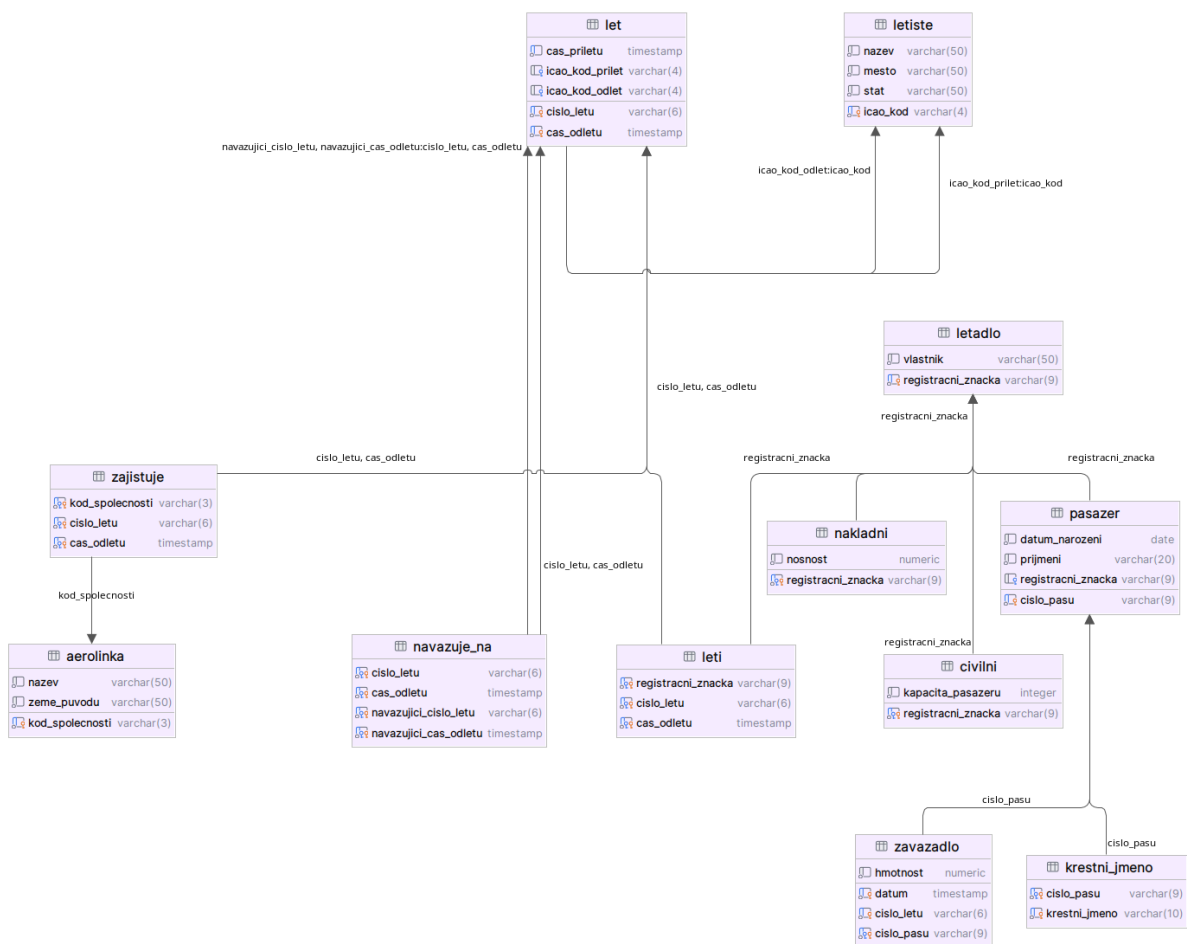


# CP-3 Vytvoření databáze, dotazy na data

Jakub Adamec – adamej14@fel.cvut.cz

## 1. ER model a relační model



Obr.1: Entity-relationship model databáze

- Letiste(ICAO kod, nazev, mesto, stat)
- Let(cislo letu, cas odletu, cislo letu, cas priletu, ICAO\_kod\_prilet, ICAO\_kod\_odlet)
  - FK: (ICAO\_kod\_prilet)  $\subseteq$  Letiste(ICAO\_kod)
  - FK: (ICAO\_kod\_odlet)  $\subseteq$  Letiste(ICAO\_kod)
- Navazuje\_na(cislo letu, cas odletu, navazuje)
  - FK: (cislo\_letu, cas\_odletu)  $\subseteq$  Let(cislo\_letu, cas\_odletu)
  - FK: (navazuje)  $\subseteq$  Let(cislo\_letu, cas\_odletu)
- Aerolinka(kod spolecnosti, nazev, zeme puvodu)
- Zajistuje(kod spolecnosti, cislo letu, cas odletu)
  - FK: (kod\_spolecnosti)  $\subseteq$  Aerolinka(kod\_spolecnosti)
  - FK: (cislo\_letu, cas\_odletu)  $\subseteq$  Let(cislo\_letu, cas\_odletu)
- Letadlo(registracni znacka, vlastnik)
- Leti(registracni znacka, cislo letu, cas odletu)
  - FK: (registracni\_znacka)  $\subseteq$  Letadlo(registracni\_znacka)
  - FK: (cislo\_letu, cas\_odletu)  $\subseteq$  Let(cislo\_letu, cas\_odletu)
- Nakladni(registracni znacka, nosnost)
  - FK: (registracni\_znacka)  $\subseteq$  Letadlo(registracni\_znacka)
- Civilni(registracni znacka, kapacita\_pasazeru)
  - FK: (registracni\_znacka)  $\subseteq$  Letadlo(registracni\_znacka)
- Pasazer(cislo pasu, datum\_narozeni, krestni\_jmeno, prijmeni, registracni\_znacka)
  - FK: (registracni\_znacka)  $\subseteq$  Letadlo(registracni\_znacka)
- Krestni\_jmeno(cislo pasu, krestni jmeno)
  - FK: (cislo\_pasu)  $\subseteq$  Pasazer(cislo\_pasu)
- Zavazadlo(datum, cislo letu, cislo pasu, hmotnost)
  - FK: (cislo\_pasu)  $\subseteq$  Pasazer(cislo\_pasu)

