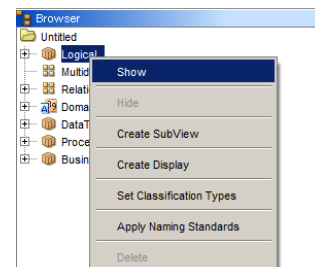


## Poznámky k používání nástroje Oracle SQL Developer Data Modeler

Dovolil jsem si sepsat pár užitečných informací k používání nástroje Oracle SQL Developer Data Modeler (dále jen Oracle DDM) před druhým zápočtovým testem z konceptuálního modelování. Doporučuju projít si tyto 4 stránky. *Ano, minimalizoval jsem okraje, ale je tady celkem dost obrázků.* Určitě si vše sami vyzkoušejte.

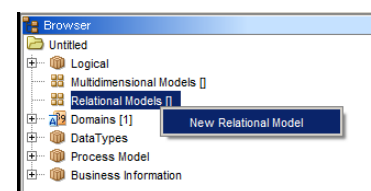
### Logický a relační datový model

Logický model odpovídá konceptuálnímu modelu, tzn. explicitě v něm neuvádíme žádné cizí klíče. Relační model upřesňuje logický model rozkladem vazeb M:N a doplněním právě atributů pro vyjádření cizích klíčů. Každý projekt v Oracle DDM obsahuje jeden logický model a jeden nebo více relačních modelů. Jednotlivé modely je možné otevírat zvlášť na záložkách.




Obrázek 1: Zobrazení koncept. modelu

Je možné, že v projektu nebudou založeny žádné relační modely, pak jednoduše nějaký vytvoříme. V relačním modelu provádíme pokud možno co nejméně zásahů, vše se snažíme namodelovat v logickém modelu, ze kterého pak relační necháme vygenerovat.



Obrázek 2: Vytvoření relačního modelu

### Entitní typ

Vytváříme konceptuální model, tzn. máme otevřenou záložku „Logical“. Klepnutím na ikonu  a následně na pracovní plochu diagramu vytvoříme nový entitní typ. Každý entitní typ je minimálně vhodné rozumně pojmenovat. Poté se ze sekce „General“ přesuneme na sekci „Attributes“, kde vyplníme atributy.

|   | Name         | Data type     |
|---|--------------|---------------|
| 1 | id_kategorie | Integer       |
| 2 | nazev        | NVARCHAR (50) |
| 3 | id_produkt   | Integer       |

**Attribute Properties**

Name:

Datatype:  Domain  Logical  Distinct

Structured  Collection

Type:  Preferred

Size:

Units:

Primary UID  Relation UID  Mandatory

Comments  Comments in RDBMS  Notes

Nový atribut

Název atributu

Nutno označit před výběrem typu z comboboxu níže

Výběr datového typu


U některých typů jako je např. NVARCHAR je potřeba specifikovat velikost

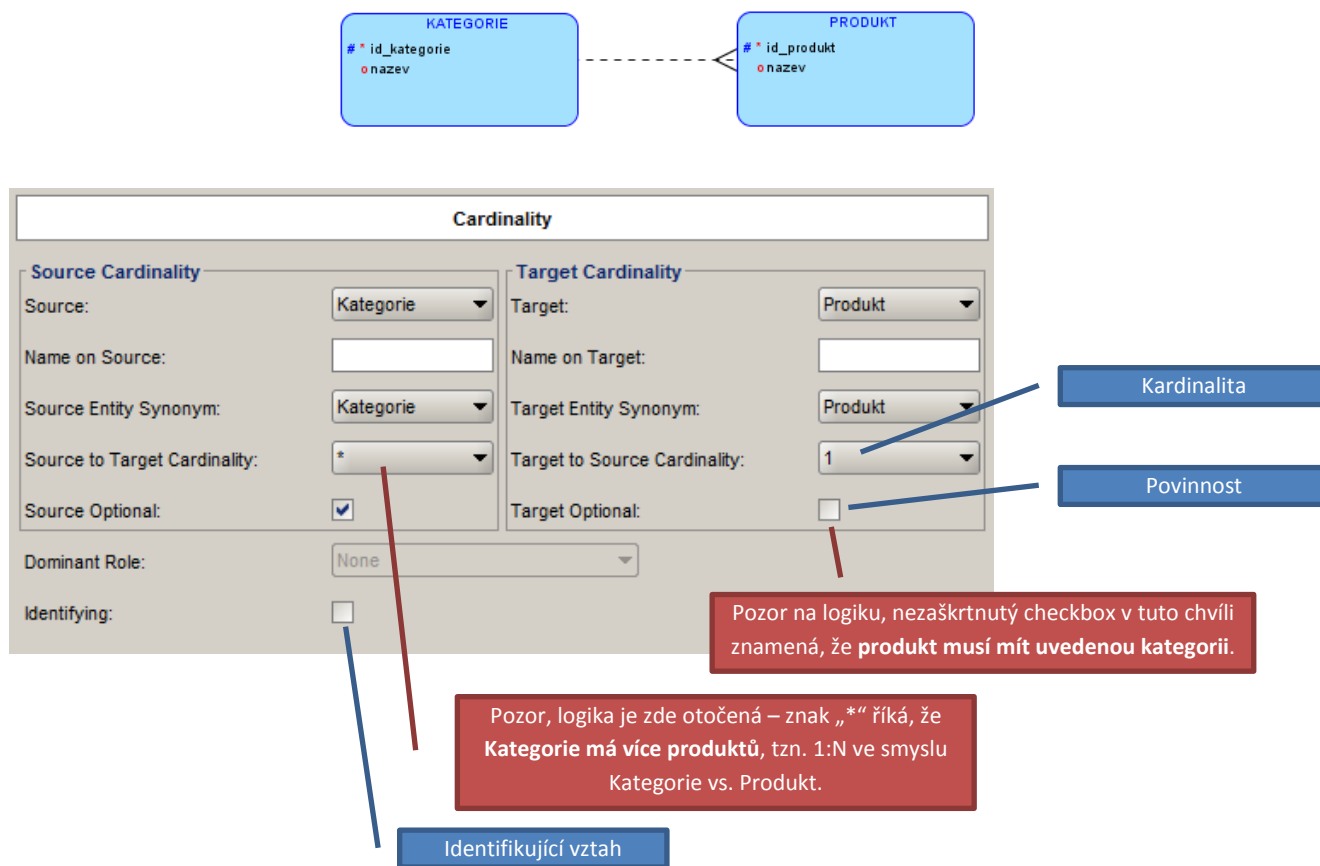
Povinný atribut

Atribut je součástí klíče

Obrázek 3: Atributy entitního typu

## Vztah

Vztah založíme klepnutím na jednu z ikon . Všechny ikony vytvoří vztah s určitým předdefinovaným nastavením. Později je možné libovolně upravit kardinalitu, povinnost členství nebo identifikující vztahy. Vztah je také vhodné pojmenovat, zejména pokud jde o M:N. Měli bychom vědět, že M:N se v relačních databázích řeší rozkladovou tabulkou. Oracle DDM rozkladovou tabulku pojmenuje právě podle názvu vztahu. Po pojmenování vztahu se přesuneme do sekce „Cardinality“. Na panelu pak máme dvě skupiny ovládacích prvků „Source“ a „Target“. Jedná se o vlastnosti prvního nebo druhého konce vztahu.



**Kardinalita**

**Povinnost**

Pozor, logika je zde otočená – znak „\*“ říká, že **Kategorie má více produktů**, tzn. 1:N ve smyslu Kategorie vs. Produkt.

Pozor na logiku, nezaškrtnutý checkbox v tuto chvíli znamená, že **produkt musí mít uvedenou kategorii**.

**Identifikující vztah**

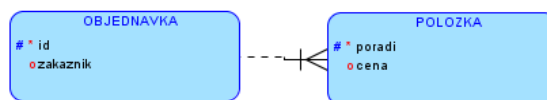
Obrázek 4: Vlastnosti vztahu

### Identifikující vztahy

Vytvoření každého vztahu 1:N (ať už identifikujícího nebo neidentifikujícího) v konceptuálním modelu zajistí, že na straně N v relačním modelu přibude atribut – cizí klíč. Typicky na obrázku 4 výše u produktu přibude atribut s číslem kategorie, kam produkt patří. Co tedy udělá identifikující vztah? Identifikující vztah navíc zahrne nově vzniklý atribut do klíčových atributů.

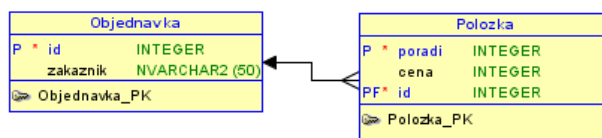
Kdy se identifikující vztah používá? Identifikující vztah se používá k provázání silného a slabého entitního typu. Vlastností slabého entitního typu je, že má složený klíč z {klíčů jednoho nebo více „nadřazených“ entitních typů} a {případně nějakých vlastních atributů}. *Doufám, že význam složených závorek je jasný ☺*. Složený klíč nepřimo vyplývá z faktu, že slabý entitní typ charakterizuje objekty, jejichž samostatná existence nemá smysl.

Modelovým příkladem je objednávka a položka objednávky. Objednávka bude jednoznačně určena svým id, položka bude určena id objednávky a pořadím na objednávce. Položka objednávky samostatně existovat nemůže – bude mít složený klíč a jedná se tedy o slabý entitní typ.



