© В.И. Суханов, В.М. Аленичев, 2015

УДК 622:528.9:912:004

***В.И. Суханов, В.М. Аленичев***

# *РАЗРАБОТКА ГОРНО-ГЕОЛОГИЧЕСКОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ ПО СТАНДАРТАМ ОТКРЫТЫХ СИСТЕМ*

*Приведены основные сведения о технологиях разработки горной информационной системы на основе стандартов открытых систем и геометрического ядра с открытым кодом. Подробные примеры кода позволяют разработчикам ГГИС реализовать основные функции хранения и манипулирования геометрическими данными об элементах карьера и модели месторождения. Использованы рекомендации консорциума OGC и программные продукты с открытым кодом: PostGIS, Open CASCADE, PythonOCC, wxPython.*

*Ключевые слова: карьер, ГИС, разработка, программирование, открытый код.*

Эффективность проектирования, планирования и управления производством по добыче полезных ископаемых при соблюдении экологических требований и обеспечении PostGIS, Open CASCADE, безопасных условии ведения горных работ в настоящее время невозможно без внедрения горно-геологических информационных систем, поскольку инженерно-геологические условия разработки месторождения определяются совокупностью следующих взаимодействующих факторов: физико-географических, геологических, литолого-петрографических, гидрогеологических и геодинамических [1].

Среди многообразия параметров месторождения и карьера строго выделить наиболее существенные геоданные и атрибутивные признаки при оценке ресурсного потенциала геологического объекта практически невозможно, поскольку они характеризуют сущность (субстанцию) недоступную для прямого наблюдения, оцениваемую по косвенно измеряемым признакам. От полноты и достоверности геоинформционного обеспечения зависит объективная оценка специалистами запасов георесурсов на конкретном месторождении. Функциональная значимость геоданных проявляется по их совокупному влиянию на выбор способа разработки и обоснование внутренней структуры прогнозируемой технологии [2]. Способ разработки определяется по совокупности пространственных геоданных, дополненных ранее наблюдаемой устойчивой статистики, структура технологии – степенью изменчивости параметров и показателей, характеризующих внутреннее строение месторождения. При разработке рациональной модели, учитывающей тип месторождения, необходимо определить набор геоданных для обоснованного выбора способа разработки и внутренней структуры прогнозируемой геотехнологии.

Указанные потребности ставят актуальную задачу разработки отечественных защищенных от возможных утечек данных горно-геологических информационных систем (ГГИС). Это можно гарантировать только с использованием технологий разработки на основе открытого кода.

Open Geospatial Consortium (OGC) – международная, добровольная организация по разработке стандартов в области геоинформационных сервисов [3]. Ею подготовлены разнообразные стандарты и рекомендации по представлению и манипулированию геопространственными данными, на основе которых

разработано большое количество общесистемного и прикладного программного обеспечения в основном с открытым кодом, с использованием которого разработка доверенных горно-геологических систем для открытой разработки месторождений посильна небольшим коллективам программистов. При реализации этих технологий достаточно использовать следующие программные обеспечения желательно последних версий: для работы с базами геоданных – СУБД PostgreSQL и ее расширение PostGIS [4], для администрирования БД и просмотра геоданных через веб-интерфейс – сервер GlassFish с установленным приложением GeoServer [5], для разработки редактора и решения технологических задач – геометрическое ядро с открытым кодом Open CASCADE (OCC) [6, 7], система программирования Python и библиотека PythonOCC.

Для иллюстрации техники хранения и работы с геометрическими данными рассмотрим таблицу edge для бровок уступов горного карьера. Операторы для создания таблицы имеют следующий вид:

```
CREATE TABLE edge
(
  id_edge serial NOT NULL,
  hor integer NOT NULL,
  edge_type integer NOT NULL,
  CONSTRAINT «PK_edge» PRIMARY KEY (id_edge),
  CONSTRAINT edg_typ FOREIGN KEY (edge_type)
      REFERENCES edge_type (id_edge_type) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT «edge-hor» FOREIGN KEY (hor)
      REFERENCES horizons (id_hor) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
  OIDS=FALSE
);
ALTER TABLE edge ADD COLUMN geom geometry(LINESTRINGZ,0);
```

Поля hor и edge_type являются внешними ключами описания горизонта и типа бровки в таблице horizons и edge_type соответственно. Основное назначение таблицы – хранить геометрию бровки уступа в поле geom.

