elib::vir — VIR lekérdezés- és kimutatásmotor

Áttekintés

Az `include/elib/vir.pm` egy **deklaratív SQL-lekérdezés generátor / OLAP réteg** a VIR (riportok, lekérdezések, kimutatások) számára. A modulok metaadatban leírják, milyen táblák, mezők és dimenziók érhetők el, a motor pedig ebből rak össze `SELECT … FROM … JOIN … WHERE … GROUP BY` lekérdezéseket — opcionálisan crosstab (pivot) táblával.

A `elib::vir` **absztrakt ősosztály**: az `_init` alapból `die „Absztrakt”`-tal áll meg, a leszármazottak (pl. `dok::szamla::vir`, `dok::bank::vir` — 19+ modulnak van saját `vir.pm`-je) töltik fel a `$self→{meta}`-t és a `$self→{lists}`-et.

A fájl négy package-et tartalmaz:

Package	Szerep
`elib::vir`	Absztrakt ős: query-generálás a metaadatból
`elib::vir::generated`	Futásidőben összerakott vir-objektum + relációs-algebra API (filter/project/aggr/join)
`elib::vir::table`	Tábla- (vagy al-lekérdezés-) leíró, ami beépíthető a `generated`-be
`elib::vir::join`	UNUSED, csak váz

Metaadat-modell

Minden mező egy **csoportba** (`groupname`) tartozik — ez a tengelyek és mértékek tipológiája:

Csoport	Szerep	Példa (számla)
`id`	egyedi azonosító mezők	`szlafej.id`, `szlaszam`, `tetelid`
`partition`	dimenzió / bontási tengely	`tipus`, `partner`, `cikk_kategoria`
`time`	idő-dimenzió	`datum`, `teljdat`, `fizhat`
`value`	mérték (számolható)	`ossz_netto`, `ossz_afa`
`check`	logikai szűrőmező	—

A `meta→{tables}` írja le a táblákat és **hogyan joinolnak** egymáshoz:

- `primary` — az elsődleges (FROM-) tábla (pontosan egy)
- `jointo` — melyik már bejoinolt táblához csatlakozik
- `joinon` / `joinusing` — a join-feltétel
- `join_type` — LEFT (alapértelmezés) / INNER / …
- `tablecalc` — opcionális callback, ami al-lekérdezést generál tábla helyett

A `$self→{lists}` **névvel ellátott nézetek**, amelyek a fenti mezőkből választanak ki egy adott listához használt halmazt (pl. `ertekesitesek_aggr`).

A generálási pipeline

new → _init → _generate_trivials

Konstrukciókor a `_generate_trivials` automatikusan legyárt **minden `time` mezőből** `_month` / `_year` / `_year_month` partíciókat (`substr(datum::text, …)`-szal), és minden `id`-ból egy `_view` partíciót. A

havi/éves bontás így „ingyen” elérhető. (A `date_project` metódussal egyedi idő-partíció is definiálható.)

generate_query_and_fields($listname, $alter)

Ez a motor szíve. Az `$alter` paraméter mezőnként vezérli, mi történjen az adott mezővel.

`process_field_group` végigmegy az `id / partition / time / value / check` csoportokon, és mezőnként dönt:

`$alter` érték	Jelentés				
`1`	sima oszlop				
`group`	GROUP BY tengely				
`sum` / `min` / `max` / `textcat_all`	aggregátum				
`count`	darabszám				
`split_day\`	`week\`	`month\`	`quarter\`	`year`	`date_trunc`-os időbontás

Segédmetódusok:

- `_add_field` — az SQL-kifejezés (`sum(…) AS mezo`)
- `_add_list_field` — a megjelenítendő oszlop metaadata (label, type, hidden, listformat); a label végét lokalizált szóval toldja meg (pl. „összesen”, „havi bontás”)
- `_group_field` — a GROUP BY kifejezés

`calc_tables` az `$alter→{where}` és a tábla-gráf alapján építi a `FROM … JOIN …` részt. Iteratív: amíg van olyan tábla, aminek a `jointo`-ja már be van joinolva, hozzáfűzi. Van egy 100-as `limit` (a szerzők saját HACK SMELL jelölésével) a végtelen ciklus ellen.

