**Пояснительная записка**

Аннотация

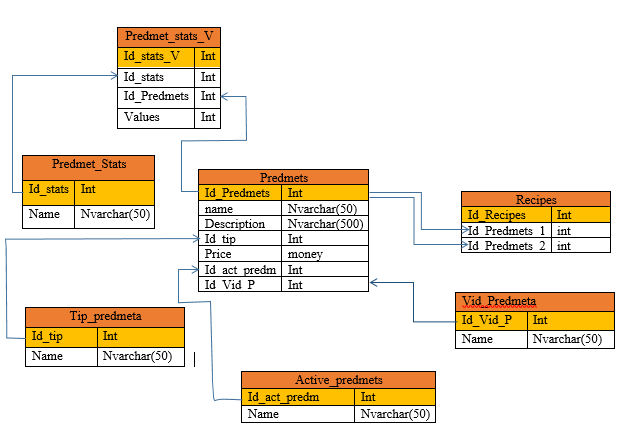
В данной пояснительной записке рассматривается создание базы данных Разработка базы данных предметов Dota 2.

Для создания базы данных была выбрана СУБД Microsoft SQL Server. Выбор обусловлен тем, что данная СУБД широка распространена как в среде разработчиков, так и в среде производителей. Она имеет бесплатную версию, отличную техническую поддержку, широкое сообщество разработчиков, множество разнообразных средств для работы с СУБД, таких как Microsoft Management Studio. Также на официальном сайте СУБД можно найти множество технической документации.

Введение

Разработать базу данных на тему предметы Dota 2, которая должна содержать в себе данные предметов, вид предметов, рецепты, тип предмета, статы предметов, статы предметов дополнительные. А также иметь триггеры, которые автоматически обновляет цену предметов , также для контроля стоимости рецепта и для отслеживания изменений предметов . Представление, которое выводит информацию о всех доступных рецептов, а также представление для отображения статистики по типам предметов и представление с отображением активных предметов с их способностью. Хранимая процедура , которая необходима для добавления нового предмета , а также процедура для расчета полной стоимости общего рецепта.

**1 Построение структуры базы данных**

Рис.1 Структура базы данных

**2 Разработка интерфейса БД**

Согласно заданию, мною были написаны следующие скрипты на языке Transact-SQL:

Для создания таблицы Predmet\_Stats

CREATE TABLE Predmet\_Stats (

Id\_stats INT PRIMARY KEY,

Name NVARCHAR(50)

);

Для создания таблицы Tip\_predmeta

CREATE TABLE Tip\_predmeta (

Id\_tip INT PRIMARY KEY,

Name NVARCHAR(50)

);

Для создания таблицы Active\_predmets

CREATE TABLE Active\_predmets (

Id\_act\_predm INT PRIMARY KEY,

Name NVARCHAR(50)

); Для создания таблицы Vid\_Predmeta

CREATE TABLE Vid\_Predmeta (

Id\_Vid\_P INT PRIMARY KEY,

Name NVARCHAR(50)

);

Для создания таблицы Predmets

CREATE TABLE Predmets (

Id\_Predmets INT PRIMARY KEY,

Name NVARCHAR(50),

Description NVARCHAR(500),

Id\_tip INT,

Price MONEY,

Id\_act\_predm INT,

Id\_Vid\_P INT,

Count INT, -- Добавлен столбец Count

FOREIGN KEY (Id\_tip) REFERENCES Tip\_predmeta(Id\_tip),

FOREIGN KEY (Id\_act\_predm) REFERENCES Active\_predmets(Id\_act\_predm),

FOREIGN KEY (Id\_Vid\_P) REFERENCES Vid\_Predmeta(Id\_Vid\_P)

);

Для создания таблицы Predmet\_stats\_V

CREATE TABLE Predmet\_stats\_V (

Id\_stats\_V INT PRIMARY KEY,

Id\_stats INT,

Id\_Predmets INT,

FOREIGN KEY (Id\_stats) REFERENCES Predmet\_Stats(Id\_stats),

FOREIGN KEY (Id\_Predmets) REFERENCES Predmets(Id\_Predmets)