Obr.2: Relační model databáze

## 2. SQL dotazy pro vytvoření databáze

```
DROP TABLE IF EXISTS Zavazadlo;
DROP TABLE IF EXISTS Krestni_jmeno;
DROP TABLE IF EXISTS Pasazer;
DROP TABLE IF EXISTS Civilni;
DROP TABLE IF EXISTS Nakladni;
DROP TABLE IF EXISTS Leti;
DROP TABLE IF EXISTS Letadlo;
DROP TABLE IF EXISTS Zajistuje;
DROP TABLE IF EXISTS Aerolinka;
DROP TABLE IF EXISTS Navazuje_na;
DROP TABLE IF EXISTS Let;
DROP TABLE IF EXISTS Letiste;

CREATE TABLE Letiste (
    ICAO_kod VARCHAR(4) CHECK (ICAO_kod LIKE '____') PRIMARY KEY,
    nazev VARCHAR(50) NOT NULL,
    mesto VARCHAR(50) NOT NULL,
    stat VARCHAR(50) NOT NULL
);

CREATE TABLE Let (
    cislo_letu VARCHAR(6) CHECK (cislo_letu LIKE '____'),
    cas_odletu TIMESTAMP,
    cas_přiletu TIMESTAMP NOT NULL,
    ICAO_kod_přilet VARCHAR(4) REFERENCES Letiste (ICAO_kod) ON UPDATE CASCADE ON DELETE CASCADE,
    ICAO_kod_odlet VARCHAR(4) REFERENCES Letiste (ICAO_kod) ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT let_pk PRIMARY KEY (cislo_letu, cas_odletu),
    CONSTRAINT let_unique UNIQUE (cislo_letu, cas_přiletu)
);

CREATE TABLE Navazuje_na (
    cislo_letu VARCHAR(6) CHECK (cislo_letu LIKE '____'),
    cas_odletu TIMESTAMP,
    navazujici_cislo_letu VARCHAR(6) CHECK (navazujici_cislo_letu LIKE '____'),
    navazujici_cas_odletu TIMESTAMP,
    FOREIGN KEY (cislo_letu, cas_odletu) REFERENCES Let (cislo_letu, cas_odletu) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (navazujici_cislo_letu, navazujici_cas_odletu) REFERENCES Let (cislo_letu, cas_odletu) ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT navazujeNa_pk_let PRIMARY KEY (cislo_letu, cas_odletu, navazujici_cislo_letu, navazujici_cas_odletu)
);

CREATE TABLE Aerolinka (
    kod_spolecnosti VARCHAR(3) CHECK (kod_spolecnosti LIKE '___') PRIMARY KEY,
    nazev VARCHAR(50) NOT NULL,
    zeme_puvodu VARCHAR(50) NOT NULL
);

CREATE TABLE Zajistuje (
    kod_spolecnosti VARCHAR(3) CHECK (kod_spolecnosti LIKE '___'),
    cislo_letu VARCHAR(6) CHECK (cislo_letu LIKE '____'),
    cas_odletu TIMESTAMP,
    CONSTRAINT zajistuje_pk PRIMARY KEY (kod_spolecnosti, cislo_letu, cas_odletu),
    CONSTRAINT kodSpol_fk_zajistuje FOREIGN KEY (kod_spolecnosti) REFERENCES Aerolinka (kod_spolecnosti) ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT let_fk_zajistuje FOREIGN KEY (cislo_letu, cas_odletu) REFERENCES Let (cislo_letu, cas_odletu) ON UPDATE CASCADE ON DELETE CASCADE
);
```

```

CREATE TABLE Letadlo (
    registracni_znacka VARCHAR(9) CHECK (registracni_znacka LIKE '__-____') PRIMARY
    KEY,
    vlastnik VARCHAR(50) NOT NULL
);
CREATE TABLE Leti (
    registracni_znacka VARCHAR(9) CHECK (registracni_znacka LIKE '__-____'),
    cislo_letu VARCHAR(6) CHECK (cislo_letu LIKE '____'),
    cas_odletu TIMESTAMP,
    CONSTRAINT leti_pk PRIMARY KEY (registracni_znacka, cislo_letu, cas_odletu),
    CONSTRAINT letadlo_fk_leti FOREIGN KEY (registracni_znacka) REFERENCES Letadlo
    (registracni_znacka) ON UPDATE CASCADE,
    CONSTRAINT let_fk_leti FOREIGN KEY (cislo_letu, cas_odletu) REFERENCES Let
    (cislo_letu, cas_odletu) ON UPDATE CASCADE
);
CREATE TABLE Nakladni (
    registracni_znacka VARCHAR(9) CHECK (registracni_znacka LIKE '__-____') PRIMARY
    KEY,
    nosnost DECIMAL NOT NULL,
    CONSTRAINT letadlo_fk_nakladni FOREIGN KEY (registracni_znacka) REFERENCES Letadlo
    (registracni_znacka) ON UPDATE CASCADE ON DELETE CASCADE
);
CREATE TABLE Civilni (
    registracni_znacka VARCHAR(9) CHECK (registracni_znacka LIKE '__-____') PRIMARY
    KEY,
    kapacita_pasazeru INT NOT NULL,
    CONSTRAINT letadlo_fk_civilni FOREIGN KEY (registracni_znacka) REFERENCES Letadlo
    (registracni_znacka) ON UPDATE CASCADE ON DELETE CASCADE
);
CREATE TABLE Pasazer (
    cislo_pasu VARCHAR(9) CHECK (cislo_pasu LIKE '____') PRIMARY KEY,
    datum_narozeni DATE NOT NULL,
    prijmeni VARCHAR(20) NOT NULL,
    registracni_znacka VARCHAR(9) CHECK (registracni_znacka LIKE '__-____'),
    CONSTRAINT letadlo_fk_pasazer FOREIGN KEY (registracni_znacka) REFERENCES Letadlo
    (registracni_znacka) ON UPDATE CASCADE ON DELETE CASCADE
);
CREATE TABLE Krestni_jmeno (
    cislo_pasu VARCHAR(9) CHECK (cislo_pasu LIKE '____'),
    krestni_jmeno VARCHAR(10),
    CONSTRAINT krestni_jmeno_pk PRIMARY KEY (cislo_pasu, krestni_jmeno),
    CONSTRAINT pasazer_fk_krestni_jmeno FOREIGN KEY (cislo_pasu) REFERENCES Pasazer
    (cislo_pasu) ON UPDATE CASCADE ON DELETE CASCADE
);
CREATE TABLE Zavazadlo (
    datum TIMESTAMP,
    cislo_letu VARCHAR(6) CHECK (cislo_letu LIKE '____'),
    cislo_pasu VARCHAR(9) CHECK (cislo_pasu LIKE '____'),
    hmotnost DECIMAL NOT NULL,
    CONSTRAINT zavazadlo_pk PRIMARY KEY (datum, cislo_letu, cislo_pasu),
    CONSTRAINT pasazer_fk_zavazadlo FOREIGN KEY (cislo_pasu) REFERENCES Pasazer
    (cislo_pasu) ON UPDATE CASCADE ON DELETE CASCADE
);

