

FAKULTA RIADENIA A INFORMATIKY
ŽILINSKÁ UNIVERZITA V ŽILINE

ANALÝZA DÁT DOPRAVNÝCH NEHÔD

SEMESTRÁLNA PRÁCA Z PREDMETU DATABÁZOVÁ ANALYTIKA A VÝKONNOSŤ

Autor: **Bc. Matej Poljak**

Učiteľ: **doc. Ing. Michal Kvet, PhD.**

Akademický rok: **2024/2025**

Obsah

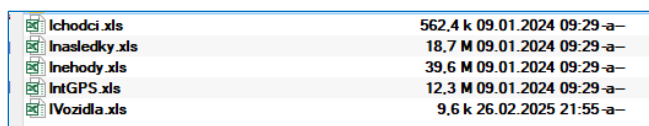
Cieľ semestrálnej práce.....	3
Popis dát	3
Transformácia dát do databázy	4
Databázová analytika	6
Prehľad o počtoch dát.....	6
TOP mesiac/e pre každý kraj podľa počtu umrtí za rok 2024	6
N-tý mesiac s najväčším počtom nehôd	9
Alkohol a dopravné nehody	11
Mesačne najčastejšie zrazené zvieratá v roku 2024	17
2,5 mesačný kĺzavý medián počtu zranených pre rok 2024 (s frekvenciou polmesiac)	17
Optimalizácia výkonnosti	20
Index pre skript získania percentuálneho podielu škôd nehôd s prítomnosťou alkoholu v TOP 1000 najvyššie ocenených škodách	20
Index pre výpis TOP 3 najčastejšie zrazených zvierat za každý mesiac	23
Optimalizácia spojenia tabuliek nehôd a vozidiel.....	25
Záver	27

Cieľ semestrálnej práce

Cieľom práce bolo získať dáta, nad ktorými následne vykonáme analýzu. Touto analýzou zistíme zaujímavé informácie vyplývajúce z dát a tak preukážeme schopnosť analytického pohľadu na ne. Na to sme mali využiť analytické a agregáčné funkcie, ktoré platforma Oracle poskytuje. Okrem toho sme sa mali pokúsiť optimalizovať rýchlosť dopytu dát s využitím indexov.

Popis dát

Na prácu sme si zvolili údaje o dopravných nehodách v Českej republike, ktoré poskytuje česká polícia na mesačnej báze. Dáta môžeme získať zo stránky <https://policie.gov.cz/clanek/statistika-nehodovosti-900835.aspx>, kde si zvolíme rok, z ktorého dáta požadujeme. Stiahneme RAR zložku, ktorá obsahuje súbory s príponou *xls*. Súbory sú rozdelené na záznamy o nehodách, o vozidlách spojených s nehodou, o chodcoch spojených s nehodou a záznamy o následkoch dopravných nehôd (obrázok 1). My si vyberieme pre ďalšie spracovanie dáta o nehodách všeobecne a dáta o vozidlách, čiže dva typy *xls* súborov, pričom použijeme dáta z 3 rôznych rokov: 2023, 2024, 2025.



lchodci.xls	562,4 k	09.01.2024 09:29 -a-
lnasledky.xls	18,7 M	09.01.2024 09:29 -a-
lnehody.xls	39,6 M	09.01.2024 09:29 -a-
lntGPS.xls	12,3 M	09.01.2024 09:29 -a-
lVozidla.xls	9,6 k	26.02.2025 21:55 -a-

Obrázok 1 – obsah stiahnutej RAR zložky

Transformácia dát do databázy

Na stránke, uvedenej v predchádzajúcej kapitole, sme si stiahli aj dokument *Položky_formuláre_WEB.xlsx* (príklad zobrazený na obrázku 2), ktorý obsahuje význam a hodnoty pre jednotlivé stĺpce, ktorými sú naplnené záznamy o nehodách a vozidlách zúčastnených pri nehode. Na základe týchto popisov si pre každý referencujúci stĺpec vytvoríme databázovú tabuľku, ktorá obsahuje významy jednotlivých ID hodnôt, na ktoré sa záznamy o nehodách a vozidlách odkazujú.

Formulár DN	Položka	Popis	Blížší definície
	p4b	okres	okresů. Útvar označuje základní
	p4c	útvár	policejní útvar, v jehož obvodě k nehodě došlo.
05a	p5a	LOKALITA NEHODY	
	1	v obci	
	2	mimo obec	
06	p6	DRUH NEHODY	
	1	srážka	s jedoucím nekoľejovým vozidlom
	2	srážka	s vozidlom zaparkovaným, odstaveným
	3	srážka	s pevnou
	4	srážka	s chodcom
	5	srážka	s lesní zvěřou
	6	srážka	s domácím zvěřatům
	7	srážka	s vlakem
	8	srážka	s tramvají
	9	havarie	
	0	jiný druh nehody	
07	p7	DRUH SRÁŽKY JEDOUČÍCH VOZIDEL	
	1	čelní	
	2	boční	
	3	z boku	

Obrázok 2 – popis hodnôt stĺpcov, na ktoré sa záznamy referencujú

Potom som si vytvoril databázovú tabuľku *cr_nehody* a *cr_vozidla* podľa štruktúry jedného riadku súboru *lnehody.xls*, resp. *lvozidla.xls*. Takýto záznam je ukázaný na obrázku 3.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1	p1	p2a	p2b	p4a	p4b	p4c	p5a	p6	p7	p8	p8a	p9	p10	p11	p11a	p12	p13a	p13b	p13c	p14*100	p15	p16	p17	p18	p19
2	50425000011	11.1.2025	2325	5	4	11	1	3	0	9	0	2	1	9	0	204	0	0	0	55000	2	8	1	1	4
3	50425000001	1.1.2025	1442	5	4	11	2	1	1	0	0	1	1	2	0	501	0	0	3	90000	2	3	1	1	3
4	50425000006	10.1.2025	720	5	4	10	2	1	4	0	0	1	1	2	0	508	0	0	1	65000	2	3	1	1	2

Obrázok 3 – ukážka záznamov o nehodách v xls súbore

Následne som mohol naplniť tabuľky o nehodách a vozidlách hodnotami zo súborov. Najprv som si napísal príkaz, ktorý mi v *xls* súbore vygeneroval sql príkaz *insert*. Všetky vytvorené sql príkazy som si uložil do súboru a potom som spustil vykonanie skriptu v nástroji *SqlDeveloper*. Tento postup je znázornený na obrázku 4.

Databázová analytika

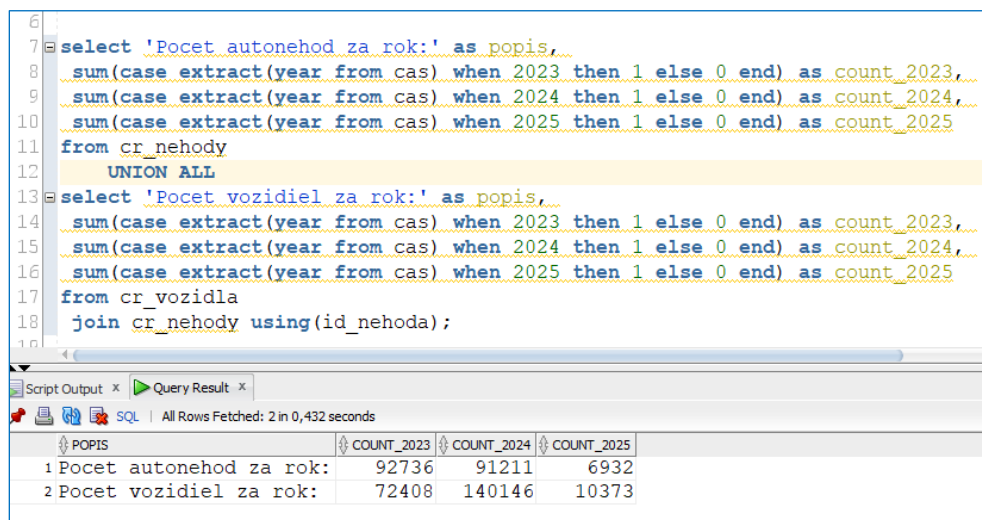
V tejto časti si ukážeme zopár zaujímavých informácií, ktoré sme získali databázovým príkazom *select* spolu s využitím analytických a agregáčnych funkcií. Pred analýzou výkonnosti si najprv nad všetkými tabuľkami, ktoré prislúchajú k schéme „POLJAK“ pregenerujeme štatistiky, pretože sme vložili veľké množstvo nových dát (obrázok 6).

```
1 BEGIN
2     dbms_stats.gather_schema_stats('POLJAK');
3 END;
4 /
5 --
```

Obrázok 6 – pregenerovanie štatistík pre aktualizáciu kvôli novým tabuľkám

Prehľad o počtoch dát

Na úvod si pre lepšiu predstavu zistíme najzákladnejšiu štatistiku a to počty záznamov o nehodách a vozidlách za jednotlivé roky. Túto informáciu vieme pekne zobrazit pomocou pivotovej transformácie dosiahnutej použitím agregáčnej funkcie *sum*, ktorá pripočíta do stĺpca záznam prave vtedy, keď spĺňa kritérium príslušnosti roku, pre ktorý je daný stĺpec určený. Stĺpec *cas* je *NOT NULL*, preto nemusíme ošetrovať podmienku s chýbajúcou hodnotou. Takéto dopyty sme vytvorili nad oboma tabuľkami nehôd aj vozidiel a následne sme ich zjednotili pre ucelenú informáciu v jednom výpise. Skript aj výpis je zobrazený na obrázku 7.



```
7 select 'Pocet autonehody za rok:' as popis,
8        sum(case extract(year from cas) when 2023 then 1 else 0 end) as count_2023,
9        sum(case extract(year from cas) when 2024 then 1 else 0 end) as count_2024,
10       sum(case extract(year from cas) when 2025 then 1 else 0 end) as count_2025
11 from cr_nehody
12 UNION ALL
13 select 'Pocet vozidiel za rok:' as popis,
14        sum(case extract(year from cas) when 2023 then 1 else 0 end) as count_2023,
15        sum(case extract(year from cas) when 2024 then 1 else 0 end) as count_2024,
16        sum(case extract(year from cas) when 2025 then 1 else 0 end) as count_2025
17 from cr_vozidla
18 join cr_nehody using(id_nehoda);
```

POPIS	COUNT_2023	COUNT_2024	COUNT_2025
1 Pocet autonehody za rok:	92736	91211	6932
2 Pocet vozidiel za rok:	72408	140146	10373

Obrázok 7 – počty nehôd a vozidel za jednotlivé evidované roky

Rok 2025 obsahuje pomerovo iba zlomok dát, pretože sme doteraz mali prístup len k záznamom za mesiac Január. Za rok 2023 je počet záznamov o vozidlách tiež zredukovaný – vybrali sme len necelú polovicu záznamov pre ukážku dát, ktoré sú vzorkou z tohto roku.

TOP mesiac/e pre každý kraj podľa počtu úmrtí za rok 2024

Prvou informáciou, ktorá nás môže zaujímať je koľko a v akých mesiacoch pre konkrétny kraj Českej republiky (ďalej ČR) bol zaznamenaný najvyšší počet úmrtí spôsobených dopravnou nehodou za

vymedzené obdobie - rok 2024. Ukážeme si tri varianty ako napísať takýto dopyt na dáta a porovnáme ich náklady.

