Medieval Arms & Armor

<u>Designed by Kenneth Brooks III</u>

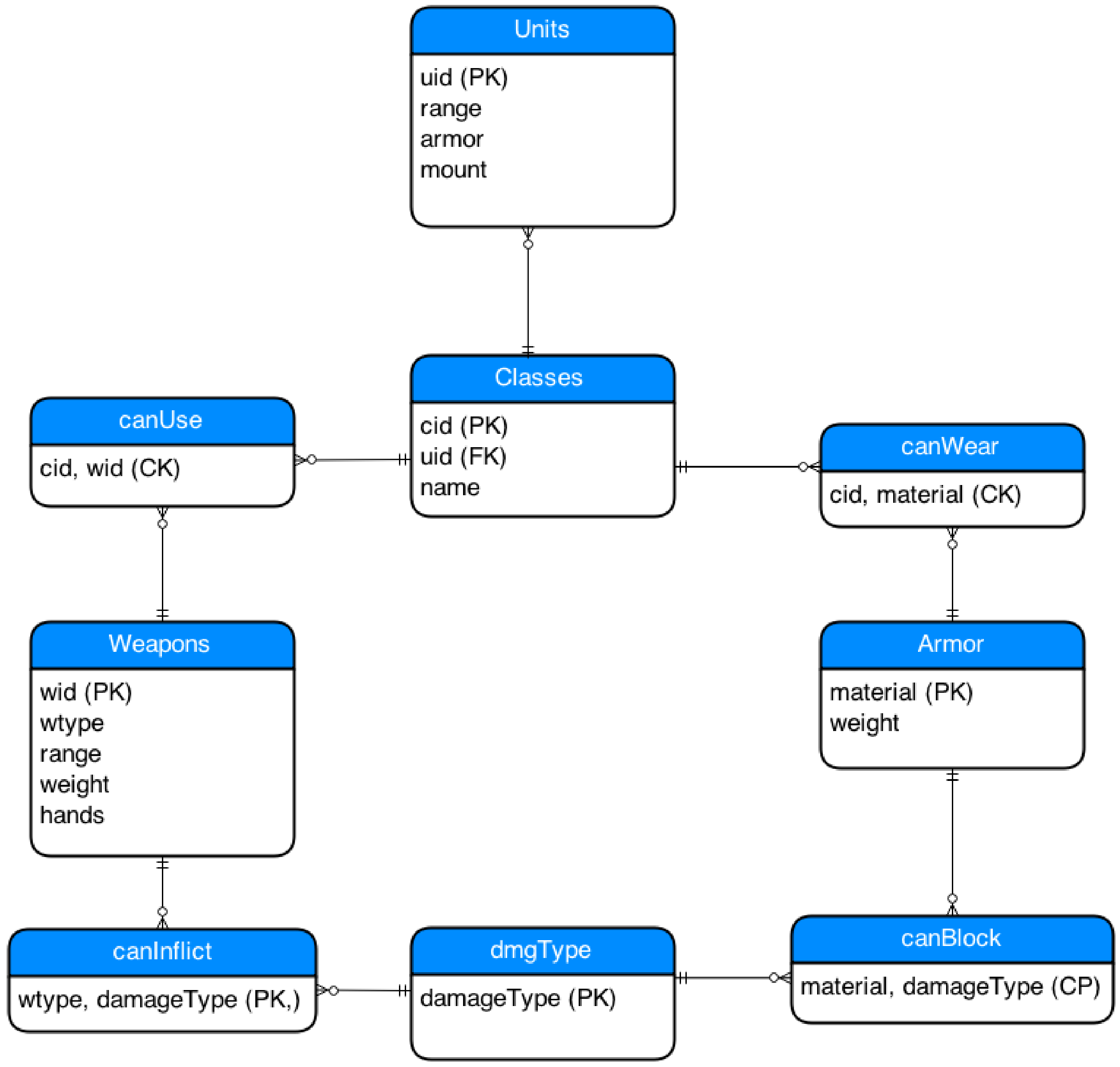
Table of Contents

Executive Summary

This document serves as my design and test data for a database on Medieval Arms and Armor. This database is a way to learn about the different classes of soldiers from this time period as well as the weapons and armor they used. potential users include students, medieval time buffs, or teachers.

