DBMS Project Design Document

Student Name: Feng Sun, Erick Muller
Contact Info:
Date: 4/2/2015

Table of Contents

This document will be an "evergreen" document. You will keep it recent, and include it (in .docx and PDF form) with every project submission. If you have a way to indicate in the margins what is new or changed relative to the previous version, that would be appreciated, but it's not required.

Your project design document will have the following sections. Please make sure to use the same section labels and numbers as presented here! You are free to add subsections as you want.

1. Project Information

Provide the following information:

- The last modified date of the design document: 4/2/2015
- Author name and contact info: Feng Sun, Erick Muller
- The project name: LOL Champions Database

2. Topic Description

This section gives a high level introduction about your project (About 250 words). It should answer the following questions:

- What is the domain of the project?

  This database is about the current popular game, league of legends. Basically it will have the information for champions, equipments, skills and etc. It will provide user detail information about champions, skills, equipments are if they want to search. Other than that, we may also provide some suggestion item builds to user.

  Some domains could be champions, skills, equipments, haveSkillOf. chamType, equipType and etc. Most of these will be character arrays in order to fit the names of the different items or equipment as well as the names of each of the champion's skills.Each champion has four skills that it can use. The skills vary greatly but have some common aspects

such as mana cost, damage, and cooldown. The items also have values such as price, effect (what attribute of the champion gets increased), and name.

- What tasks will your database system handle?

This database would store all of the relevant information in relation to a champion in the game so that someone could look up what their abilities are, how much damage they do, what their range is, how fast they move as well as several other variables. They could then look at the abilities of each of these champions in depth. One could look up the items that work well with the champion's abilities and recommended play style (Items that give health if a champion is generally played in such a manner).

3. Sample Queries - Questions in English

State at least 5 significantly different queries that your system will handle, in plain English. To make the queries significantly different, they should meet one or a combination of following elements appropriately:

- Each query should involve different objects in your project domain.
- Each query should contain different type of operation criteria.
- At least three queries are pertaining to several objects simultaneously.

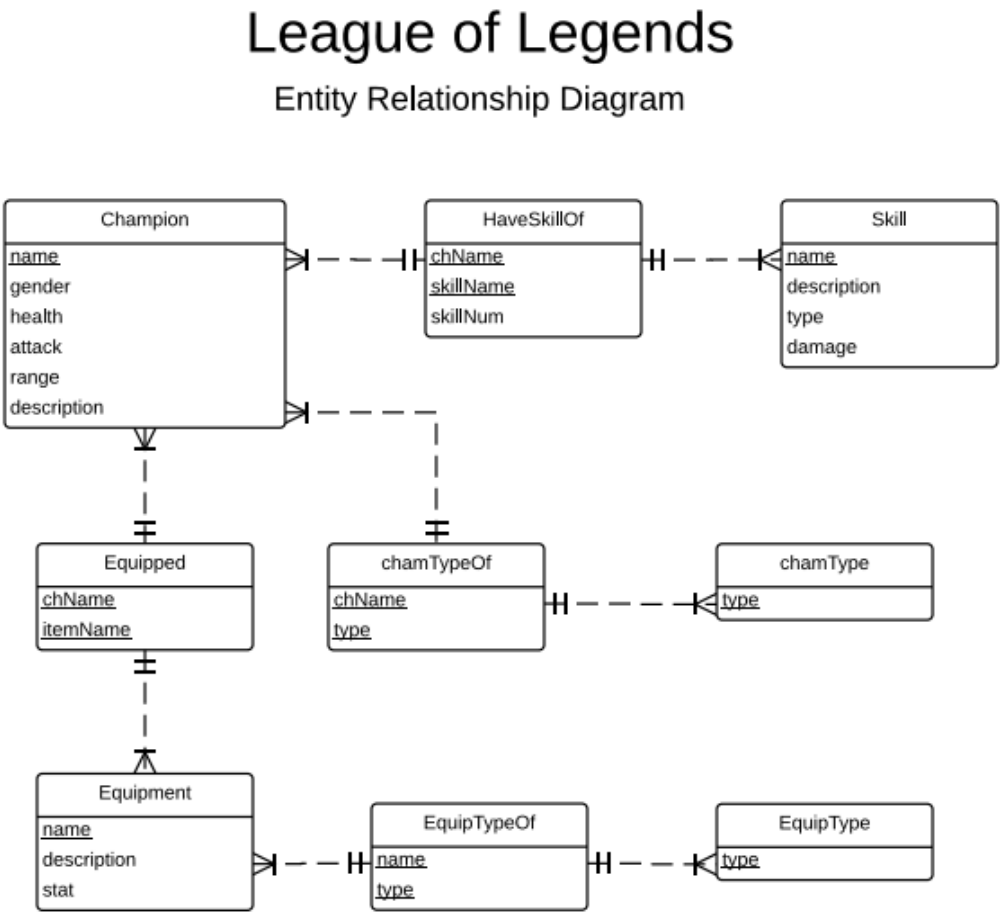
1. When user input a champion's name, everything about this champion will be displayed to user.(Champion)
2. When a skill name was given by user, not only the detail about that skill will be displayed to user, but also some kinda information will be given to indicate who has this skill.(Champion, Skill, haveSkillOf)
3. When a user wants to search all the champions whose chamType is a range, those champions will be return to user.(Champion, chamType)
4. If the user wants to know which skill has the most damage, detail information about that skill will be return.(Skill)