Variant 1 – využitie vnoreného príkazu *select* (obrázok 8)

```

21 -- mesiace s najviac umrtiami pre kazdy kraj za rok 2024
22 --vl: cost 1530
23 SELECT kraj, mesiac, mrtvych
24 FROM (
25     SELECT kk.id_kraj, kk.nazov kraj, TO_CHAR(nn.cas, 'Mon','NLS_DATE_LANGUAGE=Slovak') AS mesiac,
26           extract(month from cas) as mesiac_id,
27           sum(nn.mrtvi) AS mrtvych
28     FROM CR_NEHODY nn
29    JOIN cr_kraje kk ON nn.id_kraj = kk.id_kraj
30   where extract(year from cas)=2024
31   GROUP BY kk.id_kraj, kk.nazov, extract(month from cas), TO_CHAR(nn.cas, 'Mon','NLS_DATE_LANGUAGE=Slovak')
32 ) main
33 WHERE EXISTS (SELECT 1
34               FROM (
35                   SELECT id_kraj, max(poc_mrtvi) AS mrtvych
36                 FROM (
37                     SELECT ID_KRAJ, sum(mrtvi) AS poc_mrtvi
38                     FROM cr_nehody
39                     where extract(year from cas)=2024
40                     GROUP BY ID_KRAJ, EXTRACT(MONTH FROM cas)
41                 )
42                GROUP BY id_kraj
43               ) tmp
44              WHERE main.id_kraj = tmp.id_kraj AND main.mrtvych = tmp.mrtvych
45             )
46 ORDER BY mrtvych desc, kraj, mesiac_id;

```

KRAJ	M...	MRTVYCH
1 Stredočeský kraj	Sep	11
2 Jihomoravský kraj	Sep	10
3 Jihomoravský kraj	Okt	10
4 Plzeňský kraj	Mar	8
5 Ústecký kraj	Okt	7
6 Jihočeský kraj	Jan	6
7 kraj Vysočina	Jún	6
8 Moravskoslezský kraj	Sep	5
9 Moravskoslezský kraj	Okt	5
10 Olomoucký kraj	Aug	5
11 Královéhradecký kraj	Dec	4
12 Liberecký kraj	Okt	4
13 hl.mesto Praha	Júl	4
14 Karlovarský kraj	Jan	3
15 Karlovarský kraj	Aug	3
16 Pardubický kraj	Apr	3
17 Pardubický kraj	Aug	3
18 Pardubický kraj	Nov	3
19 Zlínský kraj	Júl	3

Obrázok 8 – najviac mŕtvych pre kraj a mesiac za rok 2024 (variant 1)

Vnorený *select*, ktorého výsledky označíme aliasom *tmp* vytvorí pre záznamy z tabuľky *cr_nehody* skupiny podľa mesiaca a kraja, ku ktorým záznamy prislúchajú. Samozrejme, pred samotným zatriedením odfiltrujeme záznamy, ktoré sú mimo roku 2024. Pre skupiny potom získame počty záznamov, ku ktorým si zapamätáme identifikátor ich skupiny – stačí kraj. Následne pre každý kraj vyberieme iba záznam s maximálnym počtom nehôd. Výsledky tohto dopytu budeme vyhľadávať v druhom *selecte* označenom aliasom *main*, kde si budeme pamätať kraj, jeho názov, mesiac a počet mŕtvych dôsledkom dopravnej nehody. Ak sa takýto záznam zhoduje so záznamom z vnoreného príkazu *select tmp*, ktorý obsahuje maximálne počty, potom tento záznam budeme vypisovať ako výsledok nášho hľadania. Nevýhodou tohto riešenia je, že vnorený príkaz *select tmp* sa musel vykonať nad každým záznamom pre *select main*.

Variant 2 – využitie analytických funkcií (obrázok 9)

```

49 --v2: cost 769
50 select kraj, mesiac, mrtvych
51 from (
52     SELECT DISTINCT kraj, mesiac, mrtvych, mesiac_id
53     FROM (
54         SELECT kraj, mesiac, mrtvych, dense_rank() OVER (PARTITION BY kraj ORDER BY mrtvych desc) AS rnk,
55             mesiac_id
56         FROM (
57             SELECT k.nazov AS kraj, TO_CHAR(n.cas, 'Mon','NLS_DATE_LANGUAGE=Slovak') AS mesiac,
58                 sum(n.mrtvi) OVER (PARTITION BY k.id_kraj, EXTRACT(MONTH FROM cas) ORDER BY null) AS mrtvych,
59                 extract (month from n.cas) as mesiac_id
60             FROM cr_nehody n
61             JOIN cr_kraje k ON n.id_kraj = k.id_kraj
62             where extract(year from cas)=2024
63         )
64     ) WHERE rnk = 1
65     ORDER BY mrtvych DESC, kraj, mesiac_id
66 );

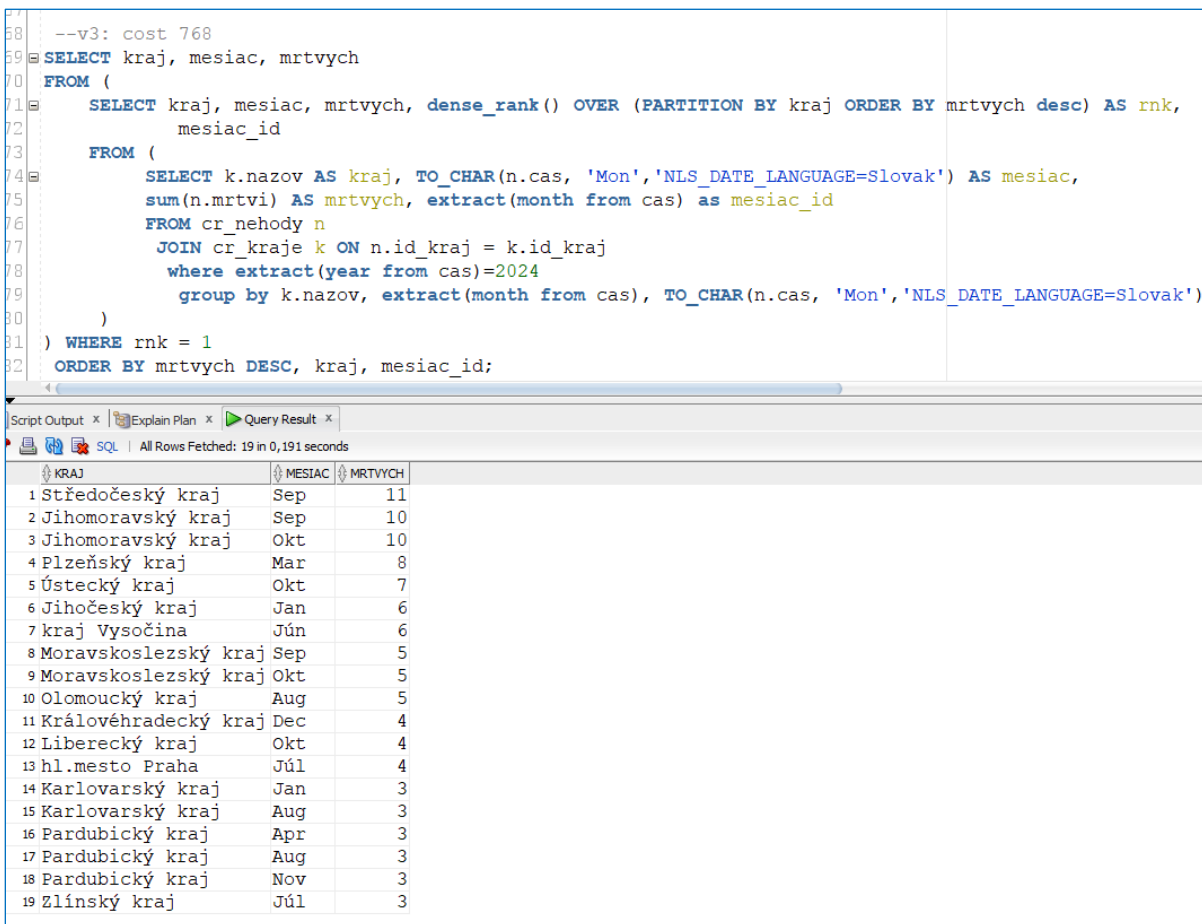
```

KRAJ	MESIAC	MRTVYCH
1 Středočeský kraj	Sep	11
2 Jihomoravský kraj	Sep	10
3 Jihomoravský kraj	Okt	10
4 Plzeňský kraj	Mar	8
5 Ústecký kraj	Okt	7
6 Jihočeský kraj	Jan	6
7 kraj Vysočina	Jún	6
8 Moravskoslezský kraj	Sep	5
9 Moravskoslezský kraj	Okt	5
10 Olomoucký kraj	Aug	5
11 Královéhradecký kraj	Dec	4
12 Liberecký kraj	Okt	4
13 hl.mesto Praha	Júl	4
14 Karlovarský kraj	Jan	3
15 Karlovarský kraj	Aug	3
16 Pardubický kraj	Apr	3
17 Pardubický kraj	Aug	3
18 Pardubický kraj	Nov	3
19 Zlínský kraj	Júl	3

Obrázok 9 - najviac mŕtvych pre kraj a mesiac za rok 2024 (variant 2)

V druhom variante využijeme namiesto vnoreného príkazu *select* analytickú funkciu *sum*, v ktorej si spočítame všetkých mŕtvych pre daný kraj a mesiac, čo definujeme v klauzule *PARTITION BY*. Keďže ide o analytickú funkciu, počet záznamov sa neredukuje, iba každému záznamu podľa jeho skupiny priradí vypočítané číslo. Každý záznam v skupine dostane rovnakú hodnotu sumy, pretože nerobíme tzv. rolling sum, lebo podľa ničoho netriedime (*order by null*). Takto získané hodnoty ohodnotíme ďalšou analytickou funkciou *dense_rank*, ktorá priradí každému záznamu v rámci skupiny (definovanej krajom) ohodnotenie podľa počtu mŕtvych s TOP ohodnotením pre záznamy s najvyšším počtom mŕtvych. Následne len vyberiem tie záznamy, ktoré dostali ohodnotenie 1 – „najlepšie“, to znamená, najvyšší počet mŕtvych pre každý kraj. Náklady v porovnaní s variantom 1 sú polovičné, pretože sme nemuseli zakaždým vyhodnocovať vnorený príkaz *select*, ale len raz sme si ku každému záznamu priradili jeho ohodnotenie v rámci skupiny na základe počtu.

Variant 3 – využitie kombinácie agregáčnych a analytických funkcií (obrázok 10)



```
--v3: cost 768
SELECT kraj, mesiac, mrtvych
FROM (
  SELECT kraj, mesiac, mrtvych, dense_rank() OVER (PARTITION BY kraj ORDER BY mrtvych desc) AS rn timer,
    mesiac_id
  FROM (
    SELECT k.nazov AS kraj, TO_CHAR(n.cas, 'Mon', 'NLS_DATE_LANGUAGE=Slovak') AS mesiac,
      sum(n.mrtvi) AS mrtvych, extract(month from cas) as mesiac_id
    FROM cr_nehody n
    JOIN cr_kraje k ON n.id_kraj = k.id_kraj
    where extract(year from cas)=2024
    group by k.nazov, extract(month from cas), TO_CHAR(n.cas, 'Mon', 'NLS_DATE_LANGUAGE=Slovak')
  )
) WHERE rn timer = 1
ORDER BY mrtvych DESC, kraj, mesiac_id;
```

KRAJ	MESIAC	MRTVYCH
1 Středočeský kraj	Sep	11
2 Jihomoravský kraj	Sep	10
3 Jihomoravský kraj	Okt	10
4 Plzeňský kraj	Mar	8
5 Ústecký kraj	Okt	7
6 Jihočeský kraj	Jan	6
7 kraj Vysočina	Jún	6
8 Moravskoslezský kraj	Sep	5
9 Moravskoslezský kraj	Okt	5
10 Olomoucký kraj	Aug	5
11 Královéhradecký kraj	Dec	4
12 Liberecký kraj	Okt	4
13 hl.mesto Praha	Júl	4
14 Karlovarský kraj	Jan	3
15 Karlovarský kraj	Aug	3
16 Pardubický kraj	Apr	3
17 Pardubický kraj	Aug	3
18 Pardubický kraj	Nov	3
19 Zlínský kraj	Júl	3

Obrázok 10 - najviac mŕtvych pre kraj a mesiac za rok 2024 (variant 3)

Tretí variant sa od druhého líši iba v spôsobe spočítania počtu mŕtvych pre skupiny definované krajom a mesiacom. Na to sme použili namiesto analytickej funkcie agregáčnú funkciu *sum* tak, že sme použili klauzulu *GROUP BY*. Vďaka tomu sme spočítali sumy pre skupiny a tak sme aj, na rozdiel od analytickej funkcie, zredukovali počet záznamov tak, že sme dostali práve jeden záznam pre každú skupinu. Následne sme, podobne ako vo variante 2, iba ohodnotili počty a vybrali pre každý kraj mesiace s najvyšším počtom úmrtí. Porovnanie nákladov na vykonanie variantu 2 a variantu 3 sa javia byť skoro rovnaké (769 verzus 768, t.j. pri variante 3 sme ušetrili 1 jednotku nákladov).

