z TXT	
LEKARI	N(6,2)	LEKARI	N(6)
JVS	N(6,2)	VS	N(6)
SZP	N(6,2)	SS	N(6)
LEKAR_INT	N(6,2)	LEKINT	N(2)
LEKAR_ATEST1	N(6,2)	ATEST1	N(3)
LEKAR_ATESTV	N(6,2)	ATESTVYS	N(3)
OPERSAL	N(1)	OPSAL	N(1)
PROVEOZ24	N(1)	PROVOZ	N(1)
LUZKA_POCET	N(4)	LUZKA	N(3)
SDILENI_LUZEK	N(1)	SDILENI	N(1)

Odbornost

Odbornost v CRPP je jednoznačně identifikována číslem smlouvy a přílohy, platností záznamu a kódem odbornosti.
Název entity: VYKPODBORNOST

Tabulka 3.51: Seznam atributů VYKPODBORNOST

Atribut v CRPP	Typ	Původní atribut	Typ
CISLOS	V(50)	Dohledání SMLOUVA přes SMLID	
CISLOP	V(50)	Dohledání CISLO přes PRACID	
PLATI_OD	D	PLATOD	D
PLATI_DO	D	PLATDO	D
ODB	V(3)	KOD	V(3)
HLAVNI_ODN	V(1)	HLAVNI	V(1)
PAUSAL_HOSP	N(13)	HOSP	N(1)
PAUSAL_AMBUL	N(13)	AMBUL	N(3)

Smluvní Výkon

Smluvní výkony jsou jednoznačně identifikovány číslem smlouvy a přílohy, platností a kódem výkonu.
Název entity: VYKPVYKON

Tabulka 3.52: Seznam atributů VYKPVYKON

Atribut v CRPP	Typ	Původní atribut	Typ
CISLOS	V(50)	Dohledání SMLOUVA přes SMLID	
CISLOP	V(50)	Dohledání CISLO přes PRACID	
PLATI_OD	D	PLATOD	D
PLATI_DO	D	PLATDO	D
VYK	V(5)	KOD	V(5)
DRUH_VYK	V(1)	TYP	V(1)

Garantované území

Garantovaná území jsou v CRPP reprezentovány pomocí NUTS[6] namísto čísla pobočky. Garantované území je jednoznačně identifikováno číslem přílohy a smlouvy, platností a kódem NUTS.
Název entity: VYKPREGION

Tabulka 3.53: Seznam atributů VYKPREGION

Atribut v CRPP	Typ	Původní atributy	Typ
CISLOS	V(50)	Dohledání SMLOUVA přes SMLID	
CISLOP	V(50)	Dohledání CISLO přes PRACID	
PLATI_OD	D	PLATOD	D
PLATI_DO	D	PLATDO	D
NUTS	V(6)	Dohledání NUTS v číselníku CISNUTS přes UP	

[6] NUTS (Nomenclature Unit of Territorial Statistic) – Územní statistická jednotka – jsou územní celky vytvořené pro statistické účely EU [11].

Zdravotnická technika

Zdravotnická technika je v CRPP identifikována číslem smlouvy a přílohy, platností, kódem a kategorií techniky.
Název entity: VYKPTECHNIKA

Tabulka 3.54: Seznam atributů VYKPTECHNIKA

Atribut v CRPP	Typ	Původní atribut	Typ
CISLOS	V(50)	Dohledání SMLOUVA přes SMLID	
CISLOP	V(50)	Dohledání CISLO přes PRACID	
PLATI_OD	D	PLATOD	D
PLATI_DO	D	PLATDO	D
KOD	V(5)	KOD	V(5)
KAT	N(2)	KATEGORIE	N(2)
CISLO_EVIDENCE	V(12)	CISLO	V(12)
NAZEV	V(50)	NAZEV	V(50)

Stomatologická kvalifikace

Stomatologická kvalifikace je jednoznačně identifikována číslem smlouvy a přílohy a kódem stomatologické kvalifikace. Stomatologická kvalifikace je tímto trochu odlišná od ostatních entit, jelikož neobsahuje platnost samotného záznamu.
Název entity: VYKPSKVALIFIKACE

Tabulka 3.55: Seznam atributů VYKPSKVALIFIKACE

Atribut v CRPP	Typ	Původní atribut	Typ
CISLOS	V(50)	Dohledání SMLOUVA přes SMLID	
CISLOP	V(50)	Dohledání CISLO přes PRACID	
KVAL	N(1)	KOD	N(1)

Volné texty

Jak již bylo zmíněno, volné texty jsou součástí přílohových tabulek. Přesto v CRPP existuje tabulka, která obsahuje všechny volné texty a slouží pouze pro zaznamenání historie volných textů. Záznamy v této tabulce jsou identifikovány pomocí čísla smlouvy a přílohy a názvem volného textu. V CRPP totiž neexistují druhy, jak tomu bylo na pobočce, ale volné texty se jmenují podle názvů sloupců, ve kterých jsou uloženy na příloze.
Název entity: VYKPTXT

Tabulka 3.56: Seznam atributů VYKPTXT

Atribut v CRPP	Typ	Původní atribut	Typ
CISLOS	V(50)	Dohledání SMLOUVA přes SMLID	
CISLOP	V(50)	Dohledání CISLO přes PRACID	
PLATI_OD	D	PLATOD	D
PLATI_DO	D	PLATDO	D
NAZEV	V(30)	DRUHTXT převeden na NAZEV	
ZNENI	V(4000)	TEXT	L

4 Návrh datové integrace

Prostředí integrovaných systému je Oracle, přesněji Oracle 9i u decentralizovaných pobočkových aplikací a 10g u centralizovaného CRPP. Proto se nabízí řešit jejich vzájemné propojení přes prostředky, které nabízí firma Oracle. Tím odpadá nejen problém se vzájemnou kompatibilitou, ale i ušetření času a peněz zákazníkovi, který nemusí vyřizovat nové licence.

V této kapitole je rozebrána obecná koncepce datové integrace. Jedná se především o princip importu stávajících dat do centrálního systému a způsobu replikace dat z centrálního systému na jednotlivé pobočky. Pro obousměrné integrační toky bude využita integrační vrstva IML (Integration Middleware Layer).

4.1 Integrační vrstva

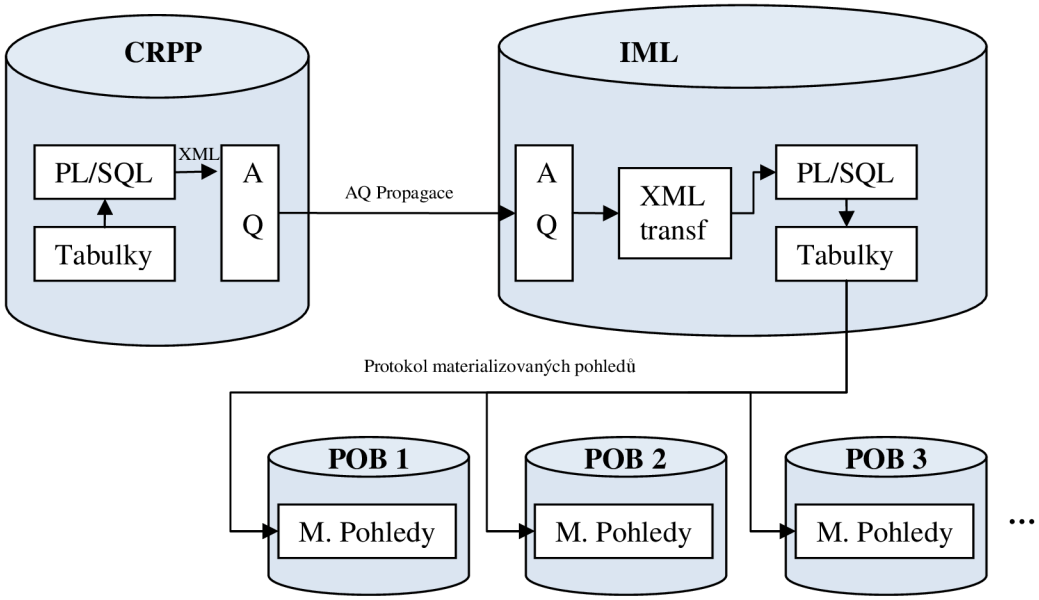
Integrační vrstva IML je článkem mezi centrální databází a pobočkovými databázovými systémy. Existují dvě možnosti, jak realizovat tuto vrstvu. První možností je vytvořit oddělené schéma v centrální databázi, které by provádělo integraci a komunikovalo s pobočkami. Tato varianta by mohla zatěžovat centrální systém a s ohledem na další centralizace ostatních aplikací by musela existovat jedna integrační vrstva pro jeden centralizovaný systém. Proto je IML implementováno jako samostatná databáze, která bude využita i pro další komunikaci a další integrace.

Integrační vrstva je tedy samostatný systém, který tvoří jakousi infrastrukturu mezi jednotlivými databázovými systémy. Pro komunikaci mezi systémy budou v IML definovány databázové linky do všech databází a ostatní databáze budou mít vytvořen databázový link do integrační vrstvy. Pro asynchronní komunikaci budou sloužit fronty Oracle Advanced Queuing (AQ), nad kterými bude fungovat směrování. IML bude mít vstupní a výstupní fronty a podle názvu zprávy, která bude uložena ve frontě, se směrovací aparát rozhodne, kam dotyčnou zprávu přeposlat.

Integrační vrstva bude také sloužit pro shromáždění všech dat ze všech poboček. Do schématu IML se přidají stejné tabulky, které jsou na pobočce, a vytvoří se mezi nimi stejné vazby popsané v kapitole 3.1. Jedinou změnou bude přidání nového sloupce UP, který bude identifikovat, z jaké pobočky záznam pochází. Tento sloupec bude také přidán do primárních klíčů, jelikož interní klíče byly jedinečné pouze v rámci jedné pobočky.