```

ON UPDATE a ON DELETE používám v kombinaci prakticky vždy proto, že vždy dává smysl buď aktualizovat „potomka“ anebo že dává smysl ho smazat, aby nezůstávali již nerelevantní data v tabulkách. ON DELETE používám v kontextu toho, že když se například přepraví Pasažér, jeho smysl v databázi zanikne, smaže se, a tedy s ním se musí smazat i jeho křestní jméno(a) a případné zavazadlo. ON UPDATE zase proto, že se například může z nějakého důvodu změnit registrační značka letadla, což se musí propsat i do civilní/nákladní tabulky, dle příslušnosti.

Všechny tabulky jsem vygeneroval pomocí přiloženého skriptu v kapitole 4. Skript vygeneroval CSV soubory, které jsem pak pomocí IntelliJ naimportoval do databáze, a tedy nikde nepoužívám INSERT INTO. Příkladem manuálního vložení řádku do tabulky by bylo například:

```
INSERT INTO Letiste VALUES ('AAAA', 'Vaclav Havel Airport Prague', 'Prague', 'Czech Republic');
INSERT INTO Aerolinka VALUES ('WXH', 'Smart Wings', 'Czech Republic');
INSERT INTO Pasazer VALUES ('P01418606', '1975-06-04', 'Dominguez', 'KM-PH04US');
INSERT INTO Krestni_jmeno VALUES ('P01418606', 'Jose');
INSERT INTO Zavazadlo VALUES ('2024-01-29 11:42:26.116000', 'iL0356', 'P01418606', '40');
```

### 3. SQL dotazy pro získání údajů z databáze

```
SELECT Letiste.nazev, Letiste.mesto, Let.cislo_letu, Let.cas_odletu
FROM Letiste
LEFT OUTER JOIN Let ON Letiste.ICA0_kod = Let.ICA0_kod_odlet;
```

Tento dotaz vrátí údaje o letištích a odpovídajících letech odletu. Využívá vnější spojení (LEFT OUTER JOIN), aby zahrnul všechna letiště a případné odpovídající lety odletu.