);

Для создания таблицы Recipes

CREATE TABLE Recipes (

Id\_Recipes INT PRIMARY KEY,

Id\_Predmets\_1 INT,

Id\_Predmets\_2 INT,

FOREIGN KEY (Id\_Predmets\_1) REFERENCES Predmets(Id\_Predmets),

FOREIGN KEY (Id\_Predmets\_2) REFERENCES Predmets(Id\_Predmets)

);VinylID Int Not null references VINYL(Id)

);

**3 Разработка запросов пользователей к базе данных**

Для автоматизации работы базы данных были написаны следующие скрипты на языке Transact-SQL:

Был создан триггер для таблицы Predmets который предназначен для автоматического обновления цены предмета :

CREATE TRIGGER UpdateItemPrice

ON Predmet\_Stats

AFTER UPDATE

AS

BEGIN

UPDATE p

SET Price = (

SELECT AVG(Price)

FROM Predmets

WHERE Id\_tip IN (SELECT Id\_tip FROM INSERTED)

)

FROM Predmets p

INNER JOIN INSERTED i ON p.Id\_tip = i.Id\_stats;

END;

GO

Был создан триггер для таблицы Recipes для контроля стоимости рецепта.

CREATE TRIGGER RecipeCostControl

ON Recipes

AFTER INSERT, UPDATE

AS

BEGIN

DECLARE @TotalPrice MONEY;

SELECT @TotalPrice = SUM(Price)

FROM Predmets

WHERE Id\_Predmets IN (SELECT Id\_Predmets\_1 FROM INSERTED)

OR Id\_Predmets IN (SELECT Id\_Predmets\_2 FROM INSERTED);

IF @TotalPrice <> (SELECT AVG(Price) FROM Predmets WHERE Id\_Predmets IN (SELECT Id\_Predmets\_1 FROM INSERTED))

BEGIN

RAISERROR (N'Стоимость рецепта не соответствует общей стоимости компонентов.', 16, 1);

ROLLBACK TRANSACTION;

END;

END;

GO

Был создан триггер для таблицы Predmets который отслеживает изменения в количестве предметов на складе:

CREATE TRIGGER UpdateInventoryCount

ON Predmets

AFTER INSERT, DELETE

AS

BEGIN

UPDATE Predmets

SET Count = (SELECT COUNT(\*) FROM INSERTED);

END;

GO

Была создана хранимая процедура, которая необходима для добавления нового предмета:

CREATE PROCEDURE AddNewItem

@Name NVARCHAR(50),

@Description NVARCHAR(500),

@Id\_tip INT,

@Price MONEY,

@Id\_act\_predm INT,

@Id\_Vid\_P INT

AS

BEGIN

INSERT INTO Predmets (Name, Description, Id\_tip, Price, Id\_act\_predm, Id\_Vid\_P)

VALUES (@Name, @Description, @Id\_tip, @Price, @Id\_act\_predm, @Id\_Vid\_P);

END;

GO

Была создана хранимая процедура, процедуры для расчета общей стоимости рецепта :

CREATE PROCEDURE CalculateRecipeCost

@RecipeId INT

AS

BEGIN

DECLARE @TotalCost MONEY;

SELECT @TotalCost = SUM(Price)

FROM Predmets

WHERE Id\_Predmets IN (SELECT Id\_Predmets\_1 FROM Recipes WHERE Id\_Recipes = @RecipeId)

OR Id\_Predmets IN (SELECT Id\_Predmets\_2 FROM Recipes WHERE Id\_Recipes = @RecipeId);

SELECT @TotalCost AS TotalCost;

END;

GO

Было создано представление, для отображения всех доступных рецептов

CREATE VIEW AvailableRecipes

AS

SELECT r.Id\_Recipes, p1.Name AS Component1, p2.Name AS Component2

FROM Recipes r

INNER JOIN Predmets p1 ON r.Id\_Predmets\_1 = p1.Id\_Predmets

INNER JOIN Predmets p2 ON r.Id\_Predmets\_2 = p2.Id\_Predmets;