4.2 Replikace dat

Replikace dat je mechanismus, který udržuje aktuální data na více místech. V případě CRPP se jedná o problém, jak pořizovat a měnit údaje o poskytovateli péče v centrální databázi a přitom mít aktuální data na pobočce, jelikož na tyto data jsou vázány jiné aplikace. Oba systémy jsou odlišné a hlavně struktura dat je logicky i fyzicky jiná. Proto musí existovat replikační aparát, který data nejen přenese na pobočku, ale musí také dojít k transformaci přenášených dat. Na **Obrázek 4.1** je znázorněn celý průběh replikace dat z CRPP na jednotlivé pobočky. Proces replikace lze rozdělit do několika fází: vytvoření notifikační zprávy, směrování AQ zpráv, transformace a replikace na pobočky.



Obrázek 4.1: Replikace dat z CRPP na pobočky

4.2.1 Vytváření notifikačních zpráv

První fází replikace dat musí být sestavení notifikační zprávy. CRPP musí informovat pobočky o změně dat, která v centrální databázi proběhla. Pro výčet změn se použije XML dokument, který bude mít totožnou strukturu se strukturou databázových tabulek v CRPP. Nedochází tedy k žádné transformaci.

Sestavení XML zprávy bude probíhat v PL/SQL balíčku, který bude obsahovat procedury pro sestavení XML jednotlivých entit v databázi. Vstupním parametrem bude uživatelsky definovaný datový typ (UDT – User-defined type) rekord dané entity, jež byl popsán v kapitole 0, a druh operace, která byla nad daty provedena. Jednotlivé položky budou opatřeny otevírací a uzavírací značkou a přidány do XML dokumentu. Pro každou entitu z registrační části, smlouvu a přílohu bude existovat zvlášť XML definice XSD (XML Schema Definition). U smlouvy a přílohy bude XML dokument obsahovat kromě položek entity VYKSMLOUVA resp. VYKPRILOHA také podřízené detailové entity VYKS% resp. VYKP%. Tyto detaily budou v XML reprezentovány komplexními položkami. Nyní nastává otázka, kdy volat takto připravené procedury. Jak již bylo uvedeno, CRPP se skládá z tří PL/SQL vrstev a právě tabulková vrstva je ideální pro sestavování notifikačních zpráv. Do jednotlivých procedur INS, UPD a DEL všech balíčků z této vrstvy se vloží volání notifikačních procedur, kde bude jako první parametr použit rekord s daty a za druhý parametr se dosadí druh operace, který je v proceduře prováděn (I – vložení, U – změna, D – smazání).

4.2.2 AQ přenos zpráv

Další fází v replikaci musí být dopravení notifikační zprávy do integrační vrstvy, kde dojde k transformaci. Pro komunikaci mezi Oracle databázemi existuje asynchronní přenos dat – Oracle Advanced Queuing. AQ bude použito i pro komunikaci mezi CRPP a IML.

Struktura AQ zpráv obsahuje mimo řídící položky, potřebné pro fungování interního aparátu AQ, také tzv. payload, který je naplněn uživatelskými daty. Tyto uživatelská data mohou být různých datových typů i UDT. K přenosu zpráv je potřeba několika údajů, jenž korespondují s UDT AQPAYLOAD_T, který zohledňuje obecné použití i pro další komunikaci.

Tabulka 4.1: Struktura UDT - AQPAYLOAD_T

Položka typu	Datový typ	Komentář
NAZEV	V(30)	Název zprávy
ODESILATEL	V(30)	Odesilatel zprávy
PRIJEMCE	V(30)	Příjemce zprávy
TEXT	CLOB	Předávaná textová data (XML, text)
DATA	BLOB	Předávaná binární data (pdf, doc, raw)

XML dokument bude tudíž vložen do atributu TEXT instance objektu AQPAYLOAD_T a ten bude použit jako payload AQ zprávy. AQ zpráva se ještě musí vložit do AQ fronty a může následovat potvrzení transakce – commit. V CRPP tedy probíhají změny dat a vytváření notifikačních zpráv v jedné transakci a nemůže tím pádem dojít k modifikaci dat bez odeslání informace o změně.

Oracle Advanced Queuing podporuje spoustu užitečných funkcí. Jednou z nich je propagace mezi AQ frontami. Propagace je mechanismus, který vyzvedává zprávy z jedné fronty a vloží je do fronty druhé. Tyto fronty mohou být v různých databázích s podmínkou, že zdrojová databáze má definovaný databázový link do databáze cílové. Tímto způsobem bude zajištěn přenos notifikačních zpráv z CRPP do IML. Nad cílovou frontou bude vytvořena callback procedura, která bude provádět dequeue zpráv a podle názvu zprávy bude řídit další zpracování notifikačních zpráv.

4.2.3 Transformace

Příchozí notifikační zprávy jsou ve tvaru centralizovaných dat. Proto je potřeba data transformovat do akceptovatelné podoby pro pobočkové aplikace. Pro snadnější práci s daty bude XML dokument transformován do databázového typu. Převod bude probíhat na databázové úrovni v PL/SQL za použití systémových nástrojů Oracle. Text přijatý z AQ zprávy bude převeden na XMLType a pomocí XPath zpracován.

Naplněný databázový typu bude předán PL/SQL procedurám, které budou provádět operace nad tabulkami v integrační vrstvě. Mimo modifikaci replikovaných dat budou procedury také dohledávat interní klíče, které v CRPP neexistují, a dosazovat je do referenčních vazeb převzatých z pobočkové aplikace.

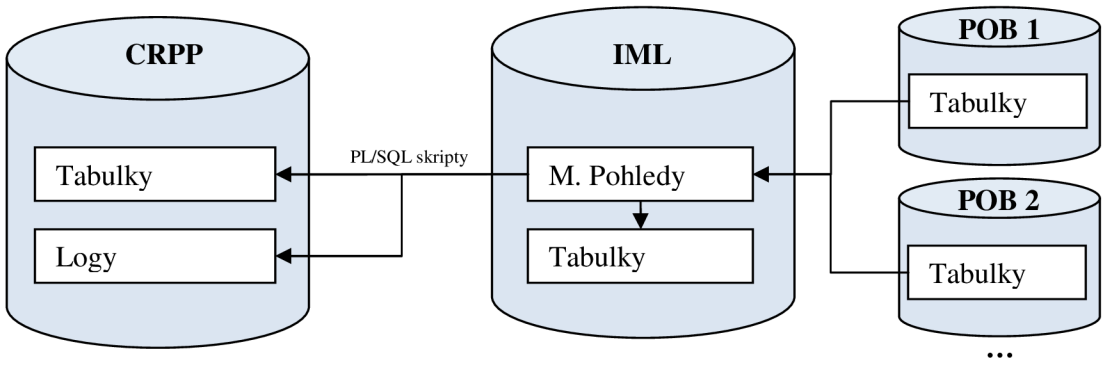
4.2.4 Přenos na pobočky

Posledním krokem v replikaci je přenos dat na pobočky. Všechna data jsou uložena v integrační vrstvě a záznamy na pobočce jsou již neměnné. Nabízí se možnost přistupovat k datům v IML online, pomocí databázového linku. To však přináší značné riziko výpadku sítě a neúměrně by docházelo k zatěžování integrační vrstvy. Vhodnějším řešením jsou materializované pohledy (Materialized Views - MV) na straně pobočky, které si budou udržovat aktuální data z IML pomocí protokolu pro rychlou obnovu (fast refresh). Materializované pohledy vytváří lokální kopie dat a přenos dat se provádí dávkově.

Pro realizaci replikace dat pomocí materializovaných pohledů je potřeba vytvořit logy materializovaných pohledů na straně IML. Logy udržují přehled o změněných datech a umožňují replikovat pouze změněná zdrojová data. Na jednotlivých pobočkách se provede odstranění původních tabulek a vytvoření materializovaných pohledů pro každou entitu. Materializovaný pohled bude obsahovat záznamy ze zdrojové tabulky v IML, které přísluší dané pobočce, k čemuž bude využit sloupec UP. Nad novými objekty se dále vytvoří indexy uvedené v kapitole 3.1 aby okolní aplikace nerozeznaly rozdíl od původních tabulek.

4.3 Migrace dat

Migrace je obecně proces, kdy dochází k jednorázovému přesunu z jednoho místa na druhé. Tato událost se může týkat zvířat, lidí, ale i počítačových programů nebo binárních dat. A právě migrace dat je pro integraci velmi důležitá. Je vyvinut nový centralizovaný databázový systém, který ale neobsahuje žádná data. Důležité informace jsou uloženy na pobočkových databázových systémech. V tomto případě se jedná o informace o poskytovatelích péče, které obsahují několik gigabytů, a ruční zadávání přes formulář je absolutně nemyslitelné. Proto je potřeba vyvinout nástroj, kterým se automaticky přenesou všechny záznamy z pobočkových databází, přetransformují se a vloží do centrální databáze. Postup migrace je znázorněn na Obrázek 4.2. Migrace obsahuje tři kroky: získání dat z poboček, import dat do IML a import dat do CRPP.



Obrázek 4.2: Migrace dat z poboček do IML a CRPP

4.3.1 Získání dat z poboček

Platná data, která potřebujeme přenést do centrálního systému, se nachází na jednotlivých pobočkách. Bylo by možné přistupovat na pobočku pomocí databázových linků přímo z CRPP, ale snadnějším způsobem bude využít připravené infrastruktury v integrační vrstvě. Pro každou pobočku a každou entitu z ní se vyrobí materializovaný pohled bez obnovovacího protokolu. Takto vytvořené materializované pohledy budou sloužit jako zdrojová data pro migraci a zároveň jako centrální záloha původních pobočkových dat. Import do IML se tím také podstatně urychlí, jelikož bude probíhat lokálně.