	☐ nazev ▼	÷ ☐ mesto ▼	÷ ☐ cislo_letu ▼	÷ ☐ cas_odletu ▼	÷
1	Farmer-Chandler	Johnshire	<null>	<null>	
2	Garcia, Brooks and Walters	Port Angelafort	<null>	<null>	
3	Howell-Miller	East Sarah	<null>	<null>	
4	Greene-Hurst	Joelport	<null>	<null>	
5	Wood LLC	West Walter	<null>	<null>	
6	Stephens, Smith and McCormick	East Jeremy	<null>	<null>	
7	Turner Group	North Kathyville	<null>	<null>	
8	Benitez-Garrison	Brendamouth	<null>	<null>	
9	Wolf, Stuart and Valentine	Matthewshire	<null>	<null>	
10	Miller Group	Ericton	<null>	<null>	
11	Marks, Wright and Turner	North Brendastad	<null>	<null>	
12	Rodriguez LLC	Catherinemouth	<null>	<null>	
13	Parrish-Garcia	Williamtown	<null>	<null>	
14	Brown, Kelly and Guerra	New Jeffreyfort	<null>	<null>	
15	Hoover-Bennett	West Linda	os1350	2024-03-16 04:16:07.172344	
16	Jackson-Baker	New Misty	<null>	<null>	
17	Hernandez, Norton and Smith	Jasminemouth	<null>	<null>	
18	Cabrera-Potter	Valdezborough	<null>	<null>	
19	Dyer PLC	North Michael	<null>	<null>	
20	Rivera-Wyatt	Juliafort	<null>	<null>	
21	Richardson Inc	Reedmouth	<null>	<null>	
22	Walter-Sanchez	Lake Michaeland	<null>	<null>	
23	Hardin-Winters	Melvinborough	<null>	<null>	
24	Lynch, Hayden and Leonard	Katrinaburgh	<null>	<null>	
25	Clark, Collins and Williams	Hendricksland	RZ2574	2024-03-04 00:59:34.929383	

Obr.3: dotaz s vnějším spojením tabulek

```
SELECT Let.cislo_letu, Let.cas_odletu, Letiste.mesto AS odlet_mesto,
Prilet_letiste.mesto AS prilet_mesto
FROM Let
INNER JOIN Letiste ON Let.ICAO_kod_odlet = Letiste.ICAO_kod
INNER JOIN Letiste AS Prilet_letiste ON Let.ICAO_kod_prilet = Prilet_letiste.ICAO_kod;
```

Tento dotaz vrací údaje o letech včetně informací o městě odletu a přiletu. Využívá vnitřní spojení (INNER JOIN) mezi tabulkami Let, Letiste a opět Letiste.

	✕ cislo_letu ▼	÷ cas_odletu ▼	÷ odlet_mesto ▼	÷ prilet_mesto ▼	÷
1	sn4899	2024-03-13 19:10:00.822324	Beverlyville	Johnshire	
2	VY8266	2024-02-01 01:16:24.147909	Johnsonchester	East Jeremy	
3	oM5276	2024-02-25 21:22:44.070276	Lake Charles	North Kathyville	
4	mW1090	2024-04-14 20:45:52.716428	Valerieville	North Andrea	
5	et4575	2024-04-09 15:59:15.136744	Cassandrashire	North Jeremy	
6	ML9515	2024-01-28 11:36:06.264035	New NichoLeview	Lake Chad	
7	aM7075	2024-03-23 23:41:34.489310	Fletcherburgh	Amyfort	
8	yN1600	2024-03-25 20:52:03.833052	Brittanyport	Alejandrofurt	
9	NA9267	2024-02-17 00:26:10.812237	East Markburgh	Ryanport	
10	JL5474	2024-01-05 04:55:12.370671	Hammondton	North James	
11	Mj4232	2024-03-12 16:58:27.338938	West Marcborough	South Robin	
12	r10958	2024-02-12 19:15:35.636603	New Bryan	Kingville	
13	LM1125	2024-03-31 01:55:28.270564	New Pedro	Kathrynport	
14	yM3141	2024-01-05 23:12:37.960143	Melendezton	South Brittneyshire	
15	pu6281	2024-01-20 19:49:02.341143	North Toddland	Sarahton	
16	6L4905	2024-01-03 12:24:29.021790	Jackport	Sarahton	
17	Cv6758	2024-02-03 14:15:31.045140	West Gina	Alexandratown	
18	e19031	2024-02-12 08:09:06.060417	Joshuaburgh	Lake Rodneyville	
19	hd8029	2024-01-01 21:56:12.021348	Doylebury	Michelleshire	
20	Qx1964	2024-01-12 06:56:24.170471	West Patricia	Erinland	
21	iR7392	2024-01-11 06:15:56.190850	West Deborah	New Ralphtown	
22	QK7218	2024-03-10 18:16:11.817312	West Tiffany	South Michael	
23	AR5812	2024-01-28 18:39:38.773480	West Stephenville	Valeriehaven	
24	VF9907	2024-01-09 21:04:17.330866	Elijahshire	Brettview	
25	DD0005	2024-01-17 08:24:27.807127	Lake Bethanyborough	South Jamesshire	

Obr.4: dotaz s vnitřním spojením tabulek

```
SELECT pasazer.cislo_pasu, datum_narozeni, Krestni_jmeno.krestni_jmeno, prijmeni,
registracni_znacka AS registracni_znacka_letadla
FROM Pasazer JOIN Krestni_jmeno USING(cislo_pasu)
WHERE datum_narozeni >= '1990-01-01'
ORDER BY datum_narozeni ASC;
```