Obrázek 5: Konceptuální model s identifikujícím vztahem

Objednávku propojíme s položkou identifikujícím vztahem (obr. 5), který zajistí, že součástí položky v relačním modelu (obr. 6) bude atribut id zároveň jako cizí klíč do tabulky objednávek a zároveň jako součást primárního klíče položky. V položce tedy nebudeme ručně vyplňovat atribut id objednávky (byl by to cizí klíč a ten v konceptuálním modelu nesmí být), jediným klíčovým atributem položky v konceptuálním modelu bude její pořadí. Oracle DDM sám přidá cizí klíč do relačního modelu.



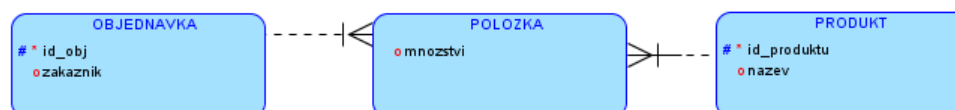
Obrázek 6: Relační model vzniklý použitím identifikujícího vztahu. Všimněte si hlavně označení "P" u atributu id v tabulce Polozka

### Rozklad vztahu M:N

Jak už bylo řečeno, vztah M:N je rozložen vždy pomocí rozkladové tabulky, která přebere všechny klíčové atributy z entitních typů, které spojuje, a tyto atributy utvoří její primární klíč. Pokud nerozumíte principu rozkladu M:N, doporučuji např. materiály z přednášek. Jestliže ze zadání víme, že dva entitní typy budou provázány vztahem M:N bez žádných doplňujících informací, pak v konceptuálním modelu jednoduše použijeme vztah M:N a necháme Oracle DDM, ať si s ním poradí.

Jsou ale případy, kdy to tak jednoduché nebude. Např. máme objednávku a produkty. Víme, že objednávka může zahrnovat více produktů a je asi pochopitelné, že produkt se může nacházet na více objednávkách. Teď se jedná o typický vztah M:N. Menší problém nastane, pokud budeme chtít navíc uchovávat informaci o počtu produktů na objednávce. Tzn. ne jen, že produkt je na objednávce, ale že ho tam je třeba 100 ks. Kam tedy atribut o počtu kusů umístíme? Určitě ne do objednávky a určitě ne přímo do produktu.

V tuto chvíli si v podstatě sami provedeme už v konceptuálním modelu jakýsi rozklad vztahu M:N tím, že zavedeme entitní typ např. „Položka objednávky“. Položka bude jednoznačně určena id objednávky a id produktu, ale pozor, id objednávky ani id produktu nebudeme vyplňovat ručně, použijeme identifikující vztahy. Jediným atributem, který do položky objednávky doplníme, bude právě množství (obr. 7).



Obrázek 7: Použití slabého entitního typu místo vztahu M:N

Nejedná se o nic složitého, pouze je potřeba si uvědomit, že mezi rozkladovou tabulkou a slabým entitním typem je někdy poměrně tenká hranice a pokud víme, že v průnikové tabulce bude nějaký doplňující atribut, nevystačíme si jednoduše s vztahem M:N. Není tedy chyba, pokud se už v konceptuálním modelu díváme na vztah M:N jako na dva vztahy 1:N a N:1 a „rozkladovou tabulku“ zavedeme už v konceptuálním modelu slabým entitním typem.

### Vygenerování relačního modelu z konceptuálního

Máme aktivní záložku „Logical“ a klepneme na ikonu . Ponecháme vše vybrané a klepneme na „Engineer“.

## Generování DDL skriptů

Máme vygenerovaný relační datový model, z menu vybereme View -> DDL File editor, v combo boxu zvolíme „SQL Server 2005“ a klepneme na „Generate“. V případě chyb doporučuji zkontrolovat, zda máme u všech atributů uvedené datové typy.

## Příklad

Nakonec malý test, zda jste pochopili význam identifikující vazby, resp. slabých entitních typů a jak je můžeme využít při rozkladu vztahu M:N.

Chceme vytvořit systém pro evidenci produktů, zákazníků a nákupů. U zákazníka evidujeme jeho jedinečné číslo, jméno, pohlaví, rok registrace, e-mail a informaci, zda zákazník chce odebírat reklamu. U produktu sledujeme opět jeho jedinečné číslo, označení, obchodní značku a rok ukončení výroby. Jeden zákazník si může koupit více produktů a jeden produkt si samozřejmě může koupit více zákazníků. Dále chceme evidovat, kolik kusů produktu zákazník koupil a za jakou cenu.

Pokud si chcete zkontrolovat řešení, zkuste se podívat na tabulky prvního zápočtového testu na SQL SELECT.

Může v tuto chvíli koupit jeden zákazník stejný produkt vícekrát? Pokud náhodou ne, zkuste provést malou úpravu v modelu tak, aby to bylo možné.