5. If the user wants to know the information about one specific equipment or all the equipments in the database, all details about that or those equipments will be return.(Equipment, EquipType)

4. Entity-Relationship (E/R) Diagram

Create an Entity/Relationship Diagram for your database. The diagram should satisfy following requirements:

- The diagram contains at least 5 but not to exceed 8 entity sets.
- The number of attributes contained in all the entity sets should be at least 20.
- Each entity set should connect to at least one other entity set.
- The degree constraints (many-many, many-one, and one-one), and primary keys should be properly indicated.



5. Relational Schema - initial

- Champion(name,gender, health, attack, range,description);
- Skill(name, description, type, damage);
- HaveSkillOf(chName, skillName, skillNum);
- chamType(type);

- chamTypeOf(chName, type); //used to connect champion and chamType, since one champion could have many different types
- Equipment(name, description, stat);
- EquipType(type);
- EquipTypeOf(name, type); //used to connect Equipment and EquipType, since one equipment could have many different types
- Equipped(chName, itemName);

6. Sample Queries - Relational algebra 1

1. R1:= σ(name = 'chamName')(Champion); //chamName is a placeholder

   R2:= π(*)(R1);

2. R1:= σ(skillName = 'skillName')(HaveSkillOf); //skillName is a placeholder

   R2:= π(*)(R1);

   R3:= σ(name = 'skillName')(Skill); //skillName is a placeholder

   R4:= π(*)(R3);

   R5:= R2 ⋈(R2.skillName = R4.name) R4;

   R6:= π(*)(R5);

3. R1:= σ(type = 'range')(chamTypeOf);

   R2:= π(*)(R1);

4. R1:= π(max(damage) -> highestDamage)(Skill);

   R2:= R1 ⋈(R1.highestDamage = Skill.damage) Skill;

R3:= π(*)(R2);

5. R1:= σ(name = 'equipName')(Equipment); //equipName is a placeholder

R2:= π(*)(R1);

7. Functional Dependencies 1

State all the functional dependencies that exist in your domain, given your initial schema from the initial schema specified in Section 4.

Champion(name,gender, health, attack, range,description);

FDs: name-> description name,description->gender name,health,attack->range name->health name->attack

Skill(name, description, type, damage);

FDs: name->damage name->type name, description, type->damage

Equipment(name, description, stat);

FDs: name->description name->stat name,description->stat

8. Proof of Normal Form 1

Prove that your original schema is in Third or Fourth Normal Form (3NF/4NF), or indicate why it isn't.