Dotaz vybere všechny informace o pasažérech, včetně jejich křestních jmen, kteří se narodili po a v roce 1990. Data jsou seřazena od nejstaršího pasažéra.

	cislo_pasu	datum_narozeni	krestni_jmeno	prijmeni	registracni_znacka_letadla
1	232382370	1990-01-01	Michael	Sharp	LJ-BMTC0H
2	035083726	1990-01-01	Robert	Martinez	WJ-QN8XPC
3	095933441	1990-01-02	Susan	Briggs	QR-ABMHX4
4	341421228	1990-01-03	Kevin	Norris	HZ-6Y57NW
5	066290227	1990-01-05	Christophe	Walker	HH-V3CXIS
6	978233538	1990-01-08	Angela	Roberts	HB-Y31AKW
7	286599440	1990-01-09	Russell	Massey	XG-RPZ3XI
8	M55176230	1990-01-11	Matthew	Richardson	UE-HSU6IU
9	B48151291	1990-01-14	Blake	Spencer	FD-NR000Z
10	270897197	1990-01-15	Ray	Turner	WT-64X3WF
11	E76274367	1990-01-17	Shaun	Kerr	NB-TVQK4M
12	542441687	1990-01-19	Bobby	Brown	ZN-QZAN5M
13	377182386	1990-01-19	Terri	Martin	IB-K2I9S1
14	L16144253	1990-01-21	Robert	Moreno	HR-EU87CQ
15	479990024	1990-01-22	Joseph	Barton	NP-FPB550
16	X43099135	1990-01-23	Juan	Aguilar	JL-2YSTKY
17	A34426857	1990-01-23	Tina	Cameron	ST-92XVIO
18	A34426857	1990-01-23	Brandon	Cameron	ST-92XVIO
19	510083649	1990-01-24	Charles	Mcdaniel	WZ-29S1RE
20	510083649	1990-01-24	Jessica	Mcdaniel	WZ-29S1RE
21	Z79605283	1990-01-25	Sarah	Kent	TD-5AEKPP
22	Z79605283	1990-01-25	Donna	Kent	TD-5AEKPP
23	023848151	1990-01-25	Kayla	Boyer	UN-BF14U6
24	023848151	1990-01-25	Darryl	Boyer	UN-BF14U6
25	M13282016	1990-01-25	Stephen	Vasquez	BK-7288JB

Obr.5: dotaz s podmínkou na data

```
SELECT cislo_letu, round(AVG(hmotnost), 2) AS prumer_hmotnosti
FROM Zavazadlo
GROUP BY cislo_letu
HAVING AVG(hmotnost) > 20;
```

Dotaz spočítá průměrnou váhu zavazadel na jednoho člověka a vypíše všechny lety, kde je průměr více jak 20.

	cislo_letu	prumer_hmotnosti
1	ZE3849	48.63
2	Kj4905	41.17
3	Sx0058	52.22
4	AW7241	52.71
5	ZF8589	41.8
6	YS3791	55.09
7	Tg5345	36.9
8	pf4849	43.54
9	Er1409	59.64
10	Sc2482	55.93
11	Zn2590	51.96
12	Jx6494	48.28
13	ex7703	43.25
14	tx1785	55.61
15	Dx7867	54.96
16	NU0104	53.45
17	rJ2762	47.68
18	RL0859	47.88
19	Kp8523	58.1
20	Py7946	47.67
21	Jr9554	65.95
22	WR4079	60.48
23	pC6051	54.31
24	XI1217	57.81
25	aV4200	45.16

Obr.6: dotaz s agregací a podmínkou v agregační funkci



```
SELECT * FROM Let ORDER BY cas_odletu DESC LIMIT 10 OFFSET 0;
```

Pomocí dotazu se vyberou časově nejvzdálenější lety a zobrazí se pouze prvních 10 výsledků. Způsobem, kterým generuji data ve skriptu jsou ty nejpozdější data v den, kdy jsem databázi generoval. Nejnovější jsou pak začátkem roku, tedy tabulka je z pohledu zaměstnance 1. 1. 2024.

	cislo_letu	cas_odletu	cas_přiletu	icao_kod_přilet	icao_kod_odlet
1	EX7585	2024-04-19 12:48:53.240347	2024-04-19 15:48:53.240347	VCOC	CIVV
2	HU3650	2024-04-19 18:03:41.723800	2024-04-19 18:03:41.723800	JFVD	MUHY
3	ZI6578	2024-04-19 08:47:33.684471	2024-04-19 20:47:33.684471	AYPH	UGYS
4	ag6509	2024-04-19 07:05:40.710800	2024-04-19 17:05:40.710800	NIQB	JTDZ
5	HK9539	2024-04-19 06:09:28.569730	2024-04-19 18:09:28.569730	UXZN	JLJG
6	xE6166	2024-04-19 05:12:26.421500	2024-04-19 16:12:26.421500	CZ6C	EXMH
7	Tb9226	2024-04-19 01:10:09.936312	2024-04-19 12:10:09.936312	DYCY	SVCK
8	Yb7773	2024-04-19 00:47:44.455142	2024-04-19 05:47:44.455142	JCPH	VARU
9	NA5169	2024-04-19 00:35:13.580297	2024-04-19 08:35:13.580297	PDFR	ISBG
10	yI9220	2024-04-18 21:51:56.484752	2024-04-19 08:51:56.484752	BYXR	DXDK