4.3.2 Import dat do IML

Importem dat do IML je myšleno naplnění tabulek v integrační vrstvě. Na pobočce dojde ke změně tabulek na materializované pohledy a tabulky v IML se stanou zdrojem pro tyto pohledy. Proto musí být tyto tabulky naplněny platnými daty, aby na pobočkách nedošlo ke ztrátě dat.

Proces importu dat do IML bude poměrně nenáročný. Struktura tabulek je shodná se zdrojovými materializovanými pohledy, kde jsou uložena data z poboček. Transfer dat proběhne PL/SQL skripty, které vyberou platná data, provedou kontrolu vnitřních vazeb a vloží data do tabulek v integrační vrstvě. Skripty budou dodržovat pořadí naplňování entit podle vzájemné provázanosti a budou volány s jedním parametrem – číslem pobočky. Podle tohoto čísla se budou vybírat zdrojová data a zároveň se toto číslo bude ukládat do sloupce POB v naplňovaných tabulkách.

4.3.3 Import dat do CRPP

Nejdůležitější částí migrace bude import dat do tabulek v centralizovaném databázovém systému. Zdrojová data budou připravena v integrační vrstvě, a jelikož mají jinou strukturu než cílové tabulky, musí dojít k transformaci dat.

Import bude probíhat pomocí PL/SQL skriptů, která si načtou platná zdrojová data přes databázový link a uloží si je do kolekce stejné struktury. Další kolekce bude mít strukturu cílové entity v CRPP. Mezi těmito kolekcemi proběhne transformace dat, kde se využije mapování sloupců z kapitoly 0. Pro každou entitu bude existovat zvláštní skript se vstupním parametrem – číslo pobočky. Výjimku bude tvořit smlouva a příloha, protože z důvodů zachování vnitřních vazeb se budou migrovat všechny entity podřízené pod smlouvou resp. přílohou. Data z naplněné cílové kolekce se pak vloží do příslušné tabulky. Pokud dojde při ukládání dat k jakékoli výjimce, dotyčný záznam se uloží do logovacích tabulek, které budou mít stejnou strukturu jako je v CRPP, ale nebudou obsahovat žádné integritní omezení a všechny sloupce budou nepovinné. Záznamy, které propadnou do logu, se budou moci analyzovat, opravit a přímo pak přenést do tabulek s platnými daty.

5 Implementace

Následující text popisuje samotnou implementaci systému, který byl navržen v předchozí kapitole. Posloupnost textu odpovídá jednotlivým krokům, které bylo potřeba provést k realizaci datové integrace systému poskytovatelů péče. Mimo slovní popis jsou uváděny i datové struktury nově vytvořených tabulek a ukázky zdrojových kódů, které byly použity při implementaci.

5.1 Vytvoření integrační vrstvy

Prvním krokem pro spojení centrálního systému a decentralizovaných systémů bylo vytvoření integrační vrstvy IML. Ta především slouží jako infrastruktura mezi databázemi a poskytuje data pro jednotlivé pobočky. Proto bylo zapotřebí vytvořit databázové linky a tabulky odpovídající struktuře pobočkové databáze.

5.1.1 Vytvoření databázových linků

V integrační vrstvě existuje poměrně velké množství databázových linků. Proto jsem pro zvýšení přehlednosti a udržovatelnosti vytvořil mechanismus pro správu databázových linků. Všechny údaje potřebné pro vzdálené připojení jsou uloženy v tabulce IMLDBLINK[7], jejíž struktura je popsána v Tabulka 5.1

Tabulka 5.1: Struktura tabulky IMLDBLINK

Název sloupce	Typ	Popis
ID	V(32)	Název databázového linku
HOST	V(256)	Adresa nebo název databázového serveru
PORT	V(6)	Port, na kterém naslouchá databáze
SID	V(50)	Identifikátor systému
UZIVATEL	V(30)	Jméno uživatele pro přístup do databáze
HESLO	V(30)	Heslo uživatele

Pro administraci databázových linků jsem vytvořil balíček IMLDBLINK_ADM, který obsahuje procedury pro vytvoření, smazání, ověření a přetváření databázových linků z tabulky IMLDBLINK. Procedury v administračním balíčku mají vždy jeden parametr, který představuje ID databázového linku. Podle tohoto parametru se vyhledá záznam z tabulky IMLDBLINK a pomocí údajů proběhne příslušná operace s využitím EXECUTE IMMEDIATE pro sestavení a provedení dynamického SQL příkazu. Jako příklad je uvedena procedura vytvářející databázový link. Pro sestavení DDL příkazu je využita konkatenace. Příkaz na vytvoření databázového linku v sobě obsahuje apostrofy, a tudíž se při zápisu musí použít trojitý apostrof (`'''`), aby byl textový řetězec správně interpretován.

[7] Objekty, které slouží pro správu příslušné databáze, obsahují ve svém názvu prefix rovnající se názvu databáze.

```
-- procedura pro vytvoření DB linku
PROCEDURE cre_dblink(p_id IN VARCHAR2)
IS
  -- zaznam z tabulky imldblink
  v_link imldblink%ROWTYPE;
BEGIN
  -- vyber záznam
  SELECT dbl.* INTO v_link
    FROM imldblink dbl
   WHERE id = p_id;
  -- vytvoření linku
  EXECUTE IMMEDIATE
    'CREATE DATABASE LINK '||p_id||' TO '||v_link.uzivatel||
    ' IDENTIFIED BY ' ||v_link.heslo||' USING '||
    '''(DESCRIPTION = (ADDRESS_LIST = (ADDRESS = (PROTOCOL = TCP)
    (HOST = '||v_link.host||')(PORT = '||v_link.port||')))'||
    '(CONNECT_DATA = (SERVER = DEDICATED)(SID = '||
    v_link.sid||')))''';
--
END cre_dblink;
```

Ve všech systémech jsem dále vytvořil nového uživatele USERPROPAG, přes kterého bude realizována veškerá komunikace se vzdálenými systémy. Tento uživatel má implicitně přiděleno právo na připojení a další práva budou poskytnuta později vzhledem ke konkrétnímu systému.

Vytvoření databázové linku pak probíhá následovně.

```
-- naplnění tabulky IMLDBLINK
-- konvence pro jméno linku: systém(číslo UP)_uživatel
INSERT INTO imldblink(id, host, port, sid, uzivatel, heslo) VALUES
('UP53_userpropag','localhost,1521,'UP','userpropag','userpropag');

-- vytvoření databázového linku
BEGIN
  Imldblink_adm.cre_dblink('UP53_userpropag');
END;
/
```

5.1.2 Vytvoření zdrojových tabulek

Dle návrhu poskytuje integrační vrstva data jednotlivým pobočkám. Za tímto účelem musí v IML existovat tabulky, které mají stejnou strukturu jako tabulky na pobočkách, aby nad nimi mohly být sestaveny materializované pohledy, jejichž pomocí budou replikována data na pobočky. Pro vytvoření tabulek jsem využil offline model pobočkové databáze, kterou jsem ovšem musel upravit. Aby byla určena příslušnost jednotlivých záznamů k příslušné pobočce, přidal jsem do každé tabulky nový sloupec UP. O tento sloupec jsem dále rozšířil stávající primární klíče. Záznamy v pobočkových tabulkách jsou jednoznačně identifikovány pomocí interních ID, která se generují z databázových sekvencí. Tyto identifikátory jsou ale bohužel unikátní pouze v rámci jedné pobočky. V kombinaci s přidaným číslem pobočky je zajištěna jednoznačnost v rámci celé republiky.

Některé z pobočkových tabulek neobsahují omezení primárního klíče, což vede k problému s definováním materializovaných pohledů a jejich obnovy. Materializované pohledy sice lze nastavit pro kompletní obnovu, ale tyto pohledy zbytečně zatěžují centrální server a komunikační síť. Proto jsem do tabulek zavedl nové primární klíče podle centrální aplikace CRPP. Tímto omezením se nejen vyřešil problém s materializovanými pohledy, ale kromě toho byla zajištěna i transparentní replikace dat z CRPP. Dále došlo při migraci dat k vyčištění duplicit, které v pobočkové aplikaci existovaly.

Nově vytvořené záznamy v CRPP, jež budou přeneseny do IML, budou v tabulkách reprezentovány novými řádky. Těmto vzniklým záznamům musí být přidělen nový interní identifikátor. Aby byla zachována unikátnost interních klíčů, vytvořil jsem v IML databázové sekvence, jimiž budou nové klíče generovány. Pro korektní generování jsem nastavil počáteční čísla sekvencí na nejvyšší hodnotu v příslušné skupině objektů, které danou sekvenci využívají, a zaokrouhlil ji nahoru. Tím je zajištěna jedinečnost identifikátoru v rámci jedné pobočky.

5.2 Migrace dat

Jedním z cílů této práce je převedení dat z decentralizovaných systémů do centrální databáze. Celý proces se skládá z několika dílčích kroků, které budou podrobně popsány z pohledu implementace. Jelikož je tento proces v rámci celé migrace dat opakován pro každou jednotlivou pobočku, bylo pro mě výhodné zajistit jeho automatizaci.

5.2.1 Získání dat z poboček

Platná zdrojová data pro import se nachází v decentralizovaných databázích. Aby byla zajištěna jejich viditelnost ze vzdálených systémů, musí existovat databázový link směřující do příslušné databáze. V této databázi pak existují tabulky popsané v kapitole 3.1, ve kterých jsou uložena relevantní data. Pro vzdálený přístup k datům v těchto objektech je zapotřebí příslušnému uživateli přidělit oprávnění.

