DJ2 – rekurze v SQL

slajdy k přednášce NDBI001

Jaroslav Pokorný

Obsah

Rekurze v SQL

- Intuitivně: dotaz je *rekurzívní*, pokud je použit ve své vlastní definici
- tato vazba může být jak přímá, tak i přes několik tabulek
- Výhody: v jistých případech jediný efektivní způsob získání výsledku
- Nevýhody: často horší čitelnost a srozumitelnost

Kde použít rekurzi SQL

- efektivní pro jakákoliv data s hierarchickou strukturou
  - vztahy ve stromových strukturách
  - hledání v cyklických i acyklických grafech
- příklady z praxe
  - hledání spojů v jízdních řádech
  - organizační struktura firmy
  - struktura výrobku
  - složky v systému správy dokumentů atd...

Lze se obejít bez rekurze

- SQL před standardem SQL:99 neobsahoval možnost konstrukce rekurzívních dotazů
- neprocedurální řešení: s přidáním jistých „grafových informací“
- procedurální řešení: použití kurzorů, cyklů,

  Jiné: ORACLE: proprietární řešení + PL/SQL,
- ztráta efektivnosti a optimalizace
  - kód není tak „elegantní“

Aplikace rekurze

- Pro procházení grafem získáváme:
  - dosažitelnost

    D1. Najdi všechny podřízené dané osoby
  - vyčíslitelnost cest

    D2. Pro daný výrobek najdi jeho celou strukturu se všemi jejími součástkami
  - spojování cest

    D3. Pro daný výrobek proveď výčet a najdi počet všech součástek, potřebných pro jeho sestavení

Další výhody a nevýhody rekurze

- výhody:
  - vše jedním dotazem
  - lze využít velkou část výsledku
- nevýhody
  - časté využití pouze malé části výsledku
  - možnost zacyklení rekurze

Obsah

Common Table Expression

- jde o zobecnění tabulkového výrazu z SQL:92
- deklaruje se klíčovým slovem WITH
- slouží jako náhrada ve vnořených dotazech
- ze SELECT, INSERT, UPDATE, DELETE
- části za WITH jsou volány právě jednou

```
WITH [RECURSIVE] CTE [, CTE]…
CTE ::=jméno_CTE[(jméno_sl[,jméno_sl]…)] AS
        (CTE_definice_dotazu)
```

Příspěvky(ID, fórum, otázka)

Skládání agregací – bez CTE

D4: Najdi fórum s nevyšším počtem příspěvků

```
SELECT  COUNT(ID) AS počet, fórum
FROM  Příspěvky
GROUP BY  fórum
HAVING  COUNT(ID) = (
      SELECT  MAX(počet)
      FROM (SELECT  COUNT(ID) AS počet, fórum
                  FROM   Příspěvky
                  GROUP BY  fórum)
```

Pz.: Hledáme vlastně MAX(COUNT(...))

Skládání agregací – s CTE

```
WITH
   Počet_příspěvků(počet, fórum)
   AS ( SELECT COUNT(ID), fórum)
                FROM Příspěvky
                GROUP BY fórum )
SELECT počet, fórum
   FROM Počet_příspěvků
   WHERE počet = (SELECT MAX(počet)
                        FROM Počet_příspěvků)
```

Více CTE v jednom dotazu

```
WITH
   Počet_příspěvků(počet, fórum)
   AS (SELECT COUNT(ID), fórum
         FROM Příspěvky
         GROUP BY fórum ),
   Max_počet_příspěvků(počet)
   AS (SELECT MAX(počet)
         FROM Počet_příspěvků)
SELECT P1.*
   FROM Počet_příspěvků P1 INNER JOIN
            Max_počet_příspěvků P2  ON
                                  P1.počet = P2.počet
```

Pz.: CTE fungují stejně jako odvozené tabulky (dané pomocí SELECT za FROM)

Přechod k rekurzi

č_zam	jméno	funkce	č_nad
1	Novák	ředitel	NULL
2	Srb	náměstek	1
3	Lomský	vedoucí	2
4	Bor	vedoucí	2

D5.

```
WITH Nadřízení(jméno, č_nad, č_zam) AS
   (SELECT jméno, č_nad, č_zam
   FROM Zaměstnanci
   WHERE funkce = 'vedoucí'
   )
SELECT * FROM Nadřízení
```

jméno	č_nad	č_zam
Lomský	2	3
Bor	2	4

Rekurzívní dotazy

- V CTE pro tabulku R se lze odkazovat na R
- vytvoří se dočasná tabulka (existuje pouze v době vyhodnocování dotazu)
- Tři části

  WITH

  ukotvení (inicializační poddotaz)

  UNION ALL

  rekurzívní člen

  - rekurze běží pokud není přidán žádný další záznam anebo není překročený limit rekurze (MAXRECURSION)
  - pozor na zacyklení rekurzívního členu

  SELECT

  - Vnější SELECT - dá výsledek dotazu

Příklad

ukotvení: spouští se jednou

rekurzívní člen: opakovaně

spojení s minulým krokem

výstup

```
WITH RECURSIVE Nadřízení(jméno, č_nad, č_zam) AS
(SELECT jméno, č_nad, č_zam
        FROM Zaměstnanci
        WHERE jméno = 'Nový'
        UNION ALL
SELECT Z.jméno, Z.č_nad, Z.č_zam
        FROM Zaměstnanci AS Z
                INNER JOIN
                Nadřízení AS N
                ON N.č_nad = Z.č_zam)
SELECT * FROM Nadřízení
```

jméno	č_nad	č_zam
Nový	11	13
Ryba	6	11
Rak	5	6
Syka	4	5
Bor	2	4
Srb	1	2
Novák	NULL	1

Co řešíme?

