Дополнения для WEB интерфейса

Автоматическое заполнение полей "Номер контракта" и "Дата контракта"

Не заполненные поля «Номер контракта» и «Дата контракта» заполняются автоматически системой. Полю «Номер контракта» присваивается следующий порядковый номер, а полу «Дата контракта» текущая дата.

Для включения данной возможности нужно создать триггер для MySQL (Данная возможность работает начиная с версии MySQL 5.0).

```
DELIMITER //
CREATE TRIGGER next_contract_id BEFORE INSERT ON users_pi
FOR EACH ROW BEGIN
  DECLARE new_id INTEGER;

  SET new_id=1;

  IF NEW.contract_id = '' THEN
       SELECT CAST(contract_id AS UNSIGNED)+1 INTO new_id
         FROM users_pi WHERE contract_sufix=NEW.contract_sufix
         ORDER BY 1 DESC  LIMIT 1;
       SET NEW.contract_id=new_id;

  END IF;

  IF NEW.contract_date = '' or NEW.contract_date = '0000-00-00' THEN
    SET NEW.contract_date=curdate();
  END IF;

END;
//
DELIMITER ;
```

```
DELIMITER //
CREATE TRIGGER next_contract_id_change BEFORE UPDATE ON users_pi
FOR EACH ROW BEGIN

  DECLARE new_id INTEGER;
  DECLARE old_contract_sufix CHAR;

  IF NEW.contract_id = '' THEN
    SET new_id=1;

    SELECT contract_sufix INTO old_contract_sufix
         FROM users_pi WHERE uid=NEW.uid;
    SELECT CAST(contract_id AS UNSIGNED)+1 INTO new_id
       FROM users_pi WHERE contract_sufix=NEW.contract_sufix
       ORDER BY 1 DESC  LIMIT 1;
    SET NEW.contract_id=new_id;

    IF NEW.contract_date = '' or NEW.contract_date = '0000-00-00' THEN
      SET NEW.contract_date=curdate();
    END IF;
  END IF;
END
//
DELIMITER ;
```

Автосоздание счета или квитанции при пополнении счёта.

Автосоздание счета или квитанции при пополнении счёта для платежей через терминалы.

```
DELIMITER //
CREATE TRIGGER add_docs AFTER INSERT ON payments

FOR EACH ROW BEGIN

  DECLARE payment_id   INTEGER;
  DECLARE payment_sum  INTEGER;
  DECLARE admin_id     INTEGER;
  DECLARE next_acct_id INTEGER;
  DECLARE doc_id       INTEGER;
  DECLARE next_invoice_id INTEGER;

  SET payment_id  = NEW.id;
  SET payment_sum = NEW.sum;
  SET admin_id    = NEW.aid;
  SET next_acct_id= 0;
  SET doc_id      = 0;
  SET next_invoice_id= 0;


  IF NEW.method > 40 THEN
    SELECT if(max(acct_id) IS NULL, 0, max(acct_id))+1 INTO next_acct_id
      FROM docs_acct WHERE DATE_FORMAT(date,'%Y')=DATE_FORMAT(curdate(), '%Y');

    INSERT INTO docs_acct (acct_id, date, created, customer, phone, aid, uid, payment_id)
      values (next_acct_id, now(), now(), '', '', admin_id, NEW.uid, payment_id);
    SET doc_id      = LAST_INSERT_ID();
    INSERT INTO docs_acct_orders (acct_id, orders, counts, unit, price)
         values (doc_id, NEW.dsc,  1, 0, payment_sum);


    SELECT if(max(invoice_id) IS NULL, 0, max(invoice_id))+1 INTO next_invoice_id
      FROM docs_invoice WHERE DATE_FORMAT(date, '%Y')=DATE_FORMAT(curdate(), '%Y');

    INSERT INTO docs_invoice (invoice_id, date, created, customer, phone, aid, uid, payment_id)
      values (next_invoice_id, now(), now(), '', '', admin_id, NEW.uid, payment_id);


    SET doc_id      = LAST_INSERT_ID();

    INSERT INTO docs_invoice_orders (invoice_id, orders, counts, unit, price)
      values (doc_id, NEW.dsc, 1, 0, payment_sum);

  END IF;

END;
//
DELIMITER ;
```