Do databáze je přistupováno z integrační vrstvy, ve které jsou databázové linky již připraveny. Přidělil jsem tedy oprávnění uživateli USERPROPAG ke čtení potřebných tabulek. Data jsou tímto připravena ke vzdálenému čtení. Aby nedocházelo k opakovanému pomalému přístupu přes síť pomocí databázového linku, vytvořil jsem v IML lokální kopie pobočkových dat. Nejelegantnějším způsobem bylo použití materializovaných pohledů, které mají shodnou strukturu s pobočkovými tabulkami a nebudou se v budoucnosti nikdy obnovovat ani přepisovat. Jediným problémem byly tabulky, které obsahují databázový typ LONG. Tento typ není podporován materializovanými podhledy. Proto jsem na pobočce vytvořil pomocnou tabulku, ve které jsem typ LONG nahradil typem VARCHAR2(4000), a převedl do ní data. Pomocná tabulka pak sloužila jako zdroj dat pro materializované pohledy. Výsledkem bylo shromáždění dat ze všech poboček na jediném místě, které může v budoucnu sloužit jako záloha původních dat. Materializované pohledy jsem pojmenoval dle zdrojových dat s připojeným číslem pobočky, ze které data pochází. Pro další použití jsem pohledům přidělil příslušná oprávnění.

```
-- vytvoření MV kopie pro subjekt
-- parametrem &1 je číslo UP, ze kterého jsou data stáhnuta
CREATE MATERIALIZED VIEW subjekt&1 NEVER REFRESH AS
  SELECT * FROM up.subjekt@up&1._userpropag;
```

5.2.2 Import dat do integrační vrstvy

Tabulky v integrační vrstvě slouží jako zdroj pro materializované pohledy na pobočkách. Proto musí být tyto tabulky naplněny platnými daty. Kdyby tomu tak nebylo, při vytvoření materializovaných pohledů na pobočkách by vznikly pohledy prázdné, což by vedlo ke kolapsu pobočkového informačního systému.

Před zavedením replikací jsem proto zmiňované tabulky v IML naplnil pobočkovými daty, které jsou uloženy v materializovaných pohledech s příslušným sufixem. Pro naplnění jsem vytvořil PL/SQL skripty (pro každou tabulku jeden skript). Vstupním parametrem skriptu je číslo pobočky, která je migrována. Podle tohoto čísla jsou data vyhledávána v příslušném materializovaném pohledu a zároveň se číslo pobočky vloží do sloupce UP. Data jsou také filtrována pomocí sloupce DEL, jelikož se přenáší pouze nezrušené záznamy, a dále podle existence nadřízených záznamů. Při vkládání záznamů do tabulek jsem do skriptu přidal zachycení výjimky na duplicitu záznamu, aby nedošlo k zastavení zpracování celé migrace. Duplicita záznamů je bohužel se zavedením nových primárních klíčů nevyhnutelná, ale nepřináší žádnou ztrátu informace. Skript pouze vypíše upozornění na standardní výstup a pokračuje v importu dál.

Pobočková data byla zadávána do systému s poněkud benevolentními kontrolami. Detail smlouvy či přílohy mohl například časově přesahovat smlouvu, resp. přílohu. Navíc detaily, které jsou na pobočce vázány na pracoviště, jsou v CRPP vázány na přílohu smlouvy. To znamená, že jeden detail přesahuje platností několik příloh. Tyto odlišnosti mohou způsobit nekompatibilitu dat při replikaci z CRPP do IML, a proto musejí být odstraněny. Migrační skripty nabízí možnost upravit data při vkládání takovým způsobem, aby odpovídala datům v CRPP. Do skriptů jsem přidal kontroly na časový přesah detailů. Pokud měl detail začátek platnosti menší než začátek platnosti smlouvy (přílohy), byla platnost nastavena na začátek nadřízeného záznamu. Obdobným způsobem byl upraven i konec platnosti. Při importu detailů pracoviště jsem nevkládal záznamy přímo, ale dohledával jsem přílohy pro dané pracoviště. K těmto přílohám jsem pak časově přiřazoval načtené detaily. Tím vzniklo z jednoho detailu několik záznamů, které na sebe časově navazují a korespondují s daty uloženými v CRPP.

```
-- skript pro import výkonů do IML
-- parametrem je číslo UP
DECLARE
  cnt  NUMBER;
  v_od DATE;
  v_do DATE;
BEGIN
  -- dohledání všech platných výkonů
  FOR par IN (SELECT smlid,pracid,&1 up,kod,platod,platdo,kodon
                    ,typ,del,uzivatel,zmeneno
                FROM vykon&1
               WHERE NVL(del, 'N') = 'N'
                 AND smlid IN (SELECT id FROM smlouva WHERE up = &1)
                 AND pracid IN (SELECT id FROM prac WHERE up = &1)
             )
  LOOP
    -- pro všechny přílohy
    FOR pri IN (SELECT platod,platdo
                  FROM priloha
                 WHERE smlid  = par.smlid
                   AND pracid = par.pracid )
    LOOP
```

```
      BEGIN
        -- úprava platnosti
        IF par.platod < pri.platod THEN
          v_od := pri.platod;
        ELSE
          v_od := par.platod;
        END IF;
        IF par.platdo > pri.platdo THEN
          v_do := pri.platdo;
        ELSE
          v_do := par.platdo;
        END IF;
        --
        -- filtr na nesmyslné platnosti
        IF v_od < v_do THEN
          -- vložení záznamu do tabulky
          INSERT INTO vykon (smlid,pracid,up,kod,platod,platdo,
                             kododn,typ,del,uzivatel,zmeneno
                            )
          VALUES (par.smlid,par.pracid,par.up,par.kod,v_od,v_do,
                  par.kododn,par.typ,par.del,par.uzivatel,par.zmeneno
          );
        END IF;
        -- odchycení duplicity
        EXCEPTION
          WHEN DUP_VAL_ON_INDEX THEN
            dbms_output.put_line('Duplicita u vykonu smlid: '||par.smlid||
                                 ' pracid: '||par.pracid||' kod: '||par.kod);
      END;
    END LOOP;
  END LOOP;
  -- počet vložených záznamů
  SELECT COUNT(*) INTO cnt FROM vykon WHERE up='&1';
  dbms_output.put_line(chr(10)||cnt||' radku vytvoreno');
END;
/
```

5.2.3 Import dat do CRPP

Migrace do centrálního CRPP byla poněkud složitější, jelikož struktura cílových tabulek je jiná než zdrojová data. Zdrojová data jsou uložena v IML a přístup k nim je zajištěn přes databázový link. Tato data musí být přetransformována do patřičného tvaru a následně vložena do tabulek v CRPP. Pro import jsem vytvořil PL/SQL skripty, které vždy zpracovávají data jedné entity. Výjimku tvoří skript pro smlouvu a přílohu, který zároveň provede import i podřízených detailů. Toto spojení bylo vytvořeno kvůli vazbám mezi detailem a nadřízeným záznamem. Ve zdrojových datech neexistuje vazba mezi přílohou a detaily pracoviště, ale v CRPP je tato návaznost vyžadována. Proto se při zpracování přílohy nabízí přímo zpracovávat i detaily, které jsou závislé na stejném pracovišti. Dalším problémem u smluv a příloh je jejich číslování. Číslo smlouvy musí být v CRPP unikátní. Na pobočce muselo být taktéž unikátní, což ale neplatí v rámci všech poboček. Číslo přílohy je unikátní v rámci jedné smlouvy. Aby toto bylo dodrženo, ke každému číslu smlouvy jsem přidal sufix složený z mezery, mřížky a čísla pobočky (např. 111 #53). K číslu přílohy jsem přidal mřížku a IČP pracoviště (např. 2 #53001001).

Samotný proces importu je prováděn za pomoci PL/SQL kolekcí. Jedna kolekce slouží pro uchování načtených zdrojových dat a druhá kolekce, jež má strukturu CRPP entit, uchovává záznamy určené k uložení do databáze. Plnění zdrojové kolekce je prováděno pomocí BULK COLLECT, což

je nejrychlejší způsob naplnění. V cyklu přes všechny záznamy zdrojové kolekce se postupně plní kolekce cílová, přičemž dochází k doplňování potřebných údajů. Nejdůležitějším krokem je korektní doplnění externích identifikátorů, které ve zdrojových datech chybí. Příloha například neobsahuje číslo smlouvy nebo číslo pracoviště IČP, ale pouze jejich interní identifikátory. Pro optimální zpracování jsem vytvořil PL/SQL kolekce, které obsahují externí identifikátor na indexu, jenž je roven identifikátoru vnitřnímu. Kolekce se naplní jednou a zamezí opakovanému přístupu do vzdálené databáze, ke kterému by docházelo při zpracování každého záznamu. Dalším problémem by mohla být chybějící hodnota sloupce, který je v CRPP vyžadován. Pokud není hodnota tohoto sloupce stěžejní, doplnil jsem do sloupce hodnotu „-“ za využití funkce NVL[8].