N-tý mesiac s najväčším počtom nehôd

Ďalej by nás mohlo zaujímať, ktoré 3 mesiace bolo zaznamenaných najviac nehôd a koľko ich presne bolo. Pre prvých *n* mesiacov by bol obtiažnejší výpis bez použitia analytickej funkcie. Síce si taký príklad neskôr ukážeme, ale zamerajme sa teraz len na výpis tretieho mesiaca v poradí s počtom najviac nehôd za rok 2024.

Variant 1 – využitie vnoreného príkazu *select* (obrázok 11)

87	--v1: cost: 763 vnorene selecty na ziskanie lokalneho maxima
88	select 'Top 3. najviac nehod (2024): ' as popis,
89	TO_CHAR(cas, 'Mon', 'NLS_DATE_LANGUAGE=Slovak') as mesiac, count(*) as poc_nehod
90	from cr_nehody
91	where extract(year from cas)=2024
92	group by TO_CHAR(cas, 'Mon', 'NLS_DATE_LANGUAGE=Slovak')
93	having count(*) = (select max(count(*))
94	from cr_nehody
95	where extract(year from cas)=2024
96	group by extract(month from cas)
97	having count(*) < (select max(count(*))
98	from cr_nehody
99	where extract(year from cas)=2024
00	group by extract(month from cas)
01	having count(*) < (select max(count(*))
02	from cr_nehody
03	where extract(year from cas)=2024
04	group by extract(month from cas)
05)
06)
07);

Script Output	Explain Plan	Query Result
All Rows Fetched: 1 in 0,431 seconds		
POPIS	MESIAC	POC_NEHOD
Top 3. najviac nehod (2024):	Júl	8215

Obrázok 11 - TOP 3. mesiac s najviac nehodami v 2024 (variant 1)

Prvý variant je postavený na vnorených príkazoch *select*, ktoré získajú maximum pri splnení podmienky roku 2024 a zároveň maximum nesmie prekročiť hranicu definovanú vnorenejším príkazom *select*. Takto si vytvoríme toľko vnorení, ktorú n-tú najväčšiu hodnotu v poradí potrebujeme, pričom najvnorenejší *select* predstavuje globálne maximum. Najmenej vnorený *select* obsahuje našu hľadanú n-tú hodnotu, na ktorú sa pozeráme z vonkajšieho príkazu *select*. Toto porovnávanie počtov na najvrchnejšej úrovni nerobíme kvôli získaniu daného počtu, pretože ten sme získali už prvým vnoreným príkazom *select*. Robíme to, aby sme zistili, pre ktorý mesiac je tento počet priradený a tak ho mohli spolu s počtom vypísať.

Variant 2 – využitie klauzuly *fetch* (obrázok 12)

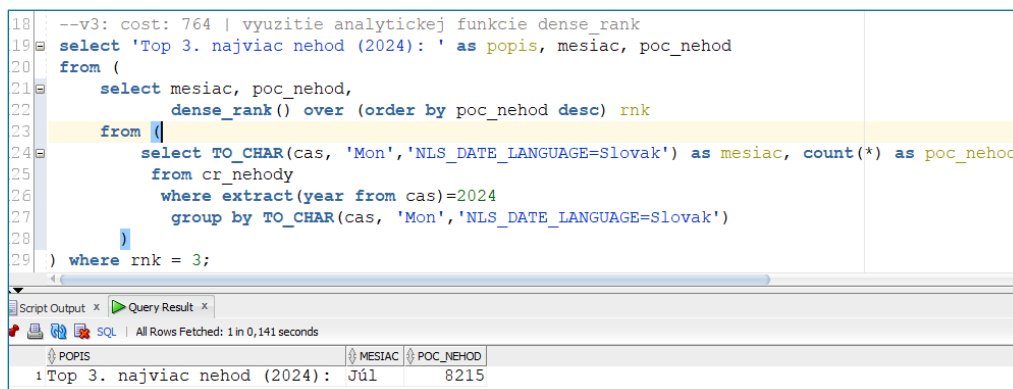
108	--v2: cost: 766 pouzitie fetch so zoradenim
109	select 'Top 3. najviac nehod (2024): ' as popis, TO_CHAR(cas, 'Mon', 'NLS_DATE_LANGUAGE=Slovak') as month,
110	count(*) as poc_nehod
111	from cr_nehody
112	where extract(year from cas)=2024
113	group by TO_CHAR(cas, 'Mon', 'NLS_DATE_LANGUAGE=Slovak')
114	order by poc_nehod desc
115	offset 2 rows fetch first row with ties;
116	

Script Output	Explain Plan	Query Result
All Rows Fetched: 1 in 0,145 seconds		
POPIS	MONTH	POC_NEHOD
Top 3. najviac nehod (2024):	Júl	8215

Obrázok 12 - TOP 3. mesiac s najviac nehodami v 2024 (variant 2)

V druhom variante si pre každý mesiac roku 2024 zistíme počet nehôd a následne je dôležité zoradiť výsledky príkazu *select* vzostupne podľa počtu nehôd. Potom využijeme klauzulu *offset*, pomocou ktorej preskočíme prvé dva najlepšie mesiace a zoberiem s využitím *fetch first row with ties* iba riadky, ktoré podľa počtu predstavujú náš hľadaný top 3. mesiac. Toto riešenie ale nemusí byť správne, pokiaľ napríklad na druhom mieste sme mali dva mesiace. Vybrali by sme teda jeden záznam z dvoch, ktoré prislúchajú k druhému mesiacu a nedozvedeli by sme sa, že sme sa dopustili chyby. Náklady sa zhoršili voči prvému variantu iba o 3 jednotky, takže variant 1 a 2 sú podobne výkonné.

Variant 3 – využitie analytickej funkcie *dense_rank* (obrázok 13)



```
18 --v3: cost: 764 | vyuzitie analytickej funkcie dense_rank
19 select 'Top 3. najviac nehod (2024): ' as popis, mesiac, poc_nehod
20 from (
21   select mesiac, poc_nehod,
22          dense_rank() over (order by poc_nehod desc) rnk
23   from (
24     select TO_CHAR(cas, 'Mon', 'NLS_DATE_LANGUAGE=Slovak') as mesiac, count(*) as poc_nehod
25     from cr_nehody
26     where extract(year from cas)=2024
27     group by TO_CHAR(cas, 'Mon', 'NLS_DATE_LANGUAGE=Slovak')
28   )
29 ) where rnk = 3;
```

POPIS	MESIAC	POC_NEHOD
Top 3. najviac nehod (2024):	Júl	8215

Obrázok 13 - TOP 3. mesiac s najviac nehodami v 2024 (variant 3)

V poslednom variante si pomocou agregáčnej funkcie *count* pre každý mesiac spočítame počet záznamov a následne tieto počty zostupne ohodnotíme pomocou analytickej funkcie *dense_rank* a vyberieme záznamy s ohodnotením 3. Výhodou tohoto variantu oproti prvým dvom je, že malou zmenou môžeme získať ľubovoľné *n*-té umiestnenie, resp. ľubovoľný interval za sebou idúcich umiestnení. Zmenili by sme iba podmienku, ktoré ohodnotenia chceme získať (t.j. aké čísla môže obsahovať stĺpec *rnk*). Čo sa týka nákladov, tie sa nezmenili (náklady sa zhoršili len o 1 jednotku oproti variantu 1), ale silno sa nám zvýšila miera možnej modifikácie výberov na základe umiestnenia, ktoré je výstupom analytickej funkcie.

Alkohol a dopravné nehody

Niektoré dopravné nehody bývajú zbytočne zapríčinené požitím alkoholu pred jazdou, v dôsledku čoho sa schopnosti účastníkov cestnej komunikácie byť pozorný výrazne znižujú.

Ročné početnosti podľa percentuálnej prítomnosti alkoholu v krvi pre rok 2024

Ako prvé by nás mohlo zaujímať pri koľkých nehodách bola zistená prítomnosť alkoholu a v akom veľkom množstve. Pre zistenie sme vypočítali agregáčnou funkciou *count* počty podľa druhov prítomnosti alkoholu v krvi pri nehodách, ako je toobrazené na obrázku 14 spolu s výsledkami. Z výsledkov môžeme vyčítať, že okolo 50% prípadov alkohol nebol prítomný pri nehode. Zaujímavé je, že pri vyše 40% nehôd sa nezistovala prítomnosť alkoholu v krvi. V približne 5% nehôd bola potvrdená prítomnosť alkoholu, pričom ak bola prítomnosť potvrdená, potom je vyše 60% pravdepodobnosť, že obsah alkoholu v krvi je viac ako 1.5%.

```

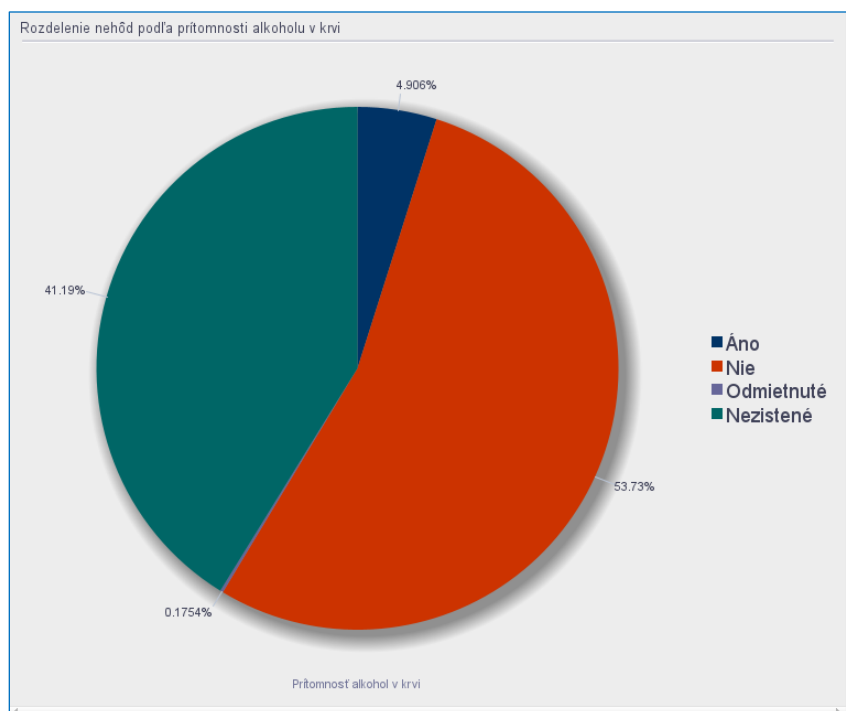
24 -- statistika pre obsah alkoholu pocas nehody
25 select case when al.pritomny='A' then 'Ano'
26           when al.pritomny='N' then 'Nie'
27           when al.pritomny='O' then 'Odmietnute'
28           when al.pritomny='X' then 'Nezistovane'
29           else 'Nezname' end as alkohol_bol_pritomny,
30       al.obsah_perc as obsah_alkoholu_v_krvi_pri_nehode,
31       count(*) as poc_zavinenych_nehod
32 from cr_nehody n
33 join cr_pritomnost_alko al on n.id_alko_prit = al.id_stav
34 where extract(year from cas)=2024
35 group by al.id_stav, al.pritomny, al.obsah_perc
36 order by poc_zavinenych_nehod desc;

```

ALKOHOL_BOL_PRITOMNY	OBSAH_ALKOHOLU_V_KRVI_PRI_NEHODE	POC_ZAVINENYCH_NEHOD
1 Nie	(null)	49004
2 Nezistovane	(null)	37572
3 Ano	obsah alkoholu v krvi 1.5 % a více	2734
4 Ano	obsah alkoholu v krvi od 1.0 % do 1.5 %	751
5 Ano	obsah alkoholu v krvi od 0.24 % do 0.5 %	315
6 Ano	obsah alkoholu v krvi od 0.5 % do 0.8 %	282
7 Ano	obsah alkoholu v krvi od 0.8 % do 1.0 %	210
8 Ano	obsah alkoholu v krvi do 0.24 %	183
9 Odmietnute	(null)	160

Obrázok 14 – ročné počty nehôd podľa prítomnosti alkoholu v krvi

Na vizuálnu ukážku sme si v programe *SqlDeveloper* vytvorili report (obrázok 15), ktorý zobrazuje percentuálne podiely nehôd kde alkohol bol prítomný v krvi (modrá), kde nebol (červená), kde to nebolo zistené (zelená) a kde to bolo odmietnuté (jemne fialová). Vidíme, že skoro 5% nehôd bolo s nejakým obsahom alkoholu v krvi (tu abstrahujeme mieru obsahu). Zaujímavé je, že vyše 41% prípadov to nebolo zistené z rôznych dôvodov, ktoré nepoznáme, lebo dáta v *xls* súboroch túto informáciu neobsahovali.



