# Database Exercises – Design

|  |  |
| --- | --- |
| **Exercise** | DBDesign.1 |
| **Model** |  |
| **Purpose** | Use rules for converting a Domain Model to a set of tables. |
| **Description** | The given Domain Model should correspond fairly closely to the model discussed in the **RelMod.1** exercise. |
| **Steps** | 1. Use the relevant step(s) for converting the given Domain Model into a set of tables. |

|  |  |
| --- | --- |
| **Exercise** | DBDesign.2 |
| **Model** |  |
| **Purpose** | Use rules for converting a Domain Model to a set of tables. |
| **Description** | The game designers have discovered a flaw in the previous Domain Model!  They discovered that they also need to have a way to repre­sent a single, physi­cal weapon. This changes the Domain Model, as shown above.  **Weapon** now represents a single, physical instan­ce of a weapon, while **Weapon­Model** represents a general model of a weap­on, of which many physical instan­­ces can exist. For now, all instances of a specific model are however still identi­cal, but this may now change later. |
| **Steps** | 1. Use the relevant step(s) for converting the given Domain Model into a set of tables. 2. Compare the resulting tables with the resulting tables from the previous exercise (**DBDesign.1**). What are the differences? How do the differences relate to the changes in the Domain Model? |

|  |  |
| --- | --- |
| **Exercise** | DBDesign.3 |
| **Model** |  |
| **Purpose** | Use rules for converting a Domain Model to a set of tables. |
| **Description** | The game designers now introduce the concept of **jewels** into the game. Jewels can be put into jewel sockets on weapons, thereby increasing the damage dealt by the weapon. Jewels are defined as follows:   1. A number of **jewel models** exist. Each jewel model contains a name, a rarity, and a value for added damage. 2. A number of **physical jewels** exist. A physical jewel will always be an instance of a jewel model. 3. Many physical jewels of the same jewel model can exist. All physical instances of a specific jewel model are identical.   This causes the Domain Model to change, as shown above. |
| **Steps** | 1. Use the relevant step(s) for converting the given Domain Model into a set of tables. 2. How many extra tables did you need, as compared to the result from the previous exercise (**DBDesign.2**)? |

|  |  |
| --- | --- |
| **Exercise** | DBDesign.4 |
| **Model** |  |
| **Purpose** | Use rules for converting a Domain Model to a set of tables. |
| **Description** | The game designers now introduce the concept of **matching**. Matching covers the idea that a jewel model and a weapon model form a **match** (or mismatch), which results in a modification of the damage added by socketing a jewel of a given jewel model onto a weapon of a given weapon model. This modification is expressed as a **match factor**, which is a decimal number, e.g. **1.2**.  **Example**: A jewel instance of a jewel model with **AddedDamage** = 25 is added to a weapon instance of a weapon model for which the match factor is **1.2** for the given jewel model. The total damage added by socketing the given jewel on the given weapon is therefore 25 x **1.2** = 30. |
| **Steps** | 1. Use the relevant step(s) for converting the given Domain Model into a set of tables. 2. How is the new table different from the existing tables (i.e. the tables resulting from solving **DBDesign.3**)? 3. [Difficult] The game designers have defined that if no match factor is defined for a weapon/jewel model pair, the match factor is 1. Could this choice be problematic with regards to implementation? |

|  |  |
| --- | --- |
| **Exercise** | DBDesign.5 |
| **Model** |  |
| **Purpose** | Use rules for converting a Domain Model to a set of tables. |
| **Description** | The game designers now introduce **jewel cut quality**, defined as follows:   1. A jewel can be cut in one of these qualities: **Poor**, **Average**, **Good** or **Excellent**. 2. A jewel cut quality applies to **individual jewels**. 3. The cut quality modifies the damage added by the jewel with a factor. The factors are: **Poor**: 0.8, **Standard** 1.0, **Good** 1.2, **Excellent** 1.5.   **Example**: A specific jewel has AddedDamage = 30 (implied by the model of the jewel). The specific jewel has a cut quality of **Good**. The total damage added by socketing the given jewel on the given weapon is therefore 30 x **1.2** = 36. Note that the damage factor may increase this number further. |
| **Steps** | 1. **NB**: The given Domain Model is identical to the Domain Model given in the exer­cise **DBDesign.4**! Before proceeding, you should update the Domain model to include jewel cut quality. 2. Use the relevant step(s) for converting the updated Domain Model into a set of tables. |

|  |  |
| --- | --- |
| **Exercise** | DBDesign.6 |
| **Model** |  |
| **Purpose** | Use rules for converting a Domain Model to a set of tables. |
| **Description** | As a final change, the game designers decide that a character can own seve­ral weapons, i.e. not just the (up to) two weapons held in hands. This results in the addition of the relation **owns** between **Character** and **Weapon**. |
| **Steps** | 1. How does this addition affect the definition of the corresponding tables? 2. The model is now considered complete. We do however also need to consider potential problems concerning **referential integrity** and **busi­ness logic**.    1. What rules should be enforced in order to ensure **referential integrity**? To start with, think about what (if any) problems might be caused by deleting a row from each of the tables.    2. What rules should be enforced in order to ensure that the **busi­ness logic** is notviolated (hint: which weapons can you use)? |

|  |  |
| --- | --- |
| **Exercise** | DBDesign.7 |
| **Model** |  |
| **Purpose** | Use rules for converting a Domain Model to a set of tables. |
| **Description** | The above Domain Model is a model for an online recruitment tool. It contains three conceptual entities:   * **Person**: A person can create a profile on the website, including a name and e-mail address (a real system will probably contain additional data) * **Company**: The system will contain representations of various companies, and a person can then specify on his profile that he has worked for spe­cific companies during specific time periods. * **Competence**: The system also contains representations of various com­pe­tences, e.g. “programming”, “accounting”, etc.. A person can then specify on his profile that he has a number of specific competences.   The model also contains three relations. The Person-Competence and Person-Company relations should be straightforward. The Person-Person relation **recommends** is a bit special, however. The idea is that one person can recom­mend another person. Both persons must have profiles in the system. |
| **Steps** | 1. Work through the entire process of creating a set of tables from the given Domain Model, including considerations for referential integrity and business logic. 2. Take a closer look at the representation of the three relations. Could it be relevant to add additional attributes to the relations them­selves (how did you e.g. represent the period a specific person has worked for a spe­cific company)? 3. Suppose that the recommendations are no longer just general recom­menda­tions, but rather recommendations on a specific compe­tence. How would that change the Domain Model? And the tables? |