Uložení do tabulek jsem realizoval pomocí instrukce FORALL, která zajistí rychlé uložení obsahu kolekce do cílové tabulky. Abych eliminoval případné chyby při ukládání vzniklé děrami v kolekci, použil jsem výraz FORALL i IN INDICES OF. Jestliže nastane chyba při ukládání dat, je výjimka uložena do pole výjimek a zpracování pokračuje dál. Na konci zpracování jsou vzniklé výjimky procházeny, vypisovány na standardní výstup a ukládány do logovacích tabulek. Pro tyto případy jsem vytvořil zvláštní schéma v CRPP, které obsahuje stejné tabulky, do nichž se migrují data. Těmto tabulkám jsem zrušil všechna omezení a ke každé jsem přidal tři nové sloupce: UP - číslo pobočky ze které záznam pochází, CHYBA - text chyby, která nastala a VLOZENO s časem vzniku chyby. Takto uchované záznamy budou sloužit pro další analýzu, proč vnikla daná chyba, a po případné opravě budou data přichystána ke snadnému přesunu do CRPP.

```
-- skript pro import příloh a detailu do CRPP
DECLARE
  -- deklarace typu kolekce
  TYPE t_src_tab     IS TABLE OF iml.priloha&1.@iml_userpropag%ROWTYPE;
  TYPE t_dest_tab    IS TABLE OF vykpriloha%ROWTYPE;
  TYPE t_icz_pole    IS TABLE OF iml.subjekt&1..icz@iml_userpropag%TYPE
    INDEX BY BINARY_INTEGER;
  TYPE t_sub_tab     IS TABLE OF iml.subjekt&1.@iml_userpropag%ROWTYPE;
  TYPE t_src_v_tab   IS TABLE OF iml.vykon&1.@iml_userpropag%ROWTYPE;
  TYPE t_dest_v_tab  IS TABLE OF vykpvykon%ROWTYPE;
  …
  -- kolekce
  src_tab    t_src_tab;
  dest_tab   t_dest_tab := t_dest_tab();
  icz_pole   t_icz_pole;
  sub_tab    t_sub_tab;
  chyby EXCEPTION;
  PRAGMA EXCEPTION_INIT(chyby, -24381);
BEGIN
  -- načtení zdrojových dat
  SELECT * BULK COLLECT INTO src_tab
    FROM iml.priloha&1.@iml_userpropag
   WHERE NVL(del,'N') = 'N';
  --
  -- načtení subjektů
  SELECT * BULK COLLECT INTO sub_tab
    FROM iml.subjekt&1.@iml_userpropag;
  -- prevod internich ID na ICZ
  FOR i IN 1 .. sub_tab.COUNT LOOP
    icz_pole(sub_tab(i).id) := sub_tab(i).icz;
  END LOOP;
```

[8] NVL je systémová funkce, která má dva vstupní parametry. Pokud je první parametr roven NULL, je vrácen parametr druhý. V opačném případě je vrácen parametr první.

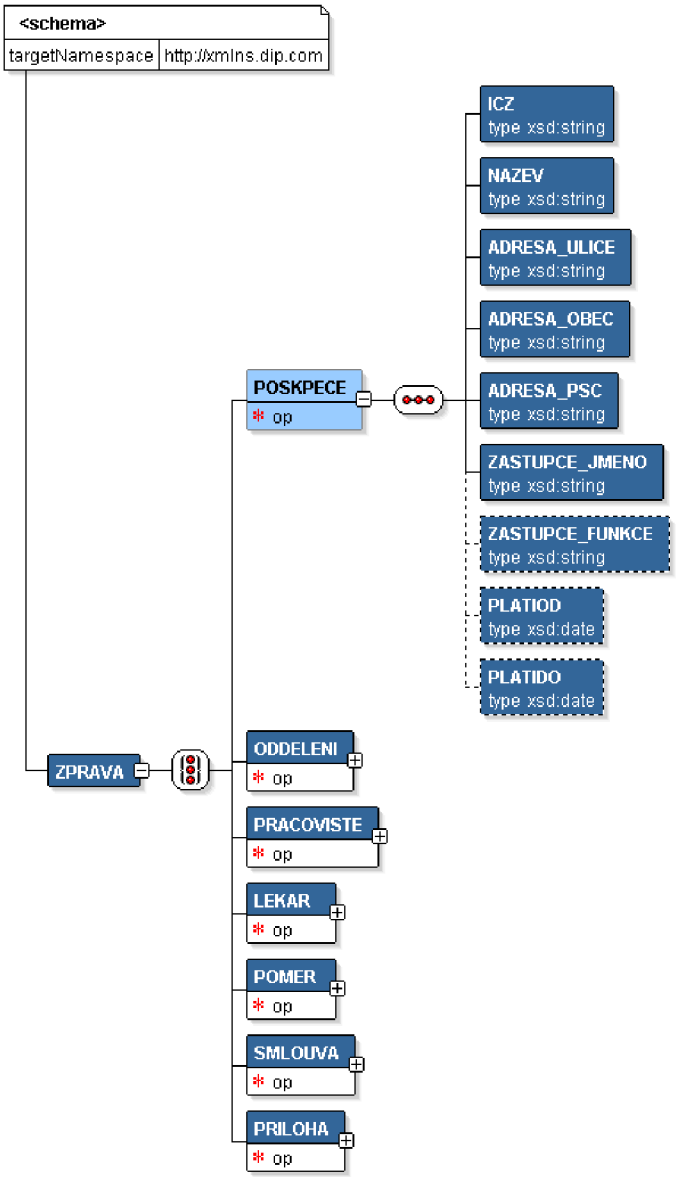
```
  -- příprava kolekce
  dest_tab.EXTEND(src_tab.COUNT);
  i := src_tab.FIRST;
  -- transformace dat pro každou přílohu
  WHILE i IS NOT NULL LOOP
    -- dohledání smlouvy
    SELECT smlouva,platod,platdo,subid INTO v_cislos,v_od,v_do,v_subid
      FROM iml.smlouva&1.@iml_userpropag
     WHERE id = src_tab(i).smlid;
    …
    -- hlavicka prilohy
    dest_tab(i).cislo:=src_tab(i).cislo||'#'||icp_pole(src_tab(i).pracid);
    dest_tab(i).cislos := v_cislos||' #&1';
    dest_tab(i).prityp := src_tab(i).typ;
    dest_tab(i).plati_od := v_od;
    dest_tab(i).plati_do := v_do;
    dest_tab(i).icz      := icz_pole(v_subid); -- doplnění dle pole icz
    …
    -- zpracování detailů
    -- načtení výkonů pro danou smlouvu a pracoviště
    SELECT * BULK COLLECT INTO src_v_tab
      FROM iml.vykon&1.@iml_userpropag
     WHERE smlid = src_tab(i).smlid
       AND pracid = src_tab(i).pracid
       AND NVL(del,'N') = 'N';
    -- zpracování do cílové kolekce
    …
    -- iterace
    i := src_tab.NEXT(i);
  END LOOP;
  -- zápis do databáze
  BEGIN
    FORALL i IN INDICES OF dest_tab SAVE EXCEPTIONS
      INSERT INTO vykpriloha values dest_tab(i);
  EXCEPTION
    WHEN chyby THEN
      FOR i IN 1..SQL%BULK_EXCEPTIONS.COUNT LOOP
        -- zápis do logu
        migracelogs.ins.priloha(dest_tab(SQL%BULK_EXCEPTIONS(i).
          error_index),SQLERRM(-SQL%BULK_EXCEPTIONS(i).ERROR_CODE),&1);
        dbms_output.put_line('CHYBA: ' || SQLERRM(-SQL%BULK_EXCEPTIONS(i)
          .ERROR_CODE));
      END LOOP;
  END;
  …
END;
```

5.3 Replikace dat

Druhou částí této práce je vytvoření replikačního aparátu, který bude udržovat aktuální stav dat ve všech pobočkových databázích. Data, která se pořídí nebo modifikují v centrálním CRPP, se musí promítnout v decentralizovaných systémech. Tato kapitola popisuje implementaci jednotlivých kroků z návrhu v kapitole 4.2.

5.3.1 Vytváření notifikačních zpráv

Notifikační zprávy jsou zprávy, které oznamují, že došlo k nějaké skutečnosti. V této práci mají notifikační zprávy účel sdělit integrační vrstvě, že došlo ke změně dat v CRPP. Kromě sdělení také přináší do IML data k zapracování do tabulek. Data jsou mezi systémy přenášena ve formátu XML. Pro jasnou syntaxi XML jsem vytvořil definiční schéma (XSD). Kořenovým elementem notifikační zprávy je ZPRAVA, která obsahuje elementy, reprezentující entity v CRPP s atributem OP, který nese informaci o operaci provedené nad daty. Tyto elementy jsou komplexního typu a obsahují dílčí elementy odpovídající sloupcům jednotlivých tabulek (smlouva a příloha opět zastřešuje všechny své detaily z důvodu vnitřních vazeb). Pro přehlednost a transparentnost jsem názvy elementů ponechal shodné s názvy sloupců v tabulkách. Vizuální zobrazení XSD je znázorněno na Obrázek 5.1.



Obrázek 5.1: XSD notifikační zprávy

Sestavení XML

Pro sestavování XML zpráv jsem v CRPP zavedl tři balíčky, které vytváří XML dokumenty nad tabulkami s daným prefixem (reg, vyks a vykp). Každá tabulka má připravenou svou proceduru, která na základě vstupního parametru, jímž je ROWTYPE dané tabulky, připraví XML a uloží jej do globální proměnné typu CLOB. Proměnná je globální, jelikož jedna XML zpráva může být složená z několika zdrojových tabulek. Přílohy a smlouvy se notifikují jako celek, se všemi měněnými detaily. Ve stejném pořadí, jak se data ukládají do databáze, XML postupně narůstá. Nejprve se vytvoří hlavička s daty smlouvy, pak následují informace o výdajových fondech či bankovním spojení. Ukončení zprávy je realizováno procedurou ODESLAT, která XML uzavře, zavolá proceduru ENQ_ZPRAVA (popsána v kapitole 5.3.2) a vyčistí globální proměnnou.

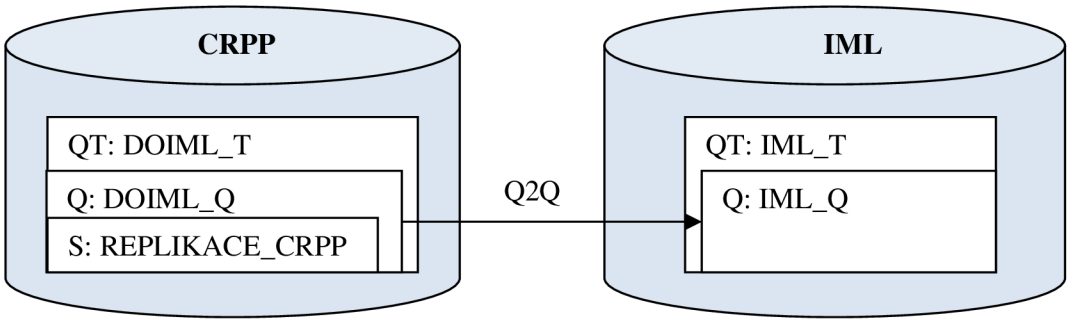
Architektura balíčků CRPP umožňuje pracovat se záznamy tabulek jako s PL/SQL kolekcemi a záznamy. Tabulky jsou modifikovány pouze touto cestou. Této skutečnosti jsem využil a do nejnižší vrstvy balíčků, které provádí vkládání a modifikaci záznamů, jsem přidal volání replikačních procedur. Tím je zajištěno, že pro všechny aplikační změny budou vytvářeny XML notifikační změny v jedné transakci. Navíc existuje jistá volnost pro přímé zásahy do databáze, aniž by vznikaly neúplné či nevalidní notifikační zprávy.