Příklad

D6.: Najdi všechny nadřízené Nového (včetně něho sama)

ukotvení: spouští se jednou

rekurzívní člen: opakovaně

spojení s minulým krokem

výstup

```
WITH RECURSIVE Nadřízení(jméno, č_nad, č_zam) AS
(SELECT jméno, č_nad, č_zam
        FROM Zaměstnanci
        WHERE jméno = 'Nový'
        UNION ALL
SELECT Z.jméno, Z.č_nad, Z.č_zam
        FROM Zaměstnanci AS Z
        INNER JOIN
        Nadřízení AS N
        ON N.č_nad = Z.č_zam)
SELECT * FROM Nadřízení
```

jméno	č_nad	č_zam
Nový	11	13
Ryba	6	11
Rak	5	6
Syka	4	5
Bor	2	4
Srb	1	2
Novák	NULL	1

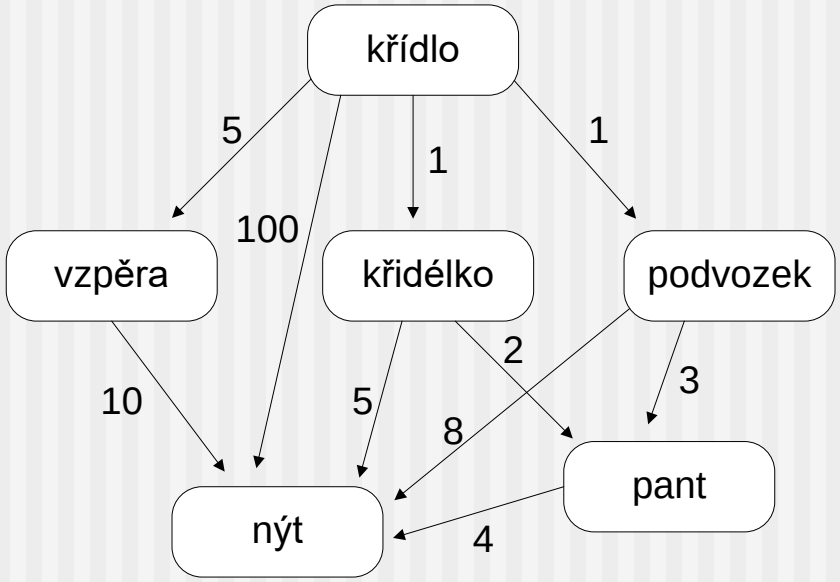
Omezení rekurzívních dotazů

- V ukotvení se nelze odkazovat na CTE
- Rekurzívní část vždy samo-odkazuje na CTE
  - SQL:99 podporuje pouze "lineární" rekurzi: každé FROM má nejvýše jeden odkaz na rekurzívně definovanou relaci.
- Rekurzívní část nemůže obsahovat
  - SELECT DISTINCT
  - GROUP BY
  - HAVING
  - skalární agregace
  - TOP
  - OUTER JOIN
- každý sloupec rekurzívního poddotazu musí být typově kompatibilní s příslušným sloupcem v inicializačním poddotazu
  - používá se přetypování – CAST

Obsah

Počítaní v rekurzi

D7. Jaké součástky (včetně počtu) jsou potřeba pro výrobu křídla



Počítaní v rekurzi

zjednodušené uložení v DB (relace Komponenty) s počtem jednotlivých součástek v daném dílu

Díl	**ČástDílu**	**Množství**
křídlo	vzpěra	5
křídlo	křidélko	1
křídlo	podvozek	1
křídlo	nýt	100
vzpěra	nýt	10
křidélko	pant	2
křidélko	nýt	5
podvozek	pant	3
podvozek	nýt	8
pant	nýt	4

Počítaní v rekurzi – dotazy

D8. Kolik nýtů je použito celkem na výrobu jednoho křídla ?

D9. Seznam všech součástek včetně počtu potřebných na výrobu jednoho křídla ?

Počítaní v rekurzi – řešení

- Na co nesmíme zapomenout ?
  - základem je použití rekurze (průchod grafem)
  - musíme sečíst počet nýtů v jednotlivých součástkách
  - a zároveň musíme uvažovat počty jednotlivých součástek

Počítaní v rekurzi – D8

- CTE

```
WITH RECURSIVE
    ČástiKřídla(ČástDílu, Množství)
    AS
  (( SELECT ČástDílu, Množství
    FROM Komponenty
    WHERE Díl = 'křídlo' )
   UNION ALL
   ( SELECT k.ČástDílu, č.Množství
     * k.Množství
    FROM ČástiKřídla č,
    Komponenty k
    WHERE č.ČástDílu = k.Díl ));
```

[inicializační poddotaz]

[rekurzívní poddotaz]