GO

Была создана процедура для поиска предметов по определенным характеристикам:

CREATE PROCEDURE SearchItemsByCharacteristic

@Characteristic NVARCHAR(50)

AS

BEGIN

SELECT \*

FROM Predmets p

INNER JOIN Predmet\_stats\_V ps ON p.Id\_Predmets = ps.Id\_Predmets

WHERE ps.Id\_stats = @Characteristic;

END;

GO

Было создано представление для отображения статистики по типам предметов :

CREATE VIEW ItemStatisticsByType

AS

SELECT t.Name AS ItemType, COUNT(\*) AS ItemCount, AVG(p.Price) AS AvgPrice

FROM Predmets p

INNER JOIN Tip\_predmeta t ON p.Id\_tip = t.Id\_tip

GROUP BY t.Name;

GO

Было создано представление для отображения активных предметов с их статистикой :

CREATE VIEW ActiveItemsWithStats

AS

SELECT p.Name AS ItemName, p.Description, a.Name AS Activity, ps.Name AS Statistic

FROM Predmets p

INNER JOIN Active\_predmets a ON p.Id\_act\_predm = a.Id\_act\_predm

INNER JOIN Predmet\_stats\_V psv ON p.Id\_Predmets = psv.Id\_Predmets

INNER JOIN Predmet\_Stats ps ON psv.Id\_stats = ps.Id\_stats;

**4 Организация защиты информации, хранящийся в базе данных**

Для базы данных были созданы: Admin\_login с паролем 999 и User\_login с паролем 111. У Admin\_login есть права на выборку, вставку, обновление и удаление записей из таблицы, а у User\_login только на выборку данных

create login Admin\_login

with password = '999' must\_change,

check\_expiration = on;

go

create user AdminLog for login Admin\_login;

go

grant select, insert, update,

delete on object::HillClimbRacing2.dbo.Predmet\_Stats to AdminLog

go

grant select, insert, update,

delete on object::Dota2PredmetsBD;.dbo.Tip\_predmeta to AdminLog

go

grant select, insert, update,

delete on object::Dota2PredmetsBD;.dbo.Active\_predmets to AdminLog

go

grant select, insert, update,

delete on object::Dota2PredmetsBD;.dbo.MAP to AdminLog

go

grant select, insert, update,

delete on object::Dota2PredmetsBD**;**.dbo.Vid\_Predmeta to AdminLog

go

grant select, insert, update,

delete on object:: Dota2PredmetsBD.dbo.Predmets to AdminLog

go

grant select, insert, update,

delete on object:: Dota2PredmetsBD.dbo.VINYL to AdminLog

go

grant select, insert, update,

delete on object:: Dota2PredmetsBD.dbo.ZAEZD to AdminLog

create login User\_login

with password = '111' must\_change,

check\_expiration = on;

go

create user UserLog for login User\_login;