Při sestavování XML se dodržují pravidla daná navrhnutým schématem. Pokud by element měl být prázdný a zároveň je jiného typu než textový řetězec, je při zpracování vyloučen. Notifikační XML zprávy musí být vždy validní, aby nedošlo k chybnému zpracování v integrační vrstvě.

```
-- složeni XML pro poskytovatele péče
  PROCEDURE poskpece( p_op  IN VARCHAR2
                     ,p_rec IN regposkpece%ROWTYPE
                     ,p_od  IN DATE DEFAULT NULL
                     ,p_do  IN DATE DEFAULT NULL
  )
  IS
  BEGIN
    -- začátek s zprávy s xml hlavičkou a elementem <ZPRAVA>
    start_xml;
    g_zprava := g_zprava
      ||'<POSKPECE op="'||p_op||'">'
      ||'<ICZ>' ||p_rec.icz|| '</ICZ>'
      ||'<NAZEV>'||p_rec.nazev||'</NAZEV>'
      ||'<ADRESA_ULICE>'||p_rec.adresa_ulice||'</ADRESA_ULICE>'
      ||'<ADRESA_OBEC>'||p_rec.adresa_obec||'</ADRESA_OBEC>'
      ||'<ADRESA_PSC>'||p_rec.adresa_psc||'</ADRESA_PSC>'
      ||'<ZASTUPCE_JMENO>'||p_rec.zastupce_jmeno||'</ZASTUPCE_JMENO>'
      ||'<ZASTUPCE_FUNKCE>'||p_rec.zastupce_funkce||'</ZASTUPCE_FUNKCE>';
    IF p_od IS NOT NULL THEN
      g_zprava := g_zprava ||'<PLATOD>'||xsd_date(p_od)||'</PLATOD>';
    END IF;
    IF p_od IS NOT NULL THEN
      g_zprava := g_zprava ||'<PLATDO>'||xsd_date(p_do)||'</PLATDO>';
    END IF;
    g_zprava := g_zprava ||'</POSKPECE>';
  END poskpece;
```

5.3.2 AQ přenos zpráv

Přenos notifikačních zpráv z CRPP do IML je zajištěno pomocí AQ zpráv. Pro sestavení funkčního aparátu jsem musel vytvořit databázový typ AQPAYLOAD_T navržený v kapitole 4.2.2, AQ fronty s frontovými tabulkami v obou databázích a nastavit propagaci Q2Q zpráv z CRPP do IML. Celá AQ komunikace je znázorněna na Obrázek 5.2 i s podpisovatelem (subscriber) definujícím vzdáleného konzumenta.



Obrázek 5.2: Schéma AQ komunikace

Enqueue zpráv

V CRPP jsem pro účel vkládání zpráv do front vytvořil proceduru, která je obecně využitelná pro enqueue zpráv do front. Procedura má čtyři parametry, které určují, do jaké fronty se zprávy budou ukládat, jaký je příjemce a název vkládané zprávy a samotný XML dokument. Pro zajištění bezpečnosti při přenosu zpráv jsem zavedl šifrování obsahu AQ zprávy. Vstupní XML ve formátu CLOB zašifruji pomocí systémové procedury DBMS_CRYPTO.ENCRYPT a výsledný BLOB vložím do položky DATA typu AQPAYLOAD_T. Jako šifrovací algoritmus jsem zvolil AES s náhodně vygenerovaným klíčem.

```
CREATE OR REPLACE PROCEDURE enq_zprava(p_fronta   IN VARCHAR2
                                      ,p_nazev    IN VARCHAR2
                                      ,p_prijemce IN VARCHAR2
                                      ,p_xml      IN CLOB
)
IS
  v_enq_opt    dbms_aq.enqueue_options_t;
  v_msg_prop   dbms_aq.message_properties_t;
  v_msg_id     RAW(16);
  v_zprava     AQPAYLOAD_T := AQPAYLOAD_T(null,null,null,null,null);
  v_key        RAW(32) := HEXTORAW(…);
  v_blob       BLOB;
BEGIN
  v_zprava.nazev := p_nazev;
  v_zprava.odesilatel := 'CRPP';
  v_zprava.prijemce := p_prijemce;
  DBMS_LOB.CREATETEMPORARY(v_blob,true);
  -- sifrovani
  dbms_crypto.encrypt(v_blob,p_xml,dbms_crypto.AES_CBC_PKCS5,v_key);
  --
  v_zprava.data := v_blob;
  --
  DBMS_AQ.ENQUEUE( queue_name          =>  p_fronta
                  ,enqueue_options     =>  v_enq_opt
                  ,message_properties  =>  v_msg_prop
```

```
                  ,payload             =>  v_zprava
                  ,msgid               =>  v_msg_id);
END enq_zprava;
/
```

5.3.3 Transformace dat

Notifikační zprávy z CRPP se automaticky pomocí propagace přenesou do fronty IML_Q v integrační vrstvě. Pro zpracování takto příchozích zpráv jsem vytvořil proceduru, která provádí nad frontou dequeue a dešifruje obsah zprávy podle symetrického klíče, který sdílí s CRPP. Podle názvu zprávy pak dochází k předání získaného XML příslušné proceduře, jež zajistí transformaci XML. Aby bylo zpracování plně automatické, zaregistroval jsem tuto proceduru jako callback nad frontou IML_Q.

Transformace XML

Data, která se musejí zapracovat do tabulek v IML, jsou uložena v XML dokumentu. To zaručuje univerzální interpretaci přenášených dat, ale XML je stále pouhý text. Pro další zpracování notifikačních zpráv jsem vytvořil databázové typy, které odpovídají struktuře dat CRPP, a balíčky s procedurami, jež budou tyto typy plnit daty získanými z XML dokumentů. Vstupním parametrem procedur je XML zpráva ve formátu CLOB. Pro získání dat z XML jsem zvolil XMLTABLE. XMLTABLE je systémová SQL/XML funkce, která z XML vytvoří pseudo tabulku, naplněnou pomocí XPath, nad níž lze provádět běžné dotazy. Takto naplněné databázové typy jsou předány dalšímu balíčku procedur, které na základě operace provedené v CRPP, provádějí modifikaci dat v tabulkách v IML.

```
-- převod XML na typ POSKPECE_REC_T
PROCEDURE poskpece(p_xml IN OUT NOCOPY CLOB)
IS
  v_rec POSKPECE_REC_T := POSKPECE_REC_T(null,null,null,…);
    v_id NUMBER;
    v_op VARCHAR2(1);
  BEGIN
    -- evidence
    vloz_zaznam('POSKPECE',p_xml,v_id);
    --
    -- naplnění typu
    SELECT
        icz,nazev,adresa_ulice,…
      INTO
        v_rec.icz,v_rec.nazev,v_rec.adresa_ulice,…
      FROM XMLTABLE( XMLNAMESPACES(DEFAULT 'http://xmlns.dip.com')
        , 'for $i in /ZPRAVA/POSKPECE return $i'
          PASSING XMLType(p_xml)
        COLUMNS
            icz             NUMBER(12)          PATH 'ICZ'
          , nazev           VARCHAR2(100 BYTE)  PATH 'NAZEV'
          , adresa_ulice    VARCHAR2(100 BYTE)  PATH 'ADRESA_ULICE'
          …);
    --
    -- zapracování do tabulky
    replikace_reg.poskpece(v_op,v_rec);
END poskpece;
```

Evidence replikačních zpráv

Za účelem sledování a kontrolování procesu replikace jsem v integrační vrstvě vytvořil dvě logovací tabulky. Do jedné se ukládají všechny zprávy, které z CRPP přišly. Do druhé se ukládají chyby, které vznikly při transformaci XML a modifikaci dat v databázových tabulkách. Struktura s popisem sloupců je popsána v následujících tabulkách.