- výsledek

ČástDílu	Množství	
vzpěra	5	*přímé použití*
křidélko	1	
podvozek	1	
nýt	100	
nýt	50	*z vzpěry*
pant	2	*z křidélka*
nýt	5	*z křidélka*
pant	3	*z podvozku*
nýt	8	*z podvozku*
nýt	8	*z pantu křidélka*
nýt	12	*z pantu podvozku*

Počítaní v rekurzi – D8

- a nakonec sečteme jednotlivý počty

```
WITH RECURSIVE ČástiKřídla(ČástDílu, Množství) AS
  (( SELECT ČástDílu, Množství
     FROM Komponenty
     WHERE Díl = 'křídlo' )
    UNION ALL
    ( SELECT k.ČástDílu, č.Množství * K.Množství
      FROM ČástiKřídla č, Komponenty k
      WHERE č.ČástDílu = k.Díl ))
SELECT sum(Množství) AS Množství
FROM ČástiKřídla
WHERE ČástDílu = 'nýt';
```

Výsledek

Množství
183

Počítaní v rekurzi – D9

- Pro řešení D9 stačí změnit finální dotaz

```
WITH RECURSIVE ČástiKřídla(ČástDílu, Množství) AS
  (( SELECT ČástDílu, Množství
     FROM Komponenty
     WHERE Díl = 'křídlo' )
    UNION ALL
   ( SELECT K.ČástDílu, Č.Množství * K.Množství
     FROM ČástiKřídla Č, Komponenty K
     WHERE Č.ČástDílu = K.Díl ))
SELECT ČástDílu, sum(Množství) AS Množství
FROM ČástiKřídla
GROUP BY ČástDílu;
```

Výsledek

ČástDílu	Množství
vzpěra	5
křidélko	1
podvozek	1
pant	5
nýt	183

Syntaxe průchodu stromů v Oracle 9i

SELECT *sloupce* FROM *tabulka*
[WHERE *podmínka3*]
START WITH *podmínka1*
CONNECT BY *podmínka2*
[ORDER BY …]

- Řádky vyhovující podmínce ve START WITH jsou považovány za kořenové řádky na první úrovni vnoření
- Pro každý řádek na úrovni *i* se rekurzivně hledají přímí potomci vyhovující podmínce v klauzuli CONNECT BY na úrovni *i*+1
  - Řádek předka se v podmínce označuje klíčovým slovem PRIOR

Syntaxe průchodu stromů v Oracle 9i

- Na závěr jsou odstraněny řádky nevyhovující podmínce ve WHERE
- Pokud není definováno třídění, odpovídá pořadí průchodu pre-order
- Každý řádek obsahuje pseudosloupec LEVEL, obsahující úroveň řádku v hierarchii

Zam(č_zam, jméno, vedoucí)

Oracle 9i vs. SQL:99

vkládá mezery v počtu 2*Level

- Oracle 9i:

```
SELECT LPAD(' ', 2*Level) || jméno, Level
FROM Zam
START WITH vedoucí IS NULL
CONNECT BY vedoucí = PRIOR č_zam;
```

- SQL:99

```
WITH RECURSIVE Zam1 AS (
    SELECT x.jméno AS jméno, 0 AS Level
    FROM Zam x WHERE vedoucí IS NULL
  UNION ALL
    SELECT y.jméno, Level+1
    FROM Zam y JOIN Zam1 ON y.vedoucí =
            Zam1.č_zam)
   SELECT * FROM Zam1;
```