Триггер авто заполнения поля ext_id для оплаты

если надо синхронизировать биллинг с внешними программами и вести параллельный учёт автозаполнение поможет вести параллельную нумерацию документов

```
DELIMITER //
CREATE TRIGGER next_ext_id BEFORE INSERT ON payments
FOR EACH ROW BEGIN
  DECLARE new_id INTEGER;

  SET new_id='1C:1';
```

```
  IF NEW.ext_id = '' THEN
       SELECT CAST(SUBSTRING_INDEX(ext_id, ':', -1) AS UNSIGNED)+1 INTO new_id
         FROM payments WHERE ext_id LIKE '1C%' and date_format(date, '%Y-%m-%d')=curdate()
         ORDER BY 1 DESC  LIMIT 1;
       SET NEW.ext_id=CONCAT('1C:', new_id);
  END IF;

END;
//
DELIMITER ;
```

Функция создания квитанции для выбранных счетов.

Функция создания квитанции для выбранных счетов

```
DELIMITER //
CREATE FUNCTION add_docs (new_date datetime, admin_id int, new_uid INT, new_payment_id INT, payment_sum
DOUBLE(10,2), new_dsc VARCHAR(50))
  RETURNS INT
   DETERMINISTIC
    BEGIN
     DECLARE doc_id INT;
     DECLARE next_invoice_id INT;
     DECLARE exists_doc INT;
     DECLARE next_acct_id INT;

     SELECT count(*) INTO exists_doc FROM docs_acct WHERE payment_id=new_payment_id;

     IF exists_doc > 0 THEN
       RETURN 0;
     END IF;

     SELECT if(max(acct_id) IS NULL, 0, max(acct_id))+1 INTO next_acct_id
       FROM docs_acct WHERE DATE_FORMAT(date, '%Y')=DATE_FORMAT(curdate(), '%Y');

     INSERT INTO docs_acct (acct_id, date, created, customer, phone, aid, uid, payment_id)
       VALUES (next_acct_id, new_date, now(), '', '', admin_id, new_uid, new_payment_id);

     SET doc_id      = LAST_INSERT_ID();

     INSERT INTO docs_acct_orders (acct_id, orders, counts, unit, price)
       VALUES (doc_id, new_dsc,  1, 0, payment_sum);

     SELECT count(*) INTO exists_doc FROM docs_invoice WHERE payment_id=new_payment_id;

     IF exists_doc > 0 THEN
       RETURN 0;
     END IF;
     SELECT if(max(invoice_id)is NULL, 0, max(invoice_id))+1 INTO next_invoice_id
        FROM docs_invoice WHERE DATE_FORMAT(date, '%Y')=DATE_FORMAT(curdate(), '%Y');
     INSERT INTO docs_invoice (invoice_id, date, created, customer, phone, aid, uid, payment_id)
        values (next_invoice_id, new_date, now(), '', '', admin_id, new_uid, new_payment_id);
     SET doc_id      = LAST_INSERT_ID();
     INSERT INTO docs_invoice_orders (invoice_id, orders, counts, unit, price)
        values (doc_id, new_dsc, 1, 0, payment_sum);
     RETURN 1;
    END
//
DELIMITER ;
```

Создание почтовых ящиков для всех абонентов

При создании ящика пароль берётся из логина абонента, домен по умолчанию 1. Реально ящик появится

на диске после отправки пользователю первого письма.

```
INSERT INTO mail_boxes (uid, username, password, domain_id, descr, create_date)
 SELECT uid, id,
 ENCODE(DECODE(password, 'test12345678901234567890'), 'test12345678901234567890'),
 1,
 'User mailbox',
 now()
FROM users
```

Изменение секретного слова шифрования паролей

Секретный ключ сохраняется в переменной **$conf{secretkey}** конфигурационного файла **config.pl**