go

grant select on object:: Dota2PredmetsBD.dbo. Predmet\_Stats to UserLog

go

grant select on object:: Dota2PredmetsBD.dbo. Tip\_predmeta to UserLog

go

grant select on object:: Dota2PredmetsBD.dbo. Active\_predmets to UserLog

go

grant select on object:: Dota2PredmetsBD.dbo. Vid\_Predmeta to UserLog

go

grant select on object:: Dota2PredmetsBD.dbo. Predmets to UserLog

go

grant select on object:: Dota2PredmetsBD.dbo. Predmet\_stats\_V to UserLog

go

grant select on object:: Dota2PredmetsBD.dbo. Recipes to UserLog

go

**5 Модель структуры базы данных**

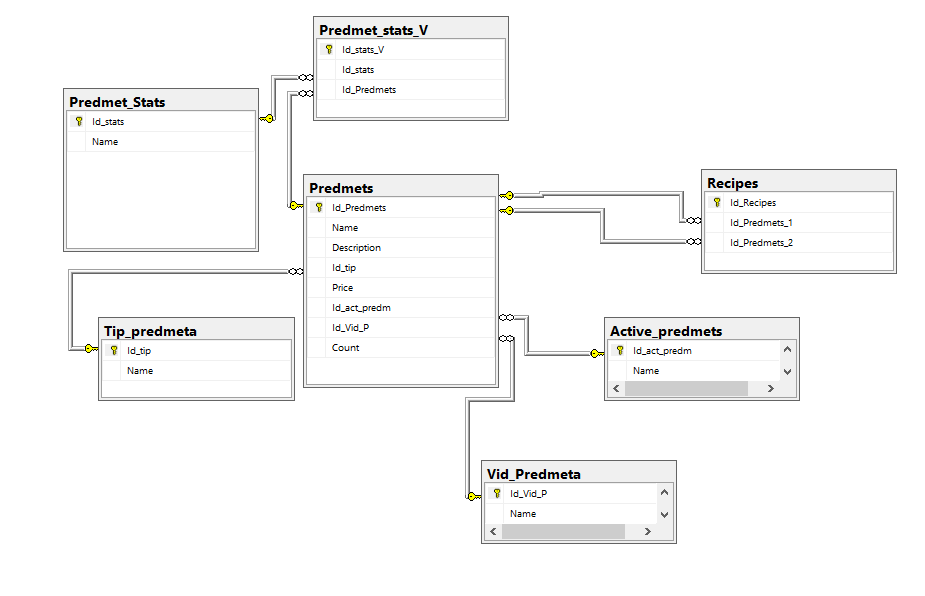


Рис.2 Модель структуры базы данных

**Список использованных источников**

1. Разработка, администрирование и защита баз данных: учебник для студ. учреждений сред. проф. образования / Г.Н. Федорова. – 5-е изд., стер. – М.: Издательский центр «Академия», 2021. – 288 с.
2. Основы проектирования баз данных: учебник для студ. учреждений сред. проф. образования / Г.Н. Федорова. – 5-е изд., стер. – М.: Издательский центр «Академия», 2021. – 224 с.
3. Проектирование и разработка информационных систем: учебник для студ. учреждений сред. проф. образования / О.Н. Перлова, О.П. Ляпина, А.В. Гусева – 4-е изд., испр. и доп. – М.: Образовательно-издательский центр «Академия», 2023. – 256 с.
4. Материалы сайта https://metanit.com/sql/sqlserver/
5. Материалы сайта https://docs.microsoft.com/ru-ru/sql
6. Материалы сайта https://metanit.com/sql/tutorial/

Приложение

Листинг интерфейса базы данных

-- Создание базы данных Dota2PredmetsBD

CREATE DATABASE Dota2PredmetsBD;

GO

USE Dota2PredmetsBD;

-- Создание таблицы Predmet\_Stats

CREATE TABLE Predmet\_Stats (

Id\_stats INT PRIMARY KEY,

Name NVARCHAR(50)

);

-- Создание таблицы Tip\_predmeta

CREATE TABLE Tip\_predmeta (

Id\_tip INT PRIMARY KEY,

Name NVARCHAR(50)

);

-- Создание таблицы Active\_predmets

CREATE TABLE Active\_predmets (

Id\_act\_predm INT PRIMARY KEY,

Name NVARCHAR(50)

);

-- Создание таблицы Vid\_Predmeta

CREATE TABLE Vid\_Predmeta (

Id\_Vid\_P INT PRIMARY KEY,

Name NVARCHAR(50)

);

-- Создание таблицы Predmets

CREATE TABLE Predmets (

Id\_Predmets INT PRIMARY KEY,

Name NVARCHAR(50),

Description NVARCHAR(500),

Id\_tip INT,

Price MONEY,

Id\_act\_predm INT,

Id\_Vid\_P INT,

Count INT, -- Добавлен столбец Count

FOREIGN KEY (Id\_tip) REFERENCES Tip\_predmeta(Id\_tip),

FOREIGN KEY (Id\_act\_predm) REFERENCES Active\_predmets(Id\_act\_predm),

FOREIGN KEY (Id\_Vid\_P) REFERENCES Vid\_Predmeta(Id\_Vid\_P)

);