Obrázok 15 – podiely počtov nehôd podľa zistenia obsahu alkoholu v krvi pre rok 2024

Mesačné početnosti podľa percentuálnej prítomnosti alkoholu v krvi pre rok 2024

Okrem početností percentuálnej prítomnosti alkoholu v krvi ročne nás môžu zaujímať aj mesačné početnosti alkoholu v krvi, ako sa celkovo počas roka vyvíjala. Pre lepšie pochopenie uvedieme aj relatívne percentuálne početnosti podielu zisteného alkoholu v krvi pre nehody v rámci mesiaca. Teraz už budeme abstrahovať od miery alkoholu v krvi a bude nás len zaujímať či alkohol bol prítomný, alebo nie. Výsledky spolu so skriptom sú uvedené na obrázku 16. Je vidno, že najviac nehôd so zisteným alkoholom v krvi sa zaznamenal v letných mesiacoch. Môže to byť spojené s tým, že v lete chodia ľudia väčšinou na dovolenky a viacej ľudí berie zodpovednosť vtedy menej vážne. Na konci roka v decembri môžeme tiež vidieť nárast prípadov, čo môže byť opäť istým spôsobom spojené s časom sviatkov.

```

-- pivotova transformacia pre mesiace
select mesiac, ' Pockty: ' as popis1, alko_ano, alko_nie, alko_odm, alko_nez,
       ' Percenta: ' as popis2,
       round(100*alko_ano/mes_all,2)||'%' as p_alko_ano,
       round(100*alko_nie/mes_all,2)||'%' as p_alko_nie,
       round(100*alko_odm/mes_all,2)||'%' as p_alko_odm,
       round(100*alko_nez/mes_all,2)||'%' as p_alko_nez
from (
  select extract(month from cas) as mesiac_id,
         TO_CHAR(cas, 'Month', 'NLS_DATE_LANGUAGE=Slovak') as mesiac,
         sum(case al.pritomny when 'A' then 1 else 0 end) as alko_ano,
         sum(case al.pritomny when 'N' then 1 else 0 end) as alko_nie,
         sum(case al.pritomny when 'O' then 1 else 0 end) as alko_odm,
         sum(case al.pritomny when 'X' then 1 else 0 end) as alko_nez,
         count(*) as mes_all
  from cr_nehody n
  join cr_pritomnost_alko al on n.id_alko_prit = al.id_stav
  where extract(year from cas)=2024
  group by extract(month from cas), TO_CHAR(cas, 'Month', 'NLS_DATE_LANGUAGE=Slovak')
)
order by mesiac_id;

```

MESIAC	POPIS1	ALKO_ANO	ALKO_NIE	ALKO_ODM	ALKO_NEZ	POPIS2	P_ALKO_ANO	P_ALKO_NIE	P_ALKO_ODM	P_ALKO_NEZ
1 Január	Pockty:	282	4041	11	3095	Percenta:	3.8%	54.39%	.15%	41.66%
2 Február	Pockty:	258	2989	11	2693	Percenta:	4.34%	50.23%	.18%	45.25%
3 Marec	Pockty:	330	3473	11	2982	Percenta:	4.86%	51.1%	.16%	43.88%
4 Apríl	Pockty:	316	3952	14	3334	Percenta:	4.15%	51.89%	.18%	43.78%
5 Máj	Pockty:	387	4421	14	3484	Percenta:	4.66%	53.23%	.17%	41.95%
6 Jún	Pockty:	423	4442	19	3199	Percenta:	5.23%	54.95%	.24%	39.58%
7 Júl	Pockty:	493	4581	11	3130	Percenta:	6%	55.76%	.13%	38.1%
8 August	Pockty:	535	4327	24	3129	Percenta:	6.67%	53.99%	.3%	39.04%
9 September	Pockty:	382	4761	10	3228	Percenta:	4.56%	56.81%	.12%	38.52%
10 Október	Pockty:	358	4352	10	3489	Percenta:	4.36%	53.01%	.12%	42.5%
11 November	Pockty:	354	3876	12	3097	Percenta:	4.82%	52.81%	.16%	42.2%
12 December	Pockty:	357	3789	13	2712	Percenta:	5.2%	55.14%	.19%	39.47%

Obrázok 16 – mesačné počty nehôd podľa prítomnosti alkoholu v krvi

Percentuálny podiel škôd nehôd s prítomnosťou alkoholu v TOP 1000 najvyššie ocenených škodách

Môžeme predpokladať, že nehody s prítomnosťou alkoholu môžu mať veľmi katastrofálne následky, preto nás bude zaujímať všeobecné ohodnotenie TOP 1000 najnákladnejších nehôd, čo je približne 1% všetkých nehôd za rok 2024. Z nich identifikujeme všetky, ktoré boli s prítomnosťou alkoholu a ich ohodnotenie dáme do pomeru s celkovými škodami TOP 1000 nehôd, ako to vysvetľuje šablóna príkazu `select` na obrázku 17.

```

select skoda_alko, skoda_celkom, round(100*skoda_alko/skoda_celkom, 2) as perc_podiel from (
select
(select x from dual) as skoda_alko,
(select y from dual) as skoda_celkom
from dual
);

```

Obrázok 17 – abstrakcia výpočtu percentuálneho podielu pre hľadajú skutočnosť v jazyku SQL

Rozoberme si postupne príkaz *select* pre čitateľ a menovateľ zvlášť.


1. Menovateľ (obrázok 18) – celková škoda TOP 1000 nehôd s najväčšími vyčíslenými škodami v peňažnej mene Kč. Potrebujeme si zoradiť nehody za rok 2024 zostupne podľa veľkosti škody v Kč a následne vybrať prvých 1000 záznamov. Keďže nezohľadňujeme koľko záznamov môže mať teoreticky rovnakú hodnotu škody (nehodnotíme princípom hodnotenia olympijských hier), ale stačí nám jednoducho 1000 ľubovoľných najnákladnejších nehôd, môžeme použiť analytickú funkciu *row_number* na získanie ohodnotenia pre každý záznam podľa jeho vyčíslenej škody. Tu máme 2 spôsoby sčítania prvých 1000 záznamov s najlepším ohodnotením.
 - a. Prvý spôsob je pomocou analytickej funkcie *sum* s využitím tzv. rolling sum, to znamená postupné nasčítanie hodnôt ako suma hodnôt stĺpca predchádzajúcich riadkov s vyšším ohodnotením (príp. rovnakým pri variantoch funkcie *rank*) a aktuálnym riadkom. Aby nasčítavanie fungovalo správne, musíme uviesť, že sa majú hodnoty nasčítavať zoradené podľa hodnoty škody (*order by celk_skoda_kc desc*). Potom stačí, keď si vyberieme riadok s ohodnotením *rn=1000*, kde je nasčítaných TOP 1000 škôd.
 - b. Druhý variant je po vytvorení ohodnotení škôd využitie agregačnej funkcie *sum*, ktorou zrátame po odfiltrovaní nepotrebných záznamov (cez *where* podmienku) sumu všetkých zostávajúcich hodnôt stĺpca so s hodnotou škody.

Náklady oboch variantov sú rovnaké a ich skript je zobrazený na obrázku 18.

```

93 -- v1 - celkova škoda vypocitana z 15 najvacsich nehod lubovolneho druhu
94 select suma_kc --v1(analytic) cost: 763
95 from (
96 select celk_skoda_kc as skoda,
97 row_number() over(order by celk_skoda_kc desc) as rn,
98 sum(celk_skoda_kc) over (order by celk_skoda_kc desc) as suma_kc
99 from cr_nehody
100 where extract(year from cas)=2024
101 ) where rn = 1000;
102
103 -- v2 - celkova škoda vypocitana z 15 najvacsich nehod lubovolneho druhu
104 select sum(skoda) suma_kc --v2(aggregate) cost: 763
105 from (
106 select celk_skoda_kc as skoda,
107 row_number() over(order by celk_skoda_kc desc) as rn
108 from cr_nehody
109 where extract(year from cas)=2024
110 ) where rn <= 1000;
111

```



Obrázok 18 – dve možnosti spočítania sumy TOP 1000 najvyšších škôd roku 2024

2. Čitateľ (obrázok 19) – celková škoda s prítomnosťou alkoholu spomedzi TOP 1000 nehôd

Najprv si zoradíme podľa veľkosti škody v Kč všetky nehody a priradíme im ohodnotenie pomocou analytickej funkcie `row_number`. Okrem toho si musíme zapamätať či bol alkohol prítomný v krvi počas nehody, alebo nie. Následne, keď máme záznamy zoradené podľa ohodnotenia, vyberieme v podmienke `where` všetky záznamy, ktoré sú medzi prvými TOP 1000 a zároveň bol pri nehode alkohol prítomný v krvi. Potom už stačí len spočítať celkovú sumu škôd z vybraných záznamov a to môžeme dosiahnuť dvoma spôsobmi:

- Využitie analytickej funkcie `sum`, v ktorej neuvedíme podmienku triedenia, čím zabránime využitiu `rolling sum`, ale namiesto toho sa sčíta suma zo všetkých záznamov a priradí sa ku každému záznamu. Potom nám stačí pomocou klauzuly `fetch` zobrať ľubovoľný, napr. prvý riadok, v ktorom získame hľadanú sumu.
- Druhý(jednoduchší) spôsob je využitie agregáčnej funkcie `sum` bez definovania skupiny pre všetky vybrané záznamy.

```

13 -- v1 celkova skoda vypocitana z 1000 najvacsih nehod so zistenym obsahom alkoholu v krvi
14 select sum(skoda) over() as suma_kc --v1(analytic) cost: 766;
15 from (
16   select a.pritomny as alko_pritomny, celk_skoda_kc as skoda,
17          row_number() over(order by celk_skoda_kc desc) as rn
18   from cr_nehody n
19   join cr_pritomnost_alko a on n.id_alko_prit = a.id_stav
20   where extract(year from cas)=2024
21 ) where rn <= 1000 and alko_pritomny='A'
22   fetch first row only;
23
24 -- v2 celkova skoda vypocitana z 1000 najvacsih nehod so zistenym obsahom alkoholu v krvi
25 select sum(skoda) suma_kc --v2(aggregate) cost: 766
26 from (
27   select a.pritomny as alko_pritomny, celk_skoda_kc as skoda,
28          row_number() over(order by celk_skoda_kc desc) as rn
29   from cr_nehody n
30   join cr_pritomnost_alko a on n.id_alko_prit = a.id_stav
31   where extract(year from cas)=2024
32 ) where rn <= 1000 and alko_pritomny='A';

```

SUMA_KC
53773000

Obrázok 19 - dve možnosti spočítania sumy škôd s obsahom alkoholu spomedzi TOP 1000 roku 2024