I have laid out this document to have the Entity Relationship Diagram first, indicating the full structure of the database and relations between tables. After the ER Diagram, i have all of my create statements for my tables along with descriptions, functional dependencies

Entity Relationship Diagram

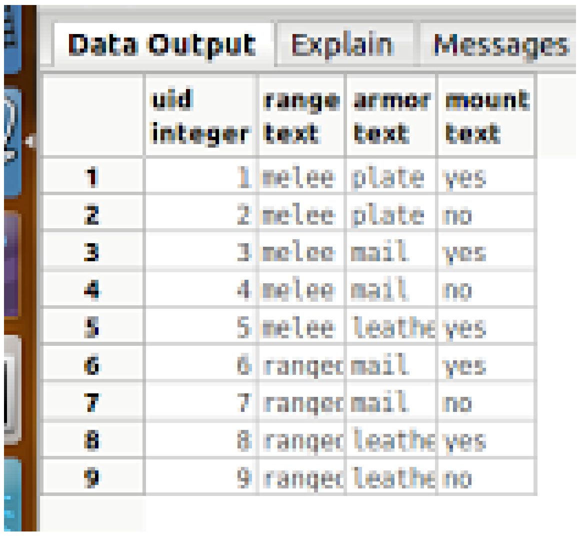


TABLES

Unit(s) table
The units table holds the different Unit types at the highest level. Its attributes indicate the range, armor type, and ability or not of the unit to mount.

```
CREATE TABLE Unit (
  uid          int NOT NULL PRIMARY KEY,
  range        text NOT NULL,
  armor        text NOT NULL,
  mount                text NOT NULL
);
```

Functional Dependencies:
    uid -> range, armor, mount
Sample Data:



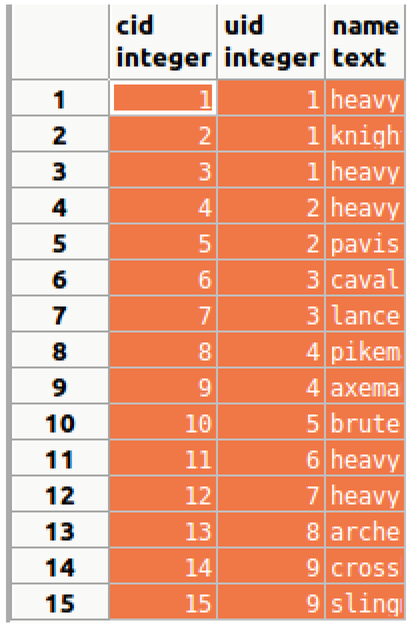
	uid integer	range text	armor text	mount text
1	1	melee	plate	yes
2	2	melee	plate	no
3	3	melee	mail	yes
4	4	melee	mail	no
5	5	melee	leathe	yes
6	6	rangec	mail	yes
7	7	rangec	mail	no
8	8	rangec	leathe	yes
9	9	rangec	leathe	no

Class(es) table
The Classes table holds information more specific to an soldiers specialization in battle. It contains the cid, uid, and class name of each unit.

```
CREATE TABLE Class(
  cid             int NOT NULL PRIMARY KEY,
  uid             int REFERENCES Unit(uid),
  name            text
);
```

Functional Dependencies:
cid -> uid, name

Sample Data:



	cid integer	uid integer	name text
1	1	1	heavy
2	2	1	knigh
3	3	1	heavy
4	4	2	heavy
5	5	2	pavis
6	6	3	caval
7	7	3	lance
8	8	4	pikem
9	9	4	axema
10	10	5	brute
11	11	6	heavy
12	12	7	heavy
13	13	8	arche
14	14	9	cross
15	15	9	sling

Weapons Table

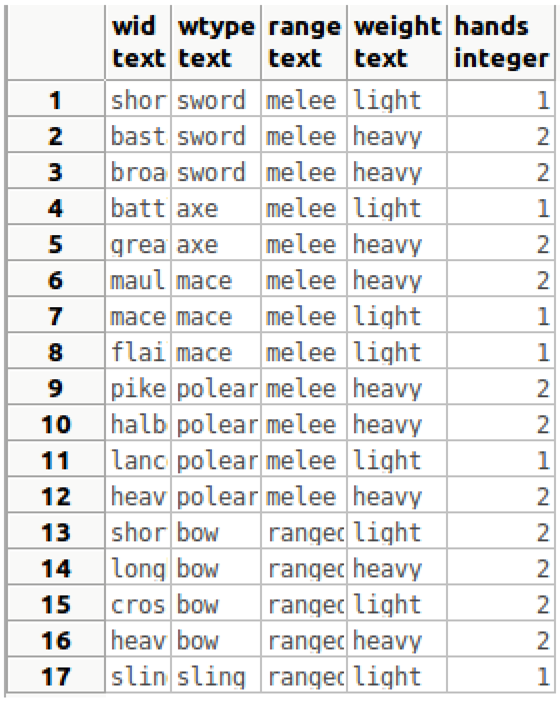
This table contains information relevant to each specific weapon. For example this contains the wid (name), type, range, weight, and required number of hands for each entry.

```
CREATE TABLE Weapons (
  wid             text NOT NULL PRIMARY KEY,
  wtype           text NOT NULL,
  range           text NOT NULL,
  weight          text NOT NULL,
  hands                   int NOT NULL
);
```