Для работы с геометрическими объектами в период исполнения редактора OCC необходима структура данных в памяти ЭВМ, содержащая текущую информацию об объекте. Для этого используем список вида:

```
[[object, id_edge, shape, id_hor, edge_type, modified], ...]
```

где object – признак типа объекта, в системе существует большое разнообразие объектов, каждый имеет свой признак типа; id_edge – ключ экземпляра бровки в БД; id_hor – ключ горизонта в БД; edge_type – ключ типа бровки в БД; modified – флаг модификации геометрии объекта в редакторе; shape – ссылка на объект OCC для визуализации на экране дисплея. Если объект был удален пользователем в диалоге с редактором, то элемент имеет значение None.

Для чтения и визуализации данных из БД следует выполнить:

```
    conn = psycopg2.connect(«dbname=»+POSTGR_DBN+» user=»+POSTGR_USR)
    curs = conn.cursor()
    query = «SELECT id_edge,hor,edge_type,ST_AsEWKT(geom),point,color FROM
edge,horizons,edge_type «
    query = query + «where (id_hor in « + setHorIds + «) and (edge.hor=horizons.
id_hor) and (edge.edge_type=edge_type.id_edge_type);»
    curs.execute(query)
    rows = curs.fetchall()
    for rec in rows: id_edge=int(rec[0]);id_hor=int(rec[1]);edge_type=int(rec[2])
        if (not rec[3]): continue
        coordsPLine = parsGeometry(str(rec[3]))
        point = float(rec[4]);color=rec[5]
        query = «SELECT red,green,blue FROM color WHERE id_color=» + str(color) + «;»
        curs.execute(query); clr = curs.fetchone()
        clrRed = clr[0]; clrGreen = clr[1]; clrBlue = clr[2]
        plgn = BRepBuilderAPI_MakePolygon()
        for pnt in coordsPLine:
            if len(pnt) < 3: pnt = pnt + [point]
            plgn.Add(gp_Pnt(pnt[0], pnt[1], pnt[2]))
        w = plgn.Wire()
        color = OCC.Quantity.Quantity_Color(int(str(clrRed))/  255.0,int(str(clrGree
n))/255.0,int(str(clrBlue))/255.0,0)
        s=self.canva._3dDisplay.DisplayColoredShape(w, color, False)
        s1 = s.GetObject()
        self.canva.dataBase = self.canva.dataBase + [[0, id_edge, s1,  id_hor, edge_
type, False]]
```

Используемая функция parsGeometry производит преобразование представления EWKT геометрии объекта в список его вершин.

```
    def parsGeometry(geom, dlt = 1):
        «»»Разобрать координаты геометрического объекта»»»
        if geom == 'LINESTRING EMPTY': return []
        coords = geom[geom.find('(')+dlt : geom.find(')')]
        lstCoords = coords.split(','); lstXYZ=[]
        for pnt in lstCoords:
            pntXYZ=[]; xyz=pnt.split(' ')
            for val in xyz:
                #if val.isdigit():
                    pntXYZ = pntXYZ + [float(val)]
            lstXYZ = lstXYZ + [pntXYZ]
        return lstXYZ
```

Для обратного преобразования экранных объектов OCC в записи таблицы БД полезными являются следующие функции.

```
    def getPoints(shape):
        «»»Получить точки выбранного объекта»»»
        result = []
```

```
        if shape:
            ex=OCC.TopExp.TopExp_Explorer()
            ex.Init(shape,OCC.TopAbs.TopAbs_VERTEX)
            te = ShapeToTopology(); bt = OCC.BRep.BRep_Tool()
            result = []
            while True:
                sv=ex.Current(); vv=te(sv)
                p1=OCC.BRep.BRep_Tool.Pnt(vv)
                p1 = [p1.X(),p1.Y(),p1.Z()]
                if (len(result)==0) or (p1 <> result[-1]):
                    result = result + [p1]
                sv=ex.Next()
                if not ex.More(): break
        return result

    def makeLINESTRING(pnts):
        geom = «GeomFromEWKT('SRID=-1;LINESTRING(«
        j = 0
        for p in pnts:
            if (j > 0): geom = geom + «,»
            j = 1
            geom = geom + «%.0f %.0f %.0f»%(p[0],p[1],p[2])
        geom = geom + «)')»
        return geom
```