Oracle 9i vs. SQL:99

Efekt funkce LPAD

Data
Novák
Srb
Lomský
Bor

Obsah

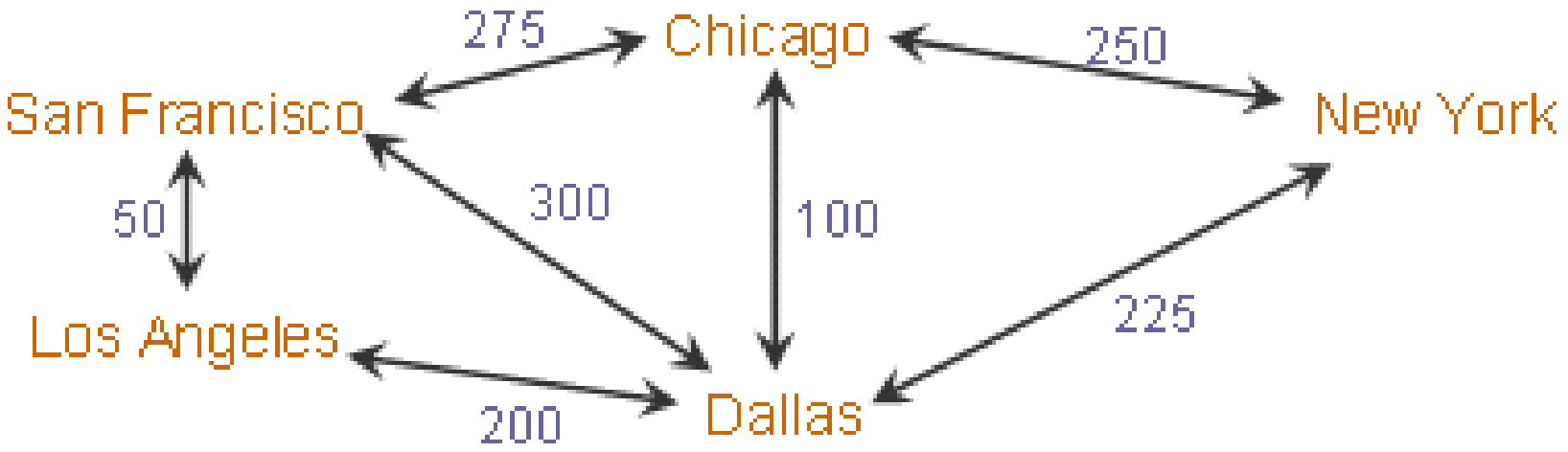
Rekurzívní vyhledávání

- Snaha nalézt nejlepší řešení na základě jistých kritérii daného problému.
- Příklad:

  uvažujme letištní systém odletů a klienta, který by rád odcestoval ze San Francisca do New Yorku za co nejnižší cenu

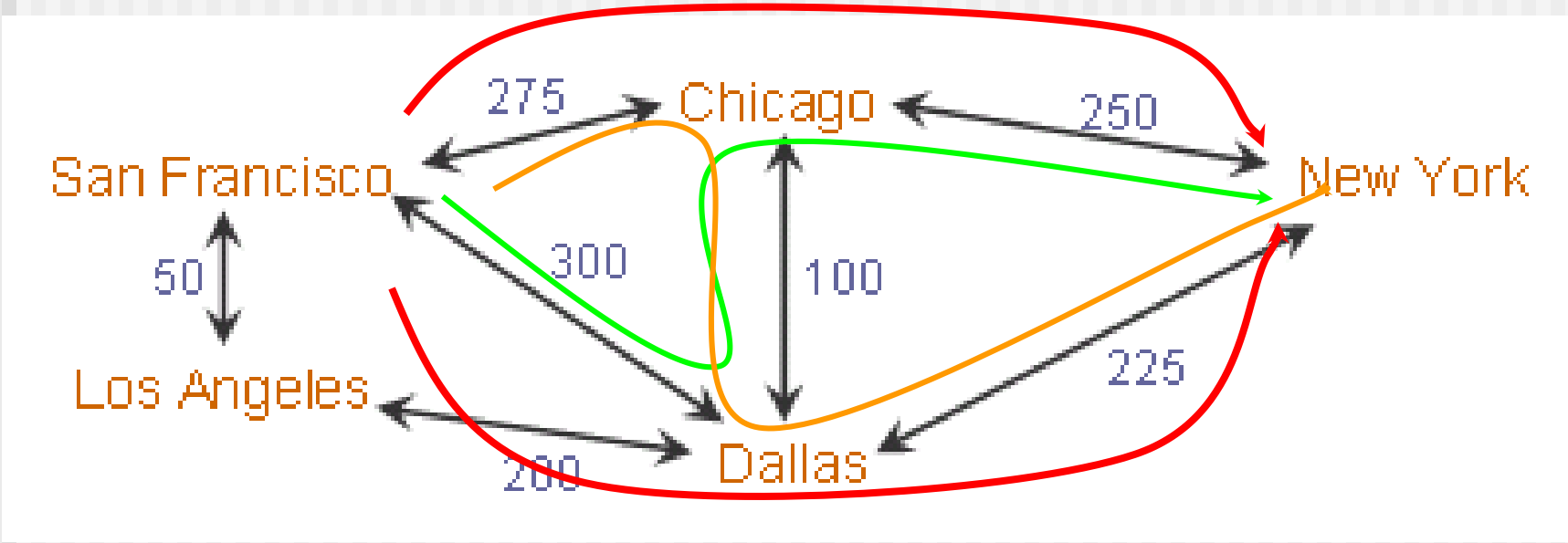
Rekurzívní vyhledávání – příklad

- letová mapa :



Rekurzívní vyhledávání – příklad

- některé možnosti letu:



Rekurzívní vyhledávání – příklad

- definice tabulky Lety

číslo_l	start	cíl	cena
xxx01	SF	CHG	275
xxx02	SF	DLS	300
…			

D10. Najdi pro San Francisko trasu do New Yorku za nejnižší cenu.

- Problém: jelikož mapa není acyklická, musíme vyřešit konec rekurze

Rekurzívní vyhledávání – 1. řešení

- Dočasnou tabulku použitou v CTE nazveme Výlety
  - jako ukotvení zvolíme poddotaz, který najde všechna města, do kterých se klient dostane ze San Francisca přímo
  - rekurzívní poddotaz bude hledat města, kam se klient dostane z již dosažených měst

Rekurzívní vyhledávání – 1. řešení

```
WITH RECURSIVE Výlety (cíl, trasa, celková_cena) AS
    ((SELECT cíl, cíl, cena
      FROM Lety
      WHERE start = 'SF' )
UNION ALL
    (SELECT l.cíl
      v.trasa || ',' || l.cíl, v.celková_cena + l.cena
    FROM Výlety v, Lety l
    WHERE v.cíl = l.start))
SELECT trasa, celková_cena
FROM výlety
WHERE cíl = 'NY';
```

Kde je chyba?

