НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
“ВЫСШАЯ ШКОЛА ЭКОНОМИКИ”

Факультет бизнеса и менеджмента
Программа «Бизнес-Информатика»

Шикунов Николай Алексеевич

ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА БАЗЫ ДАННЫХ АЭРОПОРТА

Курсовая работа студента 2 курса бакалавриата группы № ББИ172

Научный руководитель:
Доцент кафедры Бизнес-Аналитики
Ямпольский Сергей Михайлович

Москва, 2019

Оглавление

Оглавление

1. Постановка задачи

Целью данной работы является проектирование базы данных аэропорта, которая визуализирует основную работу аэропорта: табло прилётов и вылетов, регистрация пассажиров, работа диспетчера и функционирование объектов аэропорта.

1.1 Описание предметной области

С помощью базы данных можно узнать всю необходимую информацию о воздушных суднах, которые вылетают или прилетают в аэропорт. Аэропорт – крупное сооружение. Большую роль здесь занимают работники аэропорта, которые осуществляют все процессы, связанные с пассажирами и воздушными судами. С помощью базы данных многие процессы автоматизированы: менеджеры имеют возможность регистрировать пассажира на рейс, а диспетчеры могут назначить ленту выдачи багажа и взлётно-посадочную полосу для экипажа самолёта. Пассажиры наблюдают все изменения как для прилетающих рейсов, так и для вылетающих.

2. Модель базы данных

2.1 Описание сущностей

2.1.1 Сущности

- Model – сущность, содержащая информацию о модели воздушного судна;
- Airline – сущность, содержащая информацию об авиакомпании;
- Aircraft – сущность, содержащая информацию о самолёте, который принадлежит какой-либо авиакомпании. Сущность формируется из идентификатора модели самолёта и авиакомпании;
- City – сущность, содержащая информацию о городе;
- Airport – сущность, содержащая информацию об аэропорте. Сущность формируется из идентификатора города;
- Path – сущность, содержащая информацию о маршруте. Сущность формируется из идентификатора аэропорта;
- Flight – сущность, содержащая информацию о рейсе. Сущность формируется из идентификатора самолёта и маршрута;
- Person – сущность, содержащая информацию о человеке;
- Passenger – сущность, содержащая информацию о пассажире. Сущность формируется из идентификатора человека;

- Employee – сущность, содержащая информацию о работнике аэропорта. Сущность формируется из идентификатора человека;
- Manager – сущность, содержащая информацию о менеджере. Сущность формируется из идентификатора человека;
- Operator – сущность, содержащая информацию о диспетчере. Сущность формируется из идентификатора человека;
- Runway – сущность, содержащая информацию о взлётно-посадочной полосе. Сущность состоит из идентификатора диспетчера;
- Departure – сущность, содержащая информацию об отправляющемся рейсе. Сущность формируется из идентификатора рейса, взлётно-посадочной полосы и диспетчера;
- Check_in – сущность, содержащая информацию о зарегистрированном пассажире. Сущность состоит из идентификатора пассажира и отправляющегося рейса;
- Belt – сущность, содержащая информацию о ленте выдачи багажа. Сущность состоит из идентификатора диспетчера;
- Arrival – сущность, содержащая информацию о прибывающем рейсе. Сущность состоит из идентификатора рейса, взлётно-посадочной полосы, ленты и диспетчера;

2.1.2 Атрибуты сущностей

1) Model

Поля	Тип данных	Описание поля	Комментарий
Model_id	int	Идентификатор модели	PK
Name	Nvarchar(25)	Название модели самолёта	
Capacity	int	Вместимость самолёта	
Runway_min	float	Минимальная ВВП самолёта	

2) Airline

Поля	Тип данных	Описание поля	Комментарий
Airline_id	int	Идентификатор авиакомпании	PK
Name	nvarchar(20)	Название авиакомпании	
Baggage_limit	float	Максимальная масса багажа по правилам компании	
Code	nvarchar(2)	Краткий позывной авиакомпании	

3) Aircraft

Поля	Тип данных	Описание поля	Комментарий
Aircraft_id	int	Идентификатор самолёта	PK
Airline_id	int	Идентификатор авиакомпании	FK
Model_id	int	Идентификатор модели	FK
Production_date	datetime	Дата производства самолёта	

4) City

Поля	Тип данных	Описание поля	Комментарий
City_id	int	Идентификатор города	PK
Name	nvarchar(20)	Название города	

5) Airport

Поля	Тип данных	Описание поля	Комментарий
Airport_id	int	Идентификатор аэропорта	PK
City_id	int	Идентификатор города	FK
Name	nvarchar(20)	Название аэропорта	

6) Path

Поля	Тип данных	Описание поля	Комментарий
Path_id	nvarchar(4)	Идентификатор маршрута	PK
Airport_id	int	Идентификатор аэропорта	FK
Type	nchar(1)	Тип рейса (прилёт/вылет)	

7) Flight

Поля	Тип данных	Описание поля	Комментарий
Flight_id	int	Идентификатор рейса	PK
Aircraft_id	int	Идентификатор самолёта	FK
Departure_time	datetime	Время вылета	
Arrival_time	datetime	Время прилёта	
Passengers_amount	int	Количество человек на борту	
Path_id	nvarchar(4)	Идентификатор маршрута	FK

8) Person

Поля	Типа данных	Описание поля	Комментарий
Person_id	int	Идентификатор человека	PK
Name	nvarchar(20)	Имя человека	
Surname	nvarchar(20)	Фамилия человека	
Passport_number	int	Номер паспорта	
Birthdate	date	Дата рождения	
Gender	nvarchar(1)	Пол	

9) Passenger

Поля	Тип данных	Описание поля	Комментарий
Person_id	int	Идентификатор человека	PK, FK
Last_flight	date	Дата последнего перелёта	
Sky_priority	bit	Особые привилегии	

10) Employee

Поля	Тип данных	Описание поля	Комментарий
Person_id	int	Идентификатор человека	PK, FK
Position	nvarchar(20)	Позиция	
Salary	decimal	Зарплата	
Email	nvarchar(20)	Электронный ящик	
Phone	int	Номер телефона	

11) Manager

Поля	Тип данных	Описание поля	Комментарий
Person_id	int	Идентификатор человека	PK, FK

12) Operator

Поля	Тип данных	Описание поля	Комментарий
Person_id	int	Идентификатор человека	PK, FK

13) Runway

Поля	Тип данных	Описание поля	Комментарий
Runway_id	int	Идентификатор ВВП	PK
Length	float	Протяжённость ВВП	
Asphalt_type	nvarchar(20)	Тип покрытия	
Chief_operator	int	Идентификатор диспетчера	

14) Departure

Поля	Тип данных	Описание поля	Комментарий
Flight_id	int	Идентификатор рейса	PK, FK
Status	nvarchar(15)	Статус рейса	
Runway_id	int	Идентификатор ВВП	FK

Operator_id	int	Идентификатор диспетчера	FK

15) Check_in

Поля	Тип данных	Описание поля	Комментарий
Passenger_id	int	Идентификатор пассажира	PK, FK
Flight_id	int	Идентификатор рейса	PK, FK
Manager_id	int	Идентификатор менеджера	FK
Baggage_weight	float	Вес багажа	
Time	datetime	Время регистрации	

16) Belt

Поля	Тип данных	Описание поля	Комментарий
Belt_id	int	Идентификатор ленты багажа	PK
Capacity	int	Максимальный поток	
Chief_operator	int	Идентификатор диспетчера	FK

17) Arrival

Поля	Тип данных	Описание поля	Комментарий
Flight_id	int	Идентификатор рейса	PK, FK
Status	nvarchar(15)	Статус рейса	
Runway_id	int	Идентификатор ВВП	FK
Operator_id	int	Идентификатор диспетчера	FK
Belt_id	int	Идентификатор ленты багажа	FK

2.2 Описание схем отношений

Связь	Главная сущность	Подчиненная сущность	Отношение
Aircraft_ Flight	Aircraft	Flight	Один ко многим
Airline_ Aircraft	Airline	Aircraft	Один ко многим
Airport_ Path	Airport	Path	Один ко многим
Belt_ Arrival	Belt	Arrival	Один ко многим
City_ Airport	City	Airport	Один ко многим
Departure_ Check_in	Departure	Check_in	Один ко многим
Employee_ Manager	Employee	Manager	Один ко многим
Employee_ Operator	Employee	Operator	Один ко многим
Flight_ Arrival	Flight	Arrival	Один ко многим
Flight_ Departure	Flight	Departure	Один ко многим
Manager_ Check_in	Manager	Check_in	Один ко многим
Model_ Aircraft	Model	Aircraft	Один ко многим
Operator_ Arrival	Operator	Arrival	Один ко многим
Operator_ Departure	Operator	Departure	Один ко многим
Operator_ Belt	Operator	Belt	Один ко многим
Operator_ Runway	Operator	Runway	Один ко многим
Passenger_ Check_in	Passenger	Check_in	Один ко многим
Path_ Flight	Path	Flight	Один ко многим
Person_ Employee	Person	Employee	Один ко многим
Person_ Passenger	Person	Passenger	Один ко многим
Runway_ Arrival	Runway	Arrival	Один ко многим
Runway_ Departure	Runway	Departure	Один ко многим

1) Связь Aircraft_ Flight связывает самолёт и рейс. В сущность Flight передаётся идентификатор самолёта. Связь определена отношением “Один ко многим” и является не идентифицирующей;
2) Связь Airline_ Aircraft связывает авиалинию и самолёт. В сущность Aircraft передаётся идентификатор авиалинии. Связь определена отношением “Один ко многим” и является не идентифицирующей;
3) Связь Airport_ Path связывает аэропорт и маршрут. В сущность Path передаётся идентификатор аэропорта. Связь определена отношением “Один ко многим” и является не идентифицирующей;
4) Связь Belt_ Arrival связывает ленту багажа и прибывающий рейс. В сущность Arrival передаётся идентификатор ленты. Связь определена отношением “Один ко многим” и является не идентифицирующей;
5) Связь City_ Airport связывает город и аэропорт. В сущность Airport передаётся идентификатор города. Связь определена отношением “Один ко многим” и является не идентифицирующей;

6) Связь Departure_ Check_in связывает отправляющийся рейс и регистрацию пассажира. В сущность Check_in передаётся идентификатор отправляющегося рейса. Связь определена отношением “Один ко многим” и является идентифицирующей;
7) Связь Employee_ Manager связывает работника и менеджера. В сущность Manager передаётся идентификатор работника. Связь определена отношением “Один ко многим” и является идентифицирующей;
8) Связь Employee_ Operator связывает работника и диспетчера. В сущность Operator передаётся идентификатор работника. Связь определена отношением “Один ко многим” и является идентифицирующей;
9) Связь Flight_ Arrival связывает рейс и прибывающий рейс. В сущность Arrival передаётся идентификатор рейса. Связь определена отношением “Один ко многим” и является идентифицирующей;
10) Связь Flight_ Departure связывает рейс и отправляющийся рейс. В сущность Departure передаётся идентификатор рейса. Связь определена отношением “Один ко многим” и является идентифицирующей;
11) Связь Manager_ Check_in связывает менеджера и регистрацию пассажира. В сущность Check_in передаётся идентификатор менеджера. Связь определена отношением “Один ко многим” и является не идентифицирующей;
12) Связь Model_ Aircraft связывает модель самолета и самолет. В сущность Aircraft передаётся идентификатор самолета. Связь определена отношением “Один ко многим” и является не идентифицирующей;
13) Связь Operator_ Arrival связывает диспетчера и прибывающий рейс. В сущность Arrival передаётся идентификатор диспетчера. Связь определена отношением “Один ко многим” и является не идентифицирующей;
14) Связь Operator_ Departure связывает диспетчера и отправляющийся рейс. В сущность Departure передаётся идентификатор диспетчера. Связь определена отношением “Один ко многим” и является не идентифицирующей;
15) Связь Operator_ Belt связывает диспетчера и ленту багажа. В сущность Belt передаётся идентификатор диспетчера. Связь определена отношением “Один ко многим” и является не идентифицирующей;
16) Связь Operator_ Runway связывает диспетчера и ВВП. В сущность Runway передаётся идентификатор диспетчера. Связь определена отношением “Один ко многим” и является не идентифицирующей;

17) Связь Passenger_ Check_in связывает пассажира и регистрацию пассажира. В сущность Check_in передаётся идентификатор пассажира. Связь определена отношением “Один ко многим” и является идентифицирующей;
18) Связь Path_ Flight связывает маршрут и рейс. В сущность Flight передаётся идентификатор маршрута. Связь определена отношением “Один ко многим” и является не идентифицирующей;
19) Связь Person_ Employee связывает человека и работника. В сущность Employee передаётся идентификатор человека. Связь определена отношением “Один ко многим” и является идентифицирующей;
20) Связь Person_ Passenger связывает человека и пассажира. В сущность Passenger передаётся идентификатор человека. Связь определена отношением “Один ко многим” и является идентифицирующей;
21) Связь Runway_ Arrival связывает ВВП и прибывающий рейс. В сущность Arrival передаётся идентификатор ВВП. Связь определена отношением “Один ко многим” и является не идентифицирующей;
22) Связь Runway_ Departure связывает ВВП и отправляющийся рейс. В сущность Departure передаётся идентификатор ВВП. Связь определена отношением “Один ко многим” и является не идентифицирующей;

2.3 Инфологическая модель предметной области

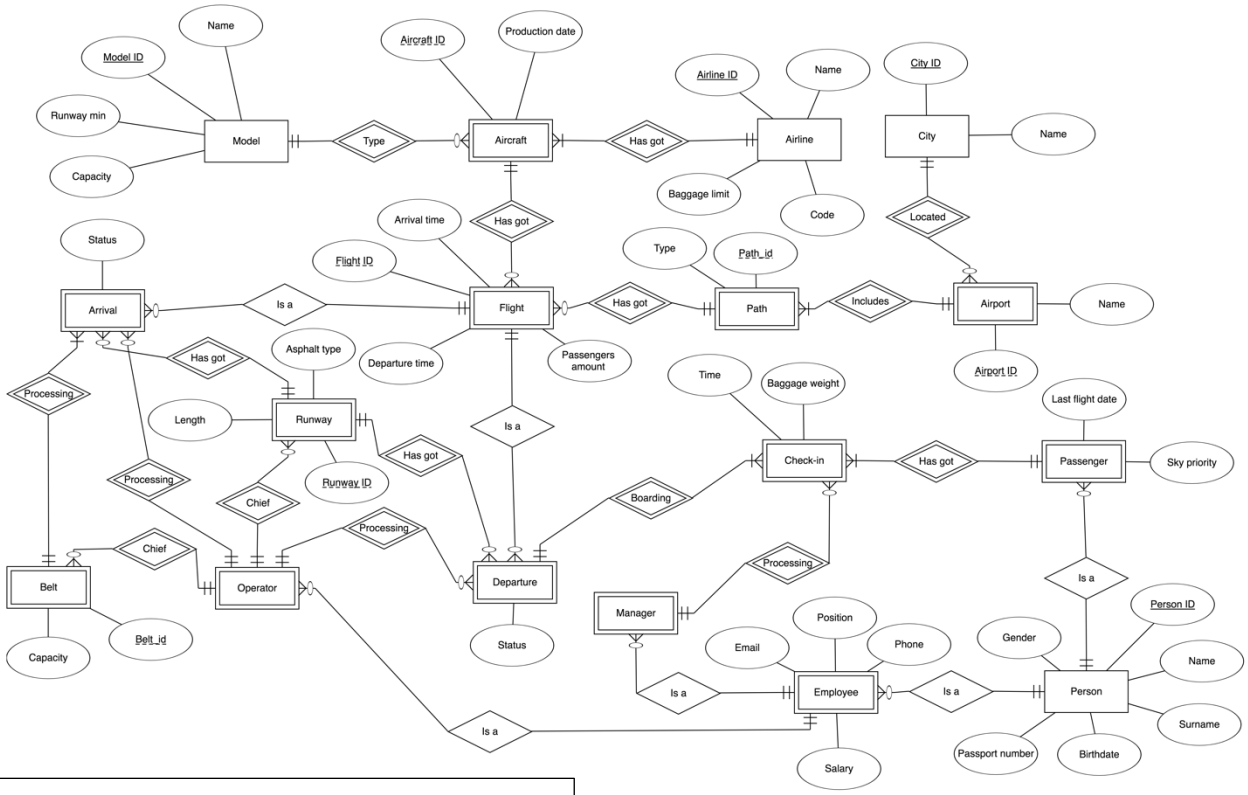

Рисунок 1. Инфологическая модель предметной

2.4 Даталогическая модель данных

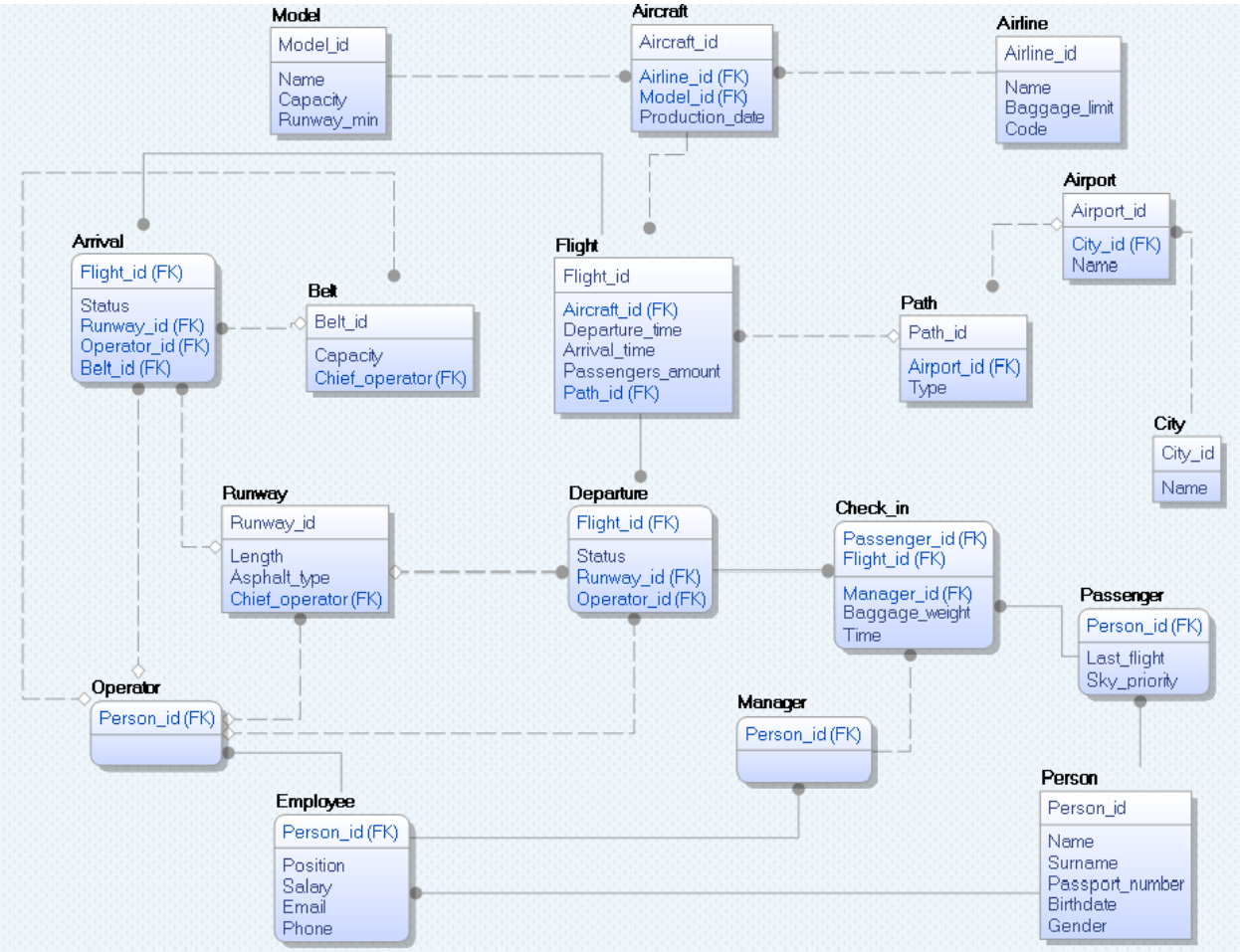

Рисунок 2. Даталогическая модель данных

3. Реализация базы данных

3.1 Реализация базы данных в формате MS SQL

База данных реализована в программе Microsoft SQL Server Management Studio 17. Данная программа позволяет выполнить все поставленные задачи.

3.2 Разработка запросов

- Запрос, выводящий табло вылетов

```
GO
SELECT CAST(f.Departure_time AS nvarchar(20)) AS [Departure], air.[Name],
air.Code + ' ' + f.Path_id AS [Flight],
c.[Name] + '(' + airp.[Name] + ')' AS [To]
FROM Flight f
JOIN Aircraft a ON a.Aircraft_id = f.Aircraft_id
JOIN Airline air ON air.Airline_id = a.Airline_id
```

```
JOIN [Path] p ON p.Path_id = f.Path_id
JOIN Airport airp ON airp.Airport_id = p.Airport_id
JOIN City c ON c.City_id = airp.City_id
WHERE p.[Type] = 'D'
ORDER BY f.Departure_time
```

Результат запроса:

	Departure	Name	Flight	To
1	дек 15 2018 4:15AM	S7	S7 7294	Nice(Nice-fly)
2	дек 15 2018 9:11AM	Aeroflot	SU 5319	London(Heathrow)
3	дек 15 2018 10:43AM	Air France	AF 2730	Bordeaux(Fine-Wine)
4	дек 15 2018 12:15PM	Air France	AF 1332	London(Gatwick)
5	дек 15 2018 12:35PM	British Airways	BA 8550	Kazan(Useinov)
6	дек 15 2018 1:23PM	British Airways	BA 7294	Nice(Nice-fly)
7	дек 15 2018 2:12PM	British Airways	BA 2730	Bordeaux(Fine-Wine)
8	дек 15 2018 3:11PM	S7	S7 5478	Marseille(Marseille-Provans)
9	дек 16 2018 1:14AM	Air France	AF 2874	Paris(Napoleon)
10	дек 16 2018 5:44AM	British Airways	BA 8550	Kazan(Useinov)
11	дек 16 2018 7:21AM	British Airways	BA 7294	Nice(Nice-fly)
12	дек 16 2018 8:52AM	Air France	AF 6338	Sochi(Sochi-City)

Рисунок 3. Результат запроса

- Запрос, выводящий табло прилётов

```
GO
SELECT CAST(f.Arrival_time AS nvarchar(20)) AS Arrival, air.[Name], air.Code
+ ' ' + f.Path_id AS [Flight],
c.[Name] + '(' + airp.[Name] + ')' AS [From]
FROM Flight f
JOIN Aircraft a ON a.Aircraft_id = f.Aircraft_id
JOIN Airline air ON air.Airline_id = a.Airline_id
JOIN [Path] p ON p.Path_id = f.Path_id
JOIN Airport airp ON airp.Airport_id = p.Airport_id
JOIN City c ON c.City_id = airp.City_id
WHERE p.[Type] = 'A'
ORDER BY f.Arrival_time
```

Результат запроса:

	Arrival	Name	Flight	From
1	дек 15 2018 11:15AM	Aeroflot	SU 5620	London(London-City)
2	дек 15 2018 11:51AM	British Airways	BA 8376	Manchester(Rooney)
3	дек 15 2018 1:24PM	Air France	AF 2231	Paris(Napoleon)
4	дек 15 2018 3:24PM	S7	S7 5793	Marseille(Marseille-Provans)
5	дек 15 2018 4:16PM	Air France	AF 7099	Barcelona(Lionel Messi)
6	дек 15 2018 9:15PM	Aeroflot	SU 7099	Barcelona(Lionel Messi)
7	дек 16 2018 8:16AM	British Airways	BA 8376	Manchester(Rooney)
8	дек 16 2018 8:34AM	S7	S7 5373	Kazan(Useinov)
9	дек 16 2018 9:42AM	British Airways	BA 2231	Paris(Napoleon)
10	дек 16 2018 10:44AM	S7	S7 5373	Kazan(Useinov)
11	дек 16 2018 11:15AM	Aeroflot	SU 5620	London(London-City)
12	дек 16 2018 7:14PM	S7	S7 8376	Manchester(Rooney)

Рисунок 4. Результат запроса

- Запрос, выводящий кол-во прилётов/вылетов каждой модели самолёта

```
GO
SELECT model.[Name], SUM([count]) [Visiting amount]
FROM (SELECT Aircraft_id, Count(Aircraft_id) [count]
      FROM Flight
      Group By Aircraft_id) plane
JOIN Aircraft air
ON air.Aircraft_id = plane.Aircraft_id
JOIN Model model
ON model.Model_id = air.Model_id
GROUP BY model.[Name]
ORDER BY [Visiting amount] DESC
```

Результат запроса:

	Name	Visiting amount
1	Airbus A320	16
2	Boeing 777	12
3	Airbus A319	8
4	Tu-154	6
5	Airbus A380	6
6	Boeing 737	6

Рисунок 5. Результат запроса

- Запрос, выводящий самые популярные направления перелётов

```
GO
SELECT city.[Name], COUNT(city.City_id) [Amount]
FROM Flight flight
JOIN [Path] p ON p.Path_id = flight.Path_id
JOIN Airport airport
ON p.Airport_id = airport.Airport_id
JOIN City city
ON city.City_id = airport.City_id
GROUP BY city.[Name]
ORDER BY Amount DESC
```

Результат запроса:

	Name	Amount
1	London	10
2	Paris	9
3	Kazan	7
4	Manchester	6
5	Nice	6
6	Marseille	4
7	Sochi	3
8	Barcelona	3
9	Bordeaux	2
10	Liverpool	2
11	Bristol	1
12	Vladivostok	1

Рисунок 6. Результат запроса

- Запрос, выводящий суммарное количество перевезённых пассажиров самолётами

```
GO
SELECT m.[Name], SUM(f.Passangers_amount) AS [Total amount of passengers]
FROM Flight f
JOIN Aircraft a ON a.Aircraft_id = f.Aircraft_id
JOIN Model m ON m.Model_id = a.Model_id
GROUP BY m.[Name]
ORDER BY [Total amount of passengers] DESC
```

Результат запроса:

	Name	Total amount of passengers
1	Boeing 777	2564
2	Airbus A320	2506
3	Airbus A380	1583
4	Tu-154	1020
5	Boeing 737	868
6	Airbus A319	734

Рисунок 7. Результат запроса

- Запрос, выводящий статистическую информацию по каждой авиакомпании

```
GO
SELECT air.[Name], air.Code, COUNT(*) AS [Flight amount],
SUM(f.Passangers_amount) AS [Total passengers amount],
AVG(f.Passangers_amount) AS [Average passengers amount]
FROM Flight f
JOIN Aircraft a ON a.Aircraft_id = f.Aircraft_id
JOIN Airline air ON a.Airline_id = air.Airline_id
GROUP BY air.Name, air.Code
```

Результат запроса:

	Name	Code	Flight amount	Total passengers amount	Average passengers amount
1	Air France	AF	16	3096	193
2	British Airways	BA	16	2424	151
3	S7	S7	12	1976	164
4	Aeroflot	SU	10	1779	177

Рисунок 8. Результат запроса

3.3 Разработка триггеров

- Триггер, который проверяет входящие данные в Path

```
GO
--Триггер проверки входных данных в Path
```

```
CREATE TRIGGER Path_check_airport ON dbo.[path]
INSTEAD OF INSERT
AS
DECLARE @Airport_id int, @Type nchar(1), @Path_id nvarchar(4)
SELECT @Airport_id = Airport_id, @Type = [Type] FROM inserted
BEGIN
	IF NOT EXISTS(SELECT * FROM Airport WHERE Airport_id =
@Airport_id) OR (@Type != 'A' AND @Type != 'D' AND @Type != 'a' AND
@Type != 'd' )
	BEGIN
		Print('VALUE ERROR')
	END
	ELSE
	BEGIN
		WHILE 1=1
		BEGIN
			SET @Path_id = CAST(LEFT(RAND() * 100000, 4)  AS
nvarchar(4))
			IF NOT EXISTS(SELECT * FROM [Path] WHERE Path_id =
@Path_id)
			BEGIN
				BREAK
			END
		END
		IF @Type = 'A' Or @Type = 'a'
		BEGIN
			SET @Type = 'A'
		END
		ELSE
		BEGIN
			SET @Type = 'D'
		END
		INSERT INTO [Path](Path_id, Airport_id, [Type])
		VALUES (@Path_id, @Airport_id, @Type)
	END
END
```

Результат работы триггера:

```
INSERT INTO [Path](Airport_id, [Type])
VALUES (1, 'E')
```

Сообщения

```
VALUE ERROR
```

Рисунок 9. Результат работы триггера

- Триггер, который проверяет входящие данные в Flight

```
GO
--Триггер проверки входных данных в Flight
CREATE TRIGGER Value_check_flight ON dbo.flight
INSTEAD OF INSERT
AS
DECLARE @Flight_id int, @Path_id nvarchar(7), @Departure_time datetime,
@Arrival_time datetime, @Passangers_amount int, @Aircraft_id int
SELECT @Flight_id = Flight_id, @Path_id = [Path_id], @Departure_time =
Departure_time, @Arrival_time = Arrival_time,
@Passangers_amount = Passangers_amount, @Aircraft_id = Aircraft_id  FROM
inserted
BEGIN
	IF @Flight_id <=0 OR @Passangers_amount <= 0 OR NOT
EXISTS(SELECT * FROM [Path] WHERE Path_id = @Path_id) OR
@Departure_time >= @Arrival_time
	BEGIN
		Print('VALUE ERROR')
	END
	ELSE
	BEGIN
		DECLARE @Amount int
		SELECT @Amount = m.Capacity
		FROM Aircraft a
		JOIN Model m
		ON a.Model_id = m.Model_id
		WHERE a.Aircraft_id = @Aircraft_id
		IF @Passangers_amount > @Amount
		BEGIN
			Print('Passangers amount error! Check the model of the
Aircraft.')
		END
		ELSE
		BEGIN
			INSERT INTO Flight(Flight_id, Path_id,
Departure_time, Arrival_time, Passangers_amount, Aircraft_id)
			VALUES (@Flight_id, @Path_id, @Departure_time,
@Arrival_time, @Passangers_amount, @Aircraft_id)
```

```
                    IF (SELECT [Type] FROM [Path] WHERE Path_id =
@Path_id) = 'A'
                    BEGIN
                        INSERT INTO Arrival(Flight_id, [Status])
                        VALUES (@Flight_id, 'processing')
                    END
                    ELSE
                    BEGIN
                        INSERT INTO Departure(Flight_id, [Status])
                        VALUES (@Flight_id, 'processing')
                    END
                END
        END
END
```

Результат работы триггера:

```
INSERT INTO Flight(Flight_id, Departure_time, Arrival_time, Passangers_amount, Aircraft_id, Path_id)
VALUES (100, '19.12.2018 7:00', '16.12.2018 17:00', 100, 1, 8376)
```

```
%
Сообщения
VALUE ERROR
```

Рисунок 10. Результат работы триггера

- Триггер, проверяющий входящие данные в Person

```
GO
--Триггер проверки входных данных в Person
CREATE TRIGGER Value_check_person ON dbo.Person
INSTEAD OF INSERT
AS
DECLARE @Person_id int, @Name nvarchar(20), @Surname nvarchar(20),
@Passport_number int, @Birthdate datetime, @Gender nvarchar(1)
SELECT @Person_id = Person_id, @Name = [Name], @Surname = Surname,
 @Passport_number = Passport_number, @Birthdate = Birthdate, @Gender =
Gender FROM inserted
BEGIN
        IF @Person_id <=0 OR LEN(@Name) <= 1 OR LEN(@Surname) <= 1 OR
@Passport_number < 1000000 OR @Passport_number > 9999999 OR @Birthdate
> GETDATE()
        OR (@Gender != 'F' AND @Gender != 'f' AND @Gender != 'M' AND
@Gender != 'm')
        BEGIN
            Print('VALUE ERROR')
```

```
        END
        ELSE
        BEGIN
                IF EXISTS (SELECT * FROM Person p WHERE p.Passport_number
= @Passport_number)
                BEGIN
                        PRINT('Passport number ''' + CAST(@Passport_number AS
nvarchar(MAX)) + ''' already exists.')
                END
                ELSE
                BEGIN
                        INSERT INTO Person(Person_id, [Name], Surname,
Passport_number, Birthdate, Gender)
                        VALUES (@Person_id, @Name ,@Surname
,@Passport_number, @Birthdate, @Gender)
                END
        END
END
```

Результат работы триггера:

```
INSERT INTO Person(Person_id, [Name], Surname, Passport_number, Birthdate, Gender)
VALUES (100, 'Ivan', 'Ivanov', 4665775, '10.10.1999', 'M')
```

```
%
Сообщения
Passport number '4665775' already exists.
```

Рисунок 11. Результат работы триггера

- Триггер, проверяющий входящие данные в Check-in

```
GO
--Триггер проверки входных данных в Check-in
CREATE TRIGGER Value_check_checkin ON dbo.Check_in
INSTEAD OF INSERT
AS
DECLARE @Flight_id int, @Manager_id int, @Baggage_weight float, @Time
datetime, @Passenger_id int
SELECT @Flight_id = Flight_id, @Manager_id = Manager_id,
@Baggage_weight = Baggage_weight, @Passenger_id = Passenger_id FROM
inserted
SET @Time = GETDATE()
BEGIN
```

```
	IF @Flight_id <=0 OR @Flight_id <=0 OR @Manager_id <= 0 OR
@Baggage_weight < 0 OR NOT EXISTS(SELECT * FROM Passenger WHERE
Person_id = @Passenger_id)
	BEGIN
		Print('VALUE ERROR')
	END
	ELSE
	BEGIN
		IF (SELECT m.Capacity FROM Flight f JOIN Aircraft a ON
a.Aircraft_id = f.Aircraft_id
		JOIN Model m ON m.Model_id = a.Model_id WHERE Flight_id =
@Flight_id) <
		1 + (SELECT Passangers_amount FROM Flight WHERE Flight_id =
@Flight_id)
		BEGIN
			PRINT('Plane is FULL')
		END
		ELSE
		BEGIN
			IF EXISTS(SELECT * FROM Check_in WHERE Passenger_id
= @Passenger_id AND Flight_id = @Flight_id)
			BEGIN
				PRINT('Passenger (id''' + CAST(@Passenger_id AS
nvarchar(MAX)) + ''') is already checked-in. ')
			END
			ELSE
			BEGIN
				DECLARE @Limit float = (SELECT air.Baggage_limit
FROM Flight f JOIN Aircraft a ON f.Aircraft_id = a.Aircraft_id
				JOIN Airline air ON a.Airline_id = air.Airline_id
WHERE f.Flight_id = @Flight_id)

				IF @Baggage_weight > @Limit
				BEGIN
				PRINT('Overweight! Your luggage: ' +
CAST(@Baggage_weight AS nvarchar(MAX)) + ' kg || Limit: ' + CAST(@Limit
AS nvarchar(MAX)) + ' kg')
				END
				ELSE
				BEGIN
					INSERT INTO Check_in(Flight_id,
Baggage_weight, [Time], Manager_id, Passenger_id)
					VALUES (@Flight_id, @Baggage_weight,
@Time, @Manager_id, @Passenger_id )
```

```
                    UPDATE Passenger
                    SET Last_flight = GETDATE()
                    WHERE Person_id = @Passenger_id
                END
            END
        END
    END
END
```

Результат работы триггера:

```
INSERT INTO Check_in(Flight_id, Baggage_weight, [Time], Manager_id, Passenger_id)
VALUES (3, 10, '10.10.2018', 3, 6 )
```

```
%
Сообщения
Passanger (id'6') is already checked-in.
```

Рисунок 12. Результат работы триггера

- Триггер для проверки входящих данных в Departure

```
GO
--Триггер проверки входных данных в Departure
CREATE TRIGGER Value_check_departure ON dbo.departure
INSTEAD OF INSERT
AS
DECLARE @Flight_id int, @Status nvarchar(15), @Operator_id int,
@Runway_id int
SELECT @Flight_id = Flight_id, @Status = [Status] FROM inserted
BEGIN
    IF @Flight_id <=0
    BEGIN
        Print('Your value <= 0')
    END
    ELSE
    BEGIN
        IF @Status != 'processing' OR (SELECT [Type] FROM Flight f JOIN
[Path] p ON f.Path_id = p.Path_id WHERE f.Flight_id = @Flight_id) != 'D'
        BEGIN
            PRINT('Wrong status/type.')
        END
        ELSE
        BEGIN
            INSERT INTO Departure(Flight_id, [Status])
            VALUES (@Flight_id, @Status)
```

```
            END
        END
END
```

Результат работы триггера:

```
INSERT INTO Departure(Flight_id, [Status])
VALUES (5, 'error')
```

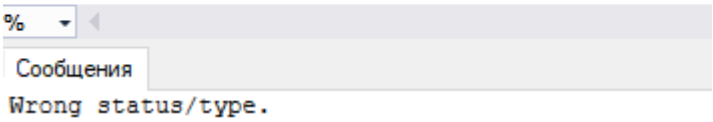

Рисунок 13. Результат работы триггера

- Триггер для проверки входящих данных в Arrival

```
GO
--Триггер проверки входных данных в Arrival
CREATE TRIGGER Value_check_arrival ON dbo.arrival
INSTEAD OF INSERT
AS
DECLARE @Flight_id int, @Status nvarchar(15), @Operator_id int,
@Runway_id int
SELECT @Flight_id = Flight_id, @Status = [Status] FROM inserted
BEGIN
        IF @Flight_id <=0
        BEGIN
                Print('Your value <= 0')
        END
        ELSE
        BEGIN
                IF @Status != 'processing' OR (SELECT [Type] FROM Flight f JOIN
[Path] p ON f.Path_id = p.Path_id WHERE f.Flight_id = @Flight_id) != 'A'
                BEGIN
                        PRINT('Wrong status/type.')
                END
                ELSE
                BEGIN
                        INSERT INTO Arrival(Flight_id, [Status])
                        VALUES (@Flight_id, @Status)
                END
        END
END
```

Результат работы триггера:

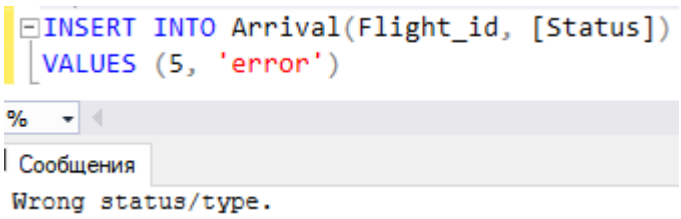

Рисунок 14. Результат работы триггера

- Триггер, для проверки входящих данных в Employee

```
GO
--Триггер проверки входных данных в Employee
CREATE TRIGGER Value_check_employee ON dbo.Employee
INSTEAD OF INSERT
AS
DECLARE @Person_id int, @Position nvarchar(20), @Phone int, @Email
nvarchar(20), @Salary decimal(18, 0)
SELECT @Person_id = Person_id, @Position = Position, @Phone = Phone,
@Email = Email, @Salary = Salary FROM inserted
BEGIN
	IF @Person_id <=0 OR LEN(@Position) <= 2 OR @Person_id <=0 OR
LEN(@Email) <=4 OR @Salary <=0
	BEGIN
		Print('VALUE ERROR')
	END
	ELSE
	BEGIN
		IF EXISTS(SELECT * FROM Employee WHERE Person_id =
@Person_id )
		BEGIN
			PRINT('Person id''' + CAST(@Person_id AS nvarchar(MAX))
+ ''' already exists.')
		END
		ELSE
		BEGIN
			INSERT INTO Employee(Person_id, Position, Phone, Email,
Salary)
			VALUES (@Person_id ,@Position, @Phone , @Email,
@Salary)
		END
	END
END
```

Результат работы триггера:

```
INSERT INTO Employee(Person_id, Salary, Email, Phone, Position)
VALUES (1, 100, 'qwerty@gm.com', 1234567, 'Intern')
```

Рисунок 15. Результат работы триггера

3.4 Разработка хранимых процедур

- Процедура для добавления нового пассажира в базу данных

```
GO
--Процедура для добавления нового пассажира
CREATE PROCEDURE [dbo].Proc_AddPassenger
	@NAME nvarchar(20),
	@SURNAME nvarchar(20),
	@PASSPORT_NUMBER int,
	@BIRTHDATE datetime,
	@GENDER nvarchar(1),
	@SKYPRIORITY bit = NULL
AS
BEGIN

	DECLARE @person_id int =
	(SELECT Person_id FROM Person WHERE [Name] =
	@NAME AND Surname = @SURNAME AND Passport_number =
@PASSPORT_NUMBER AND Birthdate = @BIRTHDATE)

	IF @person_id IS NOT NULL
	BEGIN
		--Есть ли уже такой пассажир?
		IF EXISTS (SELECT * FROM Passenger WHERE Person_id =
@person_id)
		BEGIN
			PRINT('Passanger id: '+ Cast(@PERSON_ID AS
NVARCHAR(MAX)) + '|' + ' ' + @NAME + ' ' + @SURNAME + ' already
exists.')
		END
		ELSE
		BEGIN
			-- это случай, когда работник аэропорта куда-то летит
в первый раз
```

```
                    INSERT INTO Passenger(Person_id, Sky_priority)
                    VALUES(@person_id, @SKYPRIORITY)
                END
        END
        ELSE
        BEGIN
            SET @person_id = (SELECT MAX(Person_id) FROM Person) + 1
            INSERT INTO Person(Person_id, [Name], Surname,
Passport_number, Birthdate, Gender)
            VALUES(@person_id, @NAME, @SURNAME,
@PASSPORT_NUMBER, @BIRTHDATE, @GENDER)
            --Если триггер не выдал исключение
            IF EXISTS (SELECT * FROM Person WHERE Person_id =
@person_id)
            BEGIN
                INSERT INTO Passenger(Person_id, Sky_priority)
                VALUES (@person_id, @SKYPRIORITY)
            END
        END
END
```

- Процедура для добавления нового сотрудника в базу данных

```
GO
--Процедура для добавления нового Сотрудника
CREATE PROCEDURE [dbo].Proc_AddEmployee
        @NAME nvarchar(20),
        @SURNAME nvarchar(20),
        @PASSPORT_NUMBER int,
        @BIRTHDATE datetime,
        @GENDER nvarchar(1),
        @POSITION nvarchar(20),
        @SALARY decimal(18, 0),
        @EMAIL nvarchar(20),
        @PHONE int,
        @TYPE nchar(1)

AS
BEGIN
        DECLARE @person_id int = (SELECT Person_id FROM Person WHERE
[Name] =
         @NAME AND Surname = @SURNAME AND Passport_number =
@PASSPORT_NUMBER AND Birthdate = @BIRTHDATE)
        IF @TYPE != 'o' and @TYPE != 'O' and @TYPE != 'm' and @TYPE != 'M'
        BEGIN
```

```
            PRINT('Wrong Employee type!')
        END
        ELSE
        BEGIN
            IF @person_id IS NOT NULL
            BEGIN
                -- это случай, когда пассажир становится рабочим
                IF @person_id is NULL
                BEGIN
                    SET @person_id = (SELECT MAX(Person_id) FROM
Person) + 1
                    INSERT INTO Person(Person_id, [Name], Surname,
Passport_number, Birthdate, Gender)
                    VALUES(@person_id, @NAME, @SURNAME,
@PASSPORT_NUMBER, @BIRTHDATE, @GENDER)
                    IF EXISTS (SELECT * FROM Person WHERE
Person_id = @person_id)
                    BEGIN
                        INSERT INTO Employee(Person_id, Position,
Email, Phone, Salary)
                        VALUES(@person_id, @POSITION, @EMAIL,
@PHONE, @SALARY)
                        IF EXISTS(SELECT * FROM Employee WHERE
Person_id = @person_id)
                        BEGIN
                            IF @TYPE = 'O' OR @TYPE = 'o'
                            BEGIN
                                INSERT INTO Operator(Person_id)
                                VALUES (@person_id)
                            END

                            ELSE
                            BEGIN
                                INSERT INTO Manager(Person_id)
                                VALUES (@person_id)
                            END
                        END
                    END

                END
            END
        END
END
```

- Процедура для регистрации пассажира не рейс

```
GO
--Процедура для регистрации пассажира(Основная процедура Менеджера)
CREATE PROCEDURE [dbo].Proc_Check_In
	@FLIGHT_ID int,
	@NAME nvarchar(20),
	@SURNAME nvarchar(20),
	@GENDER nvarchar(1),
	@PASSPORT_NUMBER int,
	@BIRTHDATE datetime,
	@BAGGAGE_WEIGHT float,
	@MANAGER_ID int,
	@SKYPRIORITY bit = NULL

AS
BEGIN
	IF NOT EXISTS (SELECT * FROM Departure WHERE Flight_id =
@FLIGHT_ID)
	BEGIN
		PRINT('Flight number: ' + CAST(@Flight_id AS
NVARCHAR(MAX)) + ' does not depart from this airport!')
	END
	ELSE
	BEGIN
		IF @MANAGER_ID NOT IN (SELECT Person_id FROM Manager)
		BEGIN
		PRINT('This procedure is only available for Managers. ')
		END
		ELSE
		BEGIN

			-- Сначала добавляем пассажира в таблицу Пассажиры
			EXECUTE [dbo].Proc_AddPassenger @NAME,
	@SURNAME, @PASSPORT_NUMBER, @BIRTHDATE, @GENDER,
@SKYPRIORITY
			DECLARE @Person_id int = (SELECT per.Person_id FROM
Passenger pas JOIN Person per ON pas.Person_id = per.Person_id WHERE
[Name] = @NAME AND Surname = @SURNAME AND
			Passport_number = @PASSPORT_NUMBER AND Birthdate
= @BIRTHDATE)
			IF @Person_id IS NOT NULL
			BEGIN
```

```
                    INSERT INTO Check_in(Flight_id, Passenger_id,
Baggage_weight, [Time], Manager_id)
                    VALUES(@FLIGHT_ID, @Person_id,
@BAGGAGE_WEIGHT, GETDATE(), @MANAGER_ID)
                END
            END
        END
END
```

- Процедура для назначения ВВП самолёту

```
GO
--Основная процедура Оператора. Назначение полосы/изменение статуса
CREATE PROCEDURE [dbo].Proc_Runway
        @FLIGHT_ID int,
        @OPERATOR_ID int,
        @STATUS nvarchar(15),
        @RUNWAY_ID int

AS
BEGIN
        IF @STATUS != 'processing' AND @STATUS != 'processed' AND
@STATUS != 'took off' AND @STATUS != 'landed' AND  @STATUS !=
'delayed'
        OR NOT EXISTS(SELECT * FROM Flight WHERE Flight_id =
@FLIGHT_ID)
        BEGIN
            PRINT('VALUE ERROR')
        END
        ELSE
        BEGIN

            IF @OPERATOR_ID NOT IN (SELECT Person_id FROM Operator)
            BEGIN
                PRINT('This procedure is only available for Operators.')
            END
            ELSE
            BEGIN
                DECLARE @length float = (SELECT M.Runway_min FROM
Flight F JOIN Aircraft A ON F.Aircraft_id = A.Aircraft_id
                JOIN Model M ON M.Model_id = A.Model_id WHERE
F.Flight_id = @FLIGHT_ID)
```

```
                DECLARE @runway float = (SELECT [Length] FROM
Runway WHERE Runway_id = @RUNWAY_ID)

                IF @runway < @length
                BEGIN
                        PRINT('Wrong runway! Minimum length for this
aircraft: '+ cast(@length as nvarchar(max)))
                END
                ELSE
                BEGIN
                    IF @FLIGHT_ID IN (SELECT Flight_id FROM Arrival)
                    BEGIN
                        UPDATE Arrival
                        SET [Status] = @STATUS, Operator_id =
@OPERATOR_ID, Runway_id = @RUNWAY_ID
                        WHERE Flight_id = @FLIGHT_ID
                    END
                    ELSE
                    BEGIN
                        UPDATE Departure
                        SET [Status] = @STATUS, Operator_id =
@OPERATOR_ID, Runway_id = @RUNWAY_ID
                        WHERE Flight_id = @FLIGHT_ID
                    END
                END
            END
        END
END
```

- Процедура для назначения багажной ленты самолёту

```
GO
--Основная процедура Оператора. Назначение полосы/изменение статуса
CREATE PROCEDURE [dbo].Proc_Belt
    @FLIGHT_ID int,
    @OPERATOR_ID int,
    @STATUS nvarchar(15),
    @BELT_ID int

AS
BEGIN
    IF @STATUS != 'processing' AND @STATUS != 'processed' AND
@STATUS != 'took off' AND @STATUS != 'landed' AND  @STATUS !=
'delayed'
```

```
        OR NOT EXISTS(SELECT * FROM Arrival WHERE Flight_id =
@FLIGHT_ID)
        BEGIN
                PRINT('VALUE ERROR')
        END
        ELSE
        BEGIN

                IF @OPERATOR_ID NOT IN (SELECT Person_id FROM Operator)
                BEGIN
                        PRINT('This procedure is only available for Operators.')
                END
                ELSE
                BEGIN
                        DECLARE @capacity int = (SELECT Passangers_amount
FROM Flight WHERE Flight_id = @FLIGHT_ID  )

                        DECLARE @belt_capacity int = (SELECT Capacity FROM
Belt WHERE Belt_id = @BELT_ID)

                        IF @belt_capacity < @capacity
                        BEGIN
                                        PRINT('Wrong belt! Minimum belt capacity for
this aircraft: '+ cast(@capacity as nvarchar(max)))
                        END
                        ELSE
                        BEGIN
                                UPDATE Arrival
                                SET [Status] = @STATUS, Operator_id =
@OPERATOR_ID, Belt_id = @BELT_ID
                                WHERE Flight_id = @FLIGHT_ID
                        END
                END
        END
END
```

- Процедура для выдачи детальной информации по вылетам/прилётам в конкретный день

```
 GO
CREATE PROCEDURE [dbo].Proc_Table
        @DATE date,
        @TYPE nchar(1)
```

```
AS
BEGIN
	IF NOT EXISTS(SELECT * FROM Flight WHERE cast(Arrival_time as
date) = @DATE OR cast(Departure_time as date) = @DATE)
	BEGIN
		PRINT('Wrong date.')
	END
	ELSE
	BEGIN
		IF @TYPE != 'a' AND @TYPE != 'A' AND @TYPE != 'd' AND
@TYPE != 'D'
		BEGIN
			PRINT('Wrong type.')
		END
		ELSE
		BEGIN
			IF @TYPE = 'a' OR @TYPE = 'A'
			BEGIN
				SELECT CAST(CAST(flight.Arrival_time as time) as
nvarchar(5))  as [Time], airline.[Name] as [Airline],
				airline.Code + ' ' + p.Path_id as [Flight number],
city.[Name], airport.[Name],
				model.[Name] as [Aircraft], arrival.Belt_id AS [Belt],
arrival.Runway_id AS [Runway],  arrival.[Status], arrival.Operator_id AS [Last
change by]
				FROM Arrival arrival
				JOIN Flight flight ON arrival.Flight_id = flight.Flight_id
				JOIN Path p ON p.Path_id = flight.Path_id
				JOIN Aircraft aircraft ON flight.Aircraft_id =
aircraft.Aircraft_id
				JOIN Model model ON aircraft.Model_id =
model.Model_id
				JOIN Airline airline ON aircraft.Airline_id =
airline.Airline_id
				JOIN Airport airport ON p.Airport_id =
airport.Airport_id
				JOIN City city ON city.City_id = airport.City_id

				WHERE CAST(flight.Arrival_time as date) = @DATE
				ORDER BY flight.Arrival_time
			END
			ELSE
			BEGIN
```

```
                    SELECT CAST(CAST(flight.Departure_time as time) as
nvarchar(5))  as [Time], airline.[Name] as [Airline],
                    airline.Code + '' + p.Path_id as [Flight number],
city.[Name], airport.[Name],
                    model.[Name] as [Aircraft], departure.Runway_id AS
[Runway], departure.[Status], departure.Operator_id AS [Last change by]
                    FROM Departure departure
                    JOIN Flight flight ON departure.Flight_id =
flight.Flight_id
                    JOIN Aircraft aircraft ON flight.Aircraft_id =
aircraft.Aircraft_id
                    JOIN Path p ON p.Path_id = flight.Path_id
                    JOIN Model model ON aircraft.Model_id =
model.Model_id
                    JOIN Airline airline ON aircraft.Airline_id =
airline.Airline_id
                    JOIN Airport airport ON p.Airport_id =
airport.Airport_id
                    JOIN City city ON city.City_id = airport.City_id
                    WHERE CAST(flight.Departure_time as date) =
@DATE
                    ORDER BY flight.Departure_time
                END
            END
        END
END
```

Результат выполнения процедуры:

```
exec [dbo].Proc_Table '15.12.2018', 'D'
```

121 %

Результаты | Сообщения

	Time	Airline	Flight number	City	Airport	Aircraft	Runway	Status	Last change by
1	04:15	S7	S7 7294	Nice	Nice-fly	Boeing 737	2	delayed	2
2	09:11	Aeroflot	SU 5319	London	Heathrow	Airbus A380	1	took off	1
3	10:43	Air France	AF 2730	Bordeaux	Fine-Wine	Airbus A319	2	delayed	1
4	12:15	Air France	AF 1332	London	Gatwick	Boeing 777	3	delayed	1
5	12:35	British Airways	BA 8550	Kazan	Useinov	Airbus A319	2	took off	2
6	13:23	British Airways	BA 7294	Nice	Nice-fly	Airbus A320	1	processed	2
7	14:12	British Airways	BA 2730	Bordeaux	Fine-Wine	Boeing 777	3	took off	2
8	15:11	S7	S7 5478	Marseille	Marseille-Provans	Boeing 777	1	processed	1

Рисунок 16. Результат выполнения процедуры

4. СПИСОК ЛИТЕРАТУРЫ

1. Диго С.М. Проектирование и использование баз данных - М.: Финансы и статистика, 1995.
2. Грабер М. SQL. Лори, 2003.
3. Базы данных. Проектирование и создание БД. Диго С.М. М.: ЕАОИ, 2008. – 171 с.