# Database Exercises – Relational Data Model

|  |  |
| --- | --- |
| **Exercise** | RelMod.1 |
| **Purpose** | Get started on defining a relational data model for a specific scenario. |
| **Description** | As some of the very first steps towards building a data model for a game, we consider two conceptual entities **Weapon** and **Character**.  The game designers describe a **Weapon** in this way:   * A **Weapon** represents a “weapon model”, i.e. a specification of a general weapon type, for which several physical instances can exist. All instances of a weapon type will, however, be completely identical. * A **Weapon** has a **name**, a **type** and a **rarity**. Examples of **type** are Gun, Sword and Mace. Examples of **rarity** are Common, Rare and Epic. * A **Weapon** has an **item level**. An item level is a positive integer value. * A **Weapon** has a minimal damage and a maximal damage. Both are measured in **damage points**, which are positive integer values. * A **Weapon** has 0, 1 or more jewel sockets, i.e. an integer value * A **Weapon** is either one-handed or two-handed.   The game designers describe a **Character** in this way:   * A **Character** has a **name**, a **race** and a **role**. Examples of **race** are Human, Elf and Dwarf. Examples of **role** are Wizard, War­rior and Hunter. * A **Character** has a number of **health points**. Health points is an integer value. * A **Character** may carry weapons in his/her hands. Both hands (left and right) can carry weapons. A one-handed weapon can be carried in one hand (left or right), while two-handed wea­pons require both hands. A **Character** can thus carry 0, 1 or 2 one-handed weapons, or 0 or 1 two-handed weapons.   *(Exercise continues on the next page)* |

|  |  |
| --- | --- |
| **Exercise** | **RelMod.1** [Continued] |
| **Steps** | 1. Create a table definition for **Weapon**. Consider:    1. What **columns** should be included?    2. What should the **domain** for each of the columns be (i.e. what are valid **values** for each column)?    3. Which column(s) will be a **primary key** for the table?    4. Can some of the columns contain ***null*** values? 2. Also create a table definition for **Character**, following the same conside­ra­tions as for **Weapon**. However, here you also need to consider how to model the relation between **Character** and **Weapon** (hint: foreign keys). 3. Consider how to ensure **referential integrity** between **Character** and **Weapon** rows. More specifically: do we need to put some constraints on some of the columns in **Character**, with regards to the values they may contain? 4. What should happen in case a **Character** row is deleted? What should happen in case a **Weapon** row is deleted? 5. Consider how to ensure that the **business logic** (i.e. the rules described above) for the game is maintained. This could e.g. be by placing further constraints on certain columns. It may also involve defining rules for values from multiple columns. 6. [Difficult] Implement your entire data model (including constraints and business logic) in C#. This will probably involve creating (at least) two classes **Character** and **Weapon**. |