-- Создание таблицы Predmet\_stats\_V

CREATE TABLE Predmet\_stats\_V (

Id\_stats\_V INT PRIMARY KEY,

Id\_stats INT,

Id\_Predmets INT,

FOREIGN KEY (Id\_stats) REFERENCES Predmet\_Stats(Id\_stats),

FOREIGN KEY (Id\_Predmets) REFERENCES Predmets(Id\_Predmets)

);

-- Создание таблицы Recipes

CREATE TABLE Recipes (

Id\_Recipes INT PRIMARY KEY,

Id\_Predmets\_1 INT,

Id\_Predmets\_2 INT,

FOREIGN KEY (Id\_Predmets\_1) REFERENCES Predmets(Id\_Predmets),

FOREIGN KEY (Id\_Predmets\_2) REFERENCES Predmets(Id\_Predmets)

);

-- Вставка данных в таблицу Predmet\_Stats

INSERT INTO Predmet\_Stats (Id\_stats, Name)

VALUES

(1, N'Сила'),

(2, N'Ловкость'),

(3, N'Интеллект'),

(4, N'Атака'),

(5, N'Защита');

-- Вставка данных в таблицу Tip\_predmeta

INSERT INTO Tip\_predmeta (Id\_tip, Name)

VALUES

(1, N'Оружие'),

(2, N'Броня'),

(3, N'Аксессуар'),

(4, N'Расходуемый'),

(5, N'Инструмент');

-- Вставка данных в таблицу Active\_predmets

INSERT INTO Active\_predmets (Id\_act\_predm, Name)

VALUES

(1, N'Активный'),

(2, N'Пассивный'),

(3, N'Утилитарный'),

(4, N'Расходуемый'),

(5, N'Аура');

-- Вставка данных в таблицу Vid\_Predmeta

INSERT INTO Vid\_Predmeta (Id\_Vid\_P, Name)

VALUES

(1, N'Ближний бой'),

(2, N'Дальний бой'),

(3, N'Магия'),

(4, N'Физический'),

(5, N'Утилитарный');

-- Вставка данных в таблицу Predmets

INSERT INTO Predmets (Id\_Predmets, Name, Description, Id\_tip, Price, Id\_act\_predm, Id\_Vid\_P)

VALUES

(1, N'Меч силы', N'Мощный меч, насыщенный силой.', 1, 100.00, 1, 1),

(2, N'Сапоги ловкости', N'Легкие сапоги, увеличивающие ловкость.', 2, 150.00, 1, 2),

(3, N'Палочка интеллекта', N'Волшебная палочка, усиливающая интеллект.', 3, 120.00, 1, 3),

(4, N'Зелье исцеления', N'Восстанавливает очки здоровья при употреблении.', 4, 50.00, 4, 4),

(5, N'Свиток телепортации', N'Телепортирует пользователя в целевое место.', 5, 75.00, 3, 5);

-- Вставка данных в таблицу Predmet\_stats\_V

INSERT INTO Predmet\_stats\_V (Id\_stats\_V, Id\_stats, Id\_Predmets)

VALUES

(1, 1, 1),

(2, 2, 2),

(3, 3, 3),

(4, 4, 4),

(5, 5, 5);

-- Вставка данных в таблицу Recipes

INSERT INTO Recipes (Id\_Recipes, Id\_Predmets\_1, Id\_Predmets\_2)

VALUES

(1, 1, 3),

(2, 2, 4),

(3, 3, 5),

(4, 4, 1),

(5, 5, 2);

GO -- Конец первой пакетной операции

-- Создание триггера для автоматического обновления цены предмета

CREATE TRIGGER UpdateItemPrice

ON Predmet\_Stats

AFTER UPDATE

AS

BEGIN

UPDATE p

SET Price = (

SELECT AVG(Price)

FROM Predmets

WHERE Id\_tip IN (SELECT Id\_tip FROM INSERTED)

)

FROM Predmets p

INNER JOIN INSERTED i ON p.Id\_tip = i.Id\_stats;

END;

GO

-- Создание процедуры для поиска предметов по определенным характеристикам

CREATE PROCEDURE SearchItemsByCharacteristic

@Characteristic NVARCHAR(50)