Tabulka 5.2: Struktura tabulky REPLIKACEEVID

Název sloupce	Typ	Popis
ID	N	Interní identifikátor
TYP	V (30)	Název notifikační zprávy
STAV	V (1)	Výsledek zpracování (C – úspěšně, E – s chybou)
XML	CLOB	XML notifikační zpráva
VLOZENO	D	Čas vložení záznamu

Tabulka 5.3: Struktura tabulky REPLIKACECHYBY

Název sloupce	Typ	Popis
EVID_ID	N	Vazba na záznam v evidenci
VLOZENO	D	Čas vložení chyby
CHYBA	V(1024)	Kód a text chyby

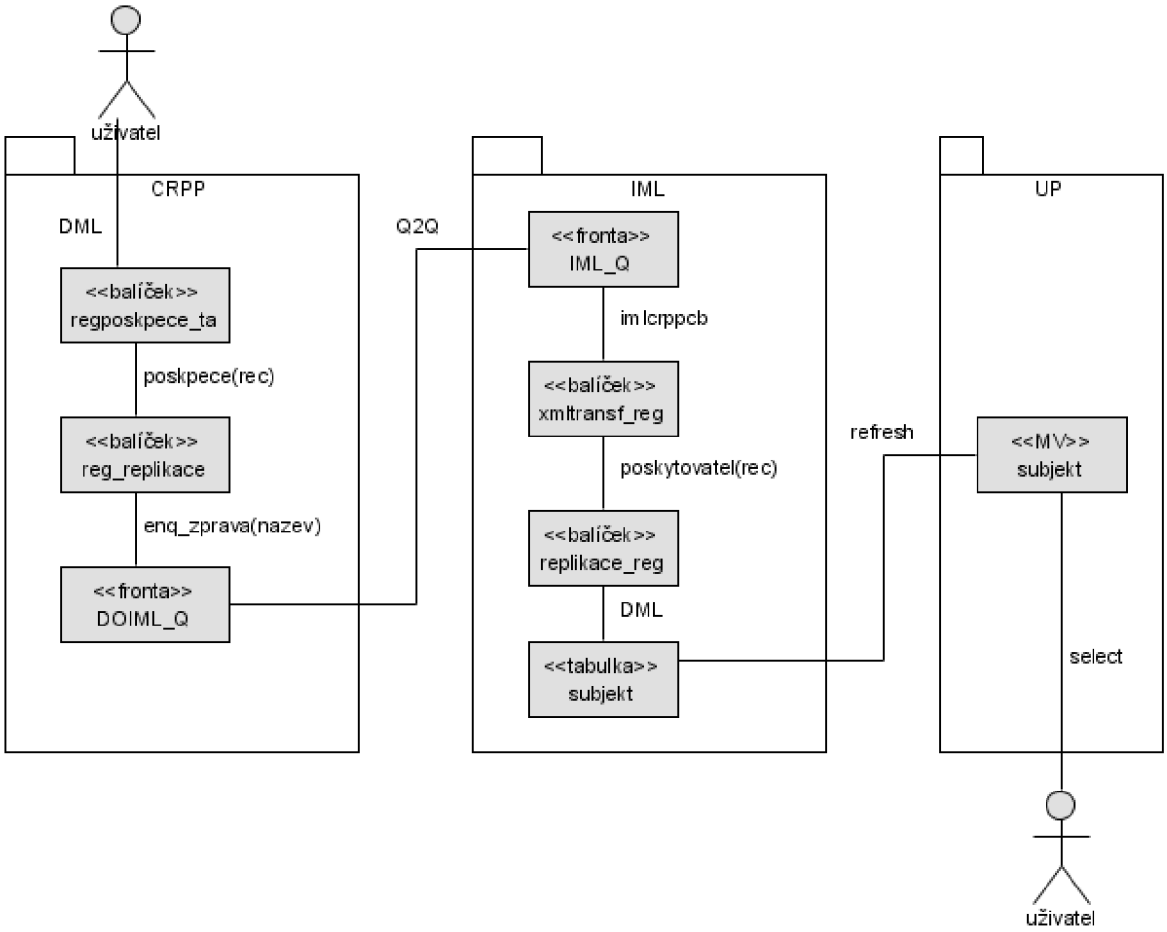
5.3.4 Přenos dat na pobočky

Posledním krokem replikace dat z CRPP je přenesení změn na jednotlivé pobočky. Dle návrhu je v této fázi použita technologie materializovaných pohledů, které si pravidelně obnovují stav z IML pomocí databázového linku. Pro realizaci nasazení materializovaných pohledů jsem v IML nad všemi tabulkami vytvořil mlogy a nastavil jim oprávnění pro uživatele USERPROPAG.

Samotná instalace na pobočce probíhala v několika krocích. Aby mohly být vytvořeny materializované pohledy, jež by nahradily původní tabulky, musel jsem původní objekty odstranit. Nejprve jsem tabulkám odstranil všechny integritní omezení, primární klíče a indexy. Vytvořil jsem skripty, které automaticky vyhledají příslušné omezení/indexy v DBA_CONSTRAINTS/ DBA_INDEXES a seskládají skripty pro jejich vymazání. Po vyčištění omezení jsem stávající tabulky přejmenoval přidáním sufixu „_stare“. Tím jsem uvolnil jméno pro nové materializované pohledy a zároveň vytvořil lokální kopii původních dat.

Vytvoření materializovaných pohledů jsem vložil do jednoho anonymního skriptu, který přijímá jeden vstupní parametr, jímž je číslo právě instalované pobočky. Toto číslo se použije v podmínce deklarace materializovaného pohledu, aby pohled zahrnoval pouze data ze své pobočky. Všechny materializované pohledy jsou obnovovány inkrementálně podle změn na primárních klíčích zdrojových tabulek. Nad materializovanými pohledy jsem vytvořil jednu skupinu pro obnovu, která bude aktualizována v pravidelných časových intervalech.

```
-- vytvoření MV subjektu
CREATE MATERIALIZED VIEW subjekt REFRESH FAST WITH PRIMARY KEY ON DEMAND
AS
  SELECT * FROM iml.subjekt@iml_userpropag WHERE up = &1;
```

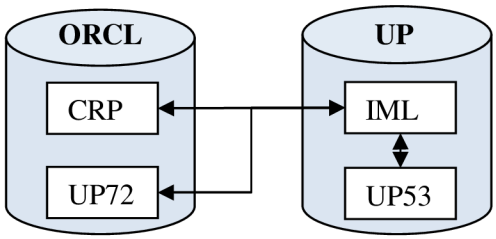


Obrázek 5.3: Diagram komunikace při replikaci dat

6 Testování

Testování probíhalo na osobním počítači, na kterém běžely dvě nezávislé databáze Oracle (ORCL a UP). V jedné databázi jsem vytvořil uživatele CRPP, který představuje centrální aplikaci, a uživatele UP72, což je jedna z decentralizovaných poboček. V druhé databázi jsem vytvořil uživatele IML s integrační vrstvou a uživatele UP53. Toto rozdělení (Obrázek 6.1) by mělo dostatečně simulovat distribuovaný systém databází, které spolu komunikují pomocí databázových linků.

Jednotlivým uživatelům jsem vyrobil potřebné tabulky a pobočkovým databázím jsem naplnil testovací data, která odpovídají menšímu okresu – přibližně 550 poskytovatelů péče (celkově cca 40000 záznamů).



Obrázek 6.1: Schéma testovacího prostředí

6.1 Testování migrace

Pro otestování procesu importu dat z poboček do IML a CRPP jsem zpustil importní skripty pro pobočku 53. Migrace dat do IML trvala přibližně 30 vteřin[9], kde časově nejnáročnější bylo vytvoření lokální kopie pobočkových dat v integrační vrstvě. Záznamy se v tabulkách vytvořily a dle očekávání nastaly chyby v duplicitě a porušení referenčních vazeb v důsledku zrušení nadřízených záznamů.

Import dat do CRPP běžel přibližně 20 vteřin. Tři čtvrtiny času zabral import příloh s detaily, které ovšem zabírají 85% importovaných dat. Při importu nastalo několik desítek chyb, které se korektně vypsaly na standardní výstup a jednotlivé záznamy byly uloženy do logovacích tabulek. Jednalo se o chyby:

Tabulka 6.1: Seznam chyb při importu dat do CRPP

Kód chyby	Text chyby	Vysvětlení
ORA-00001	unique constraint (.) violated	Duplicitní záznamy v pobočkové databázi
ORA-02291	integrity constraint (.) violated - parent key not found	Nadřízený záznam byl zrušen, ale podřízený nikoli. V CRPP neexistuje příslušné ICL či ICP.
ORA-01400	cannot insert NULL into ()	Zdrojové záznamy neměly vyplněny požadované položky.

Zaznamenané chyby jsem zanalyzoval a došel k závěru, že nejsou chybami procesu migrace, ale jsou způsobeny nekonzistentními zdrojovými daty. Tyto chyby naopak vyčistily importovaná data o nesmyslné údaje. Korektnost migrace jsem dále ověřil na počtech řádků jednotlivých tabulek v IML a CRPP. Za použití dvou různých postupů jsem neimportovat stejný počet záznamů.

[9] Všechny časy jsou zkresleny testováním na lokálních databázích. V reálném distribuovaném systému, který bude komunikovat přes internet, bude latence poněkud vyšší.

6.2 Testování replikace

Za účelem testování replikace jsem v CRPP vytvořil makety balíčků, které simulují aplikační logiku systému, a jejich spouštěcí skripty. Tyto skripty jsem postupně spouštěl a sledoval výsledky v integrační vrstvě. Propagace notifikačních zpráv mezi CRPP a IML je nastavena s minimálním zpožděním, takže změny v IML probíhají v reálném čase. Pomocí těchto testů jsem postupně opravoval chybné procedury, až bylo dosaženo zreplikování celého zařízení se všemi podřízenými náležitostmi.

```
-- příklad procedury pro vložení nového zařízení
PROCEDURE ins(p_rec IN regposkpece%ROWTYPE)
IS
BEGIN
  INSERT INTO regposkpece(
    icz
   ,nazev
   …
  )
  VALUES(
    p_rec.icz
   ,p_rec.nazev
   …
  );
  -- vytvoření notifikační zprávy
  reg_replikace.poskpece(reg_replikace.vc_oper_i,p_rec,
    ROUND(SYSDATE,'MONTH'),vc_max_date);
  reg_replikace.odeslat(reg_replikace.vc_nazev_poskpece);
END ins;
```

Při testování přenosu dat z IML do jednotlivých pobočkových databází jsem nastavil frekvenci obnovování materializovaných pohledů na jednu minutu. Databázový job pravidelně spouští aktualizaci materializovaných pohledů a změny v IML se spolehlivě promítají na pobočky.

6.3 Úzká místa systému

Replikace dat z CRPP na pobočkové databáze je distribuovaný systém a jako takový je náchylný na výpadky síťového připojení nebo dokonce na výpadky samotných databází. Pokud je vzdálená databáze nedostupná, databázový link je nefunkční a hlásí výjimku systému. To může způsobovat přerušení automatického zpracování, kterého je využito v replikaci. Jedná se jednak o propagaci AQ zpráv mezi CRPP a IML a automatickou obnovu materializovaných pohledů na pobočkách. Delší výpadek databázového linku způsobí zastavení plánovače nad odchozí frontou v CRPP. Na pobočce dojde k vypnutí databázového jobu. Tyto databázové prostředky se po obnově prostředí musejí opět ručně nastartovat. Proto by bylo vhodné tato úzká místa monitorovat vhodným monitorovacím nástrojem, který by informoval administrátory o vzniklé chybě. Využity by mohly být například Oracle BAM (Business Activity Monitoring) či Oracle Enterprise Manager Grid Control.