Functional Dependencies:

wid -> wtype, range, weight, hands

Sample data:



	wid text	wtype text	range text	weight text	hands integer
1	shor	sword	melee	light	1
2	bast	sword	melee	heavy	2
3	broa	sword	melee	heavy	2
4	batt	axe	melee	light	1
5	grea	axe	melee	heavy	2
6	maul	mace	melee	heavy	2
7	mace	mace	melee	light	1
8	flai	mace	melee	light	1
9	pike	polear	melee	heavy	2
10	halb	polear	melee	heavy	2
11	lanc	polear	melee	light	1
12	heav	polear	melee	heavy	2
13	shor	bow	rangec	light	2
14	long	bow	rangec	heavy	2
15	cros	bow	rangec	light	2
16	heav	bow	rangec	heavy	2
17	slin	sling	rangec	light	1

Armor Table

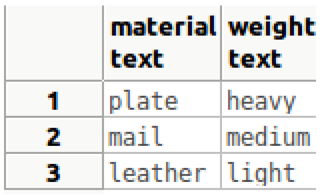
The armor table contains the material (type) of the armor and its weight, being light, medium, or heavy.

```
CREATE TABLE Armor (
  material        text NOT NULL PRIMARY KEY,
  weight          text NOT NULL
);
```

Functional Dependencies:

    material -> weight

Sample Data:



	material text	weight text
1	plate	heavy
2	mail	medium
3	leather	light

canUse Table

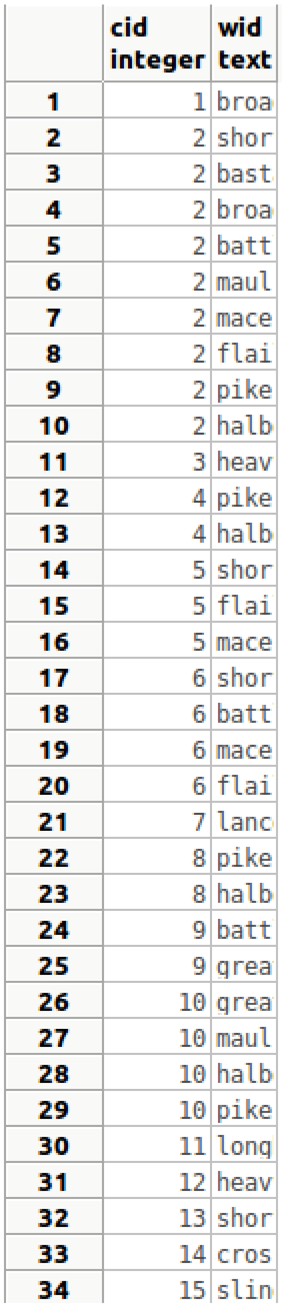
This is a weak entity for connecting Classes with Weapons.

```
CREATE TABLE canUse (
  cid             int NOT NULL,
  wid             text NOT NULL,
  PRIMARY KEY(cid, wid),
  FOREIGN KEY (cid) references class(cid),
  FOREIGN KEY (wid) references Weapons(wid)
);
```

Functional Dependencies:

Sample Data:

(sample data on next page)

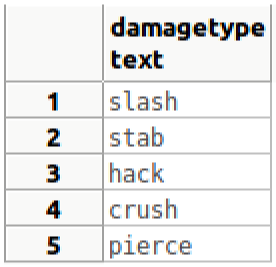


	cid integer	wid text
1	1	broa
2	2	shor
3	2	bast
4	2	broa
5	2	batt
6	2	maul
7	2	mace
8	2	flai
9	2	pike
10	2	halb
11	3	heav
12	4	pike
13	4	halb
14	5	shor
15	5	flai
16	5	mace
17	6	shor
18	6	batt
19	6	mace
20	6	flai
21	7	lanc
22	8	pike
23	8	halb
24	9	batt
25	9	grea
26	10	grea
27	10	maul
28	10	halb
29	10	pike
30	11	long
31	12	heav
32	13	shor
33	14	cros
34	15	slin

dmgType Table
This table is used to tie together weapons and armor by providing a way to show which weapons have damage that is ineffective against certain armor types.

```
CREATE TABLE dmgType (
  damageTypetext NOT NULL UNIQUE PRIMARY KEY
);
```

Functional Dependencies:
Sample Data:

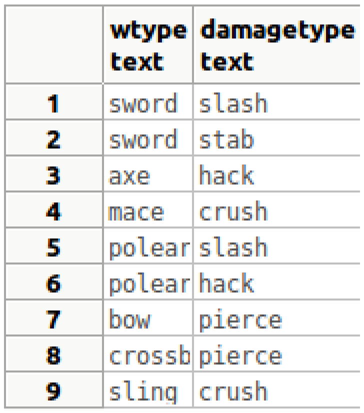


	damagetype text
1	slash
2	stab
3	hack
4	crush
5	pierce

canInflict Table
This table is a weak entity used to tie weapons to the damages they can deal and avoid many to many relationships.

```
CREATE TABLE canInflict (
  wtype              text,
  damageType    text,
  PRIMARY KEY(wtype, damageType)
);
```

Functional Dependencies:
Sample Data:

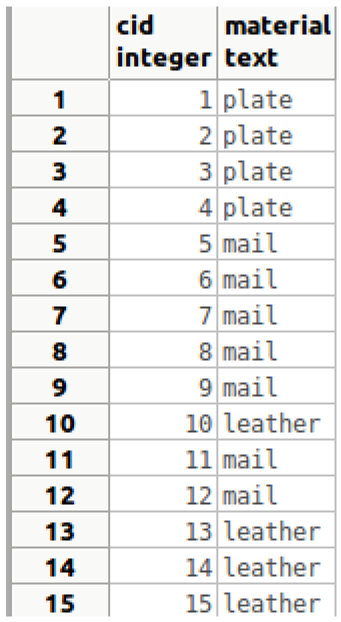


	wtype text	damagetype text
1	sword	slash
2	sword	stab
3	axe	hack
4	mace	crush
5	polear	slash
6	polear	hack
7	bow	pierce
8	crossb	pierce
9	sling	crush

canWear Table
Weak Entity for the purpose of linking different classes with different armor types without running into many to many relations.

```
CREATE TABLE canWear (
  cid            int NOT NULL references Class(cid),
  material       text NOT NULL references Armor(material),
  PRIMARY KEY(cid, material)
);
```

Functional Dependencies:
Sample Data:

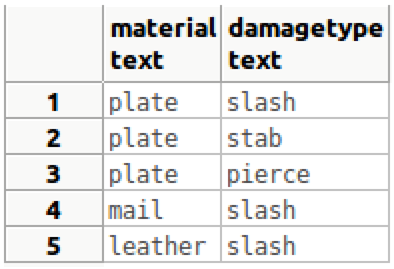


	cid integer	material text
1	1	plate
2	2	plate
3	3	plate
4	4	plate
5	5	mail
6	6	mail
7	7	mail
8	8	mail
9	9	mail
10	10	leather
11	11	mail
12	12	mail
13	13	leather
14	14	leather
15	15	leather

canBlock Table
A Weak Entity designed to link Armor and dmgType tables in order to show which armor is effective against which types of damage. This relation also avoids many to many relationships.

```
CREATE TABLE canBlock (
  material        text NOT NULL references Armor(material),
  damageTypetext NOT NULL references dmgType(damageType),
  PRIMARY KEY(material, damageType)
);
```

Functional Dependencies:
Sample Data:



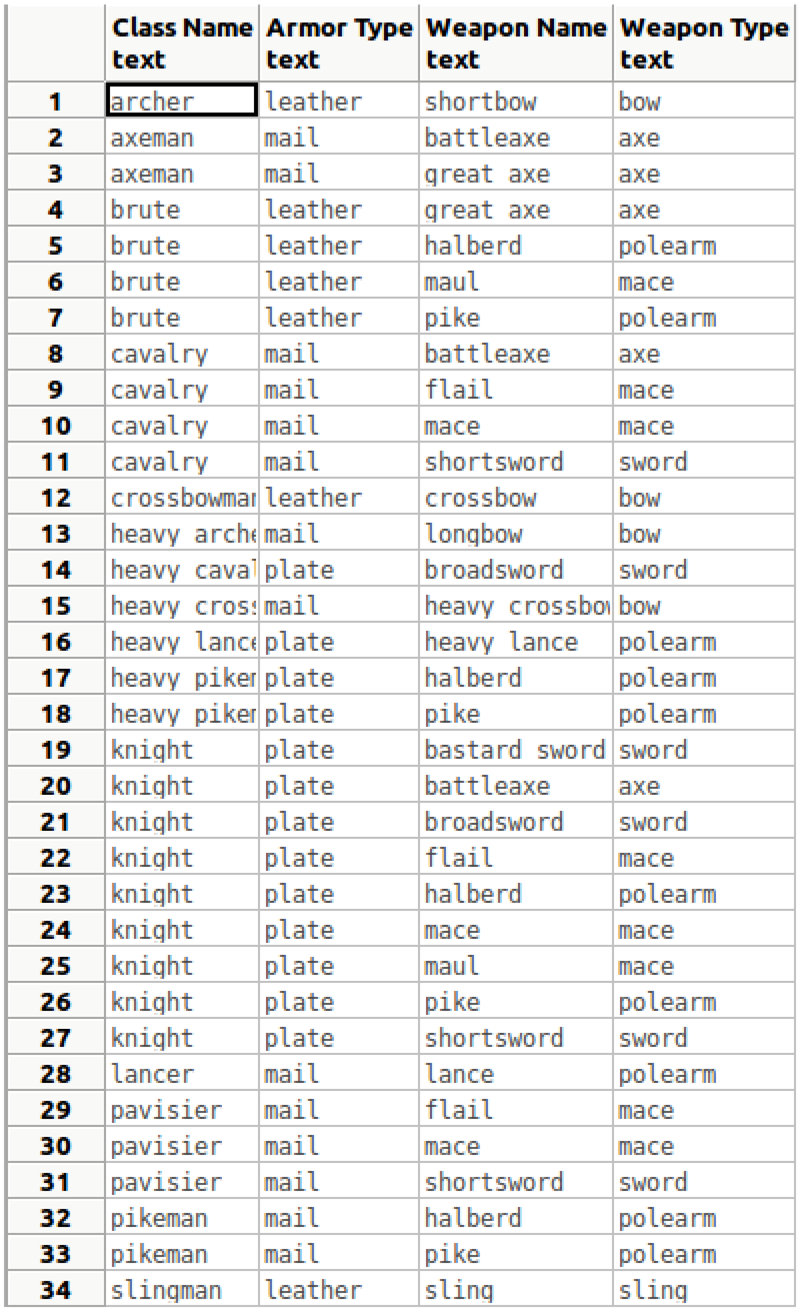
	material text	damagetype text
1	plate	slash
2	plate	stab
3	plate	pierce
4	mail	slash
5	leather	slash

VIEWS

--classWeaponsArmor view
--Displays the weapon and armor of each unit

```
CREATE VIEW classWeaponsArmor
AS
SELECT DISTINCT c.name AS "Class Name", a.material AS "Armor Type", w.wid AS
"Weapon Name", w.wtype AS "Weapon Type"
FROM Class c, canUse cu, Weapons w, canWear cw, Armor a
WHERE c.cid = cu.cid AND c.cid = cw.cid
AND cu.wid = w.wid AND cw.material = a.material
ORDER BY c.name, w.wid, w.wtype, a.material ASC;
```

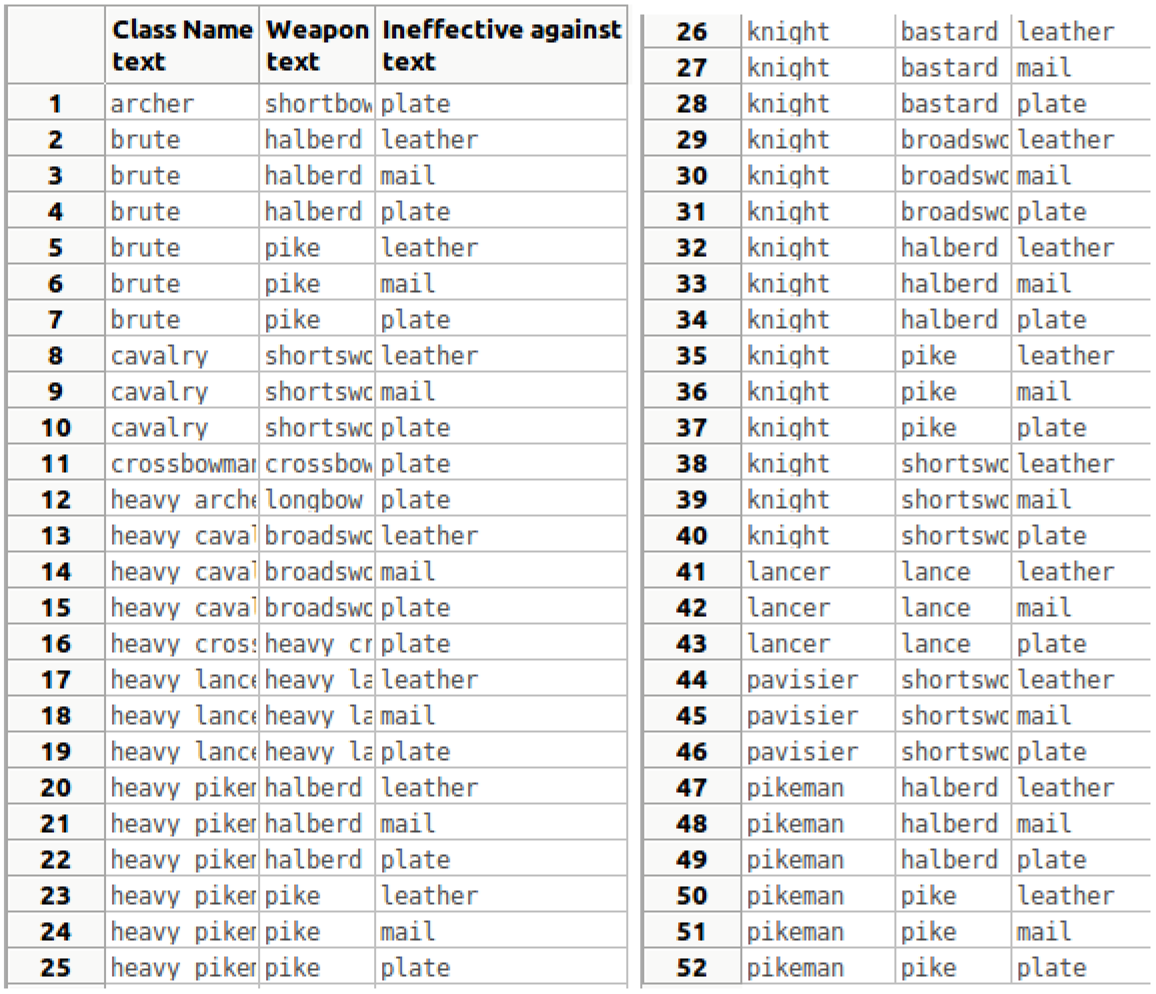
Sample Data on next page.



