SPACE DATABASE

Dmitriy Voronin & Marjorie Yeaple-Betancourt

CMSC 508 - Database Theory
Dr. Cano

Dmitriy Voronin and Marjorie Yeaple-Betancourt
Dr. Cano

Table of Contents

Dmitriy Voronin and Marjorie Yeaple-Betancourt
Dr. Cano

Phase II: Problem Definition and Database Design

1. Problem Statement

There is a contract out from NASA to help facilitate creativity and an interest in space amongst students.The Space Database will fulfill this goal, acting as a sandbox for the creation of solar systems, planets, and their satellites.

This sandbox will be targeted toward younger students to demonstrate the relationships between stars, the planets that revolve around these stars, and the satellites that revolve around the planets. Creating these objects will also encourage them to fully develop their understanding the characteristics of these objects. Stars will have their own date of discovery and classification as well as a temperature value . Additionally, the mass of planets and moons, as well as their distance from their center of orbit will be saved. To create a more detailed view, satellites can be either a moon or artificial, in which case whomever “launched” them into this universe and when can be input as well as the cost to do so.

Users of the database will be able to save their creations to accounts by associating their created stars with a password-protected profile. This will allow the user to save entire galaxies of solar systems and view others’, as well. Therefore, the Space Database will provide them with a fun way to interact with some of the most recent discoveries or creations made by their fellow researchers.

The primary goal of this sandbox is to make a place in space for the imagination of budding astronomers and their creations.

Theory
Dmitriy Voronin and Marjorie Yeaple-Betancourt
Dr. Cano

The Space Database will contain the entities: planets, stars, and satellites.

Profiles will have the attributes:

- Profile username
- Profile salt
- Profile password (hashed with SHA1)

Stars will have the attributes:

- The solar system that houses it
- Its temperature (in whole Kelvin)
- Its date of discovery (in the format j-M-y, i.e. 1-JAN-17)
- Who discovered it
- Its distance from Earth (in light years)
- Its classification, or spectral type, based on the MK system. This follows the format character, integer, character* (ie. A0IV where A is a character, 0 is an integer, I is a character, V is a character)

Planets will have the attributes:

- Name
- The solar system that houses it
- Its distance from the star it orbits(in whole kilometers)
- Its size (represented by its radius in whole kilometers)
- Who it was discovered by

Satellites will have the attributes:

- A unique identifier
- What planet it orbits
- Its distance from the planet it orbits (in whole kilometers)
- A type (Natural/Artificial)

Additionally, satellites will be either natural or artificial. Ergo:

Moons (natural) will have the attributes:

- a unique identifier
- its size (represented by its radius in kilometers)

Artificial satellites will have the attributes:

- a unique identifier
- who launched it
- when it was launched (in the format j-M-y, i.e. 1-JAN-17)
- its cost (in whole US dollar amount)

The database will include the above attributes for each entity. The summation will act as a fact sheet representing space's general information. Furthermore, the relationships between the entities reflect spatial relationships.

