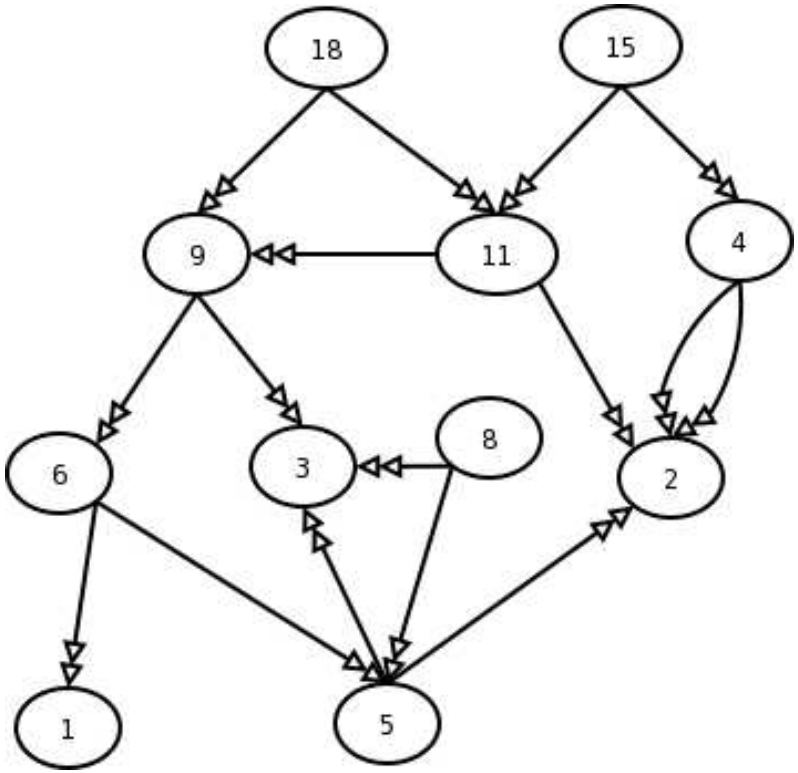
Hierarchické dotazy

- Pochádzajú z Oraclu
- Patch do PostgreSQL
- Podobný mechanizmus v SAP DB *Recursive DECLARE CURSOR*
- Ako to vyzerá:

```
SELECT name FROM employees
CONNECT BY PRIOR employee_id = manager_id
START WITH employee_id = employee_id
```

Co to robí

- Ak máme tabulku obsahujúcu hierarchické dáta, vieme si ich vypísat v poradí v akom sú v danej hierarchí.
- Presnejsie je to priechod orientovaným grafom bez cyklov do hlbky.



V tabulke

	18
18	9
18	11
	15
15	11
15	4
11	9
11	2
4	2
4	2
9	6
⋮	⋮

SQL dotaz

SELECT potomok **FROM** strom2
START WITH rodic **IS NULL**
CONNECT BY rodic = **PRIOR** potomok

potomok	_level_
15	1
11	2
9	3
6	4
⋮	⋮
2	3
4	2
2	3
2	3

Obecná syntax

SELECT a1, ... **FROM** t1, ...
START WITH condition1
CONNECT BY PRIOR condition2
WHERE condition3
HAVING condition4

START WITH Definuje podminku na vyber korenových riadkov

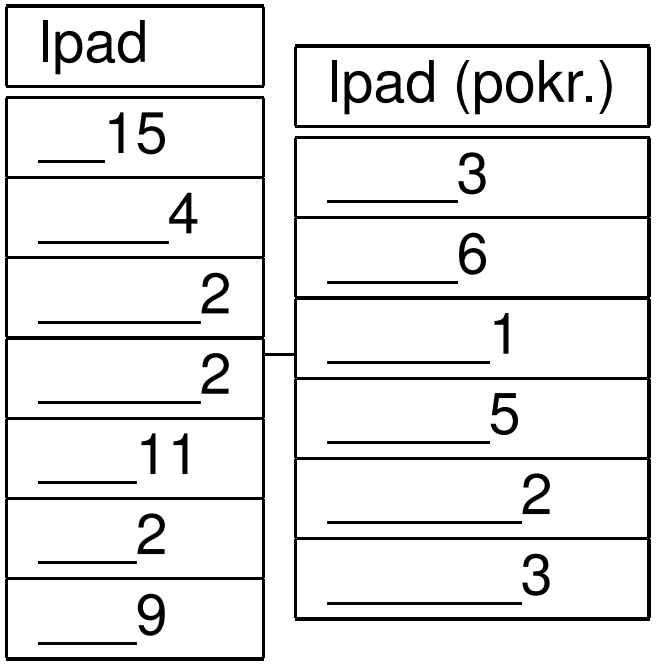
CONNECT BY Definuje podmienku na spájanie rodicov s potomkami.

