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

Diplomová práce

Datový model EEG/ERP portálu v prostředcích sémantického webu

Plzeň 2013 Filip Markvart

Prohlášení

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

V Plzni dne 19. února 2013

Filip Markvart

Abstract

This thesis DODELAT

Obsah

1 Úvod

EEG/ERP portál je webová aplikace sloužící výzkumným pracovníkům ke shromažďování a organizaci dat získaných při neuroinformatických experimentech v EEG laboratoři. Jejím cílem je ukládání naměřených dat v kontextu prováděného experimentu, který lze popsat rozsáhlou množinou různorodých údajů. Tato aplikace již prošla mnohaletým vývojem v jehož průběhu postupně docházelo ke změnám datového modelu kvůli přibývajícím požadavkům na uchovávaná data. Relační databáze jež slouží jako persistentní úložiště tak postupně byla rozšiřována o další tabulky, jejichž počet se k datu tvorby této práce pohybuje v řádu desítek. Většina realizací požadavků na ukládání dalších dat tak přímo znamená zásah nejen do databáze portálu ale také to do datové vrstvy, která ji využívá. V současné době tak databáze obsahuje velké množství tabulek uchovávající různá data, která jsou ale ve smyslu sémantiky často příbuzná a existuje mezi nimi vazba, která je prostřednictvím relačního datového modelu velmi obtížně popsatelná. Zároveň lze očekávat, že budou přibývat požadavky na uchování dalších dat, která navíc nemusejí mít jen homogenní strukturu (ve smyslu relační databáze), ale může se jednat i o množiny sémanticky příbuzných údajů – tzv. metadata, které budou vázány pouze k některým datům. Možnost ukládání strukturně heterogenních, ale sémantický příbuzných metadat je tak dalším otevřeným problémem.

Cílem této práce je prozkoumání struktury a nalezení sémantiky dat v současném datovém modelu relační databáze portálu a následná úprava tohoto modelu do podoby, která by dovolovala uchovat jak sémantiku dat, kterou není možné relačním modelem vyjádřit tak dodávat dynamicky datům přídavná metadata, aniž by muselo docházet k větším zásahům do datového modelu portálu. Pro realizaci úpravy datového modelu budou v této práci využity prostředky tzv. sémantického webu, který poskytuje množství standardů a technologií pro uchovávání organizaci a správu dat. Tyto technologie a nástroje zde budou popsány a na základě jejich analýzy budou vybrány prostředky, které se využijí pro implementaci úpravy zmiňovaného datového modelu. Poslední část práce se věnuje testování modifikovaného modelu a to především z výkonnostního hlediska. Díky této části by mělo být možné posoudit jak užitečnost samotné úpravy tak i použitelnost a efektivnost získaného modelu pro potřeby EEG/ERP portálu.

2 Sémantický web

Dnešní podoba webu, tak jak je všeobecně známa, je tvořena značným množstvím informací, které mají řadu autorů, v podobě různých organizací či jednotlivců, jež se liší jak svým obsahem tak i podobou publikace. Tyto informace jsou poměrně snadno přístupné díky jejich jednoznačné identifikaci prostřednictvím URI identifikátoru (za předpokladu, že jej známe). K usnadnění získávání dalších (často příbuzných) informací napomáhají tzv. hypertextové odkazy, jež usnadňují přístup k dalším zdrojům informací odstraněním požadavku na uživatelovu znalost identifikátoru cílového zdroje. Samotné hypertextové odkazy tak sice zajišťují provázání jednoho informačního zdroje s jiným díky znalosti jeho URI identifikátoru, ale nenesou už žádné další informace, které by například uživateli poskytly další údaje o cílovém zdroji. Takováto podoba umožňuje získávání informací jak koncovým uživatelům webu, tak v omezené podobě i vyhledávacím strojům, ale má své limity, neboť se v nepřeberném množství dat lze snadno ztratit, či se jen dostat k irelevantním informacím [23]. Základním úkolem sémantického webu, jehož první myšlenky prezentoval v roce 2001 zakladatel konsorcia W3C Tim Berners-Lee, je umožnit aby informace dostupné prostřednictvím webu byly srozumitelné nejen uživatelům, ale také počítačům, jež tato data zpracovávají [17]. Hlavním cílem je tedy vývoj standardů a technologií, které by umožňovaly přesnější a podrobnější vyhledávání, integraci dat a také automatizaci častých úkonů. Sémantický web je založen na několika principech, které budou níže uvedeny.

- **Jednoznačná identifikace entit prostřednictvím URI**

  Veškerá data, reprezentující obvykle objekty reálného světa publikovaná prostřednictvím webu je možné jednoznačně odkazovat prostřednictvím identifikátoru URI. Díky této skutečnosti je tak možné realizovat i nepřímé odkazy na objekty, například osobu Petr Novák s emailem petr.novak@w3.org je možné identifikovat jako osobou, jejíž email má URI mailto:petr.novak@w3.org.

- **Zdroje i odkazy mezi nimi je možné typovat**

  Současná podoba webu je tvořena zdroji a odkazy jež je vzájemně propojují. Zdroje, které jsou reprezentovány webovými dokumenty jsou publikovány za účelem poskytnutí informací lidskému uživateli, který dokáže ze samotného obsahu dokumentu získat i některá jeho metadata

(pokud jsou v určité formě součástí obsahu) a do jisté míry také vztah k ostatním dokumentům, na něž vedou případné odkazy. Stroje v podobě různých vyhledávačů či automatů pro shromažďování dat ale tuto schopnost nemají nebo je pro ně příliš náročná. Řešením sémantického webu je typování jak samotných zdrojů, tak i odkazů, které je provazují. Díky této skutečnosti je pak možné webovým dokumentům dodávat metadata jako např. autora, verzi či závislost na jiném dokumentu. Z hlediska typování odkazů je například možné jeden webový zdroj označit pouze jako odlišnou verzi jiného zdroje.

- **Tolerance neúplných informací**

  U současně podoby webu může nastat situace, kdy některý zdroj není dostupný. V takovém případě uživatel ztrácí přistup k danému dokumentu, ale díky koncepci webu není nikterak ohrožena dostupnost ostatních zdrojů. V případě sémantického webu se situace nemění, nedostupnost některého zdroje není žádnou překážkou, neboť nástroje sémantického webu zpracovávají pouze ty informace, které jsou dostupné a těch vytvářejí závěry. V důsledku je tak možné dojít při zpracovávání dat ke stejným výsledkům, jako v případě, když jsou zpracovávány jen některé vybrané informace, jejichž rozsah je explicitně definován.

- **Zpracování neověřených dat**

  Při zpracování informací pocházejících z neověřených zdrojů je možné dohledávat prostřednictvím typovaných odkazů důvěryhodná data, jejichž obsah a odkazy poslouží jako ověřovací prostředek. Tento princip je možné uvést na jednoduchém příkladu. Aplikace zpracovávající data sémantického webu vyhledá informace, přičemž je kladen požadavek na vysokou pravděpodobnost správnosti výsledku. Pokud část nalezených informací pochází z neověřeného zdroje, je možné vyhledávat například jejich autora v odkazech zdrojů, které jsou důvěryhodné. V případě úspěchu nalezení takového odkazu u více různých zdrojů je pak možné považovat zkoumaný zdroj s vysokou pravděpodobností rovněž za důvěryhodný a tím zajistit plnění požadavků na výsledek.