	Class Name text	Armor Type text	Weapon Name text	Weapon Type text
1	archer	leather	shortbow	bow
2	axeman	mail	battleaxe	axe
3	axeman	mail	great axe	axe
4	brute	leather	great axe	axe
5	brute	leather	halberd	polearm
6	brute	leather	maul	mace
7	brute	leather	pike	polearm
8	cavalry	mail	battleaxe	axe
9	cavalry	mail	flail	mace
10	cavalry	mail	mace	mace
11	cavalry	mail	shortsword	sword
12	crossbowma	leather	crossbow	bow
13	heavy arche	mail	longbow	bow
14	heavy caval	plate	broadsword	sword
15	heavy cros	mail	heavy crossbo	bow
16	heavy lance	plate	heavy lance	polearm
17	heavy piker	plate	halberd	polearm
18	heavy piker	plate	pike	polearm
19	knight	plate	bastard sword	sword
20	knight	plate	battleaxe	axe
21	knight	plate	broadsword	sword
22	knight	plate	flail	mace
23	knight	plate	halberd	polearm
24	knight	plate	mace	mace
25	knight	plate	maul	mace
26	knight	plate	pike	polearm
27	knight	plate	shortsword	sword
28	lancer	mail	lance	polearm
29	pavisier	mail	flail	mace
30	pavisier	mail	mace	mace
31	pavisier	mail	shortsword	sword
32	pikeman	mail	halberd	polearm
33	pikeman	mail	pike	polearm
34	slingman	leather	sling	sling

```
--classWeaponIneffective view
--Shows which class/weapon cominations are ineffective against which armor types

CREATE VIEW classWeaponIneffective
AS
SELECT DISTINCT c.name AS "Class Name", w.wid AS "Weapon", a.material AS "Ineffective against"
FROM Weapons w, canInflict ci, dmgType dt, canBlock cb, armor a, canUse cu, Class c
WHERE c.cid = cu.cid AND cu.wid = w.wid
AND w.wtype = ci.wtype AND ci.damageType = dt.damageType
AND dt.damageType = cb.damageType AND cb.material = a.material
ORDER BY c.name, w.wid, a.material ASC;
```

