Projektuppgift i Databasteknik | Samuel Palomaa

'catshelterDB'

Databasen representerar de boende på ett katthem, dvs hemlösa katter som väntar på adoption och blir omhändertagna tills dess.

Jag har inte inkluderat några separata användarinloggningar i koden, men det finns två olika vyer som kan tänkas vara tillgängliga för olika användare - den ena är ett exempel på information en arbetare på katthemmet behöver när katterna ska matas; den andra vyn är grundläggande info som kan vara tillgänglig för potentiella adoptanter.

Tabellerna som finns är:

Cats - "Profiler" av katterna som bor på hemmet; deras unika ID, namn, uppskattat födelseår, vare sig de är redo för adoption eller inte, samt vilken bur de bor i.

Cages - Burarna katter bor i, numrerade i stigande ordning fr o m 1.

FoodMenu - De olika sorters mat som finns, indelat efter märke och typ (blöt/torrfoder).

CatFood - Kopplar rätt mat till rätt katt.

Primary Keys: *Cats* har CatID som primary key och fungerar ungefär som kattens personnummer. Det behöver vara helt unikt för katten och försäkrar att andra nycklar kopplas rätt även om det t ex finns två katter med samma namn.

FoodMenu har ett FoodID som primary key, dvs identifikatorn för en specifik matkombination; i *CatFood* skapas en PK som kombinerar FoodID och CatID och, med hjälp av foreign keys, kopplar dem rätt.

I *Cages* finns bara ett värde (PK), vilket kopplas till ett motsvarande värde (CageNumber) i *Cats* via en foreign key.

1-M: Cages ↔ Cats: En katt kan bara bo i en bur; en bur kan innehålla fler katter.

M-M: Cats ↔ FoodMenu (via CatFood): En katt kan ha fler matalternativ (även om jag inte visat ett exempel i koden går det att testa) och ett specifikt matalternativ kan vara tillgängligt för fler katter.

Att ha Cages i sin egen separata tabell är praktiskt eftersom den berör information som har mer med organisation av katthemmet att göra, istället för t ex födelserår eller adoptionsstatus, som är strikt kopplat till kattens identitet. Katter kan behöva flyttas mellan burar på ett kontrollerat sätt som kan göras i den tabellen av arbetare.

FoodMenu innehåller mycket info som har att göra med vad som finns tillgängligt i matförrådet - ursprungsinformationen kommer kanske från tillgänglighet hos foderproducenter, eller vad som donerats mycket av. Yttre faktorer som inte nödvändigtvis kan kontrolleras av katthemmet. Då är det praktiskt att ha det separat.

CatFood är som sagt en mellanhandstabell och behöver vara separat för att fungera som den gör.

Databasen uppfyller 3NF eftersom informationen är uppdelad i olika tabeller (burar, mat) utan överflödiga upprepningar; varje fält är beroende av hela sin primärnyckel och ingen data riskerar att bli felaktig eller motsägelsefull vid uppdateringar.

För ID-nummer har jag använt standarden INT för att tillåta högre tal (när många katter flyttat in och ut, många matkombinationer som varit i rotation).

BirthYear och CageNumber har tilldelats SMALLINT istället eftersom årtal alltid kommer vara 4 tecken långt och det finns ett begränsat - lågt - antal burar på hemmet.

Strängarna har tilldelats VARCHAR med en generös men realistisk begränsning i relation till informationen. Något som kanske står ut är Adoptable som har VARCHAR(3) när det även skulle kunna varit en BOOLEAN - båda fyller syftet, men VARCHAR är snäppet mer flexibelt om man skulle få för sig att skriva något annat där (“TBA” kanske).

Dataintegriteten säkerställs med hjälp av primary och foreign keys (som kontrollerar att relationerna mellan tabellerna hanteras korrekt; t ex att en katt inte kan tilldelas ett matalternativ som inte finns i matmenyn), samt användandet av NOT NULL där det är av yttersta vikt att ett värde finns angivet.