Obr.7: dotaz s řazením a stránkováním

```
SELECT ICAO_kod_odlet AS ICAO_kod
FROM Let
UNION
SELECT ICAO_kod_přilet AS ICAO_kod
FROM Let;
```





Dotaz zjišťuje, která letiště jsou zapojeny do letových operací v databázi, ať už jako místo startu nebo cíle. Zkombinuje dotaz na všechna odletová a příletová letiště s tím, že odstraní duplicitní záznamy.

	icaodkod
1	OVJJ
2	LDZY
3	XJBZ
4	MNAY
5	TURW
6	PGPJ
7	PYPT
8	FALB
9	RNXP
10	FMBK
11	ZSDF
12	PCID
13	MJUN
14	ZNKS
15	QVTW
16	BWBW
17	FISY
18	QCSI
19	KXHA
20	FZVV
21	JPRY
22	MANJ
23	GDTP
24	DTJD
25	BYMS

Obr.8: dotaz s množinovou operací

```
SELECT * FROM Pasazer WHERE registracni_znacka IN (SELECT registracni_znacka FROM Nakladni);
```

Dotaz vybere všechny informace o pasažérech, jejichž letadlo je určeno pro nákladní přepravu (protože jsem neplánovaně dovolil při generaci dat, že civilisti mohou být i v nákladních letadlech :)).

	 cislo_pasu ▾	÷  datum_narozeni ▾	÷  prijmeni ▾	÷  registracni_znacka ▾	÷
1	Q50003521	1911-10-16	Day	CE-Y1I3YS	
2	T68241793	1912-11-30	Ruiz	GS-K73DTV	
3	149158840	1997-11-10	Strickland	AW-K1H4JZ	
4	Y30466727	1940-01-04	Garcia	QH-3ZPHLL	
5	028055523	2005-01-27	Delgado	ZV-T21YKI	
6	977106267	1972-10-16	Fowler	OQ-D0L04G	
7	492911853	1928-11-23	Lopez	JA-00ZWNW	
8	V09800268	1920-02-27	Thomas	OV-SV6R08	
9	M79670423	1986-07-18	Torres	QK-QC6VMW	
10	L77294538	1983-08-23	Harding	LK-9D7N2D	
11	483959092	1928-05-02	Carter	ZV-T21YKI	
12	544231658	1983-11-14	Knight	OQ-D0L04G	
13	B99386318	2006-10-14	Mueller	AH-CRE11C	
14	565475754	1926-06-05	Tran	BB-69Q00E	
15	554582361	2020-07-02	Turner	MC-9IFL6S	
16	093423274	1988-04-07	Fernandez	AW-K1H4JZ	
17	617190204	1930-03-11	Smith	VV-0BF8QY	
18	064303955	1911-09-26	Morgan	EM-0C81J6	
19	D46517796	1949-07-05	Diaz	WE-Z0UU3J	
20	875705925	1981-03-09	Murillo	AQ-9SCHQE	
21	Q15055952	1985-03-30	Johnson	MR-6RVU6D	
22	112414139	1965-03-09	Miller	KJ-H0JEFM	
23	286924135	1914-07-01	Castro	OQ-D0L04G	
24	M91953279	1967-07-16	Ewing	QX-J6S02K	
25	862435214	2005-10-09	Mcguire	WR-RCJPQH	