Po vytvorení príkazov `select` pre čitateľ aj menovateľ stačí napísať `select` podľa vzoru na obrázku 17. Príklad výsledného skriptu je na obrázku 20. Môžeme vidieť, že na TOP 1000 nehodách podľa veľkosti vyčíslených škôd mali škody s prítomnosťou alkoholu v krvi 3,81% podiel na sumárnej škode.

```

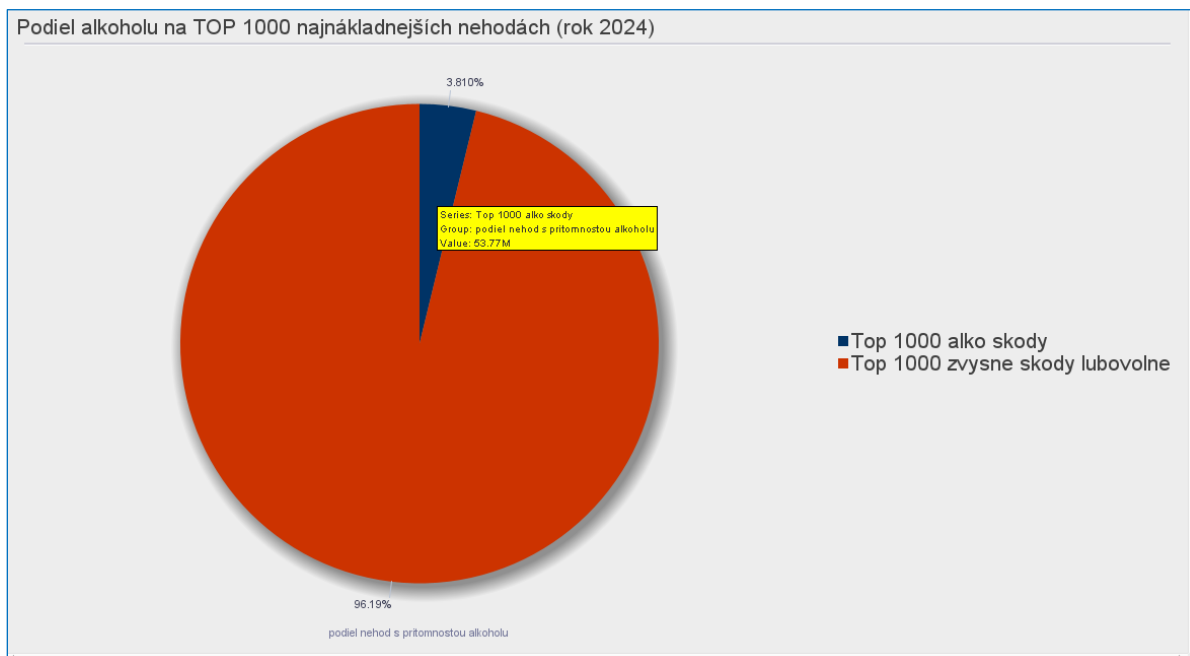
34 -- z prvych 1000 najnakladnejších nehod kolkymi percentami sa na nich podielali nehody
35 -- cost: 1531
36 select 'Podiel skody s pritom. alko vramci TOP 1000:' as popis,
37 round(100*(sum_kc_alko/sum_kc_celk),3) || '%' as perc, sum_kc_alko, sum_kc_celk
38 from (
39     select
40         (select sum(skoda) suma_kc --v2(aggregate) cost: 766
41             from (
42                 select a.pritomny as alko_pritomny, celk_skoda_kc as skoda,
43                     row_number() over(order by celk_skoda_kc desc) as rn
44                 from cr_nehody n
45                 join cr_pritomnost_alko a on n.id_alko_prit = a.id_stav
46                 where extract(year from cas)=2024
47                 ) where rn <= 1000 and alko_pritomny='A'
48             ) sum_kc_alko,
49         (select sum(skoda) suma_kc --v2(aggregate) cost: 763
50             from (
51                 select celk_skoda_kc as skoda,
52                     row_number() over(order by celk_skoda_kc desc) as rn
53                 from cr_nehody
54                 where extract(year from cas)=2024
55                 ) where rn <= 1000
56             ) sum_kc_celk
57         from dual);

```

POPIS	PERC	SUM_KC_ALKO	SUM_KC_CELK
Podiel skody s pritom. alko vramci TOP 1000:	3.81%	53773000	1411356000

Obrázok 20 – výsledný skript pre získanie perc. podielu škôd s prít. alkoholu na TOP 1000 škodách

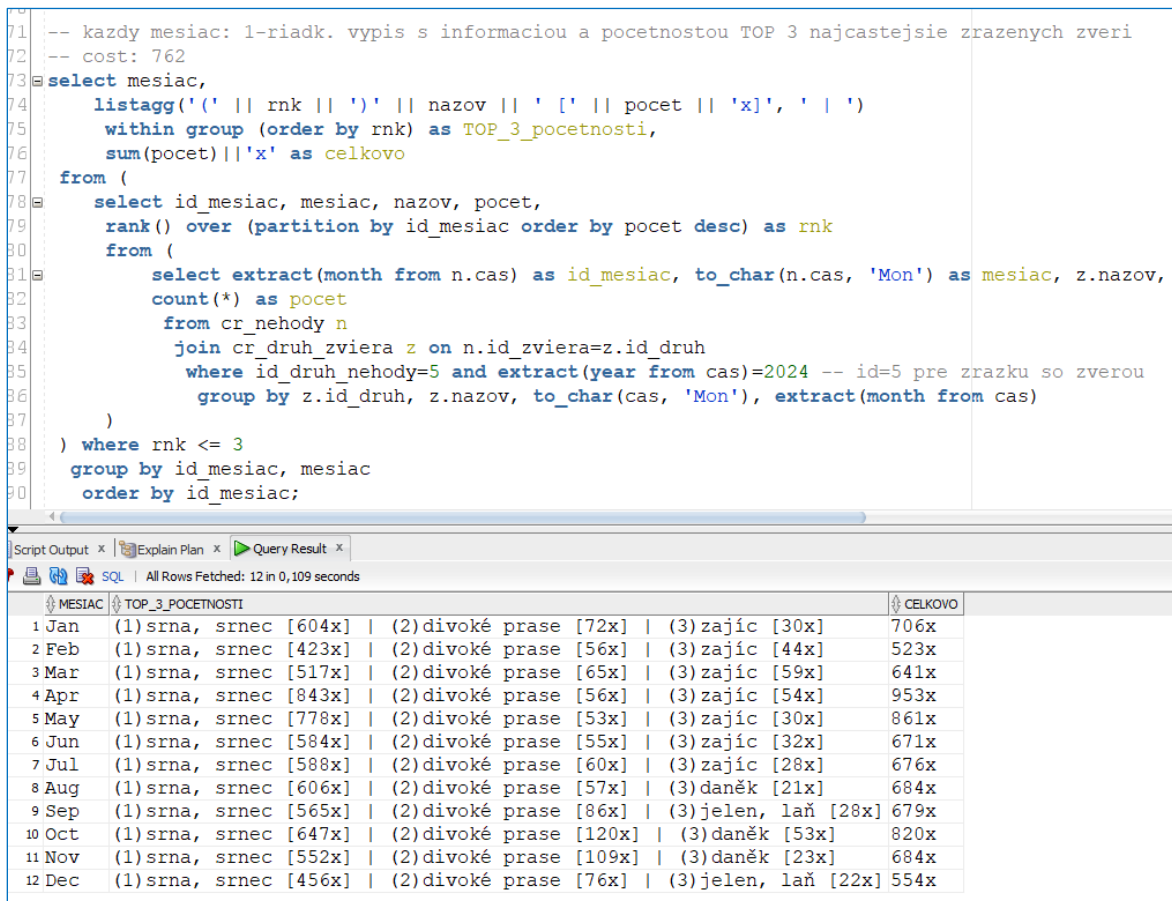
Pre lepšiu predstavu sme vytvorili aj report (obrázok 21) podielu alkoholu na TOP 1000 najnákladnejších nehodách 2024 v programe *SqlDeveloper*. Výsek vyfarbený na modro ukazuje nehody spomedzi TOP 1000, kde bol prítomný alkohol v krvi.



Obrázok 21 – report s podielom nehôd s prítomnosťou alkoholu k ostatným spomedzi TOP 1000 podľa najvyšších nákladov v Kč z roku 2024

Mesačne najčastejšie zrazené zvieratá v roku 2024

Často dochádza k zrážkam so zverou. Vypíšeme si do jedného riadku pre každý mesiac informáciu o troch najviac zrazených zvieratách v danom mesiaci a k tomu uvedenie informáciu celkového počtu zrážok v mesiaci TOP 3 zvierat. Podľa obrázku 22 je zrejmé, že všeobecne v roku 2024 dochádzalo k najviac zrážkam so srnami, diviakmi a zajacmi. Taktiež, podľa obrázku 22, najviac TOP 3 zrazených zvierat bolo v apríli a to skoro až 1000, čo je o vyše 80% viac než vo februári. Využívame tu agregačnú funkciu *listagg*, ktorou agregujeme do jedného riadku všetky vybrané záznamy pre konkrétny mesiac a vypisujeme pomocou nej nami zvolené údaje, t.j. umiestnenie podľa počtu incidentov so zverou, meno zvieratá a počty zrazení.



```
71 -- kazdy mesiac: 1-riadk. vypis s informaciou a pocetnostou TOP 3 najcastejsie zrazenych zveri
72 -- cost: 762
73 select mesiac,
74        listagg('(' || rnk || ') ' || nazov || ' [' || pocet || 'x]', ' ' || ') '
75        within group (order by rnk) as TOP_3_pocetnosti,
76        sum(pocet) || 'x' as celkovo
77 from (
78        select id_mesiac, mesiac, nazov, pocet,
79               rank() over (partition by id_mesiac order by pocet desc) as rnk
80        from (
81                select extract(month from n.cas) as id_mesiac, to_char(n.cas, 'Mon') as mesiac, z.nazov,
82                       count(*) as pocet
83                from cr_nehody n
84                join cr_druh_zviera z on n.id_zviera=z.id_druh
85                where id_druh_nehody=5 and extract(year from cas)=2024 -- id=5 pre zrazku so zverou
86                group by z.id_druh, z.nazov, to_char(cas, 'Mon'), extract(month from cas)
87        )
88        ) where rnk <= 3
89        group by id_mesiac, mesiac
90        order by id_mesiac;
```

MESIAČ	TOP_3_POCETNOSTI	CELKOVO
1 Jan	(1) srna, srnec [604x] (2) divoké prase [72x] (3) zajíc [30x]	706x
2 Feb	(1) srna, srnec [423x] (2) divoké prase [56x] (3) zajíc [44x]	523x
3 Mar	(1) srna, srnec [517x] (2) divoké prase [65x] (3) zajíc [59x]	641x
4 Apr	(1) srna, srnec [843x] (2) divoké prase [56x] (3) zajíc [54x]	953x
5 May	(1) srna, srnec [778x] (2) divoké prase [53x] (3) zajíc [30x]	861x
6 Jun	(1) srna, srnec [584x] (2) divoké prase [55x] (3) zajíc [32x]	671x
7 Jul	(1) srna, srnec [588x] (2) divoké prase [60x] (3) zajíc [28x]	676x
8 Aug	(1) srna, srnec [606x] (2) divoké prase [57x] (3) daněk [21x]	684x
9 Sep	(1) srna, srnec [565x] (2) divoké prase [86x] (3) jelen, laň [28x]	679x
10 Oct	(1) srna, srnec [647x] (2) divoké prase [120x] (3) daněk [53x]	820x
11 Nov	(1) srna, srnec [552x] (2) divoké prase [109x] (3) daněk [23x]	684x
12 Dec	(1) srna, srnec [456x] (2) divoké prase [76x] (3) jelen, laň [22x]	554x

Obrázok 22 – výpis troch najčastejšie zrazených zvierat pre každý mesiac v roku 2024

2,5 mesačný kľzavý medián počtu zranených pre rok 2024 (s frekvenciou polmesiac)

Poslednou informáciou, ktorú budeme zisťovať, je priebeh kľzavého mediánu počtu zranených (ľahko zranených + ťažko zranených) pre rok 2024 v časovom rade s frekvenciou polmesiac. Medián budeme počítať z piatich hodnôt, t.j. 5 počtov zranených zodpovedajúcich im prislúchajúcim polmesiacom. Skript rozdelíme do 4 krokov tak, ako to ukazuje obrázok 23.

```

92 -- 2.5 mesačný kľzavý medián počtu zranených(lahko + ťažko zranení) pre rok 2024 (s frekvenciou polmesiaca)
93 -- cost: 816
94 with
95     cislo_mesiac as (
96         select 1 as id_mesiac from dual
97         union select 2 from dual
98         union select 3 from dual
99         union select 4 from dual
100        union select 5 from dual
101        union select 6 from dual
102        union select 7 from dual
103        union select 8 from dual
104        union select 9 from dual
105        union select 10 from dual
106        union select 11 from dual
107        union select 12 from dual
108    ),
109    mesiace as (
110        select id_mesiac, to_date('2024-||id_mesiac||'-01', 'yyyy-mm-dd') as zac_mes
111        from cislo_mesiac
112    ),
113    polmesiace as (
114        select zac_mes as polmesiace from mesiace
115        union
116        select zac_mes + floor((last_day(zac_mes)-zac_mes+1)/2) as stred_mes from mesiace
117    ),
118    polmes_data as (
119        select polmesiace, sum(pocet_zraneni) as pocet_zraneni
120        from (
121            select polmesiace, pocet_zraneni
122            from (
123                select n.id_nehoda, trunc(n.cas) as datum, pm.polmesiace, (n.lahko_zraneni+n.tazko_zraneni) as pocet_zraneni,
124                row_number() over (partition by id_nehoda order by
125                (case when trunc(cas)-pm.polmesiace>=0 then trunc(cas)-pm.polmesiace else 10000 end)) as pm_priorita
126                from cr_nehody n
127                join polmesiace pm on trunc(n.cas, 'MM')=trunc(pm.polmesiace, 'MM')
128                where extract(year from cas)=2024
129            ) where pm_priorita=1 --kazdy zaznam bude mat teraz priradeny prave 1 polmesiace, takze pocet zaznamov = count(*)
130        )
131        group by polmesiace -- vyvoj pocet zranenych s frekvenciou polmesiaca
132    )
133    select polmesiace, median(pocet_zraneni)
134    from (
135        select polmesiace, pocet_zraneni from polmes_data
136        union all select polmesiace, lag(pocet_zraneni,1) over(order by polmesiace) from polmes_data
137        union all select polmesiace, lag(pocet_zraneni,2) over(order by polmesiace) from polmes_data
138        union all select polmesiace, lag(pocet_zraneni,3) over(order by polmesiace) from polmes_data
139        union all select polmesiace, lag(pocet_zraneni,4) over(order by polmesiace) from polmes_data
140    ) group by polmesiace order by polmesiace;