- **paralelního vývoje dat**

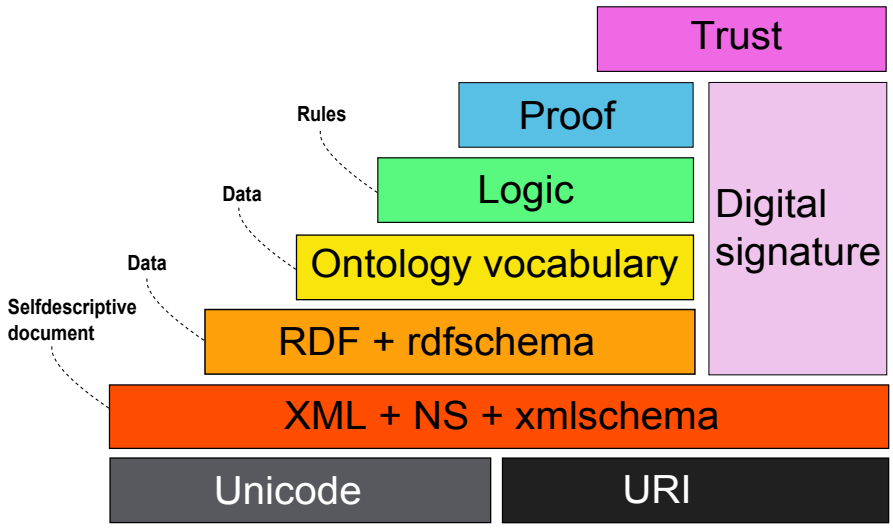
  V průběhu času nezřídka nastávají situace, kdy autoři, či skupiny autorů publikují obdobná data na různých místech nebo v odlišném čase. Obsah těchto dokumentů se může navíc lišit svým jazykem či použitou terminologií, ač význam bude shodný. S využitím prostředků sémantického webu je ale možné prostřednictvím typovaných odkazů zajistit provázanost významově obdobných či na sebe navazujících dat

i přes překážku rozdílnosti jejich podoby zápisu. Navíc je také možné dodávat nové informace bez nutnosti úpravy původních dat, která tím pádem nezmění svoji strukturu [23].

2.1 Architektura sémantického webu

Architektura sémantického webu sestává z více oddělených vrstev, mezi nimiž je zajištěna zpětná i dopředná kompatibilita [16]. Nejnižší vrstva je tvořena dvěma technologickými standardy – URI identifikátory sloužící pro jednoznačné pojmenování zdrojů dat a Unicode kódování mezinárodní znakovou sadou. Druhou vrstvu architektury, jež je patrná z obrázku 2.1, reprezentuje značkovací jazyk XML (Extensible Markup Language), který umožňuje tvorbu strukturovaného dokumentu za užití vlastních značek. Tato vrstva zároveň zajišťuje definici XML schématu včetně jmenných prostorů. RDF + rdfschema jež následuje je klíčovou vrstvou sémantického webu neboť dovoluje tvorbu vazeb a vztahů mezi jednotlivými zdroji, které jsou typované spolu s odkazy. Je tak možné definovat libovolné vztahy mezi objekty či jejich kategoriemi bez nutnosti specifikace významu samotných vazeb či objektů. Díky RDF schématu je vytvářena základní sémantika datového modelu, která už definuje význam některých elementů jako třídy či podtřídy. Vrstva ontologického slovníku, zastoupená jazykem OWL, nabízí pokročilou reprezentaci znalostí na úrovni deskripční logiky a umožňuje tak vytvářet složitější struktury sloužící k popisu různých vlastností objektů [17]. Poslední vrstvou, která je jako všechny předchozí zmíněné konsorciem W3C standardizovaná jsou digitální podpisy. Ty poskytují možnosti například pro detekci různých verzí dokumentů. Zbylé výše znázorněné vrstvy slouží pro definice a vyhodnocování odvozovacích pravidel a v současnosti jsou ve fázi vývoje [23].

Vrstvení jazyků sémantického webu je podstatné pro úroveň expresivity znalostního modelu, neboť s rostoucí vyjadřovací možností jazyka také roste složitost dotazovacích operací nad modelem. Je tedy nutné před započetím tvorby modelu nejprve zjistit jeho požadovanou expresivitu a podle té zvolit pro zápis dat jazyk, který ji dovoluje obsáhnout. Využívá se tedy skutečnosti, že jazyk vyšší vrstvy zahrnuje vyjadřovací schopnosti vrstev nižších[17]. Další podkapitoly se budou zabývat podrobněji jednotlivý zmíněnými technologiemi.



Obrázek 2.1: Architektura sémantickém webu [23]

2.2 XML

XML (eXtensible Markup Language) je značkovací jazyk sloužící pro popis hierarchických struktur textových dokumentů prostřednictvím tzv. tagů. Tag je konstrukce, která slouží k počátečnímu a koncovému ohraničení společně definovaného elementu. Tag lze chápat jako prostředek pro dodání metadat ke textové struktuře, jež ohraničuje. Příkladem může být následující zápis *<prijmeni>Novák</prijmeni>*, kde elementu *Novák* je dodána meta informace, že se jedná o příjmení. Samotné XML ale nedefinuje žádný sémantický význam tagů, slouží pouze pro specifikaci syntaxe na úrovní XML dokumentu. Pro definici (zejména hierarchické) struktury XML dokumentu slouží XML Schema, které umožňuje zápis pravidel, jež musí cílový dokument dodržovat pro zachování své validity. Ze strany sémantického webu ale nemají pravidla XML Schema žádný sémantický význam a slouží tak pouze pro definici struktury a syntaxe. Zmíněné schéma také definuje základní datové typy (čísla, řetězce, čas a pod.), které nabývají významu v sémantických jazycích jako je RDF [16].

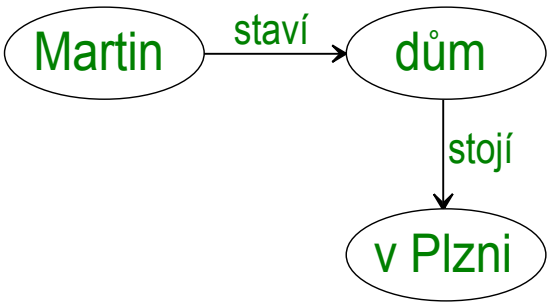
2.3 RDF

Technologický základ sémantického webu tvoří jazyk RDF (Resource Description Framework), jež slouží jako obecný rámec pro popis, výměnu a opětovné použití metadat [16]. Tento rámec poskytuje jednoduchý model sloužící pro popis zdrojů jež je nezávislý na jeho konkrétní implementaci [7]. Samotné informace o objektu jsou realizovány prostřednictvím tvrzení, jež se označují jako trojice (anglicky triple). Každou trojici tvoří spolu subjekt, predikát a objekt. Subjekt je libovolný objekt identifikovatelný pomocí URI, který se snažíme prostřednictvím trojice popisovat, zaznamenat nějakou jeho vlastnost. Tato vlastnost se popisuje prostřednictvím predikátu, který vede ve směru od subjektu ke objektu, přiřazuje tedy subjektu nějaký objekt prostřednictvím této vlastnosti. Cílový objekt pak představuje hodnotu, které předchozí objekt nabývá pro daný predikát. Tento princip je možné znázornit na jednoduchém tvrzení, zapsaném větou „Martin staví dům.“ Subjektem trojice potom bude Martin, predikátem staví a objektem dům, tak jak je znázorněno na obrázku 2.2.

Obrázek 2.2: Příklad RDF trojice

Dle definovaného standardu [21] je možné, aby objektem byl jiný subjekt. Díky této skutečnosti je možné jednotlivé trojice spojovat do většího celku, který ve výsledku tvoří strukturu orientovaného grafu, kterou lze označit jako model [16]. Jednoduchým model tvořený dvěma trojicemi lze vytvořit využitím dalšího tvrzení „Dům stojí v Plzni.“ V předchozí trojici pak bude dům subjektem namísto objektu a dojde tak ke spojení dvou trojic (za předpokladu že v obou tvrzeních je myšlen stejný fyzický dům), tak jak je znázorněno na obrázku 2.3.

Z hlediska implementace mohou být uzly orientovaného grafu datového modelu tvořeny URI identifikátorem, anonymním listem nebo literálem [19]. URI identifikátor obsahuje pouze adresu zdroje v textové podobě ve znakové sadě Unicode a zastupuje tak konkrétní jedinečný objekt. Anonymní list (anglicky blank node) je prvek nahrazující URI identifikátor při absenci unikátní adresy zdroje. Tato entita přestavuje zdroj, který je sice v rámci



Obrázek 2.3: Příklad jednoduchého RDF modelu

grafu popisován prostřednictvím trojic, ale není potřeba (často z hlediska významu), aby byl dostupný i vně grafu. S URI má společné to, že musí nést adresu zdroje (její syntaxe není implicitně definována), ale liší se skutečností, že tato adresa musí být unikátní pouze v rámci obalujícího modelu, nikoliv vně grafu. Může tak nastat situace, že dva odlišné modely budou obsahovat (ze syntaktického hlediska) dva stejné anonymní uzly, což v případě URI možné není (při respektování specifikace). Tento list tak může představovat anonymní objekt sloužící ke vytvoření vazby mezi jinými objekty, které jsou z hlediska sémantiky významné [21]. Jako příklad je možno uvést následující tvrzení. „Martinův přítel zná předpověď počasí. Počasí bude deštivé.„ Zjednodušeným převodem předchozích vět na trojice v podobě subjekt – predikát – objekt získáme: Martin – má – přítel, přítel – zná – počasí, počasí – bude – deštivé. Subjekt resp. objekt přítel zde může být reprezentován anonymním listem, v případě že jedinou signifikantní informací (z vnějšího pohledu na datový model) je Martinova získaná znalost počasí, nikoliv už jeho přítel, jež mu ji zprostředkoval. Poslední možnou reprezentací entity trojice je literál, jež nese Unicode textový řetězec, který slouží pro zápis koncové informace (užitečné pro lidského čtenáře). Literál může také navíc obsahovat položku, pro označení jazyka, ve kterém je textová informace zapsána [19].