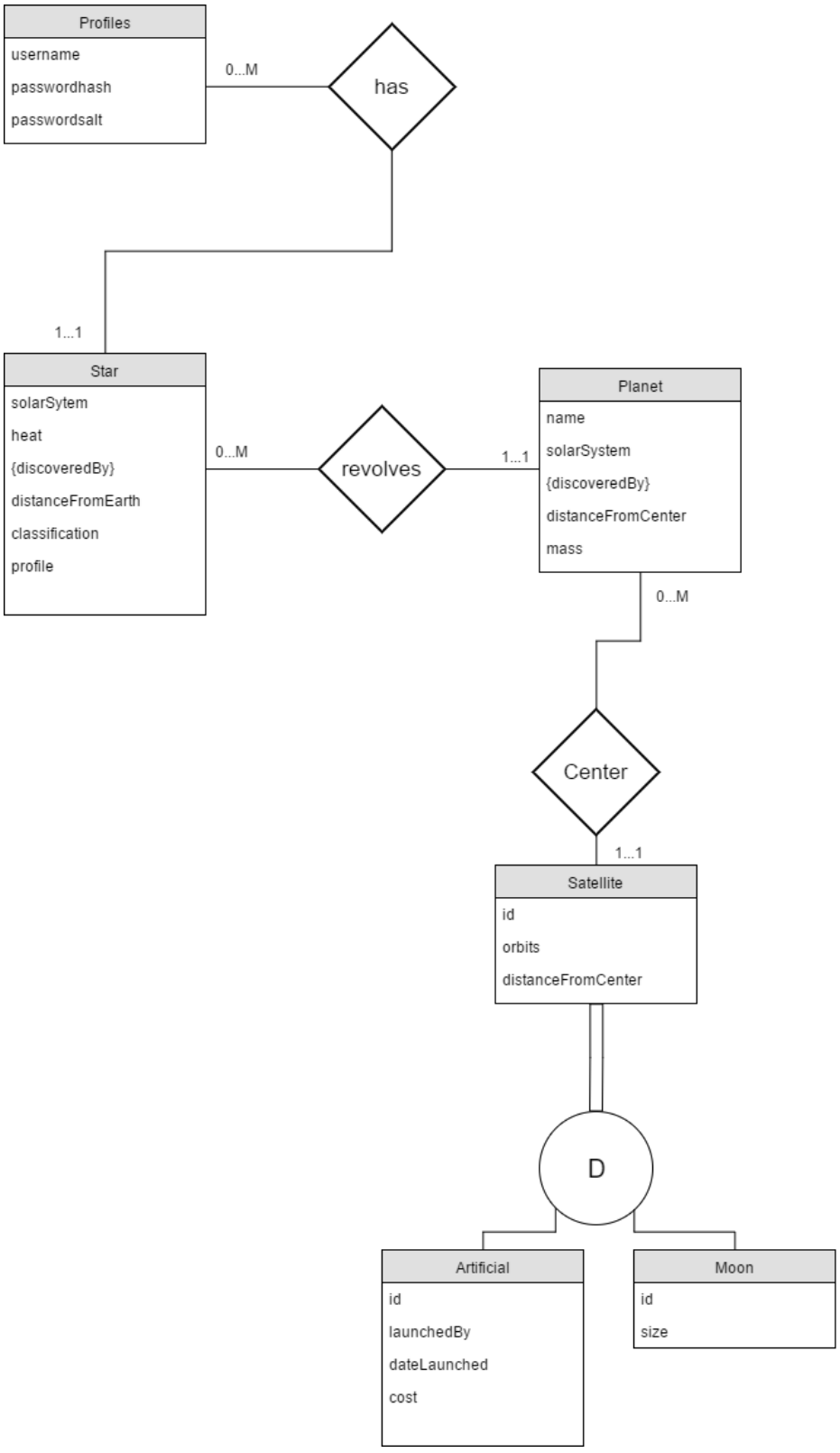
The following are the relationships and shared attributes between stars and planets that will be reflected:

- Planets revolve around a star
    - A star may have none or many planets revolving it
    - A planet will have only one star that it revolves around. This implies a planet is only part of one solar system

between satellites (natural or artificial) and planets that will be reflected:

- Satellites revolve around a planet
    - A planet may have none or many satellites
    - A satellite will have one planet

2. Entity-Relationship Diagram



3. Relational Model

Star	
PK	solarSystem
PK	userName
	heat
	distanceFromEarth
	classification
	profile

Profile	
PK	username
	passwordSalt
	passwordhash

PlanetDisc	
PK,FK	planetName
	pioneer

Planet	
PK	name
FK	solarSystem
	distanceFromCenter
	mass

StarDisc	
PK,FK	starName
	pioneer

Satellite	
PK1	id
FK1	orbits
	distanceFromCenter

Moon	
PK,FK	id
	size

Artificial	
PK,FK	id
	launchedBy
	dateLaunched
	cost

Dmitriy Voronin and Marjorie Yeaple-Betancourt
Dr. Cano

Relational Model Cont. (Attributes with types, domain, and constraints PKs and FKs (dashed underline):

STAR
Star.solarSystem VARCHAR2(20)
Star.heat VARCHAR2(10)
Star.dateDiscovered
Star.distanceFromEarth
Star.classification

STARDISC
Stardisc.starName
Stardisc.pioneer

PLANET
Planet.name
Planet.solarSystem
Planet.distanceFromCenter
Planet.size

PLANETDISC
Planetdisc.planetName
Planetdisc.pioneer

PROFILE
Profile.username
Profile.passwordSalt
Profile.passwordHash

SATELLITE
Satellite.id
Satellite.orbits
Satellite.distanceFromCenter
Satellite.type

ARTIFICIAL
Artificial.id
Artificial.launchedBy
Artificial.dateLaunched
Artificial.cost

MOON
Moon.id
Moon.size

4. Functional Dependencies

Functional dependencies include

(1)Profile.username→
Profile.passwordHash
Profile.passwordSalt

(2)Star.solarSystem →
Star.heat
Star.dateDiscovered
Star.distanceFromEarth
Star.classification

(3)StarDisc.planetName →→
StarDisc.discoveredBy

(4)Planet.name →
Planet.solarSystem
Planet.distanceFromCenter
Planet.size

(5)PlanetDisc.planetName →→
PlanetDisc.discoveredBy

(6)Satellite.id →
Satellite.orbits
Satellite.distanceFromCenter
Satellite.type

(7)Artificial.id →
Artificial.launchedBy
Artificial.dateLaunched
Artificial.cost

(8)Moon.id →
Moon.size

5. Normalization

The canonical cover of all attributes is the concatenation of all primary keys in each relation. If you provide the attributes:

Profile.username
Star.solarSystem
Planet.name
Satellite.id

You will receive a unique tuple if the whole diagram was merged into one relation. However, due to the dependencies we can minimize redundancies and create multiple tables utilizing normalization.