Dalším problém tohoto replikačního aparátu je jeho udržovatelnost. Pokud dojde v centrálním systému ke změně v datovém modelu a bude vznesen požadavek, aby byla změna provedena i na pobočce, bude se muset upravit celý systém, tzn. úprava notifikačních zpráv a jejich zpracování, zásah do datového modelu integrační vrstvy a úprava materializovaných pohledů. To znamená víc pracnosti a potenciálně větší prostor pro vytvoření chyb.

7 Závěr

Cílem této diplomové práce bylo seznámit čtenáře se systémem registru poskytovatelů péče a zrealizovat datovou integraci tohoto systému. Na základě prostudování technologií firmy Oracle, jež umožňují vybudování distribuovaných systémů, jsem vytvořil návrh systému, kterým je možné vytvořit replikační mechanismus mezi centrální databází CRPP a pobočkovými decentralizovanými aplikacemi. Dle návrhu datové integrace jsem zrealizoval implementaci importu dat do centrálního systému a replikace změn na decentralizované databázové systémy. Na závěr jsem provedl sérii integračních testů, které ověřily, že celý integrační aparát je funkční a plně automatizovaný. Všech cílů, jež byly vytyčeny zadáním práce, bylo dosaženo.

Tato práce především poskytuje jedno integrační řešení jednoho systému. V budoucnu ale může sloužit jako inspirace pro zpracování dalších projektů s podobným zaměřením. Navíc přináší pohled do problematiky distribuovaných databází, se kterými se student při programování relačních databází často nesetká.

Vytvořený replikační aparát je kompletně funkční, ale v budoucnu jej lze dále vylepšovat. Především by se jednalo o optimalizaci SQL dotazů a PL/SQL procedur. Další fází vývoje systému by mohly být automatické kontroly systémových prostředků se zotavením.

Seznam zkratek

CRPP	Centrální registr poskytovatelů péče
IML	Integrační vrstva
UP	Územní pracoviště, pobočka
MV	Materializovaný pohled
AQ	Advanced Queuing
XML	Extensible Markup Language
XSD	XML Schema Definition
PL/SQL	Procedural Language/Structured Query Language
POSKPECE	Poskytovatel péče
SML	Smlouva
PRI	Příloha
PRAC	Pracoviště
ODD	Oddělení
RG	Refresh Group
MLOG	Log nad materializovaným pohledem

Seznam obrázků

Literatura

[1] Oracle USA, Inc. *Oracle® Database Administrator's Guide 11g Release 1 (11.1)* [online]. 2001, 2008 [cit. 2010-04-10]. Database Links. Dostupné z WWW: <http://download.oracle.com/docs/cd/B28359_01/server.111/b28310/ds_concepts002.htm>.

[2] Oracle USA, Inc. *Oracle® Database Concepts 11g Release 2 (11.2)* [online]. 1993, 2010 [cit. 2010-04-25]. 10 Transactions. Dostupné z WWW: <http://download.oracle.com/docs/cd/E11882_01/server.112/e10713/transact.htm>.

[3] Oracle USA, Inc. *Oracle® Database Application Developer's Guide - Fundamentals 10g Release 2 (10.2)* [online]. 1996, 2005 [cit. 2010-04-25]. 7 Coding PL/SQL Procedures and Packages. Dostupné z WWW: <http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14251/adfns_packages.htm >.

[4] Oracle USA, Inc. *Oracle9i Application Developer's Guide - Advanced Queuing Release 2 (9.2)* [online]. 1996, 2002 [cit. 2010-04-25]. 1 Introduction to Oracle Advanced Queuing. Dostupné z WWW: <http://download.oracle.com/docs/cd/B10500_01/appdev.920/a96587/qintro.htm>.

[5] Oracle USA, Inc. *Oracle9i Supplied PL/SQL Packages and Types Reference Release 2 (9.2)* [online]. 2000, 2002 [cit. 2010-04-25]. Advanced Queuing Types. Dostupné z WWW: <http://download.oracle.com/docs/cd/B10501_01/appdev.920/a96612/t_aq2.htm>.

[6] Oracle USA, Inc. *Oracle® Database PL/SQL Packages and Types Reference 10g Release 2 (10.2)* [online]. 1996, 2007 [cit. 2010-04-27]. 16 DBMS_AQ. Dostupné z WWW: <http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14258/d_aq.htm>.

[7] Oracle USA, Inc. *Oracle9i Application Developer's Guide - Advanced Queuing Release 2 (9.2)* [online]. 1996, 2002 [cit. 2010-04-27]. 8 A Sample Application Using AQ. Dostupné z WWW: <http://download.oracle.com/docs/cd/B10500_01/appdev.920/a96587/qsample.htm>.

[8] Oracle USA, Inc. *Oracle9i Advanced Replication Release 2 (9.2)* [online]. 1996, 2002 [cit. 2010-04-27]. 3 Materialized View Concepts and Architecture. Dostupné z WWW: <http://download.oracle.com/docs/cd/B10500_01/server.920/a96567/repmview.htm>.

[9] Oracle USA, Inc. *Oracle9i Replication Management API Reference Release 2 (9.2)* [online]. 1996, 2002 [cit. 2010-05-04]. 19 DBMS_REFRESH. Dostupné z WWW: <http://download.oracle.com/docs/cd/B10500_01/server.920/a96568/rarrefre.htm>.

[10] Oracle USA, Inc. *Oracle® Database SQL Reference 10g Release 2 (10.2)* [online]. 1996, 2005 [cit. 2010-01-01]. ROWID Pseudocolumn . Dostupné z WWW: <http://download.oracle.com/docs/cd/B19306_01/server.102/b14200/pseudocolumns008.htm>.

[11] NUTS In *Wikipedia : the free encyclopedia* [online]. St. Petersburg (Florida) : Wikipedia Foundation, 2005, 2010 [cit. 2010-01-3]. Dostupné z WWW: <http://cs.wikipedia.org/wiki/NUTS>.

Seznam příloh

Příloha 1

Struktura zdrojových kódů

Celá práce je shrnuta v jednom pracovním prostoru JDeveloperu (jws), který je rozdělen na tři projekty korespondující s třemi databázemi, jež figurují v distribuovaném systému. Jednotlivé projekty mají následující strukturu:

CRPP

- Model – složka obsahující ER diagramy centrální databáze.
- Database – složka obsahující offline model schématu CRPP a MIGRACELOGS
  - Migracelogs.sql – skript pro vytvoření uživatele MIGRACELOGS s přidělením práv
  - aq_typ.sql – definice databázového typu AQPAYLOAD_T
  - dblink.sql – vytvoření DB linku do IML
  - prolinani.sql – funkce pro zjištění časového překrytí
  - userpropag.sql – vytvoření uživatele USERPROPAG
  - propagace.sql – vytvoření AQ fronty s propagací do IML
- Migrace – složka obsahující skripty pro import dat do CRPP
  - xxx.sql – naplnění tabulky XXX
  - migrace.sql – skript pro hromadné spuštění importu dat
  - del_crpp.sql – vyčištění CRPP
- Replikace – složka PL/SQL balíčků zajišťující replikaci
  - typy – sbor použitých databázových typů
  - xxx_replikace – vytváření XML zpráv
  - Ens_zprava – vložení zprávy do AQ fronty
- Test – testy replikací
  - regposk_ta.sql – table api pro poskytovatele péče
  - replikace.sql – simulace aplikační logiky
- Doc – programová dokumentace

IML

- Database – offline model schématu IML
  - aq_typ.sql – databázového typu AQPAYLOAD_T
  - cre_fronty.sql – vytvoření fronty IML_Q
  - cre_seq.sql – vytvoření sekvencí
  - userpropag.sql – vytvoření uživatele USERPROPAG

- grants.sql – přidělení oprávnění
- cre_mlogs.sql – vytvoření logů materializovaných pohledů

- Util – skripty pro správu databáze
  - imldnlink_adm – balíček pro správu DB linků
  - ins_imldblink.sql – naplnění tabulky IMLDBLINK
  - imlcrppcb.sql – procedura callback nad frontou IML_Q
  - reg_imlcrppcb.sql – registrace callback
- Migrace – skripty pro naplnění IML
  - cre_zdroj_dat.sql – vytvoření MV, lokální kopie pro import
  - xxx.sql – naplnění tabulky XXX
  - migrace.sql – skript pro hromadné spuštění
  - del_iml.sql – vyčištění IML
- Replikace – balíčky pro replikaci
  - xmltranf_xxx – zpracování XML zpráv
  - replikace_xxx – zapracování změna do tabulek
  - typy.sql – definování databázových typů
- zprava.xsd – definice schématu XML zpráv
- Doc – programová dokumentace

UP

- Model – složka obsahující ER diagramy pobočkové databáze.
- Database – složka obsahující offline model schématu UP
  - cre_txt_v.sql - vytvoření tabulky textu s V(4000) místo LONG
  - dblink.sql - vytvoření DB linku do IML
  - userpropag.sql – vytvoření uživatele USERPROPAG
- replikace - instalace replikací na UP
  - find_xxx.sql - vyhledání omezení a vytvoření skriptů a jejich zrušení
  - prejmenovani.sql - přejmenování stávajících tabulek
  - cre_mv.sql - vytvoření materializovaných pohledů
  - cre_indexes.sql - vytvoření indexů nad MV
  - cre_rg.sql - vytvoření refresh group s plánováním
  - build.sql - skript pro hromadné spuštění instalace