- do sloupce trasa vkládáme stále větší výraz
- zacyklení

Rekurzívní vyhledávání – 1. řešení + oprava

- porušení pravidla, že hodnota v sloupci rekurzívního pododotazu nesmí být delší v odpovídajícím sloupci inicializačního poddotazu (ukotvení)

  Řešení:
  - změníme datový typ u obou poddotazů (inicializační i rekurzívní) na VARCHAR(50)
  - k tomu slouží CAST výraz
- funkce CAST

  Příklady:

  CAST (c1 + c2 AS Decimal(8,2))

  CAST (jméno||adresa AS Varchar(255))

  string
  - delší je doplněn mezerami
  - kratší se uřízne a vrátí upozornění

CAST (výraz AS typ_dat)

Rekurzívní vyhledávání – 1. řešení + oprava

- problém zacyklení

  Řešení:

  - nebudeme brát v úvahu lety z počátku, tedy ze San Francisca
  - nebudeme brát v úvahu lety z cíle, tedy z New Yorku
  - a zajímají nás pouze lety, které mají maximálně 2 úseky

Rekurzívní vyhledávání – finální řešení

```
WITH RECURSIVE Výlety (cíl, trasa, #úsek, celková_cena) AS
  ((SELECT cíl,  CAST(cíl AS Varchar(50)), 1,  cena
     FROM Lety
     WHERE start = 'SF'
UNION ALL
   (SELECT l.cíl, CAST(v.trasa || ',' || l.cíl AS Varchar(50)), v. #úsek + 1,
     v.celková_cena + l.cena
    FROM Výlety v, Lety l
    WHERE v.cíl = l.start
     AND l.cíl <> 'SF'
     AND l.start <> 'NY'
     AND v. #úsek < 2))
SELECT trasa, celková_cena
FROM Výlety
WHERE cíl = 'NY '  AND celková_cena=(SELECT min(celková_cena)
                                    FROM Výlety
                                    WHERE cíl='NY');
```

Výsledek

trasa	celková_cena
DLS, NY	525
CHG,NY	525

Obsah

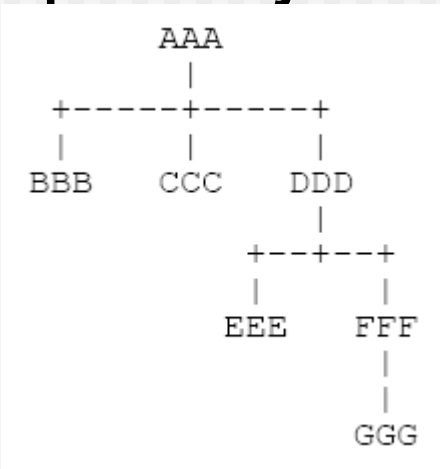
Klasifikace hierarchií

- podle grafových vlastností
  - Konvergentní
  - Divergentní
  - Rekurzívní
- podle vyváženosti
  - Vyvážené
    - všecky listy na stejné úrovni
    - na každé úrovni jiné objekty (např. geografická struktura)
  - Nevyvážené
    - listy na různých úrovních
    - jednotné objekty (např. organizační struktura)
- Problém: reprezentace pomocí relací

Divergentní hierarchie

- každý uzel kromě kořene má právě jednoho rodiče

  Př.: geografické hierarchie
  - země, stát, město, ulice
- implementace
  - Hrana (PKEY, KEYO)
  - Primární klíč KEYO
  - Tabulka s referenční integritou PKEY$\subseteq$ KEYO



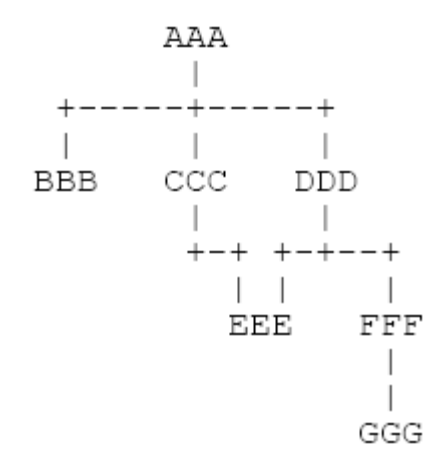
KEYO	PKEY	NUM	PRICE
AAA			$10
BBB	AAA	1	$21
CCC	AAA	5	$23
DDD	AAA	20	$25
EEE	DDD	44	$33
FFF	DDD	5	$34
GGG	FFF	5	$44

Konvergentní hierarchie

- Každý objekt může mít libovolně předků a potomků

  Př.: Oddělení firmy
- Třeba definovat výsledky dotazu

  D11. Kolik potomků má “AAA”?
  - 6, 7, 8?
- implementace
  - tabulka objektů
  - tabulka vztahů



OBJECTS

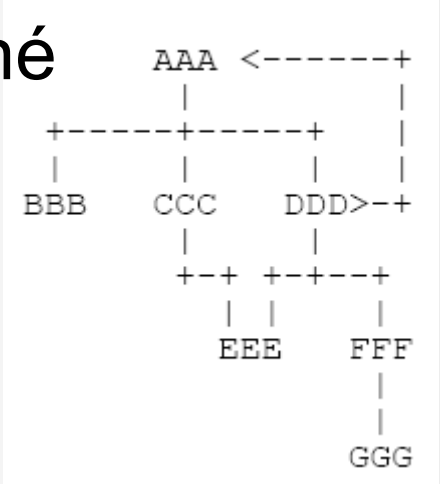
KEYO	PRICE
AAA	$10
BBB	$21
CCC	$23
DDD	$25
EEE	$33
FFF	$34
GGG	$44