Champion(name,gender, health, attack, range,description);

1: Key: name K={name}

2: F = {name-> description name,description->gender name,health,attack->range name->health name->attack}

3: R1 = {name, description, gender} R2 = {name, health, attack, range}

4: Nothing to remove

5: Nothing to add, already contains the key name

So champion is not in 3NF. and change it to the following

ChampBasic = {name, description, gender} ChampStat = {name, health, attack, range}

Skill(name, description, type, damage);

1: Key: name K={name}

2: F = {name->description, name->type, name, description, type->damage}

3: R1 = {name, description} R2 = {name, type} R3 = {name, description, type, damage}

4: R1 and R2 get removed because they are subsets of R3

5: Nothing to add, already contains the key name

Skill(name, description, type, damage) is in 3NF.

Equipment(name, description, stat);

1: Key = name K = {name}

2: F={name->description, name->stat, name, description->stat}

3: R1 = {Name, description} R2 = {Name, description, stat}

4: remove R1, because it is a subset of R2.

5: contain the key name, so nothing to add.

Equipment(name, description, stat) is in 3NF.

9. Relational Schema - final

If your initial schema is not in 3NF/4NF, follow the process described in Chapter 3 to convert your initial schema into your final schema, provided with the functional dependencies identified in Section 7.

- ChampBasic{name, description, gender}
- ChampStat{name, health, attack, range}
- Skill(name, description, type, damage);
- HaveSkillOf(chName, skillName, skillNum);
- chamType(type);
- chamTypeOf(chName, type); //used to connect champion and chamType, since one champion could have many different types
- Equipment(name, description, stat);
- EquipType(type);
- EquipTypeOf(name, type); //used to connect Equipment and EquipType, since one equipment could have many different types
- Equipped(chName, itemName);

10. Functional Dependencies 2

Show how all the functional dependencies identified in Section 7 are preserved in your final schema.

ChampBasic = {name, description, gender} ChampStat = {name, health, attack, range}

FDs: name-> description ok name,description->gender ok name,health,attack->range ok

name->health ok name->attack ok

Skill(name, description, type, damage);

FDs: name->type ok name, description, type->damage description->type ok

Equipment(name, description, stat);

FDs: name->description ok name->stat ok name,description->stat ok

11. Proof of Normal Form 2

Prove that your final schema is in 3NF/4NF.

- ChampBasic = {name, description, gender}

FDs: name-> description name,description->gender

1: key = name K = {name}

2: F = {name -> description name,description-> gender}

3: R1 = {name, description} R2 = {name, description, gender}

4: remove R1, a subset of R2

5: R = {name, description, gender}

ChampBasic is in 3NF

All the FDs are preserved. And no need to perform chase test, since we only have one table.

- ChampStat = {name, health, attack, range}

FDs: name,health,attack->range name->health name->attack

1: Key: name, K = {name}

2: F = {name,health,attack->range name->health name->attack}

3: R1 = {name,health,attack,range} R2 = {name, health}

4: remove R2, a subset of R1

5: nothing to add, contain the key already

ChampStat is in 3NF.

All the FDs are preserved. And no need to perform chase test, since we only have one table.

- Skill(name, description, type, damage);

FDs: name->description, name->type, name, description, type->damage

1: Key: name K={name}

2: F = {name->description, name->type, name, description, type->damage}

3: R1 = {name, description} R2 = {name, type} R3 = {name, description, type, damage}

4: Remove R1 and R2, subsets of R3

5: Key is present so nothing to add

Skill is in 3NF.

All the FDs are preserved. And no need to perform chase test, since we only have one table.

- Equipment(name, description, stat);

FDs: name->description name->stat name,description->stat

1: Key = name K = {name}

2: F = { name->description name->stat name,description->stat }

3: R1 = {name, description, stat} R2 = { name, stat}

4: remove R2, a subset of R1

5: nothing to add, contain the key already

Equipment is in 3NF.

All the FDs are preserved. And no need to perform chase test, since we only have one table.

12. Sample Queries - Relational algebra 2

Restate the queries from Section 5 using relational algebra notation, given the final schema specified in Section 9.

