Przewodnik wdrożeniowca > Tworzenie raportów w SQL

Do tworzenia zaawansowanych raportów SQL przydatna będzie oprócz PGAdmina również dokumentacja bazy danych. Znajduje się ona w sekcji download jako plik DokumentacjaBazyDanycheDokumenty.zip.

Instrukcja tworzenia nowego raportu

1. Przechodzimy do eDokumentów, moduł Raporty > Nowy raport (min. Nazwa, Grupa) - zapisujemy, otwierają się dodatkowe zakładki
1. W zakładce Definicja wpisujemy Kwerendę SQL "SELECT * FROM events" (UWAGA! Zapytanie musi zwracać przynajmniej 1 rekord)
1. Zapisujemy, Otwieramy do edycji ponownie. Na zakładce Wybór kolumn przenosimy klikając w przyciski te pola które chcemy wyświetlić.
1. Zapisujemy raport.
1. Otwieramy raport aby obejrzeć wyniki

W kwerendach można stosować parametry

```
{USR_ID} (string) - przecinkami rozdzielona lista użytkowników do którym ma dostęp zalogowany user
{LOGGED_USR_ID} (int) - usr_id aktualnie zalogowanego pracownika
{DATE_FROM} (string) - np. adddat::date >= '{DATE_FROM}'
{DATE_TO} (string) -np. adddat:: <= '{DATE_TO}'
{CONTID} (int) - id kontatku
{PRC_ID} (int) - id sprawy
{CONTIDS} (string) - id wybranych kontaktów z listy kontaktów dla których ma zostać wydrukowany wybrany raport
```

będą one mapowane na formularzu "Parametry raportu" i z odpowiednich pól pobierane będą wartości.

Konfiguracja automatycznych raportów

Jeżeli chcielibyśmy otrzymywać regularnie jakiś raport "na biurko" mamy taką możliwość poprzez funkcjonalność planowania raportów. W efekcie np. raz w tygodniu w piątek otrzymamy raport wysłanych ofert w formacie PDF na naszą skrzynkę dokumentów.

Dowolny raport można dodać do zaplanowanych na zakładce "Planowanie" formularza edycji raportu.

Ponieważ można dodać tylko jedno zadanie dla jednego raportu dlatego nie jest możliwa edycja zadania. Zawsze nowo-wstawiony rekord nadpisze poprzedni.

Aby raporty wykonywane były automatycznie należy upewnić się że skrypt CronRunner.php jest dodany do zaplanowanych zadań systemu operacyjnego.

W systemie Linux można dodać go poprzez skopiowanie pliku cronrunner do katalogu /etc/cron.d/ lub poprzez edycję pliku /etc/cron.d/crontab.

```
*/1 * * * * www-data cd /home/edokumenty/public_html/apps/edokumenty && php -f CronRunner.php >> /var/log/cronrunner.log
```

Ponieważ skrypt tworzący plik PDF generuje dużo ostrzeżeń, jeżeli nie chcemy debugować skryptu nie jest zalecane przekierowywanie wyników działania do maila (opcja MAILTO powinna być wyłączona)

Użyteczne konstrukcje i funkcje językowe SQL

```
-- Zwraca sformatowaną kwotę
SELECT bmoney(100.90, 'PLN')

-- Formatuje datę
SELECT to_char(d.adddat, 'YYYY-MM-DD') FROM documents d;

-- Pobiera rok
select extract(YEAR from CURRENT_DATE);

-- Formatuje kwotę z pól tekstowych np. z formularzy customowych
select cast(regexp_replace(regexp_replace('301 110,43',' ',''),',', '.') as numeric(12,2)) + 12.50;
```

```
-- Warunkowo koloruje pole
CASE WHEN p.pr_sta = 1 THEN '<div style="color: red">Rozpoczęte</div>' WHEN p.pr_sta = 2 THEN 'W trakcie realizacji' WHEN

-- Selectuje tydzień
to_char(CURRENT_DATE, 'WW') = to_char (adddat, 'WW')
```

Przykładowe użyteczne zapytania do bazy edokumenty

```
--
-- Pobranie danych z formularzy dynamicznych
--
SELECT to_char(d.adddat, 'YY-MM-DD') AS day,
-- rozmowy
(SELECT count(*) /10
FROM events_view e WHERE e.trmtyp = 'PHONECALL' AND emp_id IN (84,62) AND e.start_ >= d.adddat::date - 7
AND e.start_ <= d.adddat) AS rozmowy_handlowe,
-- spotkania
(SELECT count(*)
FROM events_view e WHERE e.trmtyp = 'MEETING' AND emp_id IN (84,62) AND e.start_ >= d.adddat::date - 7
AND e.start_ <= d.adddat) AS spotkania,
-- wartość pola featid 98
f3.data__::int AS odwedok
FROM documents d
INNER JOIN features_text_view f3 ON d.doc_id = f3.tbl_id AND f3.featid = 98;
-- cecha z listy wyboru
INNER JOIN features_opt_view f3 ON d.is_del IS NOT true AND d.gostof IS NULL AND d.doc_id = f3.tbl_id AND f3.ftopid = 119

--
-- Wybiera symbol teczki z numeru sprawy oo formacie ''2/03/08/UP/AW''
--
SELECT substring(symbol from '[0-9]*/[0-9]*/[0-9]*/([A-Z]*)') FROm processes;

--
-- Wybiera krótką nazwę klienta jeśli jest, a jeśli jest pusta to długą
--
SELECT COALESCE(c.name_2, c.name_1) FROM contacts c;

--
-- Wybiera sprawy z wartościami cech (tekstowe i opcje)
--
SELECT p.dscrpt, p.symbol, 'PROCESS'::text AS clsnam, prc_id AS keyval,
f1.data__ AS opinia, f2.ftopnm AS reklamacja, f3.data__ AS dzialanie_korygujace,
f4.data__ AS dzialanie_Klient, f5.data__ AS przyczyna
FROM processes_view p
LEFT JOIN features_text_view f1 ON p.prc_id = f1.tbl_id AND f1.featid = 14
LEFT JOIN features_opt_view f2 ON p.prc_id = f2.tbl_id AND f2.featid = 15
LEFT JOIN features_text_view f3 ON p.prc_id = f3.tbl_id AND f3.featid = 16
LEFT JOIN features_text_view f4 ON p.prc_id = f4.tbl_id AND f4.featid = 17
LEFT JOIN features_text_view f5 ON p.prc_id = f5.tbl_id AND f5.featid = 18
WHERE p.prtpid = 1

--
-- Wybiera dane do trendu - do wykresu
--
SELECT extract(month from rlstrt) as m,
sum((time__::numeric(12,2)/3600)::numeric(12,2)) AS sum
FROM rcp_cards_view
WHERE rlstrt IS NOT NULL AND is_fak = TRUE AND is_del = FALSE AND tpstid = 9
AND rlstrt + interval '1 year' >= CURRENT_DATE
```

```
GROUP BY extract(year from rlstrt),
extract(month from rlstrt)
ORDER BY extract(year from rlstrt), extract(month from rlstrt)
```