Obr.9: dotaz s vnořeným SELECT

## 4. Skript pro naplnění databáze

```
#!/usr/env/bin python3
# GNU General Public License v3.0
# knedl1k 2024

import csv
import random
import string
from datetime import timedelta
from faker import Faker

fake=Faker()

generated_ICAO_codes=set()
generated_kodspol_codes=set()
generated_aeroname_codes=set()
generated_registrznacka_codes=set()
taken_registrznacka_codes=set()

generated_cisloletu=list()
generated_pas_codes=set()

odlety={}
priletu={}

"""
HELPER FUNCTIONS
"""

def generate_unique_ICAO_code():
    while True:
        code = ''.join(random.choices(string.ascii_uppercase, k=4))
        if code not in generated_ICAO_codes:
            generated_ICAO_codes.add(code)
            return code

def generate_ICAO_code():
    code = ''.join(random.choices(string.ascii_uppercase, k=4))
    return code

def generate_unique_aerokod_code():
    while True:
        code = ''.join(random.choices(string.ascii_uppercase, k=3))
        if code not in generated_kodspol_codes:
            generated_kodspol_codes.add(code)
            return code

def generate_unique_aeroname_code():
    while True:
        code=fake.company()
        if code not in generated_aeroname_codes:
            generated_aeroname_codes.add(code)
            return code

def generate_registrznacka():
    while True:
        letters=string.ascii_uppercase
        numbers=string.digits
        code=''.join(random.choices(letters, k=2)) + '-' + ''.join(random.choices(letters
+ numbers, k=6))
        if code not in generated_registrznacka_codes:
            generated_registrznacka_codes.add(code)
            return code

def pick_registrznacka(fr):
    while True:
        code=random.choice(tuple(generated_registrznacka_codes))
        if code not in taken_registrznacka_codes:
            taken_registrznacka_codes.add(code)
            return code
```

```

def generate_cisloletu():
    code=fake.bothify(text='??####')
    generated_cisloletu.append(code)
    return code

def generate_pas():
    while True:
        code=fake.passport_number()
        if code not in generated_pas_codes:
            generated_pas_codes.add(code)
            return code

def generate_lety(n):
    for _ in range(n):
        cislo_letu=generate_cisloletu()
        odlet=fake.date_time_this_year(before_now=True, after_now=False)
        odlety[cislo_letu]=odlet
        prilet=prilet=odlet + timedelta(hours=random.randint(1, 12))
        prilety[cislo_letu]=prilet

"""
HEAD FUNCTIONS
"""

def generate_letiste_data(n):
    with open('letiste_data.csv', 'w', newline='') as csvfile:
        fieldnames = ['ICAO_kod', 'nazev', 'mesto', 'stat']
        writer=csv.DictWriter(csvfile, fieldnames=fieldnames)
        #writer.writeheader()
        for _ in range(n):
            writer.writerow({
                'ICAO_kod': generate_unique_ICAO_code(),
                'nazev': fake.company(),
                'mesto': fake.city(),
                'stat': fake.country()
            })

def generate_let_data(n):
    ls=list(generated_cisloletu)
    with open('let_data.csv', 'w', newline='') as csvfile:
        fieldnames = ['cislo_letu', 'cas_odletu', 'cas_priletu', 'ICAO_kod_prilet',
'ICAO_kod_odlet']
        writer=csv.DictWriter(csvfile, fieldnames=fieldnames)
        #writer.writeheader()
        for i in range(n):
            #odlet=fake.date_time_this_year(before_now=True, after_now=False)
            #prilet=odlet + timedelta(hours=random.randint(1, 12))
            writer.writerow({
                'cislo_letu': ls[i],
                'cas_odletu': odlety[ls[i]],
                'cas_priletu': prilety[ls[i]],
                'ICAO_kod_prilet': random.choice(tuple(generated_ICAO_codes)),
                'ICAO_kod_odlet': random.choice(tuple(generated_ICAO_codes))
            })

def generate_aerolinka_data(n):
    with open('aerolinka_data.csv', 'w', newline='') as csvfile:
        fieldnames = ['kod_spolecnosti', 'nazev', 'zeme_puvodu']
        writer=csv.DictWriter(csvfile, fieldnames=fieldnames)
        #writer.writeheader()
        for _ in range(n):
            writer.writerow({
                'kod_spolecnosti': generate_unique_aerokod_code(),
                'nazev': generate_unique_aeroname_code(),
                'zeme_puvodu': fake.country()
            })

```

```

def generate_letadlo_data(n):
    with open('letadlo_data.csv', 'w', newline='') as csvfile:
        fieldnames=['registracni_znacka', 'vlastnik']
        writer=csv.DictWriter(csvfile, fieldnames=fieldnames)
        #writer.writeheader()
        for _ in range(n):
            writer.writerow({
                'registracni_znacka': generate_registrznacka(),
                'vlastnik': fake.company(),
            })

def generate_nakladni_data(n):
    midpoint=len(list(generated_registrznacka_codes)) // 2
    ls=list(generated_registrznacka_codes)[:midpoint]
    with open('nakladni_data.csv', 'w', newline='') as csvfile:
        fieldnames=['registracni_znacka', 'nosnost']
        writer=csv.DictWriter(csvfile, fieldnames=fieldnames)
        #writer.writeheader()
        for _ in range(n):
            writer.writerow({
                'registracni_znacka': pick_registrznacka(ls),
                'nosnost': random.randint(5, 100),
            })

def generate_civilni_data(n):
    midpoint=len(generated_registrznacka_codes) // 2
    ls=list(generated_registrznacka_codes)[midpoint:]
    with open('civilni_data.csv', 'w', newline='') as csvfile:
        fieldnames=['registracni_znacka', 'kapacita_pasazeru']
        writer=csv.DictWriter(csvfile, fieldnames=fieldnames)
        #writer.writeheader()
        for _ in range(n):
            writer.writerow({
                'registracni_znacka': pick_registrznacka(ls),
                'kapacita_pasazeru': random.randint(20, 800),
            })

def generate_pasazer_data(n):
    midpoint=len(list(generated_registrznacka_codes)) // 2
    ls=list(generated_registrznacka_codes)[midpoint:]
    #print("otviram soubor")
    with open('pasazer_data.csv', 'w', newline='') as csvfile:
        fieldnames=['cislo_pasu', 'datum_narozeni', 'prijmeni', 'registracni_znacka']
        writer=csv.DictWriter(csvfile, fieldnames=fieldnames)
        #writer.writeheader()
        for _ in range(n):
            #print("zapisuju")
            writer.writerow({
                'cislo_pasu': generate_pas(),
                'datum_narozeni': fake.passport_dob(),
                'prijmeni': fake.last_name(),
                'registracni_znacka': ls[random.randint(0, len(ls)-1)]
            })

