Functional Dependencies:
Let:
ID = A, City = B, Species = C, Appearance = D, Name = E, Speed = F, Type = G, Market_Price = H, Riding_Proficiency = I

r_{Mounts} (ABCDEFGI) = F(A->BCDEFGHI, F->GI, I->GF, BCE->ADFGHI)

Below you will find the normalization process for one of our more complicated relations.

3NF:

relation Mounts
r(ABCDEFGHI)
FD's (A->BCDEFGHI, F->GI, I->GF, BCE->ADFGHI)

Candidate keys: A & BCE

Test A->BCDEFGHI
- Trivial, fails
- super key, passes

Test F->GI
- Trivial, fails
- super key, fails
- contained in candidate key, fails

Canonical Cover:
- A->BCDEFGHI
- Is B extraneous?
  - No
- Is C extraneous?
  - No
- Is D extraneous?
  - Yes, remove D
- Is E extraneous?
  - Yes, remove E
- Is F extraneous?
  - Yes, remove F
- Is G extraneous?
  - Yes, remove G
- Is H extraneous?
  - Yes, remove H
- Is I extraneous?
  - Yes, remove I

F_c(A->BCE, F->GI, I->GF, BCE->ADFGHI)

F->GI
Is G extraneous?
No
Is I extraneous?
No

I->GF
Is G extraneous?
No
Is F extraneous?
No

BCE->ADFGHI
Is B extraneous?
Yes, remove
Is C extraneous?
Yes, remove
Is A extraneous?
No
Is D extraneous?
No
Is F extraneous?
No
Is G extraneous?
No
Is H extraneous?
No
Is I extraneous?
No

F_c(A->BCE, F->GI, I->GF, E->ADFGHI)

r_1(ABCE), r_2(FGI), r_3(EADFGHI)

We decided to denormalize our model, because this decomposition would have greatly increased our number of joins.

BCNF:

Let:

ID = A, City = B, Species = C, Appearance = D, Name = E, Speed = F, Type = G, Market_Price = H, Riding_Proficiency = I

F=(A->BCDEFGHI, F->GI, I->GF, BCE->ADFGHI)
A->BCDEFGHI passes trivially
$(F\text{->}GI)^{+}$ = FGI, so F->GI does not pass
r_1(FGI),{F->GI} and r_x(ABCDE), F'=(A->BCDE, BCE->ADH)
A->BCDE passes trivially
$(BCE)^{+}$ = ABCDEH, thus is passes as well

Final change is r_1(FGI),{F->GI} and r_2(ABCDE),{A->BCDE, BCE->ADH}

In the end we decided that it would be best for our domain to keep our original model as it seemed it be the best in regards to performance.

SQl Queries:

6)
Query to return all the characters in the realm “Aerie Peak” who are not in a guild and whose level is above 50.

```
SELECT name
FROM Characters
WHERE realm = 'Aerie Peak'
        EXISTS (
                SELECT name
                FROM Characters
                WHERE guild IS NULL
                   AND name NOT IN (
                      SELECT name
                      FROM Characters
                      WHERE level < 50));
```

7)
Lists the names and the levels of all characters who completed a quest below the required level.

```
SELECT name, level
FROM ((Characters AS c) JOIN Tracks ON name=character) JOIN (Quests AS q) ON id=quest
WHERE c.level < q.required_level
GROUP BY q.title
```

8)
List of all warrior talents that grant abilities
SELECT name
FROM (Classes JOIN Attains ON name=class) JOIN (Talents AS t) ON talent= t.name
WHERE x.name='Warrior' AND t.new_spells IS NOT NULL

Relational Algebra:

6)
Relational Algebra does not have ways of expressing NULL values

7)
title ς(π name, level(σ level < required_level ((Character X Tracks) X Quests)))

8)
Relational Algebra does not have ways of expressing NULL values

Indexing:

Profiling Queries 7 and 8 without indexes…

QUERY 7 DURATION: 15 milliseconds

QUERY 8 DURATION: 0 milliseconds

Adding Indexes to model…

Profiling Queries 7 and 8 with indexes…

QUERY 7 DURATION: 31 milliseconds

QUERY 8 DURATION: 1 milliseconds

As is shown above, we have a small enough amount of data in our model that, even when performing indexing with our more complicated queries, the added complexity actually decreased performance.

NoSQL:

Our project has a few drawbacks that would be fixed by implementing a NoSQL database instead.

<u>Joins</u>: The way our relational database is set up with the resulting logical model we require multiple joins in order to retrieve the information that we are looking for. Joins are costly, so the fewer of these that we can use the better it would be for our database.

<u>Media</u>: Certain relations for our domain have appearance keys which would be connected to certain models that would need to be loaded by the game.

<u>Scalability</u>: Since our domain represents that of a massive online multiplayer game, an obvious issue that our relational model would have would be scalability. In the real world the

database would need to store over a million different characters, which would means multiple millions of rows.