```

Obrázok 23 – skript pre 2,5 mesačný kľzavý medián počtu zranených pre rok 2024

- **Krok 1** – V prvom kroku si vytvoríme mesiace pre rok 2024 tak, že mesiac bude reprezentovaný jeho prvým dňom. Budeme využívať klauzulu *with*, aby sme sa mohli nasledovne odkazovať na vytvorenú štruktúru a mať pritom prehľadnejší príkaz *select*.
- **Krok 2** – V druhom kroku si vytvoríme polmesiace, ktoré získame tak, že k začiatku mesiaca pripočítame polovicu dní toho mesiaca s prípadným zaokruhlením dní nadol.
- **Krok 3** – V treťom kroku každému záznamu o nehode musíme priradiť polmesiace, ku ktorému prislúcha a potom pre každý polmesiace spočítať celkovú sumu počtu zranených pri nehodách, ktoré k danému polmesiacu prislúchajú.

Príslušnosť nehody k polmesiacu dosiahneme nasledovne:

a) spojíme záznamy tabuľky nehôd so záznamami štruktúry obsahujúcej polmesiace na základe mesiaca. Z toho vyplýva, že každému záznamu o nehode vzniknú 2 spojenia: s polmesiacom začínajúcim nový mesiac a polmesiacom, ktorý je uprostred mesiaca.

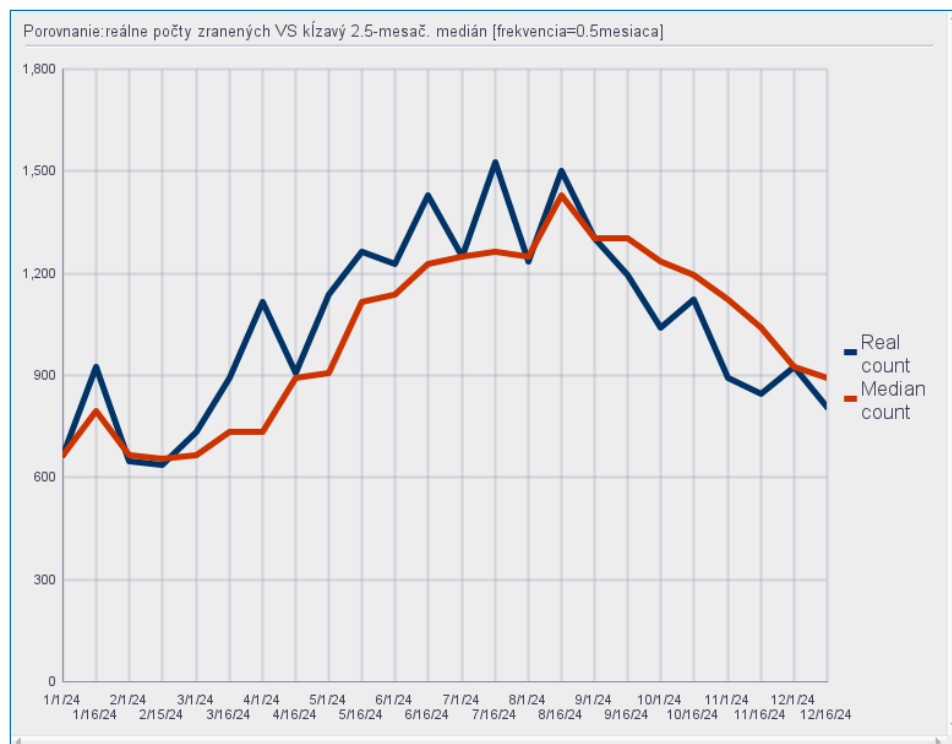
b) teraz je potrebné vybrať 1 z 2 spojení pre každý záznam. Budeme vyberať záznam s tým polmesiacom, po ktorého začiatku nehoda buď:

- 1) bezprostredne nasleduje (nemôže predchádzať, aby bolo jasne vymedzené, kam patrí)
- 2) alebo sa deň nehody zhoduje so začiatkom polmesiaca

Na priradenie polmesiaca k nehode využijeme analytickú funkciu *row_number*, ktorá ohodnotí vhodnosť priradenia daného polmesiaca pre nehodu. Hodnotenie bude len v rámci jednej nehody, t.j. ohodnotenie bude nadobúdať hodnoty 1 a 2. My priradíme ten polmesiac, ktorý dostane ohodnotenie 1. Vhodnosť priradenia polmesiaca určíme nasledovne:

- 1) Ak je deň nehody väčší alebo rovný začiatku polmesiaca, potom vhodnosť priradenia predstavuje rozdiel počtu dní dňa nehody a dňa začiatku polmesiaca
 - 2) Pokiaľ je deň nehody pred dňom začiatku polmesiaca, priradíme mu hodnotu 10000, čo je dostatočne veľká hodnota na to, aby dostala menšiu prioritu, pretože taký veľký počet dní určite nikdy nebude medzi dňom nehody a dňom začiatku polmesiaca
- **Krok 4** – Na začiatku posledného kroku už máme štruktúru, ktorá obsahuje záznamy s polmesiacom a príslúchajúcim počtom zranení pre tento polmesiac. Teraz potrebujeme ku každému polmesiacu nájsť (pokiaľ existuje v záznamoch) chronologicky prvý, druhý, tretí a štvrtý predchádzajúci polmesiac a z neho vytiahnuť informáciu o počte zranených v tom polmesiaci. Na získanie týchto počtov využijeme analytickú funkciu *lag*. Počty zranených z predchádzajúcich polmesiacov pripojíme k základnej množine zjednotením s povolením duplicit a každému záznamu priradíme polmesiac, pre ktorý je hľadaný počet viazaný. Nakoniec spočítame hodnoty mediánov cez agregačnú funkciu *median* pre skupiny definované polmesiacom.

Výsledky vypočítaných mediánov si môžeme zobrazíť v programe *SqlDeveloper* definovaním vlastného reportu (obrázok 24). Modrá krivka ukazuje skutočné počty zranených pre daný polmesiac a červená zobrazuje výpočítaný kľzavý medián, ktorý vyhladzuje priebeh modrej krivky. Môžeme vidieť, že počet zranených počas letných mesiacov je minimálne dvakrát väčší než počty zranených na začiatku roka.



Obrázok 24 – report na porovnanie počtov vyjadrených kľzavým mediánom a skutočným počtom zranených v priebehu roka 2024

Optimalizácia výkonosti

Skúsme sa teraz pozrieť či by sme mohli niektorý z predošlých dotazov rýchlostne zlepšiť vytvorením vhodného indexu.

Index pre skript získania percentuálneho podielu škôd nehôd s prítomnosťou alkoholu v TOP 1000 najvyššie ocenených škodách

Skript z obrázku 18 obsahuje podmienku *where*, v ktorej vyberáme záznamy podľa hľadaného roku.

```

    ) where rn <= 1000 and alko_pritomny='A'
) sum_kc_alko,
(select sum(skoda) suma_kc --v2(aggregate) cost: 763
 from (
  select celk_skoda_kc as skoda,
  row_number() over(order by celk_skoda_kc desc) as rn
  from cr_nehody
  where extract(year from cas)=2024
  ) where rn <= 1000
) sum_kc_celk
from dual);
```

Obrázok 25 – ukážka filtrovacej podmienky príkazu select pre získanie percentuálneho podielu škôd z obrázku 18

Vyskúšame tri varianty:

```

10 create index idx1_btree_nehody_year on cr_nehody(cas); -- nepouziť sa
11 create index idx2_btree_nehody_year on cr_nehody(extract(year from cas)); -- cost: 295
12 create index idx3_btree_nehody_year on cr_nehody(to_number(to_char(cas, 'YYYY'))); -- cost: 295
```

Obrázok 26 - varianty pre index na získanie roku

a) Index nad stĺpcom *cas* (index *idx1_btree_nehody_year* z obrázku 26)

- z časovej pečiatky potrebujeme získať rok, preto nemôžeme použiť index pre stĺpec *cas*
- takýto index vyjadruje index na celkovú hodnotu časovej pečiatky, preto sa nepoužije (obrázok 27), lebo filter záznamov sa robí až na základe výsledku funkcie spracujúcej čas, ktorý sa porovná s hľadanou hodnotou

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	LEVEL
SELECT STATEMENT				1531	0
SORT				1	1
VIEW				1000	2
Filter Predicates					3
AND					4
Filter Predicates					5
WINDOW				1909	6
Filter Predicates					7
ROW_NUMBER() OVER (ORDER BY INTERNAL_FUNCTION(N.CELK_SKODA_KC) DESC) <= 1000					8
HASH JOIN				1909	9
Access Predicates					10
N.ID_ALKO_PRIT=A.ID_STAV					11
TABLE ACCESS	CR_PRITOMNOST_ALKO	FULL	10	3	12
TABLE ACCESS	CR_NEHODY	FULL	1909	762	13
Filter Predicates					14
EXTRACT(YEAR FROM INTERNAL_FUNCTION(N.CAS))=2024					15
SORT				1	16
VIEW				1000	17
Filter Predicates					18
WINDOW				1909	19
Filter Predicates					20
ROW_NUMBER() OVER (ORDER BY INTERNAL_FUNCTION(CELK_SKODA_KC) DESC) <= 1000					21
TABLE ACCESS	CR_NEHODY	FULL	1909	762	22
Filter Predicates					23
EXTRACT(YEAR FROM INTERNAL_FUNCTION(CAS))=2024					24
FAST DUAL				1	25

Obrázok 27 – vytvorenie indexu nad stĺpcom čas nemá vplyv na kritérium filtrovania podmienky podľa hodnoty roku časovej pečiatky po spracovaní funkciou extract

b) Funkcionálny index nad výsledkom funkcie *extract(year from cas)* (index *idx2_btree_nehody_year* z obrázku 26)

- Vytvoríme klasický index, ktorý je implementovaný ako údajová štruktúra B+ strom
- celkové náklady skriptu sa z nákladov 1531 znížili na 295 jednotiek, pretože sa index použil, dokonca sa použil dvakrát – v dvoch vnorených príkazoch *select* (obrázok 28)

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	LEVEL
SELECT STATEMENT				295	0
SORT				1	1
VIEW				1000	2
Filter Predicates					3
AND					4
Filter Predicates					5
WINDOW				1909	6
Filter Predicates					7
ROW_NUMBER() OVER (ORDER BY INTERNAL_FUNCTION(N.CELK_SKODA_KC) DESC) <= 1000					8
HASH JOIN				1909	9
Access Predicates					10
N.ID_ALKO_PRIT=A.ID_STAV					11
TABLE ACCESS	CR_PRITOMNOST_ALKO	FULL	10	3	12
TABLE ACCESS	CR_NEHODY	BY INDEX ROWID BATCHED	1909	144	13
INDEX	IDX2_BTREE_NEHODY_YEAR	RANGE SCAN	764	133	14
Access Predicates					15
N.SYS_NC00035\$=2024					16
SORT				1	17
VIEW				1000	18
Filter Predicates					19
WINDOW				1909	20
Filter Predicates					21
ROW_NUMBER() OVER (ORDER BY INTERNAL_FUNCTION(CELK_SKODA_KC) DESC) <= 1000					22
TABLE ACCESS	CR_NEHODY	BY INDEX ROWID BATCHED	1909	144	23
INDEX	IDX2_BTREE_NEHODY_YEAR	RANGE SCAN	764	133	24
Access Predicates					25
CR_NEHODY.SYS_NC00035\$=2024					26
FAST DUAL				1	27