Pro komponenty každé trojice grafu platí následující pravidla [19].

- **subjekt** - může být tvořen URI identifikátorem nebo anonymním listem
- **predikát** - může být tvořen pouze URI identifikátorem
- **objekt** - může být tvořen URI, anonymním listem nebo literálem

Pro zápis trojic RDF grafu je sice možné využít grafické podoby, ale pro vyjádření sémantiky webových zdrojů je nejvhodnější využít syntaxe jazyka XML. Níže uvedená ukázka kódu reprezentuje XML zápis trojice z obrázku 2.2.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/">
  <rdf:Description rdf:about="http://www.w3.org/Person/Martin">
    <dc:stavi>dům</dc:stavi>
  </rdf:Description>
</rdf:RDF>
```

Samotné RDF nedefinuje trojicím ani jejím částem sémantiku, ale dovoluje vyjádřit základní vztahy náležitosti prvků do kategorie prostřednictvím kontejnerů a kolekcí (např. rdf:Bag, rdf:List) [17]. Pro dodání základní sémantiky slouží schéma popsané v následující podkapitole.

2.4 RDFS

RDF Schema (Resource Description Framework Schema) funguje jako základní jazyk pro tvorbu ontologií s velmi jednoduchou sémantikou. Toto schéma rozšiřuje jazyk RDF o možnosti vyjádření vlastností objektů, konstrukce tříd objektů a popis jejich hierarchie [17]. Prostředky jazyka RDFS umožňují především vyjádření vztahů mezi zdroji a lze je rozdělit na dvě skupiny – třídy a vlastnosti.

2.4.1 Třídy

Skupiny zdrojů je možné rozčleňovat do skupin označovaných jako třídy (classes), jejichž členové se nazývají instance třídy. Na úrovni RDFS se rozlišují třídy od svých instancí a každá třída jich může mít neomezený počet. Dvě třídy mohou mít navíc shodnou množinu instancí tříd a zároveň tyto třídy mohou být navzájem různé. Bude-li se tedy například definovat třída A jako osoby pracující v kanceláři *1* a třída B jako osoby žijící ve městě X. Potom je možné aby různé třídy A a B měly stejné množiny instancí, za předpokladu,

že každá osoba pracující v kanceláři *1* bydlí ve městě *X*. Pro třídy platí také dědičnost – bude li třída *B* podtřídou *A*, pak všechny instance *B* jsou zároveň instancí třídy *A*. Koncept tříd RDFS definuje následující konstrukce [22]:

- **rdfs:Resource** představuje RDF zdroj, který je obalující třídou všech prvků – je tedy nejvýše postavenou rodičovskou třídou, *rdfs:Resource* je zároveň instancí *rdfs:Class*
- **rdfs:Class** reprezentuje rodičovskou třídu všech RDF tříd zdrojů, čímž je *rdfs:Class* instancí *rdfs:Class* (sebe sama)
- **rdfs:Literal** třída je instancí *rdfs:Class*, která slouží pro reprezentaci RDF literálů a je podtřídou *rdfs:Resource*
- **rdfs:Datatype** je obalující třídou pro datové typy, které jsou její instancí, *rdfs:Datatype* je zároveň podtřídou i instancí *rdfs:Class* a každá její instance je podtřídou *rdfs:Literal*
- **rdf:XMLLiteral** je třída XML literálů, která je podtřídou *rdfs:Literal* a instancí *rdfs:Datatype*
- **rdf:Property** představuje rodičovskou třídu všech definovaných vlastností a je instancí *rdfs:Class*

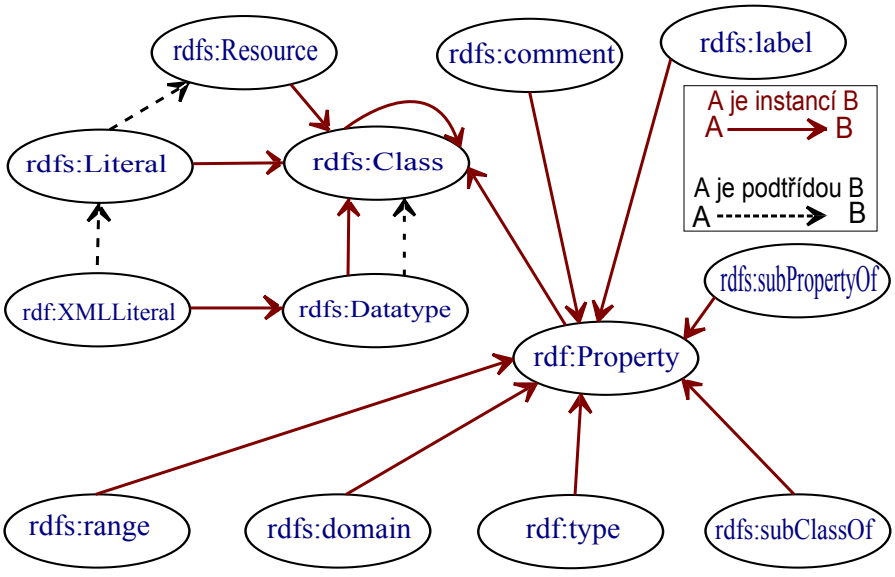
2.4.2 Vlastnosti

Vlastnosti (properties) slouží k vyjádření vztahu mezi dvěma zdroji – na úrovni trojice mezi subjektem a objektem. Koncept RDFS definuje následující vlastnosti:

- **rdfs:range** je instancí *rdf:Property*, která slouží ke vyjádření, že objekt jehož predikát má definovaný *rdfs:Range X* bude zároveň instancí *X*, například z následujících trojic *A – rdfs:Range B a X – A – C* bude vyplývat, že *C* je instancí *B*, zároveň platí, že objekt s definovaným predikátem může být instancí více tříd (pokud má predikát definováno více *rdfs:Range*).
- **rdfs:domain** představuje instanci *rdf:Property*, která vyjadřuje, že zdroj (subjekt) jehož predikát má definovaný *rdfs:Domain X* bude také instancí *X*, tento zdroj může být jako v předchozím případě instancí více tříd při definování více *rdfs:Domain*

- **rdf:type** slouží k definování, že zdroj (subjekt) je instancí třídy definované objektem, pokud tedy platí *A rdf:type B*, pak *A* je instancí *B*
- **rdfs:subClassOf** je vlastnost sloužící k vyjádření náležitosti instancí jedné třídy jako instancí jiné třídy, pokud platí *A rdfs:subClassOf B*, pak *A* je podtřídou *B* a všechny instance třídy *A* jsou zároveň instancí třídy *B*, tato vlastnost je navíc tranzitivní, takže popsanou dědičnost je možné řetězit do libovolné délky
- **rdfs:subPropertyOf** slouží k vyjádření dědičnosti vlastností, pokud platí *P1 subproperty P2*, pak pro trojici *A P1 B* platí, že subjekt *A* má pro vlastnost *P1* i *P2* pro objekt *B*
- **rdfs:label** umožňuje zdroji (subjektu) přidat textovou informaci jako lidsky čitelnou náhradu pro označení zdroje
- **rdfs:comment** dovoluje přidat zdroji popisek, který usnadňuje lidskému uživateli pochopit význam zdroje [22]

Vyjádření vztahů mezi jednotlivými vlastnostmi a třídami je patrné z obrázku 2.4. Rámec RDFS dále ještě definuje vlastnosti a třídy pro kontejnery a kolekce, které je možné naleznout ve [22].



Obrázek 2.4: Vztahy mezi vlastnosmi a třídami RDFS

2.5 Ontologie

Znalostní modely, jež jsou popisované vyššími jazyky sémantického webu se označují jako ontologie. Ontologie ovšem není pouze abstraktním znalostním modelem, ale slouží i jako prostředek pro k získání interoperability, tedy schopnosti vzájemné spolupráce oddělených systémů na úrovni sdílení dat a poskytování služeb [17]. Definicí ontologie v informatice jež uvádí Thomas Gruber, zakladatel ontologického inženýrství, je „formální specifikace sdílené konceptualizace"[12]. V této definici je formálností myšleno vyjádření znalostí prostřednictvím jazyka s formálním a logickým základem. Pojem konceptualizace znamená definování konceptů na dané úrovni abstrakce, jež odpovídá požadavkům pro model domény. Tuto doménu pak vytvářejí osoby, pro které je daná ontologie závazně (vzájemnou dohodou) definovaná jako prostředek pro popis dat, jež musejí všichni členové dodržovat [17]. V důsledku se tak jedná o tvorbu abstraktního modelu v určené oblasti, kde se definují pojmy společně se vzájemný vztahy vyjádřené logickým jazykem. Cílem explicitní specifikace pojmů a vztahů je snaha o porozumění znalostně orientovaných systémů [11]. Ontologie lze z hlediska vývoje rozčlenit do tří kategorií:

- **Terminologické (lexikální)** Slouží především ke zachycení taxonomie definovaných pojmů a popis jejich vzájemných vztahů, jejich realizace se často podobá slovníkům synonym.
- **Informační** Zastávají roli služby stojící nad databází, která zajišťuje vyšší míru abstraktnosti při databázovém dotazování.
- **Znalostní** Dovolují reprezentaci znalostí především v oblasti umělé inteligence, kde jsou chápány jako logické teorie, jejichž elementární prvky se definují prostřednictvím formálního jazyka. Využívají se zejména ve znalostně orientovaných systémech [15].

Dle míry formalizace a cílového předmětu konceptualizace se rozlišují 4 kategorie ontologie.

- **Doménové** Zabývají se specifikací určité oblasti – domény.
- **Generické** Blíží se svou podobou doménovým, ale pokrývají širší množinu dat a zachycují tak obecnější koncepty, nejdou ovšem do příliš velké hloubky. Popisují například realitu prostřednictvím vzájemných

časoprostorových pozic objektů. Často se snaží zachytit všeobecné znalosti, které odpovídají běžně používaným lidským vědomostem.

- **Úlohové** Tyto ontologie jsou zaměřené zejména na odvozovací procesy namísto záznamu znalostí a slouží především jako modely pro řešení určitých problémů – např. pro diagnostiku či plánování.
- **Aplikační** Jsou většinou vázány na specifické aplikace, kde spojují doménovou a úlohovou ontologií [15].

Pro realizaci popsaných ontologií je zapotřebí jazyka, který by dovoloval formální zápis. Výše uvedené schéma RDFS je z hlediska sémantického webu základní podobou ontologie, kterou je dále možné rozšiřovat například s využitím jazyků jako je OWL.

2.6 OWL

OWL (Web Ontology Language) je jazyk sémantického webu navržený pro získávání a zpracování informací prostřednictvím strojů (aplikací), namísto současného stavu, kdy jsou informace určeny pouze pro lidské uživatele. Tento jazyk využívá technologií XML, RDF a RDFS jakožto primárních prostředků pro tvorbu ontologických slovníků se základní sémantikou, kterou sám dále rozšiřuje o možnosti vyjádření složitějších výrazů a jejich vzájemných vztahů [20]. Mezi tato rozšíření patří například kardinalita, univerzální a existenční kvantifikace, matematické charakteristiky vlastností jako např. tranzitivnost, disjunktnost či inverze a anonymní třídy (sloužící často k jednorázovému využití) [13]. Vzhledem k tomu, OWL nabízí poměrně pokročilou úroveň expresivity jazyka, je nutné brát v potaz výpočetní složitost algoritmů odvozovacích a usuzovacích nástrojů pracujících s vytvořenou ontologií. Z tohoto důvodu jsou specifikovány 3 varianty jazyka OWL, které se vzájemně liší mírou expresivity [17].

- **OWL Lite** Umožňuje definovat hierarchický systém tříd s jednoduchými omezeními vazeb [20]. Dále poskytuje prostředky pro zaznamenání syme-
  trické, transitivní a inverzní vlastnosti a také jednoduchá omezení na velikosti množin vybraných objektů modelu – kardinalitu, která je ale v této verzi omezena pouze na přípustné hodnoty 0 a 1 [17].

- **OWL-DL** Tato varianta poskytuje maximální možnou expresivitu jazyka, u které je stále splněna podmínka rozhodovací úplnosti (jakékoliv rozhodovací pravidlo je možné použít na kteroukoliv část ontologie) a výpočetní splnitelnosti (veškeré výsledky odvozovacích operací budou získány v konečném čase) [17]. OWL-DL nabízí veškeré konstrukce jazyka OWL, které ale podléhají určitým omezením při jejich použití, např. není možné využít omezení kardinality pro vlastnosti definované jako tranzitivní. Z hlediska množiny dostupných jazykových konstrukcí dovo-
  luje tato varianta oproti OWL Lite definovat navíc sjednocení, disjunkce a doplňky tříd nebo libovolné omezení kardinality [20].

- **OWL-Full** Je variantou, která poskytuje maximální možnou expresivitu, ke které využívá stejně jako předchozí verze všech konstruktů jazyka OWL. Tato varianta neklade žádná omezení pro vyhodnocovací pravidla, díky čemuž ale není možné zaručit výpočetní splnitelnost. OWL-Full dovoluje ještě navíc uživatelskou změnu sémantického významu nativně definovaných konstrukcí jazyků RDF a OWL. Tato verze ovšem není prakticky příliš využitelná, neboť v současnosti nejsou známé žádné efektivní algoritmy, které by dovolovaly nad modelem dat provádět usuzovací operace využívající všech vlastností OWL-Full [17].

Každá verze OWL je rozšířením svého předchůdce díky čemuž je ontologie vyjádřená v jednodušší variantě platná i pro verzi vyšší, např. tedy každá ontologie zapsaná ve OWL Lite bude plně validní ve OWL-DL (což ale obráceně neplatí) [20]. Mezi typické odvozovací úlohy v kontextu OWL patří například odvozování taxonomické struktury, ověřování příslušnosti instance ke třídě, klasifikace individuí vzhledem k definované ontologii či testování splnitelnost logické teorie [13].

2.7 SPARQL

Základním prostředkem pro dotazování nad daty sémantického webu je jazyk SPARQL (SPARQL Protocol and RDF Query Language). Tento jazyk standardizovaný konsorciem W3C umožňuje vytvářet dotazy nad RDF grafy prostřednictvím vzorů RDF trojic spolu s logickými operacemi konjunkce a disjunkce [17]. Konstrukce těchto dotazů je sice složitější než v případě SQL dotazů nad relačními databázemi, ale umožňuje ze zdrojové ontologie získávat

komplexnější informace [16]. SPARQL dotaz se skládá ze 3 základních částí. První část – označovaná jako prolog slouží k definování jmenných prostorů a prefixů, které budou v dalších částech dotazu použity. Druhou částí je hlavička dotazu, kde se určuje jaký typ dotazu se bude v další části provádět. Třetí a hlavní část SPARQL slouží k definování cílové ontologie (RDF grafu) pro dotazování (v případě, že cílový graf je implicitně definován, je tato část vynechána) a proměnných včetně samotných vyhledávacích podmínek v podobě grafových vzorů (graph pattern)[17]. Tento jazyk částečně vychází ze SQL, což je patrné i na jeho syntaxi. Např. klauzule FROM slouží pro výběr cílového grafu pro dotazování, či klíčové slovo WHERE slouží pro uvození zápisu souboru RDF trojic – tzv. grafového vzorce, jež je jádrem dotazu. Vyhodnocování dotazu poté probíhá takovým způsobem, že dochází k porovnávání grafového vzorce se s RDF daty [14]. Pro názornost je níže uveden zápis jednoduchého SPARQL dotazu, který slouží k vyhledání jmen a příjmení všech osob v implicitně zadaném grafu.