1. R1:= σ(name = 'chamName')(ChampBasic); //chamName is a placeholder

   R2:= π(*)(R1);

2. R1:= σ(skillName = 'skillName')(HaveSkillOf); //skillName is a placeholder

   R2:= π(*)(R1);

   R3:= σ(name = 'skillName')(Skill); //skillName is a placeholder

R4:= π(*)(R3);

R5:= R2 ⋈(R2.skillName = R4.name) R4;

R6:= π(*)(R5);

3. R1:= σ(type = ‘range’)(chamTypeOf);

R2:= π(*)(R1);

4. R1:= π(max(damage) -> highestDamage)(Skill);

R2:= R1 ⋈(R1.highestDamage = Skill.damage) Skill;

R3:= π(*)(R2);

5. R1:= σ(name = ‘equipName’)(Equipment); //equipName is a placeholder

R2:= π(*)(R1);

13. Database Implementation Status

Build your database as described in Database Implementation. Include each table declaration SQL queries and the number of tuples in each table. (Do NOT include the data itself in the Project Design Document.)

14. SQL Queries

The set of SQL queries should include:

- SQL queries converted from the queries generated in Section 12;
- SQL queries including "insert" statements;
- SQL queries including "update" statements;

1: select * from ChampBasic where name = ‘chamName’;

2: select * from HaveSkillOf, Skill where HaveSkillOf.skillName = Skill.name;

3: select * from chamTypeOf where type = ‘range’;

4: select *
from
(select max(damage)->highestDamage from Skill)S1, Skill
where S1.highestDamage = Skill.damage;

5: select * from Equipment where name = ‘equipName’;

6: insert into ChampStat values
(‘Yasuo’, 517.76, 55.376, 175),
(‘Jinx’, 517.76, 53.04, 525),
(‘Zed’, 579.4, 54.712, 125),
(‘Lucian’, 554.4, 52.04, 500),
(‘Vi’, 582.8, 55.88, 125);

7: insert into Skill values
(‘Focus’, ‘If Ashe has not attacked for 3 seconds, she gains Focus stacks per second. At 100 stacks, Ashe will critically strike on her next basic attack. Thereafter, Focus stacks will reset to an amount equal to her critical strike chance.’, ‘passive’, 0),
(‘Frost Shot’, ‘Toggle: Ashe's basic attacks slow her targets for 2 seconds.’, ‘passive’,0),
(‘Volley’, ‘Active: Ashe fires 7 arrows in a cone, dealing physical damage. Volley also applies Frost Shot.’, ‘indirect”,),
(‘Hawkshot’,’Passive: Ashe gains 3 bonus gold each time she kills a unit or destroys a structure.’, ‘passive’, 0),
(‘Enchanted Crystal Arrow’, ‘Active: Fires a large arrow in a straight line. If it hits an enemy champion, it will deal magic damage and stun that champion for up to 3.5 seconds, based on the distance the arrow traveled. Additionally, surrounding units take half the damage and are slowed by 50% for 3 seconds.’, ‘indirect’, 40);

8: insert into Equipment values
(‘B.F. Sword’, null, ‘+50ad’),
(‘Amplifying Tome’, null, ‘+20ap’),
(‘Blasting Wand’,null,’+40ap’),
(‘Chain Vest’, null, ‘+40 armor’),
(‘Brawler’s Gloves’, null, ‘+8% critical strike chance’);

15. Client Program Status

Create a client program using C# for your project. This program should satisfy the following requirements:

- It establishes a connection to the database server.
- It defines a method where an SQL query can be sent to the database server, and the result set returned from the database server is obtained.
- It provides a way to view the content of the result set.
- It provides a way to test all the SQL queries generated from Section 14.

You can find relevant information in Chapter 9 of the textbook.

16. Additional Database Capabilities

Please make your database design more sophisticated by adding foreign key constraints, adding value constraints, adding triggers, or other features to the table schemas.

Please update your design document with this new information as well.

17. Additional Client Program Capabilities

Provide a way in your client program to show how the changes made in the previous section work.

18. Project Presentation

Give a brief introduction about what will present and demonstrate for your project.