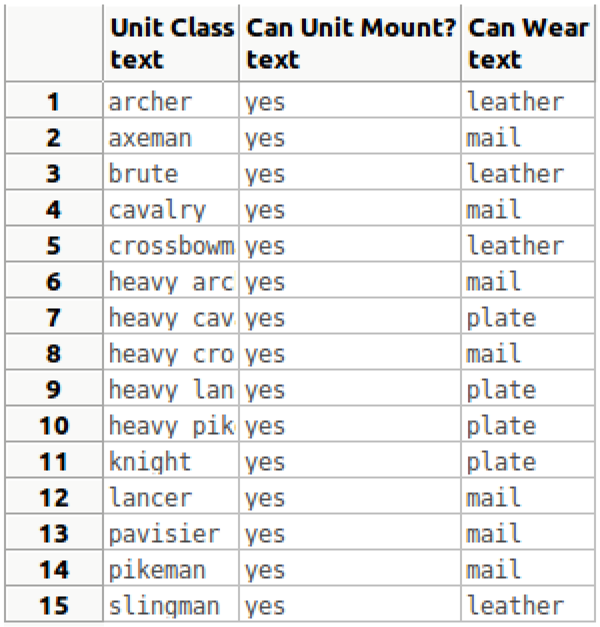


	Class Name text	Weapon text	Ineffective against text
1	archer	shortbow	plate
2	brute	halberd	leather
3	brute	halberd	mail
4	brute	halberd	plate
5	brute	pike	leather
6	brute	pike	mail
7	brute	pike	plate
8	cavalry	shortswc	leather
9	cavalry	shortswc	mail
10	cavalry	shortswc	plate
11	crossbowmar	crossbow	plate
12	heavy arche	longbow	plate
13	heavy caval	broadswc	leather
14	heavy caval	broadswc	mail
15	heavy caval	broadswc	plate
16	heavy cros:	heavy cr	plate
17	heavy lance	heavy la	leather
18	heavy lance	heavy la	mail
19	heavy lance	heavy la	plate
20	heavy piker	halberd	leather
21	heavy piker	halberd	mail
22	heavy piker	halberd	plate
23	heavy piker	pike	leather
24	heavy piker	pike	mail
25	heavy piker	pike	plate
26	knight	bastard	leather
27	knight	bastard	mail
28	knight	bastard	plate
29	knight	broadswc	leather
30	knight	broadswc	mail
31	knight	broadswc	plate
32	knight	halberd	leather
33	knight	halberd	mail
34	knight	halberd	plate
35	knight	pike	leather
36	knight	pike	mail
37	knight	pike	plate
38	knight	shortswc	leather
39	knight	shortswc	mail
40	knight	shortswc	plate
41	lancer	lance	leather
42	lancer	lance	mail
43	lancer	lance	plate
44	pavisier	shortswc	leather
45	pavisier	shortswc	mail
46	pavisier	shortswc	plate
47	pikeman	halberd	leather
48	pikeman	halberd	mail
49	pikeman	halberd	plate
50	pikeman	pike	leather
51	pikeman	pike	mail
52	pikeman	pike	plate

Reports and Queries

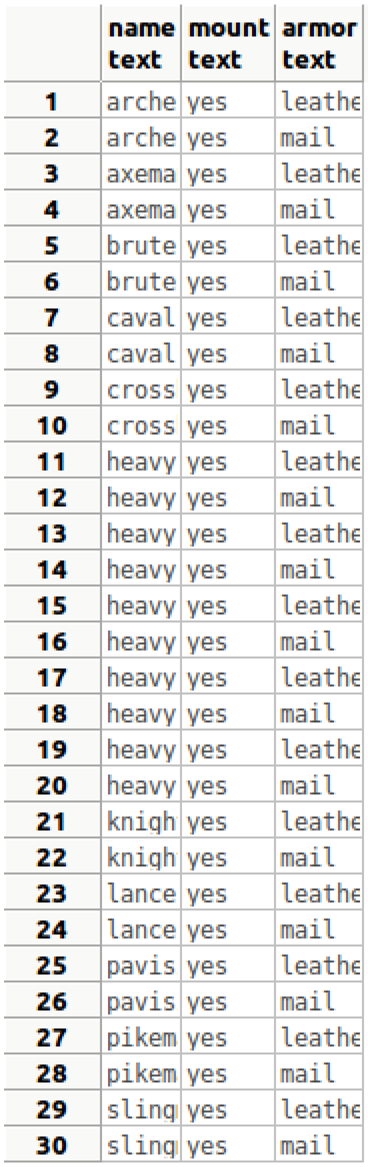
```
--Classes that can mount and use leather or mail armor
--Display all classes that can mount, wear leather or mail, along with the weapons that they can use

SELECT DISTINCT c.name AS "Unit Class", u.mount AS "Can Unit Mount?", a.material AS "Can Wear"
FROM Unit u, Class c, canWear cw, Armor a, canUse cu, Weapons w
WHERE u.mount = 'yes' AND c.cid = cu.cid
AND cu.wid = w.wid AND c.cid = cw.cid
AND cw.material = a.material
ORDER BY c.name;
```



	Unit Class text	Can Unit Mount? text	Can Wear text
1	archer	yes	leather
2	axeman	yes	mail
3	brute	yes	leather
4	cavalry	yes	mail
5	crossbowm	yes	leather
6	heavy arc	yes	mail
7	heavy cav	yes	plate
8	heavy cro	yes	mail
9	heavy lan	yes	plate
10	heavy pik	yes	plate
11	knight	yes	plate
12	lancer	yes	mail
13	pavisier	yes	mail
14	pikeman	yes	mail
15	slingman	yes	leather

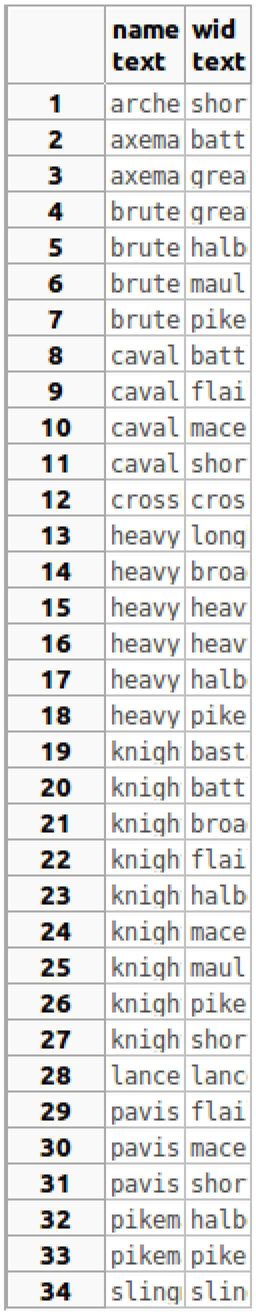
```
--Armor of Ranged units that can mount
--Display all ranged classes that can mount and the armor they can use
SELECT DISTINCT c.name, u.mount, u.armor
FROM Unit u, Class c, canWear cw, Armor a
WHERE u.mount = 'yes' AND u.range = 'ranged'
ORDER BY c.name, u.armor;
```



	name text	mount text	armor text
1	arche	yes	leathe
2	arche	yes	mail
3	axema	yes	leathe
4	axema	yes	mail
5	brute	yes	leathe
6	brute	yes	mail
7	caval	yes	leathe
8	caval	yes	mail
9	cross	yes	leathe
10	cross	yes	mail
11	heavy	yes	leathe
12	heavy	yes	mail
13	heavy	yes	leathe
14	heavy	yes	mail
15	heavy	yes	leathe
16	heavy	yes	mail
17	heavy	yes	leathe
18	heavy	yes	mail
19	heavy	yes	leathe
20	heavy	yes	mail
21	knigh	yes	leathe
22	knigh	yes	mail
23	lance	yes	leathe
24	lance	yes	mail
25	pavis	yes	leathe
26	pavis	yes	mail
27	pikem	yes	leathe
28	pikem	yes	mail
29	sling	yes	leathe
30	sling	yes	mail

```
--Weapons usable by plate wearers
--Display all classes that wear plate and the weapons they can use
SELECT DISTINCT c.name, w.wid
FROM Class c, canWear cw, armor a, canUse cu, Weapons w
WHERE a.material = cw.material AND cw.cid = c.cid
AND c.cid = cu.cid AND cu.wid = w.wid
ORDER BY c.name, w.wid;
```

Sample data on next page.



	name text	wid text
1	arche	shor
2	axema	batt
3	axema	grea
4	brute	grea
5	brute	halb
6	brute	maul
7	brute	pike
8	caval	batt
9	caval	flai
10	caval	mace
11	caval	shor
12	cross	cros
13	heavy	long
14	heavy	broa
15	heavy	heav
16	heavy	heav
17	heavy	halb
18	heavy	pike
19	knigh	bast
20	knigh	batt
21	knigh	broa
22	knigh	flai
23	knigh	halb
24	knigh	mace
25	knigh	maul
26	knigh	pike
27	knigh	shor
28	lance	lanc
29	pavis	flai
30	pavis	mace
31	pavis	shor
32	pikem	halb
33	pikem	pike
34	sling	slin

Security

I would only choose to implement two users for this database, an Admin and a basic User.

The admin has full control to change and update the database as he sees fit.

```
CREATE ROLE admin
GRANT SELECT, INSERT, UPDATE, ALTER
ON ALL TABLES IN SCHEMA PUBLIC
TO admin
```

The public user has the ability to look at the database and query it but cannot change or update anything.

```
CREATE ROLE public
GRANT SELECT
ON ALL TABLES IN SCHEMA PUBLIC
TO public
```

Implementation Notes / Known Problems / Future Enhancements

The impementation came really down to the wire on this one, as i am writing this piece at 7 A.M. before class after spending the last 16 hours (wow that figure shocked me just now) coding trying to get everything to work. The one thing i had the most trouble with and was not in the end able to figure out was stored procedures and how to actually get them to function correctly.

Future enhancements will include stored procedures and possibly expanding the database to include siege weapons and early firearms. One known problem is that occasionally when i tried to use DISTINCT it would only show one entry for a unit with only one type of armor, when two entries should be showing for mail and leather.