```
PREFIX zcu: <http://www.zcu.cz/ns/students>
SELECT ?name ?surname
WHERE  {
     ?person x zcu:Person.
     ?person zcu:Name ?name.
     ?person zcu:Surname ?surname
}
```

SPARQL podporuje 4 druhy dotazů, které se vzájemně liší v typu vráceného výsledku [14].

- **SELECT** Tento typ dotazu nejvíce odpovídá běžnému SQL dotazování, neboť zadaným proměnným nastaví příslušné hodnoty a vrací je v podobě tabulky [24].
- **ASK** Dotaz typu ASK vrací pouze boolean odpověď a slouží ke otestování, zda zadaný grafový vzorec má pro daný graf řešení. ASK tak slouží ke testování vytvořených logických hypotéz, ale neumožňuje už vrácení konkrétních řešení [14].
- **DESCRIBE** Dotazy typu DESCRIBE navrací podgraf dotazovaného grafu, který obsahuje všechny dostupné informace o zdroji, jež vyhověl zadanému grafovému vzoru [14]. Při vyhodnocování dotazů dochází k porovnávání zdrojů získaných ze vstupního grafového vzoru se zdroji

celého grafu. Pro nalezené odpovídající zdroje jsou pak získány všechny vázané relevantní informace, ze kterých je ve výsledku složen navracený graf [24].

- **CONSTRUCT** Tento speciální druh dotazu navrací jako výsledek nový RDF graf dle definované grafové šablony. Výsledný RDF graf je konstruován podle částečných řešení dotazovací sekvence, jež vyplňují proměnné ze zadané grafové šablony na základě které jsou tvořené výsledné trojice pro cílový navracený graf [14]. Tato dotazovací operace se do jisté míry podobá XSLT transformaci XML dokumentu.

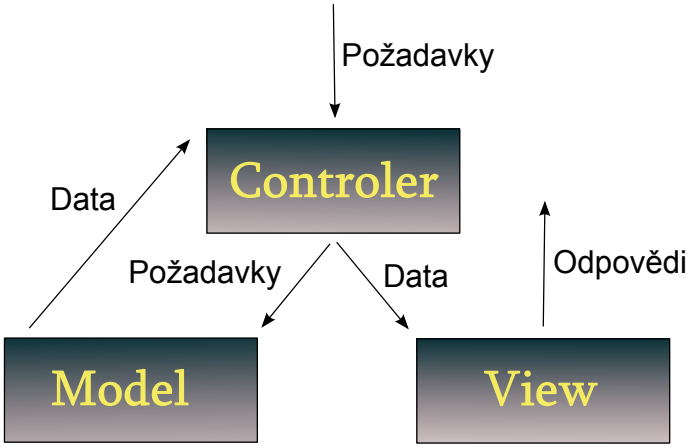
3 EEG/ERP portál

EEG/ERP portál je webová aplikace, sloužící vědeckým pracovníkům k ukládání, stahování a vyhledávání EEG/ERP experimentů včetně jejich přidružených metadat. Tato aplikace je vyvíjena pro Katedru informatiky a výpočetní techniky Západočeské univerzity v Plzni. Portál funguje jako komplexní aplikace, která neslouží jen ke zpracování dat experimentů, ale poskytuje také funkce pro správu uživatelů, jež mohou mít odlišné role a také se sdružovat do výzkumných skupin jež mohou vzájemně spolupracovat. Portál je vyvíjen jako open-source aplikace pod GNU licencí. Jeho základní funkcionalitu je možné shrnout do několika následujících bodů [10].

- **Ukládání, aktualizace a stahování naměřených dat experimentů včetně metadat**
- **Registrace nových uživatelů, sdružování do skupin včetně určení rolí**
- **Sdílení dat a metadat experimentů mezi výzkumnými skupinami**
- **Publikační a diskusní prostředky pro uživatele v podobě článků a jejich komentářů**
- **Fulltextové vyhledávání nad celou databází**
- **Přihlašování uživatelů skrze interní účty či prostřednictvím sociálních sítí**
- **Rezervace výzkumné laboratoře**

3.1 Technologie portálu

Z technologického hlediska je EEG/ERP portál webová aplikace se standardní třívrstvou architekturou MVC (Model – View - Controler) - viz obrázek 3.1, běžící na aplikačním serveru Tomcat. Portál je vyvíjen v programovacím jazyce Java EE, přičemž při implementaci je využito několika frameworků, zejména pak Spring.



Obrázek 3.1: Architektura MVC

Prezentační vrstva je realizována prostřednictvím technologie JSP (Java Servlet Pages), zabezpečení a přihlašování uživatelů pak využívá frameworku Spring Security. Dále je pluginem Sping Social zajištěno prostřednictvím definovaného API propojení webové aplikace se sociálními sítěmi Facebook, Twitter a LinkedIn. Spring Social také slouží ke integraci možnosti přihlašování uživatelů prostřednictvím účtů zmíněných sociálních sítí.

Aplikační vrstva využívá zejména technologií Spring MVC a Spring Core a zajišťuje zpracování veškerých uživatelských požadavků pro ukládání a čtení dat, která získává z datové vrstvy. Persistentní data jsou uložena v relační databázi Oracle 11g, přičemž k jejich zpracování je použito objektově-relační mapování frameworku Hibernate. Ten využívá XML souborů pro mapování tabulek na POJO objekty, jež tvoří perzistentní vrstvu. Framework Hibernate zároveň umožňuje odstínit datovou vrstvu od implementační závislosti na zvolené databázi, jejíž případná obměna by následně nevyžadovala příliš velké zásahy do zdrojových kódů portálu [4].

3.2 Persistentní data

Databáze Oracle jež slouží ke ukládání persistentních dat uchovává veškeré naměřené hodnoty zaznamenaných experimentů. Tato data mají z pohledu databáze podobu binárních souborů s velikostí řádu desítek až stovek MB.

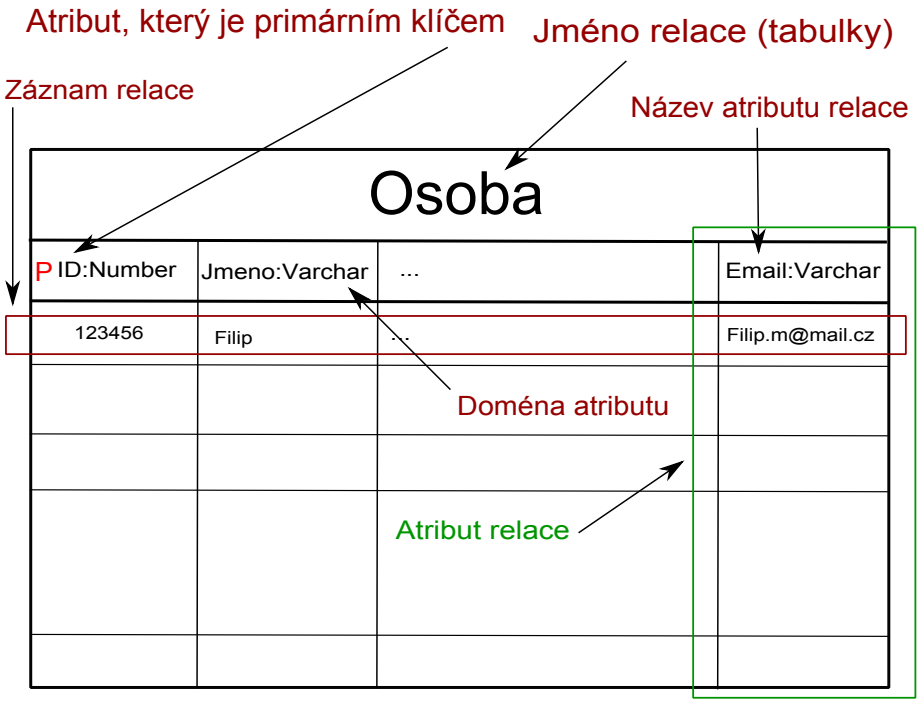
Tato naměřená data by ovšem ztrácely hodnotu, kdyby byly uchovávány jako samostatné. Proto databáze obsahuje další tabulky, jež nesou metadata prováděných experimentů. Kontext experimentu je popsán především hodnotami o prostředí laboratoře a zúčastněných osobami. Prostředí laboratoře je charakterizováno například teplotou, použitým hardwarem a softwarem. Nedílnou součástí popisu experimentu je také scénář zaznamenávající samotný postup měření, který má podobu binárního souboru. Dalšími důležitými daty jsou také informace o použitých elektrodách měřících EEG, u kterých je důležitý jejich typ a poloha. Co se týče osob účastnících se experimentu, zaznamenává se měřící a především měřená osoba. U osob jsou uchovávány osobní a kontaktní údaje a také data nutná pro přihlašování do portálu. Osoby se mohou sdružovat do různých výzkumných skupin, jež nesou seznam svých členů a vlastníka. Podstatným údajem jsou také identifikátory skupin vázané k hodnotám číselníků charakterizující vlastnosti experimentu, které tak napomáhají výzkumníkům zaznamenávat vlastní vybrané předdefinované hodnoty. Další tabulky ještě zaznamenávají historii stahování, zveřejňované články a jejich textové komentáře. Rezervace laboratoře obsahuje údaje o počátečním a koncovém času události a také rezervující osobě.

4 Relační datový model

Jak již bylo zmíněno v předchozí kapitole, webová aplikace EEG/ERP portál uchovává data v relační databázi – lze na ně tedy pohlížet jako na relační datový model. Vzhledem k tomu, že další část práce se bude zabývat úpravou současného datového modelu portálu, bude vhodné uvést zde základní popis a vlastnosti obecného relačního modelu.

Základem relačního modelu dat, který byl popsán již v roce 1970 je takzvaná relace neboli tabulka [3]. Z matematického hlediska můžeme n-ární relaci chápat jako libovolnou podmnožinu kartézského součinu n množin [18]. Množinou je zde myšlena doména jednoho konkrétního sloupce tabulky, tedy všechny přípustné hodnoty, kterých může libovolný řádek v daném sloupci tabulky nabývat. V důsledku tedy máme tabulku tvořenou n sloupci (n-ární relace) a x řádky, kde každý řádek reprezentuje jeden prvek z množiny tvořené kartézským součinem n množin. V případě, že by každá z n množin byla konečná (řádky daných sloupců mohou nabývat konečného počtu hodnot), bude konečný i kartézský součin těchto množin a tabulka tak může mít jen konečný a tudíž omezený počet neopakujících se řádek (duplicity v tabulce neuvažujeme, neboť by neměly žádný smysl – jednalo by se o redundantní data) [18].