Visszatérés: `{ query, fields, debug }`.

crosstable

Ha `$alter→{crosstable}` meg van adva, a query köré PostgreSQL `crosstab(…)` hívást rak (`tablefunc` extension), és dinamikusan generálja az oszlopdefiníciókat egy „columns” segéd-query eredményéből. Ez a pivot/kimutatás.

Belépési pontok a hívóknak

Metódus	Visszatérés
`generate_query`	nyers SQL string
`generate_query_for_list`	lista-felülethez illeszkedő struktúra: `Tables` / `Mezok` / `AllKeys`
`generate_vir`	jelenleg `undef` (csonk)

elib::vir::generated — kompozíciós algebra

A `elib::vir::generated` (`@ISA=('elib::vir')`) egy **futásidőben, dinamikusan összerakott** vir-objektum, ami relációs-algebra-szerű műveleteket ad. Minden művelet **új** `generated` példányt ad vissza (immutábilis, láncolható):

Metódus	Művelet
`filter(todo⇒'keep'|'drop', fields⇒[…])`	mezők szűrése
`project(…)`	új származtatott mezők hozzáadása

`aggr($todo, $where)`	aggregálás: GROUP BY-ra állítja a mezőket, a `value`-kat összegezhetővé teszi
`where($where)`	szűrőfeltétel (az `aggr`-ra épül)
`left_join_table` / `inner_join_table` / `gen_join_table`	két vir-objektum összejoinolása (`vir`⇒, `joinon`⇒, `jointo`⇒, `rename_left/right`⇒)
`revir` / `get_table` / `get_meta`	nézetből visszatabularizált meta vagy `elib::vir::table` (al-lekérdezésként újra felhasználható)

A `state` mező (`meta` / `aggr`) és az `_assert` őrzi, hogy a műveletek csak a megengedett állapotban fussanak (csak `$SYS{devel}` alatt aktív).

A `_add_meta` egyesíti több forrás metaadatát (a join-okhoz), `rename`-mel a mezőnév-, `field`-, `label`- és `label_prefix`-ütközések feloldására.

elib::vir::table — táblaleíró

Egy tábla (vagy al-lekérdezés `expr`⇒-rel) leírója, `describe(…)`-szal feltöltve:

- `_tablify` — minden mezőre rányomja a tábla-aliast (`as_table` vagy `table`)
- `get_meta` — olyan meta-struktúrát ad, amit a `generated` be tud kebelezni
- `merge_desc` — descriptor adatszerkezetek mergelése (copy-paste elkerülésére)

Ez teszi lehetővé, hogy egy korábban generált query-t al-lekérdezésként egy új vir építőelemeként használjunk.

Példa: dok::szamla::vir

A `dok::szamla::vir::_init` tölti fel a metaadatot:

```
$meta->{tables}={
    szlafej => { table=>'szlafej', primary=>1 },
    szlatet => { table=>'szlatet', jointo=>'szlafej',
                 joinon=>'szlafej.szlaszam=szlatet.szlaszam', type=>'1N' },
    ...
};
$meta->{id}        = { id=>{...}, szlaszam=>{...}, tetelid=>{...} };
$meta->{partition} = { tipus=>{...}, partner=>{...}, cikk_kategoria=>{...} };
$meta->{time}      = { datum=>{...}, teljdat=>{...}, fizhat=>{...} };
$meta->{value}     = { ossz_netto=>{ field=>'szlafej.ossz_netto',
                                     me=>'szlafej.penznem', ... }, ... };
$self->{lists}     = { ertekesitesek_aggr=>{ tables=>[...], id=>[...],
                       partition=>[...], time=>[...], value=>[...] }, ... };
```

Megjegyzések

A kód tele van a fejlesztők saját SMELL / HACK jelöléseivel és kikommentált debug-sorokkal (`#debug(…); use devel;`). Ezek a szerzők kétségei, nem hibák, de jelzik, hogy egyes részek (pl. a `calc_tables` limit-hack, a `where` az `aggr`-ra hackelve) tudottan ideiglenesek.

Generálva kódelemzésből — `include/elib/vir.pm` (r22049, 2025-04-30).