RELATIONSHIPS

PKEY	CKEY	NUM
AAA	BBB	1
AAA	CCC	5
AAA	DDD	20
CCC	EEE	33
DDD	EEE	44
DDD	FFF	5
FFF	GGG	5

Rekurzívní hierarchie

- jako konvergentní
  - Navíc: rodič může být potomkem sebe sama (přímo nebo nepřímo)
  - Příklad: nadřízeny-podřízený vs. vedoucí grantu a ředitel jako řešitel
- způsobují zacyklení
- v praxi je její použití většinou chybné
- implementace
  - jako konvergentní (obyčejně)



Obsah

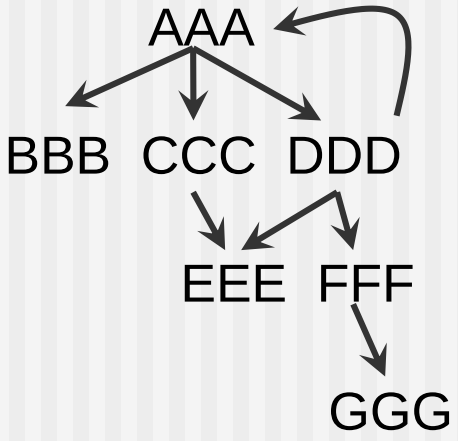
Zastaveni rekurze

- Jak odstranit zacyklení v rekurzívních hierarchiích?
- Možnosti zastavení
  - DB Server
    - V MS SQL 2005 po dosažení hodnoty MAXRECURSION (default 100)
  - po dosáhnutí n-té úrovně
  - pamatovat si cestu a vynechat navštívené uzly

Problém – rekurzívní hierarchie

Tabulka RH

PKEY	CKEY
AAA	BBB
AAA	CCC
AAA	DDD
CCC	EEE
DDD	AAA
DDD	FFF
DDD	EEE
FFF	GGG



D12. Najdi potomky AAA do úrovně 4

Zastavení po dosažení n-té úrovně (atribut LVL)

```
WITH RECURSIVE RODIČ(CKEY, LVL) AS
(SELECT DISTINCT PKEY, 0
        FROM RH
        WHERE PKEY = 'AAA'
 UNION ALL
 SELECT T.CKEY, R.LVL+1
        FROM RH H,RODIČ R
        WHERE R.CKEY = H.PKEY
        AND R.LVL + 1 < 4
)
SELECT CKEY, LVL
FROM RODIČ;
```

N = 4

	CKEY	LVL	
1	AAA	0	
2	BBB	1	
3	CCC	1	
4	DDD	1	
5	AAA	2	⇦
6	EEE	2	
7	FFF	2	
8	GGG	3	
9	BBB	3	⇦
10	CCC	3	⇦
11	DDD	3	⇦
12	EEE	2	

- co s duplicitami ve výsledku?

Vynechání duplicit (pomocí 2 CTE)

```
WITH RECURSIVE RODIČ(CKEY, LVL) AS
(SELECT DISTINCT PKEY, 0
            FROM RH
            WHERE PKEY = 'AAA'
UNION ALL
SELECT H.CKEY, R.LVL+1
            FROM RH H, RODIČ R
            WHERE R.CKEY = H.PKEY
            AND R.LVL + 1 <4
),
BEZ_DUPL(CKEY, LVL, NUM) AS
(SELECT CKEY, MIN(LVL), COUNT(*)
FROM RODIČ
GROUP BY CKEY)

SELECT CKEY, LVL, NUM FROM BEZ_DUPL
```

	CKEY	LVL	NUM
1	AAA	0	2
2	BBB	1	2
3	CCC	1	2
4	DDD	1	2
5	EEE	2	2
6	FFF	2	1
7	GGG	3	1

Ignorování již navštívených

```
WITH RODIČ (CKEY, LVL, CESTA) AS
(SELECT DISTINCT PKEY, 0, VARCHAR(PKEY, 20)
        FROM RH
        WHERE PKEY = 'AAA'
 UNION ALL
 SELECT H.CKEY, R.LVL + 1,
      R.CESTA || '>' || H.CKEY
   FROM RH H, RODIČ R
   WHERE R.CKEY = H.PKEY
   AND
   LOCATE(H.CKEY || '>', R.CESTA) = 0
)
SELECT CKEY, LVL, CESTA
FROM RODIČ;
```

dá pozici vzoru v řetězci

Výsledek		
CKEY	LVL	CESTA
AAA	0	AAA
BBB	1	AAA>BBB
CCC	1	AAA>CCC
DDD	1	AAA>DDD
EEE	2	AAA>CCC>EEE
EEE	2	AAA>DDD>EEE
FFF	2	AAA>DDD>FFF
GGG	3	AAA>DDD>FFF>GGG