OLD_SECRET_KEY	старый ключ
NEW_SECRET_KEY	Новый ключ

Пользователи:

```
UPDATE users SET password=ENCODE(DECODE(password, 'OLD_SECRET_KEY'), 'NEW_SECRET_KEY');
```

Администраторы:

```
UPDATE admins SET password=ENCODE(DECODE(password, 'OLD_SECRET_KEY'), 'NEW_SECRET_KEY');
```

Сервера доступа:

```
UPDATE nas SET mng_password=ENCODE(DECODE(mng_password, 'OLD_SECRET_KEY'), 'NEW_SECRET_KEY');
```

Почтовые ящики:

```
UPDATE mail_boxes SET password=ENCODE(DECODE(password, 'OLD_SECRET_KEY'), 'NEW_SECRET_KEY');
```

Карточная платформа:

```
UPDATE cards_users SET pin=ENCODE(DECODE(pin, 'OLD_SECRET_KEY'), 'NEW_SECRET_KEY');
```

Если используется модуль sql для FreeRadius, нужно также поменять секретный ключ в конце файла **/usr/local/etc/raddb/sql.conf**

Изменение логина абонента

```
update users set id='new_login' where id='old_Login'
```

Автоматическое создание логинов и контрактов идентичным UID

в **config.pl** установить переменную проверки логинов в

```
$conf{USERNAMEREGEXP}=".{0,20}";
```

параметр `TABLE_SCHEMA`='abills' должен соответствовать названию базы в параметре

$conf{dbname}

MYSQL тригер

```
DELIMITER //
CREATE TRIGGER login_id BEFORE INSERT ON users
FOR EACH ROW BEGIN
  DECLARE user_id INT;

  IF NEW.id = '' THEN
    SELECT `AUTO_INCREMENT` INTO user_id  FROM `information_schema`.TABLES WHERE
`TABLE_SCHEMA`='abills' AND `TABLE_NAME`='users';
    SET NEW.id=CONCAT(if(@login_prefix IS NOT NULL, @login_prefix, ''), user_id);
  END IF;

END;
//
DELIMITER ;

DELIMITER //
CREATE TRIGGER next_contract_id BEFORE INSERT ON users_pi
FOR EACH ROW BEGIN

  IF NEW.contract_id = '' THEN
    SET NEW.contract_id=NEW.uid;
  END IF;

  IF NEW.contract_date = '' or NEW.contract_date = '0000-00-00' THEN
    SET NEW.contract_date=curdate();
  END IF;

END;
//
DELIMITER ;
```

Удаление нуля как первого символа пин на карте пополнения

проверка перекодировки

```
SELECT id,
 serial,
 number,
 DECODE(pin, 'test12345678901234567890') as pin ,

if (DECODE(pin, 'test12345678901234567890') REGEXP '^0' =1,
  SUBSTRING(DECODE(pin, 'test12345678901234567890') FROM 2),
  DECODE(pin, 'test12345678901234567890')) AS result,

ENCODE( if (DECODE(pin, 'test12345678901234567890') REGEXP '^0' =1,
  SUBSTRING(DECODE(pin, 'test12345678901234567890') FROM 2),
  DECODE(pin, 'test12345678901234567890')),
 'test12345678901234567890'
) AS hex_result

FROM cards_users

WHERE serial = 'seria'

LIMIT 100;
```

если все нормально

перекодировка

```
UPDATE cards_users SET pin=ENCODE( if (DECODE(pin, 'test12345678901234567890') REGEXP '^0' =1,
  SUBSTRING(DECODE(pin, 'test12345678901234567890') FROM 2),
  DECODE(pin, 'test12345678901234567890')),
 'test12345678901234567890'
)
WHERE serial = 'seria'
```

Создание логинов с UID определённого диапазона

создание логинов в диапазоне ниже 100 000 Ghb при наличии свободных UID и существовании UID выше 100 000

```
DELIMITER //
CREATE TRIGGER create_uid BEFORE INSERT ON users
FOR EACH ROW BEGIN

  DECLARE new_uid INTEGER;

  IF NEW.uid = '' THEN
    SET new_uid=1;
    SELECT uid+1 INTO new_uid
       FROM users WHERE uid < 100000
       ORDER BY uid DESC LIMIT 1;
    SET NEW.uid=new_uid;

  END IF;
END
//
DELIMITER ;
```