Uvedená reprezentace matematického pojmu relace jako databázové tabulky je základem takzvaného schéma relace. Toto schéma obsahuje název relace, jména všech atributů (sloupců tabulky) a jejich integritních omezení – tedy domény [2]. Domény jednotlivých atributů jsou v systémech řízení báze dat reprezentovány datovými typy, tedy např. číselný typ, textový řetězec či datum. Tabulku lze tedy chápat jako určitou formu znázornění relace [5]. Jedna tabulka slouží zpravidla k uchování údajů o jednom druhu objektů [6]. V případě portálu tak například tabulka *Person* uchovává pouze data o osobách – jeden řádek neboli záznam tedy reprezentuje údaje o jednom člověku, které jsou strukturované do sloupců, např. jméno, příjmení, email apod. Názornější představu konkrétní relace je možné získat z obrázku 4.1. Aby bylo možné s daty tabulky pracovat, zavádí se aparát, který zajišťuje jednoznačnou identifikaci každého záznamu - primární klíč. Primární klíč je atribut či skupina atributů, která je pro každý záznam tabulky jedinečná (stejně jako název tabulky v databázi) [6]. V případě zmíněné tabulky *Person* by tak mohl posloužit například atribut email, neboť žádné dvě osoby nemohou logicky mít stejnou emailovou adresu (což by mělo být zajištěno ze strany domény



Obrázek 4.1: Znázornění relace

dané emailové adresy). V případě že tabulka neobsahuje žádný atribut, který by potřebnou jedinečnost záznamů zajišťoval, je nutné jej přidat. To je realizováno např. atributem ID, obsahujícím číslo z číselníku, jehož hodnota je inkrementována po každém přidání dalšího záznamu do tabulky *Person*. Tento identifikátor položky sice jednoznačně určuje daný záznam v tabulce, nikoliv už však v rámci celé databáze (pokud uvažujeme více tabulek) či vně databáze. V jedné databázi je tedy možné mít například tabulky *Person* a *Experiment*, jejichž primárními klíči bude atribut s názvem ID typu celého čísla. V takovém případě je pak i možné aby obě tabulky obsahovaly 2 různé záznamy jež budou jednoznačně identifikovatelné stejnou číselnou hodnotou (ale v kontextu dané tabulky). Z toho důvodu je pro jednoznačnou identifikaci záznamu nutné znát jak hodnotu příslušného atributu obalující relace, tak i její název.

4.1 Limity relační databáze

4.1.1 Dědičnost

Relační databáze může samozřejmě obsahovat větší množství tabulek (za předpokladu že mají vzájemně různé názvy). Každá tabulka tak sdružuje záznamy, jejichž obsah si je v určitém smyslu blízký. Z hlediska objektově orientovaného přístupu, který je realizovaný např. nástrojem Hibernate, lze tabulku chápat také jako třídu a její záznamy jako instance dané třídy. Každá takováto instance má pevně daný počet atributů, který je definován obalující třídou. Uvažujme například tabulku *Person*, která ponese údaje jak o osobách které jsou cílovým objektem experimentů tak údaje o výzkumných pracovnících. U obou skupin nás budou zajímat společné atributy jako jméno a příjmení, ale některé atributy budou různé. U testovaných osob to bude např. věk, zatímco u výzkumníků např. titul. Uvažujeme-li objektový přístup k takovéto tabulce, může využít dědičnosti. Vytvoříme tedy třídu *Person*, která ponese všechny společné atributy a od ní bude zděděna třída pro výzkumníky a třída pro testované osoby. Tyto třídy budou obsahovat atributy, které jsou specifické pro danou skupinu. Z pohledu relačního modelu je tento problém řešitelný obtížněji. Je možné například vytvořit 2 samostatné tabulky pro dané skupiny či ponechat jednu tabulku, která ponese všechny atributy a každá skupina bude mít vyplněna jen jejich část. Podobný problém by vznikl například při potřebě dodat další atribut jen omezenému počtu osob jedné ze skupin. Z uvedeného příkladu je patrné, že dynamické rozšiřování záznamů obsažených v jednotlivých tabulkách je poměrně obtížné.

4.1.2 Vztahy mezi relacemi

Relační model dovoluje také zaznamenávat vztahy mezi jednotlivými tabulkami a provazovat tak záznamy z různých tabulek. Tyto vazby se realizují prostřednictvím atributů označovaných jako cizí klíč [6]. Hodnotou cizího klíče je pak hodnota primárního klíče tabulky, která je cílem vytvářené vazby. Z hlediska kardinality vztahu je možné rozlišit 3 základní případy *1:1* , *1:N* a *M:N* [1]. Vztah *1:1* vyjadřuje případ, ve kterém záznam jedné tabulky odpovídá záznamu jinému tabulky. Tento případ není příliš častý a většinou se řeší umístěním obou záznamů do jediné tabulky. Vztah *1:N* reprezentuje situaci, kdy se jednomu záznamu tabulky přiřazuje více záznamů jiné tabulky. Příkladem takové vazby může být vztah záznamů tabulek *Person*

a *Experiment*. Jedna osoba tedy může provádět *N* experimentů, ale každý experiment je prováděn právě jednou osobou. V modelu je tato informace realizovaná přidáním atributu např. zodpovědná osoba do tabulky *Experiment*. Tento atribut se tak stává cizím klíčem a ponese hodnotu primárního klíče tabulky *Person* pro příslušný záznam cílové osoby. Samotná vazba tedy obsahuje informaci o zdrojové záznamu, cílovém záznamu a také název vazby který je reprezentován názvem atributu pro cizí klíč tabulky.

Posledním druhem vazby je vztah *M:N*, jež reprezentuje stav, kdy *M* záznamům jedné tabulky odpovídá *N* záznamů jiné tabulky. Tento stav je v relačním modelu řešen rozložením na dvě vazby *1:N* a *1:M*, které jsou pak realizovány vytvořením další tabulky, jejíž záznamy budou obsahovat cizí klíče obou tabulek. Pro realizaci takové vazby je tedy nutné vložit do databáze další tabulku a v případě potřeby rozlišení druhu jednotlivých vazeb je zapotřebí buď dalšího atributu jež by vztah popisoval, či pro každý druh vazby mít vlastní tabulku (což se týká i případu vazby *1:N*). Pokud by ovšem bylo zapotřebí popsat také vztahy mezi jednotlivými vazbami nastává problém, který je řešitelný velmi obtížně [6].

4.2 Datový model EEG/ERP portálu

Současná podoba relační databáze portálu je tvořena přibližně 80 tabulkami. Naměřená data experimentů (hodnoty EEG signálů) jsou uložena v podobě binárních souborů v tabulce *DATA_FILE*, spolu se popisnými údaji jako je název či datový typ mající podobu textového řetězce nebo čísla. Tyto popisné údaje lze označit jako metadata binárního souboru naměřených dat. Metadata scénáře experimentu uložená v tabulce *SCENARIO* obsahují popisné informace dalších binárních souborů, jež jsou uloženy v tabulce *SCENARIO_TYPE_NONXML* obsahující popis průběhu měření. Obě tabulky *SCENARIO* a *DATA_FILE* obsahují vazbu na tabulku *EXPERIMENT*, která obsahuje buď přímo metadata experimentu (nikoliv však souboru naměřených dat) nebo zajišťuje provázání s jinou tabulkou jež je nese. Z aktuálního modelu je patrné, že jeden experiment je popsán jedním scénářem, ale může obsahovat více (binárních) souborů naměřených dat.

Tabulka *EXPERIMENT* s klíčovými metadaty obsahuje také dvě vazby na tabulku *PERSON*, jež slouží ke zaznamenání měřené osoby a experimentátora (vlastníka experimentu). Z hlediska datového modelu není mezi těmito osobami rozdíl, neboť jsou všechny uloženy v jediné tabulce a tudíž jsou všechny