Umożliwienie otwierania dialogów (formularzy) z wyników raportu

Tworzymy zapytanie które w kolumnach o nazwach *clsnam* i *keyval* będą zawierać odpowiednio NAZWĘ_FORMULARZA i wartość klucza podstawowego rekordu np.

```
SELECT
'PROCESS' AS clsnam,
prc_id AS keyval,
dscrpt FROM processes;
```

Następnie na zakładce *Definicja* raportu wpisujemy aliasy tych pól.

- Typ z pola: clsnam
- ID z pola: keyval

Dostępne formularze i ich przykładowe klucze podstawowe:

```
PROCESS    - prc_id   - Kartoteka sprawy
CONTACT    - contid   - Kartoteka klienta
DOCUMENT   - doc_id   - Formularz dowolnego typu dokumentu
EVENT      - evntid   - Formularz dowolnego typu zdarzenia ??
RCP        - rcp_id   - Formularz karty pracy
```

Wykresy

Dostępna jest możliwość drukowania wykresów, należy jednak odpowiednio sformułować zapytanie, tak aby wyniki możliwe były do wyświetlenia na wykresie: słupkowym, liniowym i kołowym (Pie).

Aby wydrukować wykres kołowy jego definicja musi być tak ułożona, aby pierwsza kolumna wskazywała na opis a druga na wartość !!

Menu raportów

Wybrane raporty można udostępnić w formularzu sprawy lub klienta. Będą dostępne na dole kartoteki pod przyciskiem *Raporty*.


Rys. 1

Aby dodać raport można użyć przycisków *Dodaj raport* (Rys. 1), można również to zrobić w Ustawieniach. W tym celu należy wybrać Ustawienia -> Menu raportów. W polu "Nazwa dialogu" wybieramy nazwę formatki do której chcemy dodać raport najczęściej będą to kartoteka klienta lub kartoteka sprawy. W liście wyboru ustawiamy interesujący nas raport. W definicji wywoływanego w ten sposób raportu używamy odpowiednio parametru {CONTID} albo {PRC_ID} - aby raport użył otwartej kartoteki jako parametru.

Biblioteka raportów

Raporty z biblioteki można pobierać poprzez przeglądarkę lub klienta webdav z serwera support.

Autoryzowani partnerzy mogą również współtworzyć raporty, uzyskując dostęp do biblioteki poprzez SVN. Polecany klient TortoiseSVN.

Na razie repozytorium dostępne jest pod adresem *https://edokumenty.beta:444/svn/repos/Wdrozenia* Przykładowa konfiguracja dla tunelu z deva na lokalny port 44443. Hasła takie jak do traca.

```
ssh -N -f -L 44443:localhost:44444 tunnel@dev.bnet.pl
svn co https://localhost:44443/svn/repos/Wdrozenia
```

Każdy katalog zawierać może jeden raport, każdy raport reprezentowany musi być przez co najmniej 3 pliki.

- *Zestawienie spotkan.report* - nazwa opisowa
- *Spotkania.sql* - może zawierać również dodatkowe kwerendy ale wyraźnie oddzielone od właściwej oraz komentarze
- *Spotkania.png* - screenshot dla łatwiejszej orientacji
- opcjonalnie pliki HTML dla raportów z szablonami np. *szablonAudytu.html*

Tips & Tricks

Dla łatwiejszego tworzenia raportów można użyć narzędzia PgAdmin, wówczas dla sieci lokalnej konfiguracja pg_hba.conf powinna wyglądać dla sieci w której serwer ma adres przykładowo:

```
[root@edokumenty ~]# ip a | grep eth0
inet 10.8.16.33/24 brd 10.8.16.255 scope global eth0

# TYPE  DATABASE    USER        CIDR-ADDRESS          METHOD
host    oblig       all         10.8.16.255/24        trust
```

W postgresql.conf należy ustawić nasłuchiwanie na wszystkich interfejsach sieciowych.

```
listen_addresses = '*'
```

Po skończeniu wdrożenia koniecznie przywrócić do pierwotnej postaci.