Сохранение результатов создания и редактирования бровок в БД может быть выполнено следующим фрагментом программы.

```
    def SaveDB(self):
        «»» Сохранить изменения в БД «»»
        conn = psycopg2.connect(«dbname=»+POSTGR_DBN+»
user=»+POSTGR_USR)
        curs = conn.cursor()
        for indexInfo in range(len(self.canva.dataBase)):
            element = self.canva.dataBase[indexInfo]
            if element[ – 1]:           # Был изменен
                if element[0] == 0:     # Бровка
                    id = element[1]; s1 = element[2]     # Объект
                    if s1 == None:      # Удалять из БД
                        query = «DELETE FROM edge WHERE id_edge=» + str(id) + «;»
                        curs.execute(query)
                    elif (id == -1):
                        # Добавить в БД
                        pnts = getPoints(s1.Shape())
                        id_hor = element[3]
                        edge_type =element[4]
                        geom = makeLINESTRING(pnts)
                            query = «INSERT INTO edge (hor,edge_type,geom) VALUES
(«+str(id_hor)+»,»+str(edge_type)+»,»+geom+») RETURNING id_edge;»
```

```
                curs.execute(query)
                id_edge=curs.fetchone()[0]
                element[1] = id_edge
            else:
                # Заменить в БД
                pnts = getPoints(s1.Shape())
                geom = makeLINESTRING(pnts)
              query = «UPDATE edge SET geom=» + geom + « WHERE id_edge=»
+ str(id) + «;»
                curs.execute(query)
            element[-1] = False # Снять флаг модификации
            self.canva.dataBase[indexInfo] = element
      .       .       .       .       .       .
    conn.commit()
    curs.close()
    conn.close()
```

**Программирование модулей ГГИС для работы с объектами**

Разработку головной программы целесообразно начать с адаптации демонстрационного примера с именем InteractiveViewer, входящего в комплект примеров пакета PythonOCC. Пример включает три основных модуля: InteractiveViewer.py – головная программа, browser.py – вспомогательный модуль и wxDisplay.py – модуль работы с интерактивной графикой.

Главный модуль определяет класс AppFrame с конструктором __init__, где определяются основные органы управления приложения (меню и палитры инструментов), связанные с обработчиками соответствующих событий. Обработчики событий – методы тогоже класса, описанные в тексте модуля.

Модуль wxDisplay.py отвечает за взаимодействие с графической оболочкой wxPython и содержит обработчики таких событий как перемещение указателя (мышки), нажатие и отпускание кнопок мышки и клавиатуры, изменение размеров окна и другие. Будем считать, что мировая система координат является правосторонней прямоугольной евклидовой системой с направлением осей: *X* – вправо, *Y* – вверх, *Z* – на наблюдателя. При вводе и редактировании геометрических координат объектов важным является возможность их интерактивного задания с использованием указателя мышки на экране. Обычно ввод координат выполняется нажатием левой кнопки мышки. Для этого добавим в текст обработчика события OnLeftDown следующий фрагмент.

```
    # Расчет позиции точки в мировых координатах
        self.startPt = evt.GetPosition()
      xt, yt, zt, Pt,Ut,Vt = self._3dDisplay.GetView().GetObject().ConvertWithProj(self.
startPt.x, self.startPt.y)
        resPnt = [xt,yt,zt]
        self.worldPt = resPnt
```

При вводе координат указателем мышки следует исходить из того, что точное прицеливание практически невозможно и пользователю нужно дать возможность вручную откорректировать координаты указателя при их использовании в дальнейших манипуляциях. Для этого в главном модуле в конструкторе главного окна определим визуальный редактор текстовой строки

```
self.coordXYZ = wx.NewId()
self.canva.coord = wx.TextCtrl(tb, self.coordXYZ, «x,y[,z]», wx.DefaultPosition,
(190, 30), 0)
tb.AddControl(self.canva.coord)
```
и поместим в него полученные координаты.
```
# Координаты точки в окно координат
    self.frame.canva.coord.SetValue(«%.2f,%.2f,%.2f»%(resPnt[0],resPnt[1],resP
nt[2]))
```

Пользователь может вручную редактировать все три координаты точки перед передачей их на исполнение командам построения и изменения геометрических объектов. Использование координат для геометрических построений можно подтвердить, добавив кнопку «Принять»:

```
self.put_coordOk = wx.NewId()
coord_bmp = CreateGif(os.path.join(THISPATH, 'icons', 'but_login.gif'), 16, 16)
tb.AddLabelTool(self.put_coordOk, 'Coords', coord_bmp, shortHelp='Принять')
self.Bind(wx.EVT_TOOL, self.onCoord_yes, id=self.put_coordOk)
```