osoby popsány stejnou množinou atributů. Testované osoby například vůbec nemusejí potřebovat přístup do portálu, ale tabulka ve které jsou zaznamenány obsahuje atributy uživatelského jména a hesla (i když mohou nabývat hodnoty NULL). Stejná situace nastává i pro další referencované tabulky *ARTICLES*, *ARTICLES_COMMENTS* či *RESERVATION*. Datový model tak nikterak nezabraňuje, aby měřené osoby měly své články, přidávaly k nim komentáře nebo rezervovaly laboratoř (vychází se z předpokladu, že portál slouží pouze pro výzkumné pracovníky, nikoli pro testované osoby).

Obdobná situace nastává v případě tabulky *RESEARCH_GROUP*, sloužící ke sdružování osob do výzkumných skupin. Její vazby na tabulku *PERSON* zajišťují, že každá skupina má svůj název, popis a také vlastníka. Každá skupina může mít samozřejmě více členů, což zajišťuje rozkladová tabulka *RESEARCH_GROUP_MEMBERSHIP*, jež obsahuje i roli člena ve skupině (ve smyslu uživatelský práv). Samotná tabulka ovšem poměrně značně zasahuje do celého datového modelu, díky množství rozkladových tabulek sloužících ke kategorizaci hodnot specifických metadat experimentů ke vybraným skupinám. Díky tomu je možné, aby vybraná skupina mohla definovat pro své potřeby vybrané hodnoty pro určitá metadata – např. pro tabulku *HARDWARE* definovat množinu přístrojů, které ke svým výzkumům využívá. Díky rozkladovým tabulkám může navíc více skupin sdílet stejné hodnoty položek hardwaru. Tyto tabulky s dodatečnými metadaty experimentu jako například zmíněný *HARDWARE*, *SOFTWARE* či *WEATHER* sice popisují kontext měření, ale z hlediska datového modelu se jedná pouze o nekategorizovaná data vázaná (často nepřímo) na tabulku *EXPERIMENT*. Není tedy možné sdružit některé parametry měření do skupin – např. tabulky *PHARMACEUTICAL* a *DISEASE* označit jako medicínské parametry s specifikovat tak jednoduchou hierarchickou kategorii metadat experimentů.

Další problém, který řeší současný relační datový model jen částečně je dynamická rozšiřitelnost struktury dat (ve sémantickém smyslu). V průběhu vývoje portálu se postupně zvyšoval počet tabulek nesoucích specifická metadata experimentu a je i nadále rozšiřován. Pro případ vzniku potřeby popisu daného experimentu parametrem, jenž není definován jako atribut tabulky *EXPERIMENT* ani jinou samostatnou tabulkou, obsahuje databáze tabulky *EXPERIMENT_OPT_PARAM_DEF* a *EXPERIMENT_OPT_PARAM_VAL*. Ty umožňují zaznamenat další hodnoty ve tvaru parametr-hodnota a navíc je sdílet pro více skupin. Nevýhodou tohoto řešení je ale absence možnosti přidat danému parametru více klíčů (chceme-li se vyhnout změně tabulky přidáváním dalších atributů), či tyto údaje ještě více řetězit. Obdobná situace však nastává i v případě samostatných tabulek pro konkrétní metadata

experimentu.

Pro bližší představu o současném datovém modelu portálu je možné nahlédnout do přílohy A obsahující ERA model EEG/ERP portálu.

5 Datový model sémantického webu - DODELAT

6 Nástroje pro vývoj sémantického webu

Jak již bylo zmíněno v kapitole Sémantický web, jazyk RDF je jedním ze základních stavebních prvků technologie sémantického webu. Současná podoba tohoto jazyka vychází ze specifikace W3C [19] z února 2004. Za dobu uběhlou od publikace zmíněné specifikace do současnosti (2013) vznikla poměrně široká sada nástrojů od jednoduchých utilit až po komplexní softwarová řešení jež umožňují zpracovávat data v jazyce RDF resp. OWL. Rozsáhlý a vcelku přehledný seznam těchto nástrojů (k datu vzniku této práce celkem 183) dostupný ze zdroje [**?**] poslouží jako primární seznam pro výběr vhodného nástroje. Pro účely této práce se zaměříme na hledání nástroje jež slouží primárně jako persistentní úložiště RDF dat a zároveň poskytuje veřejné Jena/Sesame API pro komunikaci a umožňuje dotazování jazykem SPARQL nad daty. Další nutnou podmínkou je, aby nástroj poskytoval hybridní úložiště umožňující uchovávat jak relační data tak data v jazyce sémantického webu (zmíněné požadavky vyplývají z následující kapitoly). Tyto poměrně náročné podmínky ovšem splňují pouze dva nalezené nástroje, jejichž popis bude následovat.

6.1 Virtuoso

Virtuoso je hybridní databázový server umožňující práci nad více datovými modely. Nástroj poskytuje klasickou relační databázi, RDF databázi trojic, XML databázi a také dokumentově orientovanou databázi. Verze aplikace se zpoplatněnou licencí poskytuje oproti GPL licencované variantě navíc virtuální databázi a také replikaci dat. Virtuální databáze umožňuje zastřešit více databázových systémů jediným centrálním se kterým klient komunikuje. Ve výsledku tak umožňuje klientovi pracovat s více databázemi, které se mu transparentně jeví jako jediná. Díky tomu je například možné prostřednictvím jediného dotazu získat výsledek ze všech databází najednou. Replikace dat slouží k zajištění konzistence dat takovým způsobem, kdy při např. dvou databázích slouží hlavní databáze k zachycení aktuálních změn dat, které postupně zasílá vedlejší databázi jež obsahuje původní data. Díky tomu obsahuje vedlejší databáze stále konzistentní data neboť při nezdařeném zápisu aktualizace může zažádat hlavní databázi o jejich opětovnézaslání.

Databázový server poskytuje grafické webové rozhraní, které slouží ke komplexní administraci a také je možné jeho prostřednictvím zadávat libovolné SQL/SPARQL dotazy. Virtuoso implementuje nad relační databází engine poskytující funkcionalitu v podobě standardu SQL-92, podporu pro pohledy, standardní datové typy, procedury, kurzory a také transakce. SPARQL dotazy je možné zadávat stejnými prostředky jako SQL dotazy, či je dokonce vnořovat do SQL dotazů. Díky tomu je možné pracovat s relačním modelem dat a RDF modelem zároveň. Samotné dotazování a další komunikace klientské aplikace se serverem může být realizováno prostřednictvím driverů ODBC/JDBC nebo Java knihovnou poskytující Jena API. Neméně důležitou součásti podpory Java aplikací je i distribuce knihoven pro Hibernate (Hibernate driver a Hibernate dialect). Virtuoso také podporuje import souborů s RDF daty ve formátu N3, Turtle a také nejběžnější RDF/XML [8].

6.2 Oracle 11g - Oracle Spatial

Oracle Spatial je rozšířením databázového systému Oracle, jež je primárně určeno pro zpracování geoprostorových dat [**?**]. Toto rozšíření s sebou ovšem přináší také podporu pro sémantické technologie, která je klíčová pro tuto práci. Standardní instalace Oracle 11g release 2 (současná databáze EEG/ERP portálu), ani žádná z předchozích verzí tuto podporu standardně po instalaci neposkytuje. Její zapnutí je ale pouze záležitostí spuštění instalačního skriptu a následného vygenerování tabulkového prostoru pro systémové tabulky a všechna následně ukládaná data. V tomto tabulkové prostoru je již možné vytvořit sémantickou síť, která ponese všechny systémové RDF tabulky a pohledy, jež jsou nezbytné pro provádění SPARQL dotazů. Při práci s databází je zapotřebí mít k dispozici datový model, který ponese všechny RDF trojice, jež budou vytvářeny. Tento model bude vázán na tabulku, která ponese všechny vytvářené trojice. Do tohoto modelu je již možné zapsat první trojici [9]. Pro názornost jednoduchosti tohoto procesu bude uvede výše popsaný postup v podobě SQL příkazů [9].