Obrázok 28 – využitie funkcionálneho indexu *idx2_btree_nehody_year* vo where podmienke *extract(year from cas)*

c) Funkcionálny index nad výsledkom funkcie *to_number(to_char(cas, 'YYYY'))* (index *idx3_btree_nehody_year* z obrázku 26)

- Musíme si dať pozor, aby sme výsledok funkcie *to_char(cas, 'YYYY')* konvertovali na číslo pre správne a dobré fungovanie
- Taktiež sme vytvorili B+ strom index

- Celkové náklady sú rovnaké ako pre variant z predchádzajúceho bodu b) (výsledok použitia variantu c) na obrázku 30)
- Ak by sme tento index chceli použiť, museli by sme upraviť podmienku *where* na tvar, v akom je definovaný index, t.j. *where to_char(cas, 'YYYY')=2024* (príklad na obrázku 29)

```

from cr_nehody
--where extract(year from cas)=2024
where to_number(to_char(cas, 'YYYY'))=2024
) where rn <= 1000
) sum_kc_celk
from dual);

```

Obrázok 29 – zámena podmienky *where* kvôli zmene funkcionálneho indexu

- Variant b) ak c) fungujú, lebo je splnená podmienka, že funkcia je deterministická pre získanie roku

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	LA
SELECT STATEMENT				295	
Sort					
VIEW					
Filter Predicates					
AND					
RN<=1000					
ALKO_PRITOMNY='A'					
WINDOW					
Filter Predicates					
ROW_NUMBER() OVER (ORDER BY INTERNAL_FUNCTION(N.CELK_SKODA_KC) DESC) <= 1000					
HASH JOIN					
Access Predicates					
N.ID_ALKO_PRIT=A.ID_STAV					
TABLE ACCESS	CR_PRITOMNOST_ALKO	FULL	10	3	
TABLE ACCESS	CR_NEHODY	BY INDEX ROWID BATCHED	1909	144	
INDEX	IDX3_BTREE_NEHODY_YEAR	RANGE SCAN	764	133	
Access Predicates					
N.SYS_NC00035\$=2024					
Sort					
VIEW					
Filter Predicates					
RN<=1000					
WINDOW					
Filter Predicates					
ROW_NUMBER() OVER (ORDER BY INTERNAL_FUNCTION(CELK_SKODA_KC) DESC) <= 1000					
TABLE ACCESS	CR_NEHODY	BY INDEX ROWID BATCHED	1909	144	
INDEX	IDX3_BTREE_NEHODY_YEAR	RANGE SCAN	764	133	
Access Predicates					
CR_NEHODY.SYS_NC00035\$=2024					
FAST DUAL					

Obrázok 30 - využitie funkcionálneho indexu *idx3_btree_nehody_year* vo *where* podmienke *to_number(to_char(cas, 'YYYY'))*

Keď si skúsime vypísať všetky možné hodnoty výsledkov funkcie *extract(year from cas)*, dostaneme len toľko rôznych hodnôt koľko rôznych rokov v hodnote stĺpca *cas* nájdeme. My sme importovali iba dáta pre rok 2023, 2024 a 2025 (obrázok 31).

```

1.7 select distinct extract(year from cas) rok -- iba 3 zaznamy: 2023, 2024, 2025
1.9 from cr_nehody;

```

ROK
1 2025
2 2024
3 2023

Obrázok 31 – všetky možné hodnoty roku pre nainportované dáta

Ako vidíme, máme len 3 možnosti a celkový počet záznamov tabuľky s nehodami podľa obrázku 7 okolo 190000. To znamená, že selektivita stĺpca je $100 * (3/190000)$, čo je približne 0.1% a mohli by sme skúsiť použiť bitmapový index, ktorý je ukázaný na obrázku 32.

```

19
20 create bitmap index idx_bitmap_nehody_year on cr_nehody(extract(year from cas)); -- cost: 640
21

```

Obrázok 32 – bitmapový funkcionálny index pre získanie roku z časovej pečiatky premennej cas

Celkové náklady s využitím bitmapového indexu sa ukázali byť vyššie (obrázok 33). Nevýhodou využitia bitmapového indexu v tomto kontexte by bolo, že by do tabuľky nehôd postupne rokmi mohli pribúdať záznamy s časovými pečiatkami z nasledujúcich rokov. Vtedy by bolo nutné vykonať nad indexom rebuild, aby sa pridal do indexu nový stĺpec (nová hodnota roku, ktorú môže index nadobúdať). Preto je v tomto kontexte pre optimalizáciu rýchlosti vykonania najlepšie využiť B+ strom funkcionálny index variantu b) alebo c). Väčšina doteraz ukázaných príkladov obsahovala filter na konkrétny rok, takže by sme tým zrýchlili vykonanie všetkých týchto skriptov.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				640
VIEW				320
WINDOW				320
HASH JOIN				319
TABLE ACCESS	CR_PRITOMNOST_ALKO	FULL	10	3
TABLE ACCESS	CR_NEHODY	BY INDEX ROWID BATCHED	1909	316
BITMAP CONVERSION		TO ROWIDS		
BITMAP INDEX	IDX_BITMAP_NEHODY_YEAR	SINGLE VALUE		
AGGREGATE			1	317
VIEW				317
WINDOW				317
TABLE ACCESS	CR_NEHODY	BY INDEX ROWID BATCHED	1909	316
BITMAP CONVERSION		TO ROWIDS		
BITMAP INDEX	IDX_BITMAP_NEHODY_YEAR	SINGLE VALUE		
FAST DUAL			1	2

Obrázok 33 – využitie bitmapového funkcionálneho indexu pre zrýchlenie filtrovania záznamov podľa roku

Index pre výpis TOP 3 najčastejšie zrazených zvierat za každý mesiac

V tomto skripte (obrázok 22) sa opäť zameriame iba na podmienku *where*, lebo nie sme schopní index definovať pre všetky hodnoty, ktoré sa v príkaze *select* používajú, pretože dochádza k spájaniu tabuliek a vyberanie atribútu z pripojenej tabuľky. V podmienke *where* filtrujeme záznamy podľa typu nehody a roku, v ktorom k nehode došlo.

Vyskúšame najprv použiť bitmapový index *idx_bitmap_neh_druh_nehody* iba na druh nehody (obrázok 34), pretože môže nadobúdať iba 10 jedinečných hodnôt, čo znamená, že selektivita tohto stĺpca v tabuľke s nehodami sa blíži k nule a taktiež nemienime pridávať nové typy nehôd, čo je pre tento typ indexu žiaduce.

```

117
118 select count(distinct id_druh_nehody) from cr_nehody; --10 roznych typov
119
120 create bitmap index idx_bitmap_neh_druh_nehody on cr_nehody(id_druh_nehody);

```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				10
VIEW				10
WINDOW				10
HASH JOIN				10
TABLE ACCESS	CR_PRITOMNOST_ALKO	FULL	10	3
TABLE ACCESS	CR_NEHODY	BY INDEX ROWID BATCHED	1909	316
BITMAP CONVERSION		TO ROWIDS		
BITMAP INDEX	IDX_BITMAP_NEH_DRUH_NEHODY	SINGLE VALUE		
AGGREGATE			1	317
VIEW				317
WINDOW				317
TABLE ACCESS	CR_NEHODY	BY INDEX ROWID BATCHED	1909	316
BITMAP CONVERSION		TO ROWIDS		
BITMAP INDEX	IDX_BITMAP_NEH_DRUH_NEHODY	SINGLE VALUE		
FAST DUAL			1	2

Obrázok 34 - bitmapový index *idx_bitmap_neh_druh_nehody*

Po vykonaní plánu však zistíme, že sa bitmapový index nepoužil a to ani vtedy, keď sme príkazu *select* pridali HINT (obrázok 35).

```

4 select id_mesiac, mesiac, nazov, pocet,
5 rank() over (partition by id_mesiac order by pocet desc) as rnk
6 from (
7 select /*+Index(CR_NEHODY IDX_BITMAP_NEH_DRUH_NEHODY)*/ extract(month from n.cas) as id_mesiac, to_char(n.cas, 'Mon') as mesiac,
8 count(*) as pocet
9 from cr_nehody n
10 join cr_druh_zviera z on n.id_zviera=z.id_druh
11 where id_druh_nehody=5 and extract(year from cas)=2025 -- id=5 pre zrazku so zverou
12 group by z.id_druh, z.nazov, to_char(cas, 'Mon'), extract(month from cas)
13 )
14 where rnk <= 3
15 group by id_mesiac, mesiac

```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				762
GROUP BY			204	762
VIEW			204	761
Filter Predicates				
RANK <= 3				
WINDOW				
Filter Predicates				
RANK() OVER (PARTITION BY EXTRACT(MONTH FROM INTERNAL_FUNCTION(N.CAS)) ORDER BY COUNT(*) DESC) <= 3			204	761
HASH				
MERGE JOIN				
TABLE ACCESS	CR_DRUH_ZVIERA	BY INDEX ROWID	23	2
INDEX	SYS_C00197588	FULL SCAN	23	1
JOIN			204	757
Access Predicates				
N.ID_ZVIERA=Z.ID_DRUH				
Filter Predicates				
N.ID_ZVIERA=Z.ID_DRUH				
TABLE ACCESS	CR_NEHODY	FULL	204	756
Filter Predicates				
AND				
N.ID_DRUH_NEHODY=5				
EXTRACT(YEAR FROM INTERNAL_FUNCTION(N.CAS))=2025				

Obrázok 35 – HINT pre použitie bitmapového indexu

Teraz skúsme vytvoriť klasický B+ strom funkcionálny index nad získaním roku zo stĺpca *cas* (obrázok 36).

```

123 create index idx_btree_neh_year on cr_nehody(extract(year from cas)); -- cost: 150
124

```

Obrázok 36 – funkcionálny index na získanie roku zo stĺpca *cas*

Tento index sa už využil vo vykonanom pláne a rapídne znížil náklady zo 762 jednotiek na 150 (obrázok 37).

```

4 select id_mesiac, mesiac, nazov, pocet,
5 rank() over (partition by id_mesiac order by pocet desc) as rnk
6 from (
7 select /*+Index(CR_NEHODY IDX_BITMAP_NEH_DRUH_NEHODY)*/ extract(month from n.cas) as id_mesiac, to_char(n.cas, 'Mon') as mesiac,
8 count(*) as pocet
9 from cr_nehody n
10 join cr_druh_zviera z on n.id_zviera=z.id_druh
11 where id_druh_nehody=5 and extract(year from cas)=2025 -- id=5 pre zrazku so zverou
12 group by z.id_druh, z.nazov, to_char(cas, 'Mon'), extract(month from cas)
13 )
14 where rnk <= 3
15 group by id_mesiac, mesiac

```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				150
GROUP BY			204	150
VIEW			204	149
Filter Predicates				
RANK <= 3				
WINDOW				
Filter Predicates				
RANK() OVER (PARTITION BY EXTRACT(MONTH FROM INTERNAL_FUNCTION(N.CAS)) ORDER BY COUNT(*) DESC) <= 3			204	149
HASH				
MERGE JOIN				
TABLE ACCESS	CR_DRUH_ZVIERA	BY INDEX ROWID	23	2
INDEX	SYS_C00197588	FULL SCAN	23	1
JOIN			204	145
Access Predicates				
N.ID_ZVIERA=Z.ID_DRUH				
Filter Predicates				
N.ID_ZVIERA=Z.ID_DRUH				
TABLE ACCESS	CR_NEHODY	BY INDEX ROWID BATCHED	204	144
Filter Predicates				
N.ID_DRUH_NEHODY=5				
INDEX	IDX_BTREE_NEH_YEAR	RANGE SCAN	764	133
Access Predicates				
N.SYS_NC00035\$=2025				