```

```

def generate_krestnijmeno_data():
    ls=list(generated_pas_codes)
    with open('krestnijmeno_data.csv', 'w', newline='') as csvfile:
        fieldnames=['cislo_pasu', 'krestni_jmeno']
        writer=csv.DictWriter(csvfile, fieldnames=fieldnames)
        #writer.writeheader()
        for i in range(len(ls)):
            jmeno=fake.first_name()
            writer.writerow({
                'cislo_pasu': ls[i],
                'krestni_jmeno': jmeno,
            })
            if(random.choices([0, 1], weights=[0.9, 0.1], k=1)[0]):
                jmeno2=fake.first_name()
                while jmeno==jmeno2:
                    jmeno2=fake.first_name()
                writer.writerow({
                    'cislo_pasu': ls[i],
                    'krestni_jmeno': jmeno2,
                })

def generate_zavazadlo_data(n):
    with open('zavazadlo_data.csv', 'w', newline='') as csvfile:
        fieldnames=['datum', 'cislo_letu', 'cislo_pasu', 'hmotnost']
        writer=csv.DictWriter(csvfile, fieldnames=fieldnames)
        #writer.writeheader()
        for _ in range(n):
            writer.writerow({
                'datum': fake.date_time_this_year(before_now=True, after_now=False),
                'cislo_letu': generated_cisloletu[random.randint(0,
len(generated_cisloletu)-1)],
                'cislo_pasu': random.choice(tuple(generated_pas_codes)),
                'hmotnost': random.randint(1, 100),
            })

def generate_leti_data(n):
    ls=list(generated_cisloletu)
    ls2=list(generated_registrznacka_codes)
    with open('leti_data.csv', 'w', newline='') as csvfile:
        fieldnames=['registracni_znacka', 'cislo_letu', 'cas_odletu']
        writer=csv.DictWriter(csvfile, fieldnames=fieldnames)
        #writer.writeheader()
        for i in range(n):
            writer.writerow({
                'registracni_znacka': ls2[i],
                'cislo_letu': ls[i],
                'cas_odletu': odlety[ls[i]]
            })

```

```

taken_navazujeCislo_code=set()
taken_navazujiciCislo_code=set()
def generate_navazujeNa_data(n):
    with open('navazujeNa_data.csv', 'w', newline='') as csvfile:
        fieldnames = ['cislo_letu', 'cas_odletu', 'navazujici_cislo_letu',
'navazujici_cas_odletu']
        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
        #writer.writeheader()

        unique_ICAO_kody_odletu = list(generated_ICAO_codes)

        for _ in range(n):
            cislo_letu = random.choice(generated_cisloletu)
            if(cislo_letu in taken_navazujeCislo_code):
                continue
            taken_navazujeCislo_code.add(cislo_letu)
            ICAO_kod_odlet = odlety[cislo_letu]
            navazujici_lety = [let for let in generated_cisloletu if let !=
cislo_letu and prilety[let] > odlety[cislo_letu] and odlety[let] - prilety[cislo_letu]
< timedelta(days=1)]

            if navazujici_lety:
                navazujici_cislo_letu = random.choice(navazujici_lety)
                if(navazujici_cislo_letu in taken_navazujiciCislo_code):
                    continue
                taken_navazujiciCislo_code.add(navazujici_cislo_letu)
                navazujici_cas_odletu = odlety[navazujici_cislo_letu]

                writer.writerow({
                    'cislo_letu': cislo_letu,
                    'cas_odletu': odlety[cislo_letu],
                    'navazujici_cislo_letu': navazujici_cislo_letu,
                    'navazujici_cas_odletu': navazujici_cas_odletu
                })

    """
def generate_zajistuje_data():
    ls=list(generated_cisloletu)
    ls2=list(generated_kodspol_codes)
    with open('zajistuje_data.csv', 'w', newline='') as csvfile:
        fieldnames=['kod_spolecnosti', 'cislo_letu', 'cas_odletu']
        writer=csv.DictWriter(csvfile, fieldnames=fieldnames)
        #writer.writeheader()
        for i in range(len(ls2)):
            writer.writerow({
                'kod_spolecnosti': ls2[random.randint(0, len(ls2)-1)],
                'cislo_letu': ls[i],
                'cas_odletu': odlety[ls[i]]
            })

```



```

if __name__ == "__main__":
    num_letadlo=1_000
    num_pasazer=40_000
    .....

    generate_letiste_data(10_000)
    generate_lety(1_200) #pregenerate combinations of odlet+cislo & prilet+cislo
    generate_let_data(num_letadlo)

    generate_aerolinka_data(1_000)
    generate_letadlo_data(num_letadlo)
    generate_nakladni_data(int(num_letadlo/2))
    generate_civilni_data(int(num_letadlo/2))
    taken_registrznacka_codes=set()
    generate_pasazer_data(num_pasazer)
    generate_krestnijmeno_data()
    generate_zavazadlo_data(int(num_pasazer//1.5))
    generate_leti_data(num_letadlo)
    generate_zajistuje_data()
    generate_navazujeNa_data(num_letadlo)

```

Kód není příliš hezký ani efektivní, ale funkční. Pro naplnění velkým množstvím dat jsem zvolil tabulku **Pasazer**, ve které je 40 tisíc záznamů. Vedlejším efektem se stalo, že tabulka **Krestni\_jmeno** je ještě větší, protože část populace má více křestních jmen (skript umí generovat pouze lidi s jedním nebo dvěma křestními jmény, ale databáze je schopná pojmout libovolné kladné množství křestních jmen).