All relations satisfy 1st normal form because all attributes are atomic. Each attribute maintains a domain specific to that attribute.

All relations satisfy 2nd normal form because the candidate key and original single relation:

Profile.username
Star.solarSystem
Planet.name
Satellite.id

was broken up into multiple relations as seen in the relational model above. This decomposition created individual candidate keys for each relation. Now all relations satisfy all FDs and there are no partial dependencies in the functional dependencies. This generated a total of four relations.

All relations satisfy 3NF--BCNF. This is because there are no transitive functional dependencies.

All relations did not satisfy 4NF because there could have been more than one discoverer of a particular planet and or star. Due to this functional dependencies, the STAR and PLANET relation was decomposed further to satisfy 4NF. The resulting relations include PLANET (with omitted “discoveredBy” attribute), STAR (with omitted discoveredBy attribute), PlanetDisc, and StarDisc. The resulting candidate keys for each decomposed relation is PLANET.name for PLANET, STAR.solarSystem for STAR, PLANETDISC.name&&PLANETDISC.discoveredBy for PLANETDISC, and lastly STARDISC.solarSystem&&STARDISC.discoveredBy for STARDISC. The final decomposition and set of relations included nine relations. These nine relations satisfy the five single variable functional dependencies and two multivariable functional dependencies.

6. Example Data

Star

solarSystem	heat	distFromEarth	classification
Home	5778	0.00001581	G2V
Carina	6998	310	F0II
Aldebaran	3910	65	K5III
Piscis	8590	25	M4V
Hamal	4480	65.8	K2III

Planet

name	solarsystem	size	distanceFromCenter
Earth	Home	6371	1496000
Bespin	Anoat	118000	1250000
Coruscant	Carina	12240	1482000000
Geonosis	Arkanis	11370	1320000000
Saturn	Home	58232.503	1429000000

Satellite

id	orbits	distanceFromCenter	type
Moon	Earth	384400	natural
Europa	Jupiter	780000000	natural
Ganymede	Jupiter	2631	natural
Enceladus	Saturn	238000	natural
Rishi	J1407b	3250	natural
Titan	Saturn	1221870	natural
DS-1 Orbital Battle Station	Geonosis	1737	artificial

Moon

id	size
Moon	1737
Europa	1500
Ganymede	2631
Callisto	2410
Io	1821
Titan	2576
Enceladus	252
Rishi	1500

Artificial

id	launchedBy	dateLaunched	cost
DS-1 Orbital Battle Station	Darth Vader	07012222	1000000000000
Cassini	Jet Prop. Laboratory	10151997	3260000000

Profile

username	passwordSalt	passwordHash
yeaplebetamn	b295d117135a9763da282e7dae73a5ca7d3e5b11	aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d
ABCDE	5d4e622b482f04417662364ae38beb79	04875C6D3735E30B1897B3BD057098FEAC873F3F
veroninda	27b1e31486f11da03843548827a29a14e36b22a7	68d5fef94c7754840730274cf4959183b4e4ec35

7. Interface Documentation

Space Database's Interface uses HTML, CSS, and PHP to display and create new stars, satellites and planets as well as new profiles. It's homepage, index.php, shows the full directory of the website's functions, including links to input forms for each of the entities. Additionally, in the directory banner in the upper right of the website, links are provided to view fully each entity's data currently stored in the Space Database. Views were created to display Moon and Artificial complete data, meaning a join of Satellite on Moon and Satellite on Artificial to display each individual Satellite's full details.

Navigation of the website is done through links using HTML. Inputs and queries are passed from HTML web forms to php "handlers" that query and insert into the database. Queries are done when displaying the database in the aforementioned links. Queries are also done when using the insert forms for Planet (input_planet.html, input_planet_handler.php), Moon (input_moon.html, input_moon_handler.php), and Artificial (input_artificial.html, input_artificial_handler.php) to query the database for existing, viable foreign keys. So, for the Moon input form, when providing its foreign key, orbits (for the planet it orbits), a livesearch is implemented to query the database for existing Planets and no value will be inserted if an existing Planet is not input.

Additionally, the login function, while not fully implemented interface-side, is set up completely within the SQL database. This was done by adding a foreign key attribute to the Star entity that is the primary key of the Profile entity. This allows each star to be linked to a user's "profile" which will then be secured with a password that is encrypted upon creation in the database. The website will verifies through the use of a global variable that is checked when altering the database.