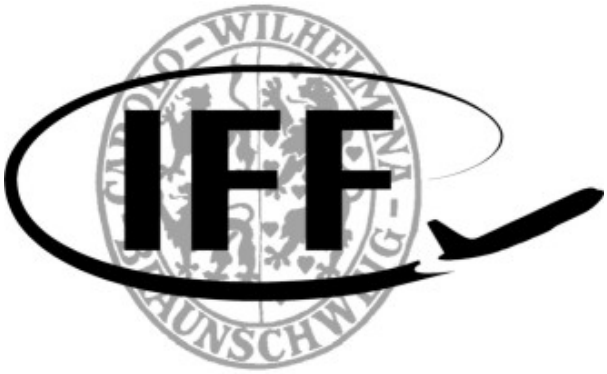
T.U. Braunschweig

Institut für Flugführung



Project: Flight plan conflicts

Student: Nuria Alsina Pujol

Matr.-Nr. 30 22 305

Supervisor: Dipl.-Ing. Eike Rehwald

Winter Semester 2008/09

Index

1. Introduction

The air transport traffic grows up everyday. One could think that in order to handle all this new traffic the only solution is to build new facilities but it's a fact that today's limited resources are not optimized, therefore the airports have less capacity than they could. The problem is not only found in the limited resources but also in the limited airspace the ATC works with and the fact that the companies try to ensure their punctuality.

All in all, every part acts like an individual prioritising their own needs in a way that creates a kind of fight between them to get what they want. The thing is that the only way to have all these needs covered without building new facilities and under valid safety regulations is making a holistic planning system and sharing information between them.

The project CLOU "Co-operative Local resOUrce planner" is the first prototype of Total Operation Planner. It makes a prognosis of the demand and capacity the airport will have and determines when the airport will face a bottleneck situation. Then optimises the planning to minimalise the waiting times and queues. In order to make it all a reallity, there is the need of a constant good comunication with the ATC, Airport and Airlines.

2. Theory

2.1 Flightplans

Every civil airplane that takes off from an airport must have a flight plan accepted before its depature. A flight plan is a written report to an air traffic control facility describing the schedule and expected route of an airplane. Every flight plan consist of the following information:

- Identification:
  - Flight plan code: it is an alphanumeric designator used to identify one flight plan from others. In the databased used, it is named the *key*, to ensure that there are no equal flight plan codes. It is usually formed by the airline name's abbreviation, the departure airport and the date and time of the departure flight.
  - Flight number: it is an alphanumeric code used to identify every flight. It is usually formed by the abbreviation of the airline's name and three or four number.
  - Call sign: it is an alphanumeric name used for radio comunications to identify one aircraft from the other ones.
  - Aircraft code: it is an alphanumeric designatore issued by the ICAO to identify aircrafts.
- Kind: it states whether the flight is an arrival or a departure.
- Airport:
  - Arrival and departure airports: they are codes from the arrival and departure airports. There are two columns to specify each airport, the IATA airport code and the ICAO airport code. The IATA airport's code uses three letters to identify airports worldwide. The ICAO airport code uses 4 letters to identify airports worldwide, the two first letters usually correspond to the country the airport is in.

- Position code: it is the alphanumeric designator of the parking place where the airplane should stay while in the airport.
- Gate: it is the code of the passageway through which the passengers proceed when boarding or leaving this flight.
- Runway code: it is the alphanumeric designator of the runway the plane should take.
- SID code: it is the alphanumeric code of the Standard Instrumental Departure.
- STAR code: it is the alphanumeric code of the Standard Terminal Arrival Route.

- Times: All the times below use the Unixtime which is a system of time mesuring defined as the number of seconds elapsed since midnight of January 1st, 1970 (UTC) without counting the leap seconds.
  - STOT: Scheduled Take-Off Time.
  - SOBT: Scheduled Off-Block Time.
  - ETOT: Estimated Take-Off Time.
  - EOBT: Estimated Off-Block Time.
  - COBT: Calculated Off-Block Time.
  - ATOT: Actual Take-Off Time.
  - AOBT: Actual Off-Block Time.
  - CTOT: Calculated Take-Off Time.
  - SLDT: Scheduled LanDing Time.
  - SIBT: Scheduled In-Block Time.
  - ELDT: Estimated LanDing Time.
  - EIBT: Estimated In-Block Time.
  - ALDT: Actual LanDing Time.
  - AIBT: Actual In-Block Time.
  - SLOT: A space of 15 minutes time in which an aircraft is authorised to departure, if it misses this oportunity it has to wait until it gets another authorised slot, which can be a very long time.
  - Taxitime: it is the time an airplane needs to get from the gate, terminal or ramp to the runway.

- Canceled: It is marked with an Y when the flight has been canceled.
- Source: it describes the data source.
- Ts: it is the timestamp of the last modification of the record.

2.2 Flightplan conflicts

The flightplans issued by the airlines are sometimes lacking information or have errors that may cause problems when they are used by the ATC. Below are described the two major problems this application solves.

2.2.1 Brief turn around time

It is a fact that the cost of an airplane is the minimum when it is flying, therefore the airlines tend to increase the frequency of their flights making the stops at the airports as short as possible. This means that the same airplane must take off shortly after its arrival.

When an airplane lands, it has to get through a lot of procedures, like disembark, fueling, cleaning, mechanical revision, boarding… All this procedures take some time, so there is a minimum time that the airplane must stay at the airport. The problem arises when the arrival flight lands with a certain delay and there is not enough time to get through all the operations and depart at the scheduled time. It is necessary to find this connection between flights and determine whether the departure can be done at the scheduled time or it wil be delayed

The delayed problem is not the only one that can be found when two flights are operated with the same aircraft. There is also the possibility that the arrival flight is canceled. That would mean that the departure flight has no airplane in that airport to take his departure, so that flight must be canceled too.

2.2.2 Airline alliances

In the actual air transport market are there too many airlines operating. That means that competition between airlines is very high and the costs of operating flight increases. In order to reduce costs, operate less airplanes but with more service, and reduce competition among others the airlines make alliances with other airlines. That means that sometimes in the same plane are people flying with different airlines tickets. That in itself would be no problem if the airlines could summit only one flightplan. But as it is a flight from two companies, two flightplans are in order. The Air Traffic Control needs only one flightplan since they will only be controling one flight. Having two flightplan can create problems or cause a waste of valuable ATC time.

3. Project

3.1 Software

The application was written in the language C++ by using the program Microsoft Visual Studio 2008. Because of the need to connect the application to a MySQL database, MySQL server 5.0 was used and the MySQL++ 3.0.6 library was also added to the common libraries.

3.2 Application's outline

To solve the proposed problems, the application follows the path presented in figure 3.1. Every flightplan is compared respectively, until it finds a complete match that solves one of the two problems presented. To differentiate the problem presented in each path, the final operation was marked either in red, blue or green. The operations in red find no match at the end of the comparisons. The operations in blue are the solution of an airline alliance conflict, which indicates that two flightplans from two different companies ended up partnering with each other to operate only one plane. The operations in green are the solution of a brief turn around time conlfict, which indicates that the departure of plane is scheduled less than one hour after its arrival.

In order to solve the brief turn around conflict proposed, the application takes the following steps:

1. Find two flights, first arrival and then departure, from the same company operated with the same airplane.
2. Determine if the arrival flight is canceled and if so, cancel the departure flight.
3. If it is not canceled, calculate the time between the arrival and the departure and determine whether it is less than one hour or not. If

it is longer than one hour the delay doesn't have to affect the departure flight.

4. If it is less than one hour, checks if the arrival flight has any delay and calculates the amount of it.
5. If it has a delay, changes the estimated time of departure to add the delay time.

In order to solve the airline alliances conflict proposed, the application takes the following steps:

1. Finds two flights operated with the same airplane.
2. Checks that both arrival and departure airports are equal.
3. Checks that all the times available of one flightplan are equal to the other flightplan times.
4. When everything is equal, ereases one of the flightplans since they are the same flight.

In both problem solving a short message is written in the *src* column indicating the changes that the flightplans have endured. When a flightplan is canceled the message written is "Canceled arrival, (arrival flightplan code)". When the Estimated Take-Off Time is changed the message written is "New etot, (arrival flightplan code)" . When the Estimated Off-Block Time is changed the message written is "New eobt, (arrival flightplan code)". When a flight is deleted, the message written in the remaining flight is "Deleted: (deleted flightplan code)".

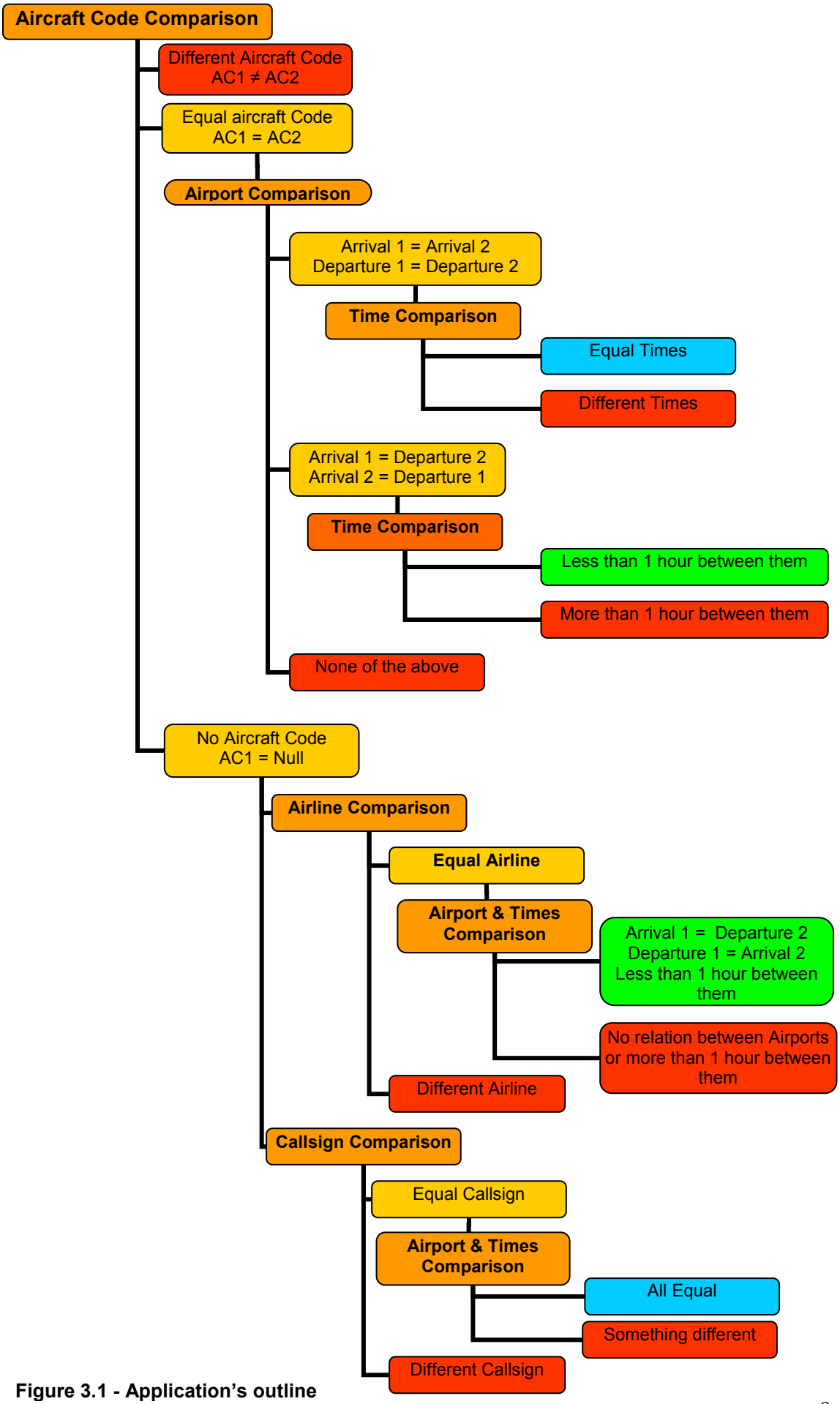


Figure 3.1 - Application's outline

3.3 Classes

The application is structured in two different classes “mysql_explore” and “basic_functions” to make it easier to work with. The basic structure of the application is found in the main, where the different functions are called when needed. Every one of these functions is situated in one class or the other depending of its uses.

3.3.1 Mysql_explore

All the functions that need a direct contact with the MySQL database are found in this class, like retrieving the initial information or writing it down again. Most of these functions are called in some of the functions in the class basic_functions, even though some of them are called directly in main.

There are two private variables in the class, *input* and *output,* which refer to the name of the two tables used, one to get the initial data and the other to write the new data with the problems solved respectively.

3.3.1.1 connect

This function establishes a connection in MySQL using the server, user, database and password provided by the user through the interface. It also asks the name of the initicial table and the new one where the results will be presented and stores it in the private variables *input* and *output.*

3.3.1.2 new_file

This function creates a new table, named as the value in *output*, copying the information in the *input* table. If said table already exists it is dropped. It also changes the datatype of the column named *src* from VARCHAR(10) to VARCHAR (100) because more space is needed later on the program.

3.3.1.3 store

This function saves all data contained in the *input* table in a StoreQueryResult variable so that the application can easily have acces and work with it.

3.3.1.4 erase_flight

This function deletes a flightplan from the *output* table. It is used when the application detects a airline alliance conflict between two flightplans. It also writes a note in the *src* column of the remaining flightplan informing that another one has been deleted, indicating the flightplan code of the erased flight.

3.3.1.5 update_conditions

This function is only used when a brief turn around conflict is detected. It changes the etot or eobt (when there is no stot available) adding to the scheduled time the expected delay. It also writes a note in the *src* column from the flightplan indicating which value was changed and the key of the arrival flightplan, which caused the delay.

3.3.1.6 update_ac_code

This function writes the missing aircraft code in one flightplan when a complete match with aircraft code is found.

3.3.1.7 cancel_flight

This function writes a *Y* in the *canceled* column from a departure flight when the matching arrival flightplan is already canceled. It also writes a comment in the *src* column of the departure flightplan indicating that it has been canceled and writes the key of the arrival flightplan that caused this cancelation.

3.3.2 Basic_functions

Here are all those functions needed for the correct operation of the application. To access the database they use the functions in mysql_explore as intermediary.

3.3.2.1 compare_ac_code

This function determines whether the first flightplan has an aircraft code available and if so, compares it with the other aircraft codes until it finds an equal one or it gets to the end of the table.

3.3.2.2 compare_airports

This function makes two comparisons between the arrival and departure airports of two different flightplans. The first comparison is to prove whether both arrivals and both departures are equal. The second one is to prove whether the arrival airport of the first flightplan is equal to the departure airport of the second flightplan and vice versa. It returns different values acording to the results from the comparisons.

3.3.2.3 compare_times

This function compares the different times (stot, sobt, sldt and sibt) of two different flightplans to see, in case that they are both available in the *input* table, if they are equal.

3.3.2.4 compare_flnr

This function compares the airlines from two flights. In order to do so it compares the first three characters of the flightnumber. This function is used

when a flightplan has no aircraft code available and too determine whether two flights are operated with the same aircraft.

3.3.2.5 compare_callsign

This function compares the callsign of two different flights to see if they are the same flight in case the aircraft code is not available.

3.3.2.6 time_between

This function determines if the time between the Scheduled Take Of Time (stot) of the departure flight and the Scheduled Landing Time (sldt) of the arrival flight is less than one hour. When there is no stot or sldt available it tries again with the estimated times (etot and eldt). It also makes sure that the arrival from the arrival flight is earlier than the departure of the departure flight.

3.3.2.7 conditions

This function determines whether the arrival flightplan has any not scheduled conditions. To begin with it checks if the arrival flight is canceled, if so it automatically cancels the departure flight. When the flight is not canceled, it calculates the delay of the arrival flight. In order to do so it compares the scheduled times with the actual or the estimated times, when the actual ones are not available. When there is a delay it is added to the departure flight. In order to add this delay to the times of the flightplan in the *output* table it uses the *update_conditions* function of *mysql_explore* class.

3.3.2.8 add_ac_code

This function is used when the first flightplan has no aircraft code. First of all it determines whether the second flightplan has an aircraft code. When the answer is positive, this second aircraft code is added to the first flightplan. In

order to do this changes to the *output* table the function *update_ac_code* of *mysql_explore* class is used.

4. Example

4.1 Initial parameters

To prove the efficiency of the application a table[1] with manipulated initial parameters has been made. This example table consists of six different flightplans, the important contenents of which are described in the table below.

Fp_code	UAL945_EDDF_2008-09-19T06:25:00	UAL901_EDDF_2008-09-19T12:00:00	UAL904_KLAX_2008-09-19T00:35:00	UAL905_EDDF_2008-09-19T13:20:00	UAL917_EDDF_2008-09-19T10:20:00	UAL927_EDDF_2008-09-19T15:25:00
Kind	D	D	A	D	A	D
Flnr	UA 945	UA 901	UA 904	UA 905	UA 917	UA 927
Callsign	UAL945	UAL945	UAL904	UAL905	UAL917	UAL927
Ap_dep_code	EDDF	EDDF	KLAX	EDDF	EDDF	KIAD
STOT	-	-	-	-	-	-
SOBT	1221805500	1221805500	-	1221826500	-	-
ETOT	1221806400	1221806400	-	1221825300	-	1221825800
EOBT	1221805500	1221805500	1221784500	1221826500	1221821000	1221824900
ATOT	1221807480	1221807480	-	-	-	-
AOBT	1221806580	1221806580	-	-	-	-
Ap_arr_code	KORD	KORD	EDDF	KLAX	KIAD	KSFO
SLDT	-	-	-	-	-	-
SIBT	-	-	1221823500	-	-	-
ELDT	1221837060	1221837060	1221822300	1221871560	1221822800	1221872060
EIBT	-	-	1221824000	-	-	-
ALDT	-	-	-	-	-	-
AIBT	-	-	-	-	-	-
Ac_code	-	N771UA	-	N206UA	N181UA	N181UA
Canceled	-	-	-	-	Y	-
Src	StanlyCDM	StanlyCDM	StanlyCDM	StanlyCDM	StanlyCDM	StanlyCDM

[1] The backup MySQL file of this example can be found in the CD provided under the name of "example.sql"

4.2 Theorical solution

The table above shows the initial parameters of the example table. Obviously that is not a real table since it is almost impossible to find three pairs of problems in six consecutive flightplans. The table below sumarises the changes that the initial table must suffer. The values in red are the ones the application should erease while the cells in green should change their values.

Fp_code	UAL945_EDDF_2008-09-19T06:25:00	UAL901_EDDF_2008-09-19T12:00:00	UAL904_KLAX_2008-09-19T00:35:00	UAL905_EDDF_2008-09-19T13:20:00	UAL917_EDDF_2008-09-19T10:20:00	UAL927_EDDF_2008-09-19T15:25:00
Kind	D	D	A	D	A	D
FInr	UA 945	UA 901	UA 904	UA 905	UA 917	UA 927
Callsign	UAL945	UAL945	UAL904	UAL905	UAL917	UAL927
Ap_dep_code	EDDF	EDDF	KLAX	EDDF	EDDF	KIAD
STOT	-	-	-	-	-	-
SOBT	1221805500	1221805500	-	1221826500	-	-
ETOT	1221806400	1221806400	-	1221825300	-	1221825800
EOBT	1221805500	1221805500	1221784500	1221826500	1221821000	1221824900
ATOT	1221807480	1221807480	-	-	-	-
AOBT	1221806580	1221806580	-	-	-	-
Ap_arr_code	KORD	KORD	EDDF	KLAX	KIAD	KSFO
SLDT	-	-	-	-	-	-
SIBT	-	-	1221823500	-	-	-
ELDT	1221837060	1221837060	1221822300	1221871560	1221822800	1221872060
EIBT	-	-	1221824000	-	-	-
ALDT	-	-	-	-	-	-
AIBT	-	-	-	-	-	-
Ac_code	-	N771UA	-	N206UA	N181UA	N181UA
Canceled	-	-	-	-	Y	-
Src	StanlyCDM	StanlyCDM	StanlyCDM	StanlyCDM	StanlyCDM	StanlyCDM

The first pair corresponds to a airline alliances conflict. There are two different flightplans, obviously with different flightplan codes, but the other parameters are equal. The aircraft code of the first flightplan has also been ereased, so as the application ereases the second flight, the aircraft code must

be copied from the second flightplan to the first one. Therefore the changes these two flightplans must face are the addition of the aircraft code, *"N771UA"*, the addition of the message *"Deleted: UAL901_EDDF_2008-09-19T12:00:00"* in the *src* column of the first flightplan and the second flightplan must be totally ereased.

The second and third pair both correspond to a brief turn around time conflict but while the first pair has a delayed arrival, the second pair has a canceled arrival. In the first case, the application should add the aircraft code *"N206UA"* to the first flightplan and change the EOBT value from *"1221826500"* to *"1221827000"* (see figure 4.1) and write the short message *"New eobt, UAL904_KLAX_2008-09-19T00:35:00"* in the *src* column of the departure flightplan. In the second case it should ony cancel the departure flightplan and write the message *"Canceled arrival, UAL917_EDDF_2008-09-19T10:20:00"* in the *src* column of the departure flightplan.

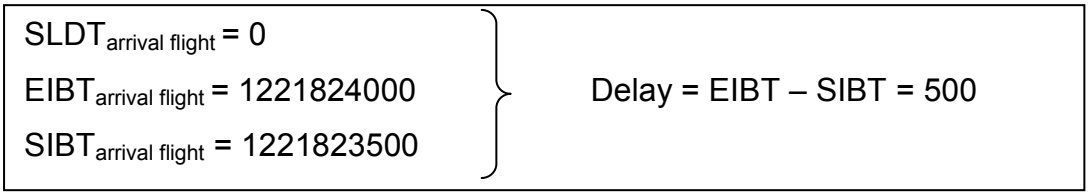


Figure 4.1 – Calculation of the delay

4.3 Solution provided by the application

After configuring the initial table, the application is run and the results obtained and stored in the output table are described below. The cells that have suffered any kind of change are coloured in green.

Fp_code	UAL945_EDDF_2008-09-19T06:25:00	UAL904_KLAX_2008-09-19T00:35:00	UAL905_EDDF_2008-09-19T13:20:00	UAL917_EDDF_2008-09-19T10:20:00	UAL927_EDDF_2008-09-19T15:25:00
Kind	D	A	D	A	D
Flnr	UA 945	UA 904	UA 905	UA 917	UA 927
Callsign	UAL945	UAL904	UAL905	UAL917	UAL927

Ap_dep_ code	EDDF	KLAX	EDDF	EDDF	KIAD
STOT	-	-	-	-	-
SOBT	1221805500	-	1221826500	-	-
ETOT	1221806400	-	1221825300	-	1221825800
EOBT	1221805500	1221784500	1221827000	1221821000	1221824900
ATOT	1221807480	-	-	-	-
AOBT	1221806580	-	-	-	-
Ap_arr_ code	KORD	EDDF	KLAX	KIAD	KSFO
SLDT	-	-	-	-	-
SIBT	-	1221823500	-	-	-
ELDT	1221837060	1221822300	1221871560	1221822800	1221872060
EIBT	-	1221824000	-	-	-
ALDT	-	-	-	-	-
AIBT	-	-	-	-	-
Ac_code	N771UA	N206UA	N206UA	N181UA	N181UA
Canceled	-	-	-	Y	Y
Src	Deleted: UAL901_EDDF_2008-09-19T12:00:00	StanlyCDM	New eobt, UAL904_KLAX_2008-09-19T00:35:00	StanlyCDM	Canceled arrival, UAL917_EDDF_2008-09-19T10:20:00

As can be appreciated in the table above, the application has added the aircraft code to the two flightplans that were missing it. It has also deleted the second flightplan that was in the initial table and changed the EOBT value for the desired one. It has also added the short messages described above in each *src* column.

5. Conclusion

As the air traffic grows, a lot of different needs arise, better planning, optimization of resources and capacity, better demand forecasts… the only way to satisfy all these needs is creating a Total Operation Planner.

Flightplans are essential for air traffic control and management. Therefore a little conflict in a flightplan can cause a great waste of important time and resources. Two of these conflicts are originated by little time for turn around and alliances between airlines. Those situations imply unexpected delays or cancelations and undesired flightplans duplicates.

The application presented with this project detects these conflicts and solves them, creating a new file where the new flightplans are stored. It deletes duplicated flightplans, adds the specified delay to a departure flightplan when the arrival flight was late or cancels the departure if the arrival was previously canceled. It also adds missing aircraft codes when it can find a flightplan that matches the first one. And last, but not least, leaves constance of all the changes in the affected flightplans.

Bibliography

Documents

- CLOU Dokumentation

Internet resources

- Microsoft Developer Network: http://msdn.microsoft.com/
- MySQL: http://mysql.com/