Adatbázis normalizálás:

1. első normálforma
   1. összetartozó adatok egy táblába kerüljenek;
   2. az egyedi adatelemeket kulccsal jelöljük meg;
   3. szűrjük ki az ismétlődő adatokat.

Ennek megfelelően kell egy IMAGE tábla a megoldandó képek adatainak, kell egy USER tábla a játékosok adatainak, és kell egy HELP tábla a kitöltés során igénybe vehető segítségeknek. Az egyedi azonosítók ennek megfelelően az IMAGE táblában egy „*Id*” lehetne, a USER táblában a *„UserName”*, hisz ez egyedi, a HELP táblában pedig a segítségek típusai („*Type”*). Az IMAGE tábla önmagában létrehozható, hiszen idegen kulcsot nem kell tartalmaznia, és módosítani sem kell a játék során.

A HELP tábla a segítségek típusait, azoknak a kép pontszámára való hatását (szorzó), valamint a tokenben kifejezett árat kell tartalmaznia minden egyes típus esetében:



A USER tábla már nem ennyire egyszerű ugyanis abban rögzíteni kellene azt, hogy melyik segítségtípusból mennyi áll rendelkezésre a játékos számára, azt azonban már nem kell rögzíteni a USER táblában, hogy ezek mennyibe kerülnek, és milyen hatással vannak az épp kitöltött képre. Ugyanakkor az 1NF-nek megfelelne az is, ha a HELP tábla elsődleges kulcsait betennénk a USER táblába további oszlopokként.

A megoldott képeket is a felhasználóhoz csatoltan kellene tárolni, azonban egy-egy felhasználóhoz biztosan több megoldott kép tartozna, így ez nem elégítené ki az első normálforma 3. kívánalmát. A megoldás az lehet, hogy a megoldott képeket és azok állapotát, ha félbehagyott, akkor a félbehagyott állapotát egy külön táblában kell tárolni, amelynek egy sorában a „UserName”, a kép „Id”, a befejezettség és a pillanatnyi tartalom, illetve pontszám tartozna:



Ennek alapján pedig a USER táblában maradó adatok így néznének ki:



1. második normálforma (**2NF**):
   1. első normálformában van;
   2. és minden nem elsődleges mező (nem kulcs attribútum) teljesen függ a kulcstól.

Az 1NF alapján összeállított tábláink megfelelnek a 2NF-nek is, hiszen mindegyik tábla oszlopai teljesen a kulcstól függnek. Például a *„FullName”* attribútum közvetlenül a *„UserName”*-től függ, mivel a *„FirstName”* és *„LastName”* is közvetlenül a *„UserName”*-től függ. Tehát a *„FullName”*megfelel a 2NF-nek. Ugyanakkor felmerülhet a gyanú, hogy mégsem teljes mértékben elégíti ki a 2NF-t, hiszen az 1H…Erase attribútumok a HELP tábla elsődleges kulcsai, tehát attól is függni látszanak. Ez azonban nincs így, hiszen ezen oszlopok értékei teljesen a „UserName”-től függnek, ugyanis csak az egyes segítségtípusok darabszáma (ami a felhasználó rendelkezésére áll) van eltárolva bennük. Ugyanakkor felvethető, hogy túl sok attribútum kerül bele a USER tábla rekordjaiba ezzel, így célszerűbb lenne külön táblába rendezni ezeket. Ennek a táblának az egyedi kulcsa a „UserName” és minden segítség egy-egy oszlop lesz benne, és az ott szereplő szám adja meg, hogy a felhasználónak melyik segítségből mennyi áll rendelkezésére. Ebben a táblában lehet növelni az értéket vásárláskor, az összeget a USER tábla Tokens oszlopából levonni a HELP táblában tárolt ár alapján. Tehát ez a USERHELP tábla így nézne ki: 

1. harmadik normálforma:
   1. második normálformában van,
   2. nem tartalmaz tranzitív függést.

A USER tábla, amiről leválasztottuk a USERHELP táblát így néz tehát most ki:



Ez a tábla azonban nem felel meg a 3NF-nek, mert a 3NF megköveteli, hogy minden nem kulcs attribútum közvetlenül az elsődleges kulcstól függjön, és ne függjön más nem kulcs attribútumtól. Mivel a *„FullName”* függ a *„FirstName”* és *„LastName”* attribútumoktól, ez egy tranzitív függőséget hoz létre, ami megsérti a 3NF-et. Ez az adat azonban nyilvánvalóan redundáns is, hiszen semmi mástól nem függ, csupán a „*FirstName”* és *„LastName”* attribútumoktól, így egyáltalán nincs is szükség ezen adat eltárolására. Ha kivesszük a kérdéses attribútumot a USER táblából, akkor az már megfelel a 3NF-nek is, így az adatbázis minden táblája megfelel a harmadik normál formának. Az új USER tábla tehát így néz ki:

