Breath of the Wild: Item DB

Designed By Faith Matthew

Table of Contents

Executive Summary

The Legend of Zelda: Breath of the Wild is a huge RPG with numerous items belonging to different categories. Players often find themselves overwhelmed with the expansive world before them and will question the use and worth of most items they encounter for the first time. This database is designed to help players stay on top of their game and will focus mainly on items. By utilizing this database, players will be able to see the breakdown of items and what they have to offer.

An E-R diagram displaying the framework for the database will be given. Followed by a slide by slide explanation on tables, stored procedures, views, triggers, and reports. A brief description of current issues with the database will be provided along with plans for future enhancement.
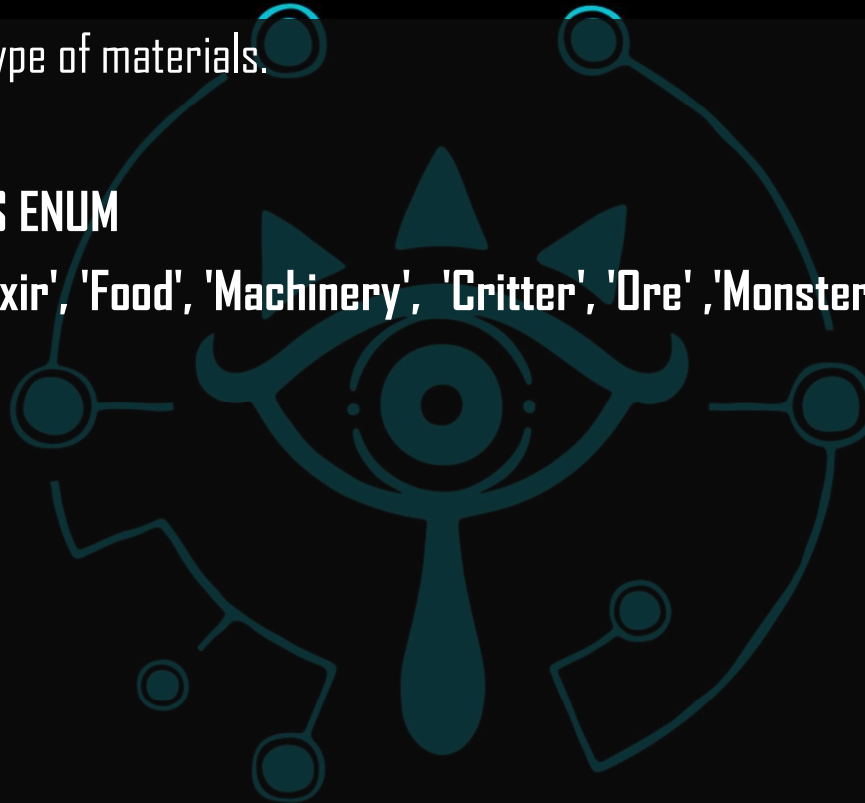
E-R Diagram

TypeOfMaterial

This type specifies the different type of materials.

CREATE TYPE typeOfMaterial AS ENUM

('Fruit', 'Vegetation' , 'Meat', 'Elixir', 'Food', 'Machinery', 'Critter', 'Ore' ,'Monster Part');

Types

wType

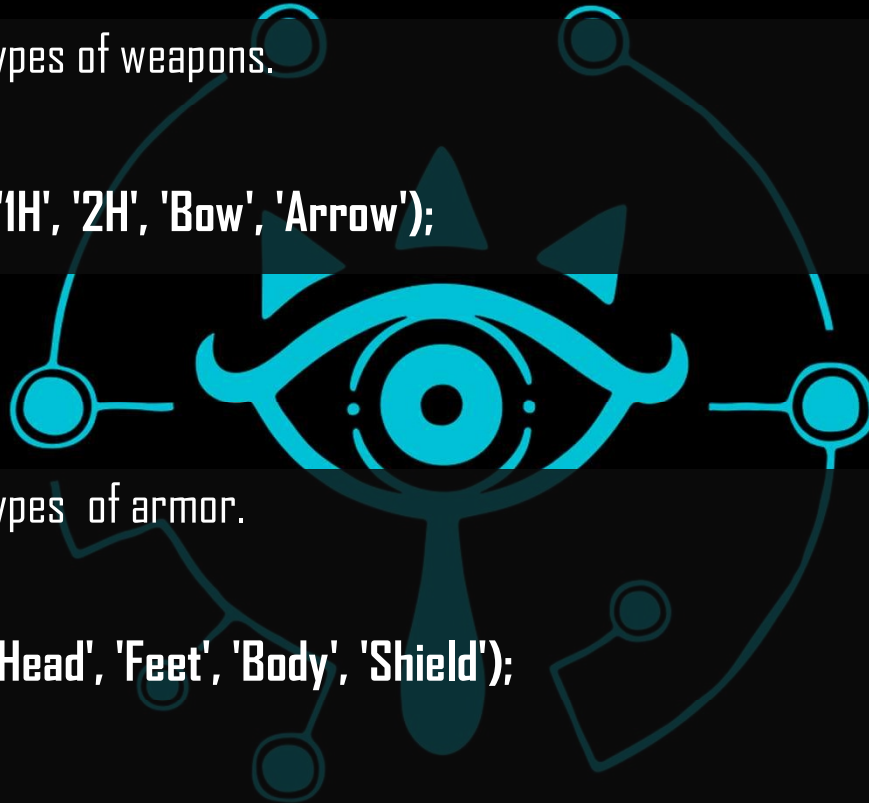This type specifies the different types of weapons.

```sql
CREATE TYPE wType AS ENUM ('1H', '2H', 'Bow', 'Arrow');
```

aType

This type specifies the different types of armor.

```sql
CREATE TYPE aType AS ENUM ('Head', 'Feet', 'Body', 'Shield');
```

Types

effect

This type specifies the different effects items can bestow.

```
CREATE TYPE effect AS ENUM ('None','Heat Resistance', 'Cold Resistance',
'Stealth', 'Shock Resistance', 'Attack', 'Defense,', 'Stamina' , 'Health Increase',
'Movement');
```

effectStr

This type specifies the various strength levels of effects.

```
CREATE TYPE effectStr AS ENUM ('None','Low', 'Med' ,'High');
```

Types

Items

This table lists all the items in BOTW.

CREATE TABLE IF NOT EXISTS Items (

    xid   INT default   nextval('xid_id_seq')UNIQUE NOT NULL,

    itemName   TEXT NOT NULL UNIQUE,

    Primary Key (xid)

    );

Dependencies:

    xid → itemName

xid integer	itemname text
1	Keese Wing
2	Keese Eyeball
3	Ruby
4	Diamond
5	Sapphire
6	Hearty Salmon
7	Raw Gourmet Meat
8	Blue Nightshade
9	Silent Princess
10	Apple
11	Wildberry
12	Ancient Spring
13	Ancient Core

Tables

KeyItems

This table lists all the important items in BOTW.

```
CREATE TABLE IF NOT EXISTS keyitems(
  xid              INT NOT NULL REFERENCES Items(xid) UNIQUE,
  description      TEXT NOT NULL,
  Primary Key  (xid)
);
```

Dependencies:

xid → description

xid integer	description text
20	Allows you to glide from high places

Tables

Equipment

Equipment is a subtype of Items. Within this table are all items Link can equip.

```
CREATE TABLE IF NOT EXISTS Materials(
  xid        INT NOT NULL REFERENCES Items(xid) UNIQUE,
  Primary Key     (xid)
);
```

Dependencies:

xid$\rightarrow$

xid integer
21
22
23
24
25
26
27
28
29
30
31

Tables

Armor

Armor is a subtype of Equipment. Within this table are all items Link can equip to raise his defense.

xid integer	basedef integer	armortype atype
21	2	Head
22	3	Body

```
CREATE TABLE IF NOT EXISTS Armor(
    xid          INT NOT NULL REFERENCES Equipment(xid)
                 UNIQUE,
    baseDef      INT NOT NULL,
    armorType    atype NOT NULL,
    Primary Key (xid)
);
```

Dependencies:
    xid → baseDef

Tables

Weapons

Weapons is a subtype of Equipment. Within this table are all items Link can equip to raise his Attack

```
CREATE TABLE IF NOT EXISTS Weapons (
  xid        INT NOT NULL REFERENCES Equipment(xid)UNIQUE,
  baseAtk    INT NOT NULL,
  WeaponType  wType
);
```

Dependencies:
         xid→baseAtk, WeaponType

xid integer	baseatk integer	weapontype wtype
23	26	2H
24	24	1H
25	30	1H
26	26	Bow
27	10	Bow
28	0	Arrow
29	0	Arrow

Tables

Materials

Materials is a subtype of Items. Within this table are all items used to craft or cook other items.

```
CREATE TABLE IF NOT EXISTS Materials(
  xid          INT NOT NULL REFERENCES Items(xid) UNIQUE,
  typeOfMaterial  typeofMaterial NOT NULL,
  salePriceRupees INT NOT NULL,
  Primary Key    (xid)
);
```

Dependencies:

xid→ typeOfMaterial, salePriceRupees

xid integer	typeofmaterial typeofmaterial	salepricerupees integer
1	Monster Part	2
2	Monster Part	20
3	Ore	210
4	Ore	500
5	Ore	260
6	Meat	10
7	Meat	35
8	Vegetation	4
9	Vegetation	10
10	Fruit	3
11	Fruit	3
12	Machinery	15
13	Machinery	80

Tables

MaterialsWithEffects

This is a subtype of Materials. This table keeps tabs on Material Items that have special effect boosts

```
CREATE TABLE IF NOT EXISTS MaterialsWithEffects(
    xid  INT NOT NULL REFERENCES materials(xid) UNIQUE,
    effectBoost  effect NOT NULL,
    effectBoostStrength    effectStr NOT NULL,
    Primary key (xid)
);
```

Dependencies
    xid → effectBoost, effectBoostStrength

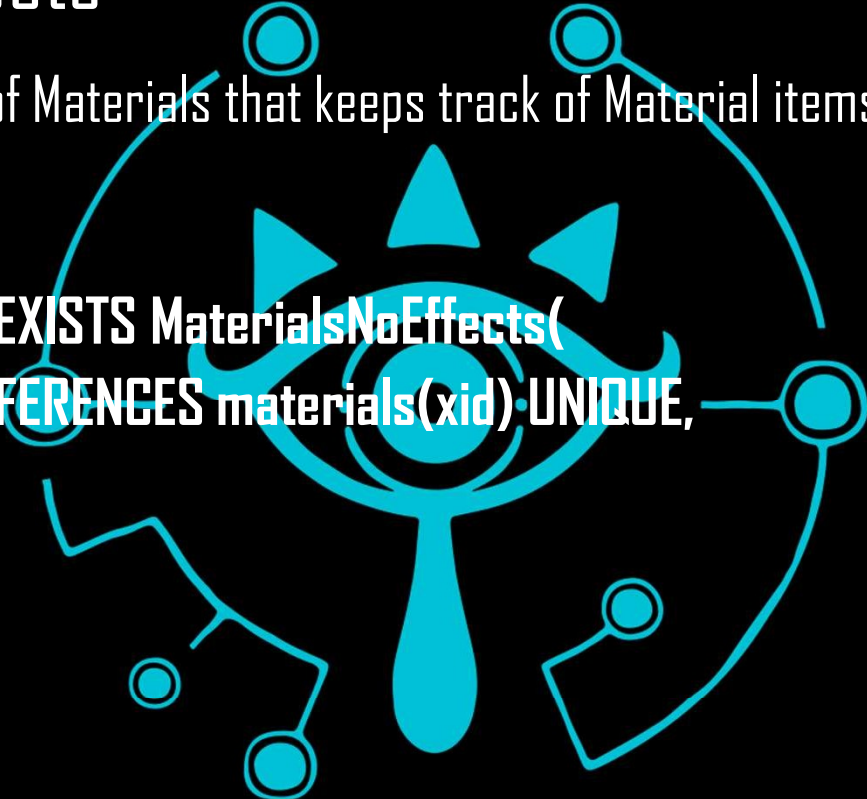| xid integer	effectboost effect	effectbooststrength effectstr
6	Health Increase	Low
8	Stealth	Low
9	Stealth	Med

Tables

MaterialNoEffects

This table is a subtype of Materials that keeps track of Material items that don't have effects.

```
CREATE TABLE IF NOT EXISTS MaterialsNoEffects(
  xid  INT NOT NULL REFERENCES materials(xid) UNIQUE,
  Primary key(xid)
);
```

Dependencies:
xid →

xid integer
5
6
12
13

Tables

EdibleMaterials

This is a subtype of both HasEffects and HasNoEffects. This keeps track of the Material items Link can eat.

```
CREATE TABLE IF NOT EXISTS EdibleMaterials(
    xid          INT NOT NULL REFERENCES materials(xid) UNIQUE,
    heartsRestored  FLOAT NOT NULL,
    Primary Key (xid)
    );
```

Dependencies:
    xid → heartsRestored

xid integer	heartsrestored double precision
6	4
7	5
10	0.5
11	0.5

Tables

NonEdibleMaterials

This is a subtype of both MaterialsWithEffects and MaterialsNoEffects. It keeps track of all the Material items Link can not eat.

CREATE TABLE IF NOT EXISTS NonEdibleMaterials(

    xid   INT NOT NULL REFERENCES materials(xid) UNIQUE,

    Primary Key (xid)

    );

Dependencies

    xid →

xid integer
1
2
3
4
5
12
13
14
15

Tables

Monsters

This table contains monsters and enemies.

```
CREATE TABLE IF NOT EXISTS Monsters(
monID          INT default nextval('monID_id_seq'), UNIQUE
monsterName    TEXT NOT NULL,
Primary Key (monID)
);
```

Dependencies:

monid→monsterName

monid integer	monstername text
1	Keese
2	Fire Keese
3	Electric Keese

Tables
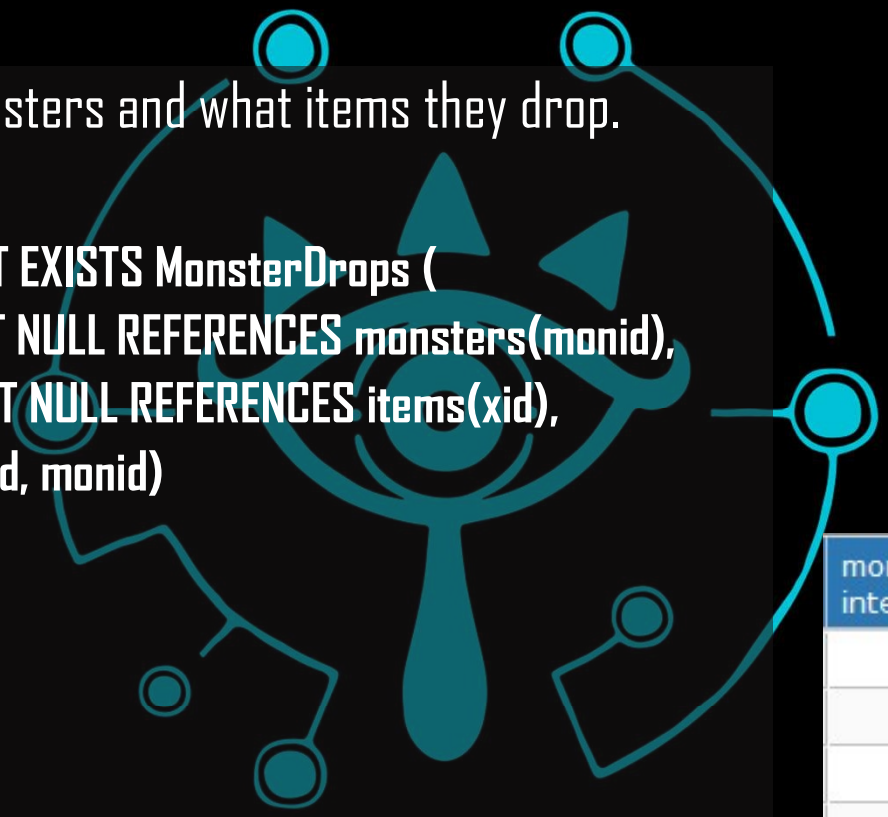
MonsterDrops

This table contains Monsters and what items they drop.

```
CREATE TABLE IF NOT EXISTS MonsterDrops (
    monid      INT NOT NULL REFERENCES monsters(monid),
    xid        INT NOT NULL REFERENCES items(xid),
    Primary Key   (xid, monid)
);
```

Dependencies:
  monid→
  xid →

monid integer	xid integer
1	1
1	2
2	1
2	2
3	2

Tables

Locations

This table contains locations and places.

```
CREATE TABLE IF NOT EXISTS Locations (
    lid INT default nextval('lid_id_seq') NOT NULL UNIQUE,
    locationName TEXT NOT NULL,
    Primary Key (lid)
);
```

Dependencies:
    lid→locationName

lid integer	locationname text
1	Zoras Domain
2	Central Hyrule
3	Tabantha Frontier
4	Hebra
5	Eldin
6	East Nucluda
7	West Nucluda

Tables

HasItem

This table tracks locations and items that can be found within them.

```
CREATE TABLE IF NOT EXISTS HasItem(
    lid          INT NOT NULL REFERENCES locations(lid),
    xid          INT NOT NULL REFERENCES items(xid),
     Primary key   (xid,lid)
 );
Dependencies:
    lid →
    xid→
```

lid integer	xid integer
1	10
1	11
5	8
7	9
3	14
2	15

Tables

HasMonster

This table tracks monsters and where they can be found

```
CREATE TABLE IF NOT EXISTS HasMonster(
    lid         INT NOT NULL REFERENCES locations(lid),
    monid         INT NOT NULL REFERENCES monsters(monid),
    Primary key   (xid,lid)
  );
Dependencies:
    lid →
    monid→
```

lid integer	monid integer
1	3
5	2
7	3

Tables

Players

Keeps track players playing on the same console.

```
CREATE TABLE IF NOT EXISTS Players(
  pid        INT default nextval('pid_id_seq')UNIQUE  NOT NULL,
  playerName  TEXT NOT NULL,
  Primary Key (pid)
);
Dependencies
    Pid → playerName
```

pid integer	playerna... text
1	Alan
2	Skull Kid
3	Tilted
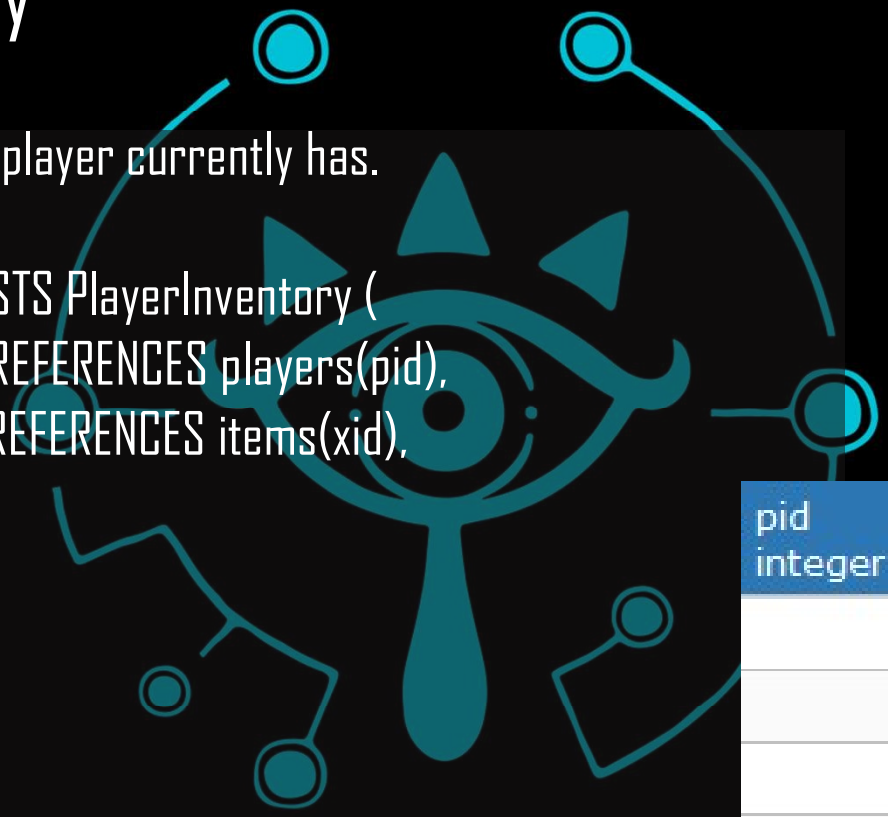
Tables

PlayerInventory

Keeps track of items the player currently has.

```
CREATE TABLE IF NOT EXISTS PlayerInventory (
  pid        INT NOT NULL REFERENCES players(pid),
  xid        INT NOT NULL REFERENCES items(xid),
  Qty        INT NOT NULL,
  Primary Key (pid, xid)
);
```

Dependencies:
    pid, xid→ qty

pid integer	xid integer	qty integer
1	31	2
1	2	20
1	4	100
2	7	1

Tables

ArmorStats

This view is an extension of Armor table that includes the Armor's name for easy identification

```
CREATE OR REPLACE VIEW ArmorStats AS
select items.xid, itemName, armorType  ,baseDef
from items, Armor
where items.xid=armor.xid;
```

xid integer	itemname text	armortype atype	basedef integer
21	Stealth Mask	Head	2
22	Zora Armor	Body	3

views

WeaponStats

This view is an extension of Armor table that includes the Armor's name for easy identification

```
CREATE OR REPLACE VIEW WeaponStats AS
    SELECT items.xid, itemName, baseAtk
    FROM weapons, items
    WHERE items.xid=weapons.xid;
```

xid integer	itemname text	baseatk integer
23	Lightscale Trident	26
24	Flameblade	24
25	Great Frostblade	30
26	Knights Bow	26
27	Lynel Bow	10
28	Ancient Arrow	0
29	Bomb Arrow	0

views

MatsEffectsInfo

This view contains all items with effects, their sale price, effect boost, effect boost strength and type of material

CREATE OR REPLACE VIEW MatsEffectsInfo   AS

SELECT materialswitheffects.xid, itemname, salepricerupees, effectboost,   effectbooststrength, typeofmaterial

FROM materials, materialswitheffects,items

WHERE materials.xid=materialswitheffects. items.xid=materials.xid

xid inte...	itemname text	sale... inte...	effectbo... effect	effectbo... effectstr	typeofmaterial typeofmaterial
6	Hearty Salmon	10	Health In...	Low	Meat
8	Blue Nightshade	4	Stealth	Low	Vegetation
9	Silent Princess	10	Stealth	Med	Vegetation

views

HealthRestoreItems

This view contains all items with effects, their sale price, effect boost, effect boost strength and type of material

```
CREATE OR REPLACE VIEW HealthRestoreItems AS
SELECT ediblematerials.xid, itemname, heartsrestored
FROM items,ediblematerials
WHERE ediblematerials.xid=items.xid
```

views

PlayerInventoryMaterialStats

This view contains an items, sale price, material type, name, and location

Create or replace VIEW PlayerInventoryMaterialStats AS

SELECT playername, playerinventory.xid, itemname, typeOfMaterial,qty, salepricerupees, lid
FROM players, playerinventory, materials, items, hasItem
WHERE playerinventory.xid=materials.xid
AND players.pid=playerinventory.pid
AND playerinventory.xid=items.xid
AND hasitem.xid=playerinventory.xid

playerna... text	xid integer	itemname text	typeofm... typeofm...	qty integer	salepriceru... integer	lid integer
Alan	10	Apple	Fruit	2	3	1
Alan	11	Wildberry	Fruit	20	3	1
Alan	15	Cold Dar...	Critter	100	2	2
Skull Kid	9	Silent Pri...	Vegetation	1	10	7

views

AtkgreaterThan

This function takes in an number and returns all weapons with attack stat greater than or equal to it

```
select weaponatk(10, 'results');
Fetch all from results;
```

xid integer	itemname text	baseatk integer
23	Lightscale Trident	26
24	Flameblade	24
25	Great Frostblade	30
26	Knights Bow	26
27	Lynel Bow	10

```
create or replace function weaponAtk (int, REFCURSOR) returns refcursor as
$$

declare
   atk int := $1;
   resultset REFCURSOR := $2;
begin
      open resultset for
select  items.xid,itemname, baseAtk
      from items, weapons
      where items.xid=weapons.xid AND baseAtk in
                                (select baseAtk
      from weapons
      where baseAtk>=atk);
      return resultset;

      end;
$$
language plpgsql;
```

Stored Procedures

DefGreaterThan

This function takes a number and returns all armor with a defense stat greater than or equal to entered number

select defgreaterthan(1, 'results');
Fetch all from results;

xid integer	itemname text	basedef integer
21	Stealth Mask	2
22	Zora Armor	3

```
create or replace function DefGreaterThan (int, REFCURSOR) returns refcursor as
$$

declare
    def int := $1;
    resultset REFCURSOR := $2;
begin
        open resultset for
        select items.xid,itemname, basedef
            from items, armor
            where items.xid=armor.xid AND basedef in
                            (select basedef
            from weapons
            where basedef>=def);
        return resultset;

    end;
$$
language plpgsql;
```

Stored Procedures

LocHasItems

This function takes in a location name and returns all items that can be found in that location

itemname text	xid integer	lid integer
Apple	10	1
Wildberry	11	1

```
create or replace function locHasItems (text, REFCURSOR) returns refcursor as
$$
declare
    loc text := $1;
    resultset REFCURSOR := $2;
begin

    open resultset for
    SELECT itemname, items.xid, locations.lid
    FROM items, locations, hasitem
    WHERE hasitem.xid=items.xid
    AND   hasitem.lid=locations.lid
    AND hasitem.lid in
        (SELECT lid
        FROMhasitem
        WHERE loc LIKE locationName);
    return resultset;
    end;
$$
language plpgsql;
```

Stored procedures

RemoveFromInvent

This trigger automatically removes an item from the PlayersInventory when the Quantity is set to 0 or below. I know it looks like I just deleted it, but I promise I didn't!

pid integer	xid integer	qty integer
1	2	20
1	4	100
2	7	1
1	31	2

pid integer	xid integer	qty integer
1	2	20
1	4	100
2	7	1

```
CREATE OR REPLACE FUNCTION removeFromInvent()
RETURNS TRIGGER AS
$$
BEGIN
IF new.Qty <= 0
THEN

DELETE FROM PlayerInventory
WHERE PlayerInventory.qty =0;
END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER removeFromInvent
AFTER UPDATE on PlayerInventory
FOR EACH ROW
EXECUTE PROCEDURE removeFromInvent();
```

Triggers

Query to View Number of Distinct Items In Inventory

SELECT playername, count(xid) As NumOfDiffItems
FROM players, playerinventory
WHERE players.pid= playerinventory.pid
GROUP BY playername

playerna... text	numofdif... bigint
Skull Kid	1
Alan	3

Reports

Query to View Total Number of Items In Inventory

playerna... text	numofite... bigint
Skull Kid	1
Alan	122

```
select playername, SUM(qty) AS numOfItems
FROM players, playerinventory
WHERE players.pid= playerinventory.pid
GROUP BY playername
```

Reports

Query To View Number Of Rupees A Bulk Of Materials Can Be Sold For.

select playername, playerinventory.xid, qty, (salepriceRupees*qty) AS BulkSellPriceRupees

FROM players, playerinventory, materials

WHERE players.pid= playerinventory.pid AND playerinventory.xid=materials.xid

playerna... text	xid integer	qty integer	bulksellpricerupees integer
Alan	2	20	400
Alan	4	100	50000
Skull Kid	7	1	35

Reports

# Database Roles

This database only has two roles.

Admin: Has access to entire database.

CREATE ROLE admin;
GRANT ALL ON ALL TABLES IN SCHEMA public TO admin;

Player: Only has access to Players and PlayerInventory.

CREATE ROLE player;
REVOKE ALL ON ALL TABLES IN SCHEMA public FROM player;
GRANT INSERT ON Players, PlayerInventory TO player;
GRANT UPDATE ON Players, PlayerInventory TO player;
GRANT SELECT ON ALL TABLES IN SCHEMA public TO player;

Triggers
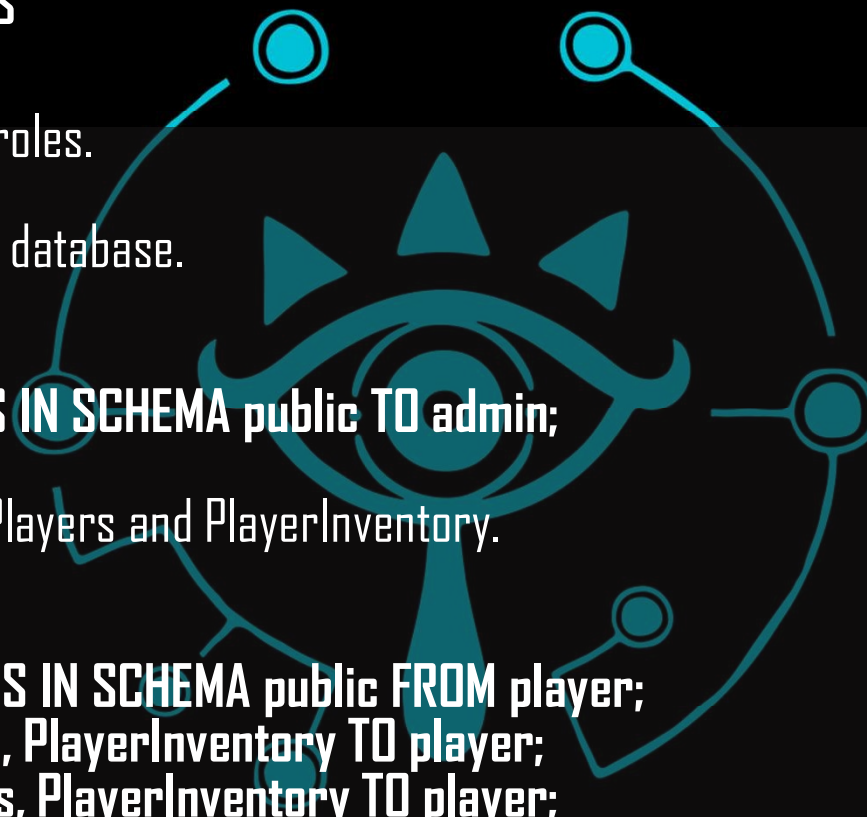
Known Issues

This database does not include or has lack of:

- The amount of energy replenished by items.
- Stats for equipment after upgrade at a fairy fountatin
- Stored Procedures to search for basic information. (I.E Enter monster and find all locations they are native to.)

Only sale prices for Materials are included. The Report Query will only return the total sale price for items that are _materials_. This may cause confusion for those who use the query but don't see totals for items such as equipment.

Future Enhancements

Future updates can include the following:

- Locations for Korok Seeds, Shrines, and Memories
- A recipe table, detailing what materials can be cooked together to make   certain   dishes. Along with a Food table subtype.
- Health of Monsters
- Buying price for materials
- Buying and selling price for Equipment. Including notation of what items can't be bought/can only be attained through main story line.
- Notation of what weapons catch on fire/will get Link electrocuted and killed during a thunderstorm
- All animals Link can ride, including 20ft horses.
- Database itself can expand beyond items and include weather conditions, NPCs, Quests and much more.