Zásobník vs. rekurze

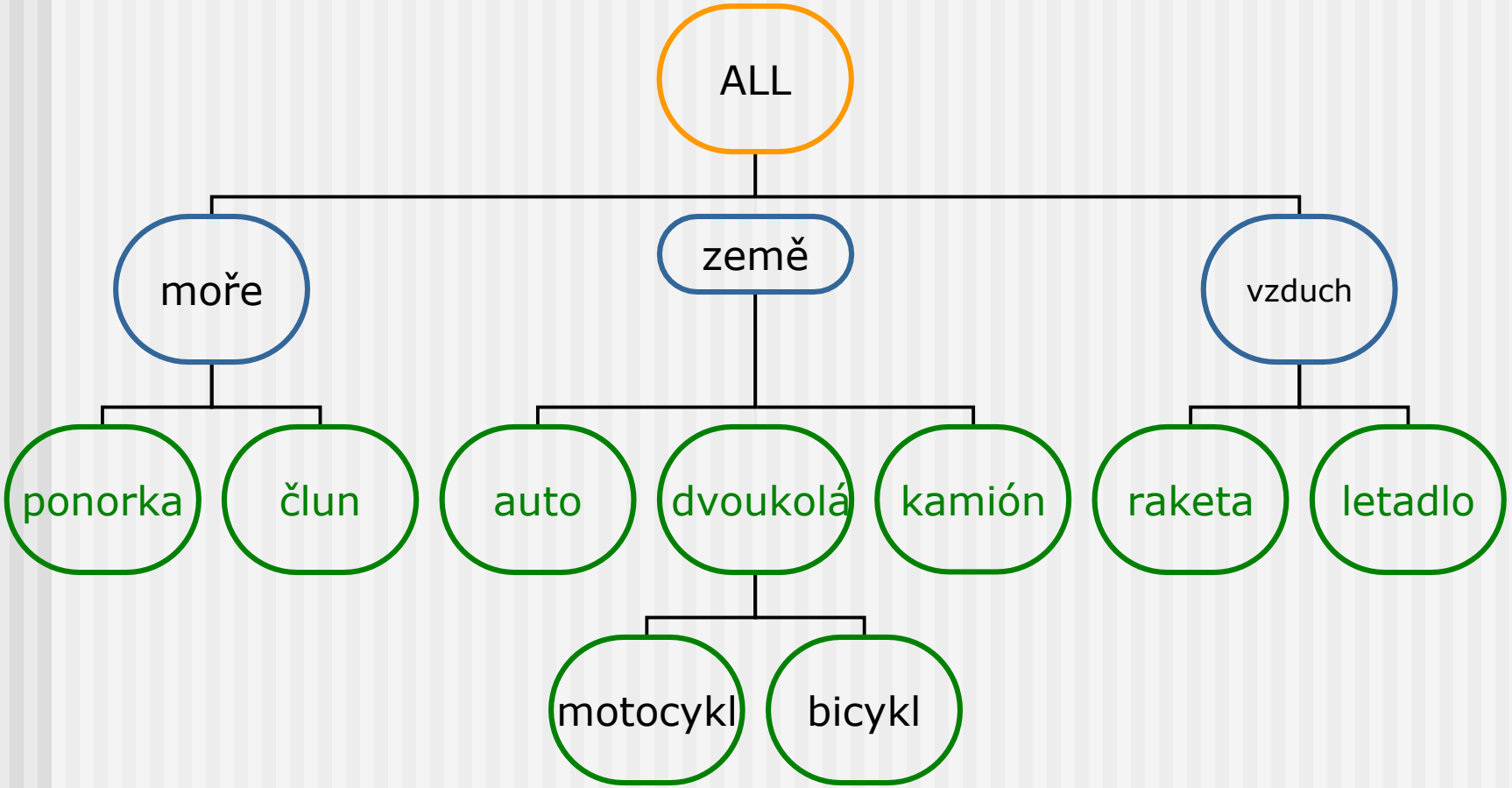
- Problém: jak efektivně implementovat rekurzi – opakování spojení může vést k tomu, že se věci počítají opakovaně
- Rekurzi lze simulovat také pomocí zásobníku
- Zásobníkový model je rychlejší než CTE
  - Dá se použít jen pro dotazovaní na hierarchická data

Vozidla(Id, rodičID, jméno)

Příklad

Id	rodičID	jméno
1	NULL	ALL
2	1	moře
3	1	země
4	1	vzduch
5	2	ponorka
6	2	člun
7	3	auto
8	3	dvoukolá
9	3	kamión
10	4	raketa
11	4	letadlo
12	8	motocykl
13	8	bicykl

Příklad


ALL
moře
země
vzduch
ponorka
člun
auto
dvoukolá
kamión
raketa
letadlo
motocykl
bicykl

Předchůdci bez rekurze (1)

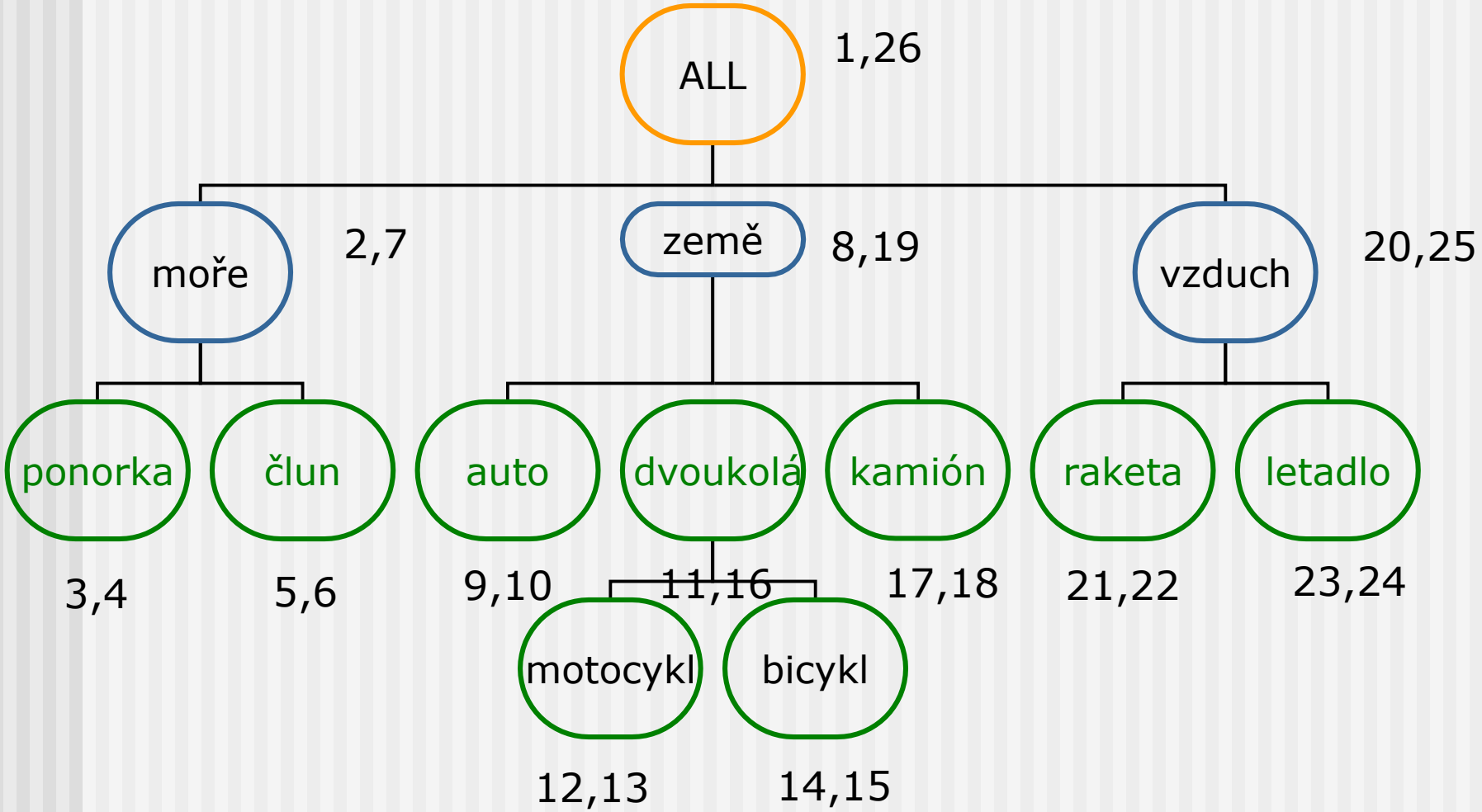
- Dá se rekurze odstranit? ANO, pomocí zásobníku.
- Do tabulky přidáme 2 nové sloupce: P_hranice a L_hranice
- Jejich hodnoty vycházejí z očíslování, které vznikne průchodem stromu,

Předchůdci bez rekurze (2)

Tabulku naplníme daty, pro nové sloupce

```
UPDATE Vozidla SET L_hranice = 1 , P_hranice = 26
   WHERE ID = 1
UPDATE Vozidla SET L_hranice = 2 , P_hranice = 7 WHERE
   ID = 2
…
UPDATE Vozidla SET L_hranice = 12 , P_hranice = 13
   WHERE ID = 12
UPDATE Vozidla SET L_hranice = 14 , P_hranice = 14
   WHERE ID = 13
```

Předchůdci - bez rekurze (3)



Příklad

Id	rodičID	jméno	L_hranice	P_hranice
1	NULL	ALL	1	26
2	1	moře	2	7
3	1	země	8	19
4	1	vzduch	20	25
5	2	ponorka	3	4
6	2	člun	5	6
7	3	auto	9	10
8	3	dvoukolá	11	16
9	3	kamión	17	18
10	4	raketa	21	22
11	4	letadlo	23	24
12	8	motocykl	12	13
13	8	bicykl	14	15

Předchůdci - bez rekurze (4)

Dotaz na předchůdce motocyklu využije intervalů.

```
SELECT *
FROM Vozidla
WHERE P_hranice > 12
   AND L_hranice < 13
```

Příklad

Id	rodičID	jméno	L_hranice	P_hranice
1	NULL	ALL	1	26
2	1	moře	2	7
3	1	země	8	19
4	1	vzduch	20	25
5	2	ponorka	3	4
6	2	člun	5	6
7	3	auto	9	10
8	3	dvoukolá	11	16
9	3	kamión	17	18
10	4	raketa	21	22
11	4	letadlo	23	24
12	8	motocykl	12	13
13	8	bicykl	14	15