- Typicky sa pouzíva operator "=", ale je mozné pouzit akýkolvek relacny operator
- Iny ako operator "=" sa vsak neodporúca, moze sposobit zacyklenie
- Moze obsahovat aj zlozené podmienky
- Jedna podminka MUSI obsahovat slovo **PRIOR**
- Nesmie obsahovat *subquery*

GROUP BY Nesmie obsahovat, nedavalo by to zmysel

Príklad

```
SELECT lpad(potomok, _level_ + 3, '_')
FROM (SELECT potomok FROM strom2
        START WITH rodic IS NULL
        CONNECT BY rodic = PRIOR potomok
        ORDER BY potomok) AS T;
```



Dalsi príklad

. . . **CONNECT BY** name != ’King’
AND PRIOR employe_id = manager_id . . .

Ako to funguje (Oracle)

1. Ak dotaz obsahuje v klauzuli **WHERE** *join predikát*, tak sa v tomto kroku pouzije
2. Nájdu sa korenové riadky (**START WITH**)
3. Nájdu sa potomkovia korenových riadkov (**CONNECT BY**)
4. Rekurzívne (do hlbky) hladáme potomkov pre riadky z bodu 3 (**CONNECT BY**)
5. Ak klauzula **WHERE** neobsahuje *join predikát*, tak sa pouzije teraz
6. Vráti sa výsledok zotriedený v poradí do hlbky

Oracle

- Sposob vyhodnocovania **WHERE**
  1. Ak predikát obsahuje *join predikát* pouzije sa predtým ako sa spraví **CONNECT BY**
  2. Ak neobsahuje *join predikát*, vsetky predikáty okrem **CONNECT BY** sa aplikujú po **CONNECT BY**
- Dotaz nesmie obsahovat **ORDER BY**, existuje varianta **ORDER SIBLINGS BY**, ktorá triedi súrodencov v grafe (potomkov jedneho rodica)
- Pouziva pseudoatribút **LEVEL**, ktorý pre kazdý riadok obsahuje stupen zanorenia pri priechode do hlbky. Pocita sa od 1 (korene grafu).
- Ak klauzula **WHERE** obsahuje podmienku typu **IN** na jej lavej strane sa nesmie nachádzat pseudoatribút **LEVEL**. Co s tym? Obalit dotaz do *subquery*.

Oracle 2

- V hierarchickom dotaze je mozné pouzit funkciu **SYS_CONNECT_BY_PATH**(*stlpec*, *znak*), ktorá vracia cestu k danému riadku od korena. Komponenty sú oddelené znakom *znak*. *stlpec* a *znak* mozu byt typu CHAR, VARCHAR2, NCHAR, NVARCHAR2 a výsledok je typu VARCHAR2 a má rovnakú znakovú sadu ako *stlpec*.
- **SELECT** LPAD(' ', 2*level-1) ||
  SYS_CONNECT_BY_PATH(last_name, '/') "Path"
  FROM employees
  START WITH last_name = 'Kochhar'
  CONNECT BY PRIOR employee_id = manager_id

/Kochhar
/Kochhar/Greenberg
/Kochhar/Greenberg/Faviet
/Kochhar/Greenberg/Chen
/Kochhar/Higgins
/Kochhar/Higgins/Gietz

PostgreSQL

- Podporuje základný tvar hierarchického dotazu . . . **CONNECT BY** . . . **START WITH** . . .
- Podporuje pseudoatribút **_LEVEL_**.
  - Vzdy sa pripojí k výsledku ako stlpec.
  - Zacína od 1.
  - Od verzie 0.4 (tohoto patchu) je mozné ho odkazovat a aliasovat aj v *target liste* (zozname atribútov príkazu **SELECT**). (Mne to nefungovalo).
- Moze sa pouzívat **ORDER BY**.
  - Funguje ako **ORDER SIBLINGS BY** v Oracle.
  - Podla dokumentácie by sa nemalo triedit podla pseudoatribútu **_LEVEL_** v dotaze v ktorom je **CONNECT BY**.
- Podmienka vo **WHERE** sa vyhodnocuje vzdy pred vyhodnotením **CONNECT BY**. Ak je potrebné vyhodnotit nejakú podmienku potom, pouzi **HAVING**.

PostgreSQL 2

- V target liste je mozné pouzit slovo **PRIOR**, ktoré odkazuje na hodnotu stlpca z rodicovského riadku. Pouzíva sa napríklad na generovanie identifikátorov alebo odkazovanie nan.

```
SELECT nextval('tseq') AS id ,
       PRIOR nextval('tseq') as pnt, c_id, c_pnt, _level_
FROM data
CONNECT BY PRIOR c_id = c_pnt
START WITH c_id='top'
```

Koniec

Otázky?