Obrázok 37 – využitie funkcionálneho indexu *idx_btree_neh_year*

Môžeme sa teraz ešte pokúsiť znížiť dobu vykonania vytvorením kompozitného funkcionálneho indexu. Index sa bude skladať z extrahovaného roku z časovej pečiatky a druhu nehody (obrázok 38). Vďaka tomu indexu sa nám podarilo zredukovať náklady na 32 jednotiek z pôvodných 762.

```

424 create index idx_btree_nehody_rok_druh on cr_nehody(extract(year from cas), id_druh_nehody); -- cost: 32
425
426

```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				32
SORT				32
VIEW				31
Filter Predicates				
RNK<=3				
WINDOW				204
Filter Predicates				
RANK() OVER (PARTITION BY EXTRACT(MONTH FROM INTERNAL_FUNCTION(N,CAS)) ORDER BY COUNT(*) DESC)<=3				
SORT PUSHED RANK				31
HASH				204
GROUP BY				31
MERGE JOIN				29
TABLE ACCESS	CR_DRUH_ZVIERA	BY INDEX ROWID	23	2
INDEX	SYS_C00197588	FULL SCAN	23	1
JOIN			204	27
SORT				
Access Predicates				
N.ID_ZVIERA=Z.ID_DRUH				
Filter Predicates				
N.ID_ZVIERA=Z.ID_DRUH				
TABLE ACCESS	CR_NEHODY	BY INDEX ROWID BATCHED	204	26
INDEX	IDX_BTREE_NEHODY_ROK_DRUH	RANGE SCAN	81	17
Access Predicates				
AND				
N.SYS_NC00035\$=2025				
N.ID_DRUH_NEHODY=5				

Obrázok 38 – aplikácia kompozitného funkcionálneho indexu pre extrahovaný rok a druh nehody

Optimalizácia spojenia tabuliek nehôd a vozidiel

Na záver vyskúšame zrýchlenie skriptu z obrázku 7 na základe optimalizácie spojenia tabuliek nehôd a vozidiel asociovaných s nehodami. Na obrázku 5 si môžeme všimnúť, že ID nehody je v tabuľke s nehodami primárnym kľúčom. V tabuľke s vozidlami je ID nehody cudzím kľúčom bez žiadneho existujúceho indexu. Bez existujúceho indexu nad cudzím kľúčom v tabuľke vozidiel sa využila metóda spojenia *HASH JOIN* (obrázok 39).

```

36 -- spajanie tabuliek cez FK
37 -- cost: 1793
38 select 'Pocet autonehod za rok:' as popis,
39 sum(case extract(year from cas) when 2023 then 1 else 0 end) as count_2023,
40 sum(case extract(year from cas) when 2024 then 1 else 0 end) as count_2024,
41 sum(case extract(year from cas) when 2025 then 1 else 0 end) as count_2025
42 from cr_nehody
43 UNION ALL
44 select 'Pocet vozidiel za rok:' as popis,
45 sum(case extract(year from cas) when 2023 then 1 else 0 end) as count_2023,
46 sum(case extract(year from cas) when 2024 then 1 else 0 end) as count_2024,
47 sum(case extract(year from cas) when 2025 then 1 else 0 end) as count_2025
48 from cr_vozidla
49 join cr_nehody using(id_nehoda);

```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				1793
UNION-ALL				
SORT				
INDEX	IDX_BTREE_NEH_YEAR	AGGREGATE	1	112
FAST FULL SCAN			190879	
AGGREGATE			1	1681
HASH JOIN				
Access Predicates				
CR_VOZIDLA.ID_NEHODA=CR_NEHODY.ID_NEHODA				
TABLE ACCESS	CR_VOZIDLA	FULL	222927	412
TABLE ACCESS	CR_NEHODY	FULL	190879	754

Obrázok 39 – zvolenie metódy spojenia tabuliek HASH JOIN bez indexu nad cudzím kľúčom

Vytvoríme teda index nad stĺpcom *id_nehoda* v tabuľke *vozidla*. Náklady sa zmenšili približne o 250 jednotiek, pričom sa opäť použil *HASH JOIN* ako metóda spojenia tabuliek.

```

49 join cr_nehody using(id_nehoda);
50
51 create index ind_btree_voz_id_nehoda_fk on cr_vozidla(id_nehoda);
52

```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	LAST_C
SELECT STATEMENT					1541
UNION-ALL					
SORT		AGGREGATE		1	
INDEX	IDX_BTREE_NEH_YEAR	FAST FULL SCAN	190879	112	
SORT		AGGREGATE		1	
HASH JOIN			222927	1429	
Access Predicates	CR_VOZIDLA.ID_NEHODA=CR_NEHODY.ID_NEHODA				
INDEX	IND_BTREE_VOZ_ID_NEHODA_FK	FAST FULL SCAN	222927	160	
TABLE ACCESS	CR_NEHODY	FULL	190879	754	

Obrázok 40 – optimalizácia spojenia tabuliek nehôd a vozidiel pridaním indexu na cudzí kľúč tabuľky vozidiel

Pokusom o zlepšenie pomocou *HINTU* na použitie *MERGE JOIN* nedosiahneme lepší výsledok (obrázok 41). *MERGE JOIN* sme skúsili použiť preto, že obe tabuľky majú nad stĺpcom *id_nehoda* index.

```

from cr_nehody
UNION ALL
select /*+Use_Merge(CR_VOZIDLA IND_BTREE_VOZ_ID_NEHODA_FK)*/ 'Pocet vozidiel za rok:' as popis, -- hint nezohľadňuje
sum(case extract(year from cas) when 2023 then 1 else 0 end) as count_2023,
sum(case extract(year from cas) when 2024 then 1 else 0 end) as count_2024,
sum(case extract(year from cas) when 2025 then 1 else 0 end) as count_2025
from cr_vozidla
join cr_nehody using(id_nehoda);

```

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST	LAST_C
SELECT STATEMENT					1541
UNION-ALL					
SORT		AGGREGATE		1	
INDEX	IDX_BTREE_NEH_YEAR	FAST FULL SCAN	190879	112	
SORT		AGGREGATE		1	
HASH JOIN			222927	1429	
Access Predicates	CR_VOZIDLA.ID_NEHODA=CR_NEHODY.ID_NEHODA				
NESTED LOOPS				222927	1429
STATISTICS COLLECTOR					
INDEX	IND_BTREE_VOZ_ID_NEHODA_FK	FAST FULL SCAN	222927	160	
INDEX	SYS_C00197691	UNIQUE SCAN			
Access Predicates	CR_VOZIDLA.ID_NEHODA=CR_NEHODY.ID_NEHODA				
TABLE ACCESS	CR_NEHODY	BY INDEX ROWID	1	754	
TABLE ACCESS	CR_NEHODY	FULL	190879	754	

Obrázok 41 – pokus o zlepšenie metódy spojenia tabuliek pomocou *HINTU* vykonania spojenia cez *MERGE JOIN*

Záver

V tejto semestrálnej práci sme v prvej časti vykonali niekoľko zaujímavých analýz nad reálnymi dátami z nehôd v Českej republike, z ktorých sme zistili nové informácie o počtoch mŕtvych a zranených, spôsobených škodách, najčastejšie zrazených zvieratách a iné. V týchto analýzach sme využili analytické a agregáčné funkcie, ktoré nám pomohli napísať jednoduchšie konštruované a ľahšie modifikovateľné skripty a často aj rýchlejšie v ich vykonaní. V druhej časti sa nám podarilo zoptimalizovať niektoré skripty využitím vhodných indexov a tak preukázať pochopenie ich účelu. Celou prácou sme sa tak utvrdili vo vedomí, že databázový systém je silný nástroj pre spracovanie veľkého množstva údajov, z ktorých dokážeme správnym postupom získať zaujímavé informácie o dátach a zároveň vieme aj optimalizovať rýchlosť ich získavania.

Zoznam obrázkov

Obrázok 1 – obsah stiahnutej RAR zložky	3
Obrázok 2 – popis hodnôt stĺpcov, na ktoré sa záznamy referencujú	4
Obrázok 3 – ukážka záznamov o nehodách v xls súbore	4
Obrázok 4 – transformácia záznamov do databázy	5
Obrázok 5 – dátový model najpodstatnejších tabuliek zvýraznených oranžovým rámkom.....	5
Obrázok 6 – pregenerovanie štatistík pre aktualizáciu kvôli novým tabuľkám	6
Obrázok 7 – počty nehôd a vozidel za jednotlivé evidované roky	6
Obrázok 8 – najviac mŕtvych pre kraj a mesiac za rok 2024 (variant 1)	7
Obrázok 9 - najviac mŕtvych pre kraj a mesiac za rok 2024 (variant 2)	8
Obrázok 10 - najviac mŕtvych pre kraj a mesiac za rok 2024 (variant 3)	9
Obrázok 11 - TOP 3. mesiac s najviac nehodami v 2024 (variant 1)	10
Obrázok 12 - TOP 3. mesiac s najviac nehodami v 2024 (variant 2)	10
Obrázok 13 - TOP 3. mesiac s najviac nehodami v 2024 (variant 3)	11
Obrázok 14 – ročné počty nehôd podľa prítomnosti alkoholu v krvi	12
Obrázok 15 – podiely počtov nehôd podľa zistenia obsahu alkoholu v krvi pre rok 2024	12
Obrázok 16 – mesačné počty nehôd podľa prítomnosti alkoholu v krvi	13
Obrázok 17 – abstrakcia výpočtu percentuálneho podielu pre hľadajú skutočnosť v jazyku SQL	14
Obrázok 18 – dve možnosti spočítania sumy TOP 1000 najvyšších škôd roku 2024	14
Obrázok 19 - dve možnosti spočítania sumy škôd s obsahom alkoholu spomedzi TOP 1000 roku 2024 ..	15
Obrázok 20 – výsledný skript pre získanie perc. podielu škôd s prít. alkoholu na TOP 1000 škodách	16
Obrázok 21 – report s podielom nehôd s prítomnosťou alkoholu k ostatným spomedzi TOP 1000 podľa najvyšších nákladov v Kč z roku 2024	16
Obrázok 22 – výpis troch najčastejšie zrazených zvierat pre každý mesiac v roku 2024	17
Obrázok 23 – skript pre 2,5 mesačný kľzavý median počtu zranených pre rok 2024.....	18
Obrázok 24 – report na porovnanie počtov vyjadrených kľzavým mediánom a skutočným počtom zranených v priebehu roka 2024	19
Obrázok 25 – ukážka filtrovacej podmienky príkazu select pre získanie percentuálneho podielu škôd z obrázku 18.....	20
Obrázok 26 - varianty pre index na získanie roku	20
Obrázok 27 – vytvorenie indexu nad stĺpom cas nemá vplyv na kritérium filtrovania podmienky podľa hodnoty roku časovej pečiatky po spracovaní funkciou extract.....	21
Obrázok 28 – využitie funkcionálneho indexu idx2_btree_nehody_year vo where podmienke extract(year from cas).....	21
Obrázok 29 – zámena podmienky where kvôli zmene funkcionálneho indexu	22
Obrázok 30 - využitie funkcionálneho indexu idx3_btree_nehody_year vo where podmienke to_number(to_char(cas, 'YYYY'))	22
Obrázok 31 – všetky možné hodnoty roku pre nainportované dáta	22
Obrázok 32 – bitmapový funkcionálny index pre získanie roku z časovej pečiatky premennej cas	23
Obrázok 33 – využitie bitmapového funkcionálneho indexu pre zrýchlenie filtrovania záznamov podľa roku	23
Obrázok 34 - bitmapový index idx_bitmap_neh_druh_nehody	23
Obrázok 35 – HINT pre použitie bitmapového indexu.....	24
Obrázok 36 – funkcionálny index na získanie roku zo stĺpca cas	24

Obrázok 37 – využitie funkcionálneho indexu idx_btree_neh_year	24
Obrázok 38 – aplikácia kompozitného funkcionálneho indexu pre extrahovaný rok a druh nehody	25
Obrázok 39 – zvolenie metódy spojenia tabuliek HASH JOIN bez indexu nad cudzím kľúčom	25
Obrázok 40 – optimalizácia spojenia tabuliek nehôd a vozidiel pridaním indexu na cudzí kľúč tabuľky vozidiel	26
Obrázok 41 – pokus o zlepšenie metódy spojenia tabuliek pomocou HINTU vykonania spojenia cez MERGE JOIN	26