```
1.  ?/md/admin/catsem11i.sql
2.  CREATE TABLESPACE rdf_tblspace DATAFILE
    '/oradata/orcl/rdf_tblspace.dat' SIZE 1024M AUTOEXTEND
    ON NEXT 256M MAXSIZE 16384M;
3.  EXECUTE SEM_APIS.CREATE_SEM_NETWORK('rdf_tblspace');
4.  CREATE TABLE family_rdf_data (id NUMBER,
```

```
    triple SDO_RDF_TRIPLE_S);
5.  execute SEM_APIS.create_sem_model('family',
   'family_rdf_data', 'triple');
```

První příkaz instaluje podporu sémantických modelů v databázi. Druhým příkazem je na disku vytvořen sobor, který ponese tabulkový prostor pro sémantickou síť a všechny datové modely. Jeho počáteční velikost je nastavena na 1 GB s možností dalšího zvětšení v případě vyčerpání dané kapacity. V tomto tabulkové prostoru je ve třetím kroku vytvořena sémantická síť, ve které je možné vytvářet jednotlivé datové modely. Ve čtvrtém kroku je vytvořena první tabulka se dvěma sloupci, kde se vyskytuje nový datový typ *SDO_RDF_TRIPLE*. Jedná se o datový typ trojice ve smyslu RDF, který v sobě obsahuje subjekt, predikát a objekt (dle očekávaného standardu). Posledním příkazem byl vytvořen datový model nad předchozí tabulkou, který poslouží pro ukládání konkrétních trojic. Samotné vkládání trojic je pak poměrně jednoduché, a je možné jej nalézt ve [9].

Po instalaci podpory sémantických technologií je již tedy možné využívat databázi Oracle jako úložiště RDF dat. Ve výsledku je tedy k dispozici hybridní databáze pro ukládání dat jak v relační podobě tak ve RDF jazyce. Nad zmíněnými sémantickými daty je možné se dotazovat jazykem SPARQL a to skrze stejné prostředky jako v případě jazyka SQL. V důsledku toho je také možné vytvářet vnořené či jinak kombinované dotazy z obou jazyků, stejně jako tomu bylo u nástroje Virtuoso. K sémantickým datům databáze je možné přistupovat samozřejmě stejnou cestou jako k relačním – tedy prostřednictvím ODBC/JDBC driveru, ale Oracle také dodává driver poskytující standardní Jena API pro Java aplikace. Databázový systém podporuje odvozovací pravidla jazyka RDF, RDFS i OWL a od verze Oracle 10g je navíc možné, aby uživatel definoval vlastní odvozovací pravidla. Při vytváření dotazu je pak možné určit jakou množinu odvozovacích pravidel (RDF / RDFS / vlastní pravidla či jejich kombinace) chceme do dotazu zahrnout a ovlivnit tím tak jeho výsledek [9].

Literatura

[1] Gilfillan Ian. *Introduction to Relational Databases*, http://www.databasejournal.com/sqletc/article.php/1469521/Introduction-to-Relational-Dat (přístup 16.2.2013).

[2] Kauler Jan. *Základní pojmy databáze*. České vysoké učení technické v Praze, webzam.fbmi.cvut.cz/szabozol/ISZ/Kauler/506.doc, (přístup 17.2.2013).

[3] Kauler Jan. *Relační datový model*. České vysoké učení technické v Praze, webzam.fbmi.cvut.cz/szabozol/ISZ/Kauler/508.doc, (přístup 17.2.2013).

[4] Kolena Jan. *EEG/ERP portal - reservation system for EEG/ERP laboratory*. ZČU, Plzeň, 2011.

[5] Zendulka Jaroslav. *Relační model dat*. Vysoké učení technické v Brně, http://www.fit.vutbr.cz/study/courses/DSI/public/pdf/nove/4_2.pdf, (přístup 5.2.2013).

[6] Kosek Jiří. *Jak pracují databáze na Webu*, http://www.kosek.cz/clanky/iweb/12.html, (přístup 15.2.2013).

[7] Pitner Tomáš Matulík Petr. *Sémantický web a jeho technologie*. Masarykova univerzita, 2004. http://www.ics.muni.cz/zpravodaj/articles/296.html, (přístup 6.2.2013).

[8] Openlink Software. *Virtuoso Universal Server*, http://virtuoso.openlinksw.com/, (přístup 19.2.2013).

[9] Oracle. *Oracle Database Semantic Technologies Tutorial*, http://download.oracle.com/otndocs/tech/semantic_web/pdf/oradb_semtech_tutorial.pdf, (přístup 19.2.2013).

[10] Brůha Petr. *EEG/ERP portál a prostředky sémantického webu.* ZČU, Plzeň, 2011.

[11] Hanyáš Petr. *Sémantický web - tutotiál a demonstrační příklady.* Vysoké učení technické v Brně, 2007. http://www.hanyas.net/download/soubor: bp_cesky.pdf, (přístup 6.1.2013).

[12] Semantic web. *Ontology*, 2012. http://semanticweb.org/wiki/Ontology, (přístup 9.2.2013).

[13] Svátek Vojtěch. *Sémantický web*, 2010. http://nb.vse.cz/ svatek/rzzw/seweb-prehled.pdf, (přístup 10.2.2013).

[14] Svátek Vojtěch. *SPARQL*, 2010. http://nb.vse.cz/ svatek/rzzw/SPARQL11.2.2013).

[15] Zanlostní technologie. *Univerzita Hradec Králové*, 2006. http://lide.uhk.cz/fim/ucitel/fshusam2/lekarnicky/zt1/zt1_kap02.html, (přístup 9.2.2013).

[16] Štencek Jiří. *Užití sémantických technologií ve značkovacích jazycích.* Vysoká škola ekonomická v Praze, 2009. http://vse.stencek.com/semanticky-web/, (přístup 4.2.2013).

[17] Vitvar Tomáš. *Sémantický web.* ČVUT, Praha, 2011, http://www.cvut.cz/pracoviste/odbor-rozvoje/stranky/habilitace-a-inaugurace/habilitacni-p

[18] Homola Vladimír. *Kartézský součin, relace.* Vysoká škola báňská, http://homel.vsb.cz/ hom50/SLBSTATS/MNO/GS01B.HTM, (přístup 17.2.2013).

[19] W3C. *RDFConcepts and Abstract Syntax*, 2004. http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/, (přístup 7.2.2013).

[20] W3C. *OWL Web Ontology Language Overview*, 2004. http://www.w3.org/TR/owl-features/, (přístup 10.2.2013).

[21] W3C. *RDF/XML Syntax Specification*, 2004. http://www.w3.org/TR/REC-rdf-syntax, (přístup 6.2.2013).

[22] W3C. *RDF Vocabulary Description Language 1.0*, 2004. http://www.w3.org/TR/rdf-schema/, (přístup 8.2.2013).

[23] W3C. *W3C Semantic Web Activity*, 2006.
http://www.w3.org/2001/12/semweb-fin/w3csw, (přístup 8.1.2013).

[24] W3C. *SPARQL Query Language for RDF*, 2008.
http://www.w3.org/TR/rdf-sparql-query, (přístup 8.2.2013).

Priloha A - ERA model portalu