AS

BEGIN

SELECT \*

FROM Predmets p

INNER JOIN Predmet\_stats\_V ps ON p.Id\_Predmets = ps.Id\_Predmets

WHERE ps.Id\_stats = @Characteristic;

END;

GO

-- Создание триггера для контроля стоимости рецепта

CREATE TRIGGER RecipeCostControl

ON Recipes

AFTER INSERT, UPDATE

AS

BEGIN

DECLARE @TotalPrice MONEY;

SELECT @TotalPrice = SUM(Price)

FROM Predmets

WHERE Id\_Predmets IN (SELECT Id\_Predmets\_1 FROM INSERTED)

OR Id\_Predmets IN (SELECT Id\_Predmets\_2 FROM INSERTED);

IF @TotalPrice <> (SELECT AVG(Price) FROM Predmets WHERE Id\_Predmets IN (SELECT Id\_Predmets\_1 FROM INSERTED))

BEGIN

RAISERROR (N'Стоимость рецепта не соответствует общей стоимости компонентов.', 16, 1);

ROLLBACK TRANSACTION;

END;

END;

GO

-- Создание триггера для отслеживания изменений в количестве предметов на складе

CREATE TRIGGER UpdateInventoryCount

ON Predmets

AFTER INSERT, DELETE

AS

BEGIN

UPDATE Predmets

SET Count = (SELECT COUNT(\*) FROM INSERTED);

END;

GO

-- Создание процедуры для добавления нового предмета

CREATE PROCEDURE AddNewItem

@Name NVARCHAR(50),

@Description NVARCHAR(500),

@Id\_tip INT,

@Price MONEY,

@Id\_act\_predm INT,

@Id\_Vid\_P INT

AS

BEGIN

INSERT INTO Predmets (Name, Description, Id\_tip, Price, Id\_act\_predm, Id\_Vid\_P)

VALUES (@Name, @Description, @Id\_tip, @Price, @Id\_act\_predm, @Id\_Vid\_P);

END;

GO

-- Создание процедуры для расчета общей стоимости рецепта

CREATE PROCEDURE CalculateRecipeCost

@RecipeId INT

AS

BEGIN

DECLARE @TotalCost MONEY;

SELECT @TotalCost = SUM(Price)

FROM Predmets

WHERE Id\_Predmets IN (SELECT Id\_Predmets\_1 FROM Recipes WHERE Id\_Recipes = @RecipeId)

OR Id\_Predmets IN (SELECT Id\_Predmets\_2 FROM Recipes WHERE Id\_Recipes = @RecipeId);

SELECT @TotalCost AS TotalCost;

END;

GO

-- Создание представления для отображения всех доступных рецептов

CREATE VIEW AvailableRecipes

AS

SELECT r.Id\_Recipes, p1.Name AS Component1, p2.Name AS Component2

FROM Recipes r

INNER JOIN Predmets p1 ON r.Id\_Predmets\_1 = p1.Id\_Predmets

INNER JOIN Predmets p2 ON r.Id\_Predmets\_2 = p2.Id\_Predmets;

GO

-- Создание представления для отображения статистики по типам предметов

CREATE VIEW ItemStatisticsByType

AS

SELECT t.Name AS ItemType, COUNT(\*) AS ItemCount, AVG(p.Price) AS AvgPrice

FROM Predmets p

INNER JOIN Tip\_predmeta t ON p.Id\_tip = t.Id\_tip

GROUP BY t.Name;

GO

-- Создание представления для отображения активных предметов с их статистикой

CREATE VIEW ActiveItemsWithStats

AS

SELECT p.Name AS ItemName, p.Description, a.Name AS Activity, ps.Name AS Statistic

FROM Predmets p

INNER JOIN Active\_predmets a ON p.Id\_act\_predm = a.Id\_act\_predm

INNER JOIN Predmet\_stats\_V psv ON p.Id\_Predmets = psv.Id\_Predmets

INNER JOIN Predmet\_Stats ps ON psv.Id\_stats = ps.Id\_stats;

GO