Изменение тарифного плана абонентам с определённой группой

```
UPDATE dv_main set tp_id=225 WHERE uid IN (SELECT uid from users WHERE gid=123);
```

Конвертация денежной единицы для абонентов (смена основной валюты или девальвация)

1.Конвертация списку/группе абонентов депозита по указанному курсу. для абонентов без компаний

```
UPDATE bills b, users u, companies c
  SET b.deposit=b.deposit * [ курс ]
  WHERE b.id=u.bill_id AND u.gid IN ([ группа ],[группа])
```

для абонентов в компаниях

```
UPDATE bills b, users u, companies company
  SET b.deposit=b.deposit * [ курс ]
  WHERE
  u.company_id=company.id
  AND company.bill_id=b.id
  AND u.gid IN ([ группа ])
```

2.Конвертация списку/группе абонентов кредита по указанному курсу.

```
UPDATE users u SET u.credit=u.credit * [ курс ] WHERE u.gid IN ([ номера групп ])
```

3. Определенному списку/группе тарифных планов. Конвертирует стоимость в тарифных планах (ежедневные и ежемесячные снятия, изменение ТП и т.п.) по указанному курсу.

для ТП

```
UPDATE tarif_plans tp
  SET
    tp.day_fee=tp.day_fee * [ курс ],
    tp.month_fee=tp.month_fee * [ курс ],
    tp.activate_price=tp.activate_price * [ курс ],
    tp.change_price=tp.change_price * [ курс ]
WHERE
   tp.id IN ([ номера тарифных планов ])
```

Группы тарифных планов

```
UPDATE tarif_plans tp, tp_groups tp_g
  SET
    tp.day_fee=tp.day_fee *  [ курс ],
    tp.month_fee=tp.month_fee *  [ курс ],
    tp.activate_price=tp.activate_price *  [ курс ],
    tp.change_price=tp.change_price *  [ курс ]
WHERE
  tp.gid=tp_g.id AND  tp_g.id IN ([ номера групп тарифных планов ])
```

Объединение 2 ABillS систем

Создаём базу abills2 и заливаем туда данные с дополнительной биллинговой системы

1. увеличиваем uid, billd_id, company_id 100000 чтобы не было конфликтов в новой системе.
2. Заливаем данные в основную систему

```
use abills2;
UPDATE users SET uid=uid+100000, bill_id=bill_id+100000, company_id=company_id+10000;
UPDATE companies SET id=id+10000, bill_id=bill_id+100000;
UPDATE users_pi SET uid=uid+100000;
UPDATE bills SET uid=uid+100000, id=id+100000;
UPDATE payments SET uid=uid+100000, bill_id=bill_id+100000, id=id+1000000, aid=aid+10000;
UPDATE fees SET uid=uid+100000, bill_id=bill_id+100000, id=id+1000000, aid=aid+10000;
UPDATE dv_main SET uid=uid+100000;
UPDATE dv_log SET uid=uid+100000;
UPDATE admin_actions SET uid=uid+100000, id=id+1000000, aid=aid+10000;
UPDATE admins SET aid=aid+1000;
```

```
use abills;
INSERT INTO users SELECT * from abills2.users;
INSERT INTO companies SELECT * from abills2.companies;
INSERT INTO users_pi SELECT * from abills2.users_pi;
INSERT INTO bills SELECT * from abills2.bills;
INSERT INTO payments SELECT * from abills2.payments;
INSERT INTO fees SELECT * from abills2.fees;
INSERT INTO dv_main  SELECT * from abills2.dv_main;
INSERT INTO dv_log SELECT * from abills2.dv_log;
INSERT INTO admin_actions SELECT * from abills2.admin_actions;
```

Приведение номеров телефонов к общему формату

перевести номера +38050xxxxxx в 050xxxxxx

```
UPDATE users_pi SET phone=REPLACE(phone, '+38050', '050')
WHERE phone like '+38050%'
```