Обработчик события onCoord_yes кнопки «Принять» будет выполнять манипуляции с координатами из окна в зависимости от исполняемой команды, задаваемой пользователем средствами меню и кнопок. Например, для построения полилинии в главном окне добавим кнопку «Полилиния»:

```
# Полилиния
self.plineID = wx.NewId()
img_pline = CreatePng(os.path.join(THISPATH, 'icons', 'linear.png'), 16, 16)
tb.AddLabelTool(self.plineID, 'PLine', img_pline, shortHelp='Полилиния')
self.Bind(wx.EVT_TOOL, self.OnPLine, id=self.plineID)
```

Обработчик кнопки для полилинии:

```
def OnPLine(self, event):
    «»» Рисование ломаной «»»
    self.canva.MakePLine = True
    self.canva.lstPnt = []
    self.SetStatusText(«Полилиния. Начало», 0)
    self._refreshui()
```

Обработчик кнопки «Принять» будет выглядеть так:

```
def Coord_yes(self):
    «»» Ввод координат из окна Point от кнопки Принять «»»
    Z = float(self.canva.coordZ.GetValue())
    coordStr = self.canva.coord.GetValue()
    if self.canva.MakePLine:
        # PolyLine
        drawP = False; closeP = False
        if (len(self.canva.lstPnt) > 2): #Close or End
```

```
        if (coordStr.upper().find('C') <>- 1):   #Close
            drawP = True; closeP = True
        if (not (coordStr.strip())):   #End
            drawP = True
        if drawP:
            # Строим объект на экране средствами OCC
            plgn = BRepBuilderAPI_MakePolygon()
            for pnt1 in self.canva.lstPnt:
                plgn.Add(gp_Pnt(pnt1[0], pnt1[1], pnt1[2]))
            if closeP: plgn.Close()
            w = plgn.Wire()
            self.canva._3dDisplay.DisplayColoredShape(w, 'YELLOW', False)
            self.SetStatusText("Готово", 2)
            self.canva.MakePLine = False
            self.canva.lstPnt = []
            return
    coord1 = coordStr.split(',')
    . . .
```

Если пользователь ввел не менее двух точек и решил закончить полилилинию, очистив координаты, или ввел символ «C», то на экране строится объект средствами OCC. Иначе новая точка добавляется в список lstPnt. Вспомогательное окно ввода coordZ позволяет сократить ввод третьей координаты в окне координат «*X,Y,Z*» за счет использования значения *Z* по умолчанию из окна coordZ.

***Булевские операции над геометрическими объектами***

Булевские операции объединения, пересечения и вычитания геометрических фигур часто возникают при выполнении анализа обстановки или метрических характеристик объектов. Разработчику программ предоставляется возможность использовать для этих целей инструментарий, предоставляемый как OCC, так и PostGIS. Функционально эквивалентные средства предоставляют различные интерфейсы для их использования. Более удобными, а стало быть и целесообразными, являются интерфейсы функций PostGIS. В приведенном ниже методе используются инструменты выполнения булевских операций PostGIS. Исходные геометрии задаются списками точек lstPnt1 и lstPnt2, а выполняемая операция параметром op='Union' для объединения, 'Intersection' для пересечения и 'Difference' для разности фигур. Результатом является список полигонов (возможно нескольких), задаваемых координатами их вершин.

```
def boolOp(lstPnt1, lstPnt2, op='Union'):
    # op ::=  Difference | Intersection | Union
    # результат [[p1,...,pn], ...]
    geom1 = «ST_AsEWKT('POLYGON((«
    cnt = len(lstPnt1); i = 0
    for pnt in lstPnt1:
        i = i + 1; geom1 = geom1 + «%.0f %.0f» % (pnt[0], pnt[1])
        if (i < cnt):geom1 = geom1 + «,»
    geom1 = geom1 + «))')»; geom2 = «ST_AsEWKT('POLYGON((«
    cnt = len(lstPnt2); i = 0
```

```
        for pnt in lstPnt2:
            i = i + 1; geom2 = geom2 + «%.0f %.0f» % (pnt[0], pnt[1])
            if (i < cnt): geom2 = geom2 + «,»
        geom2 = geom2 + «))')»
        query = «SELECT ST_AsEWKT(ST_» + op + «(« + geom1 + «,» +
geom2 + «));»
        conn = psycopg2.connect(«dbname=»+POSTGR_DBN+»
user=»+POSTGR_USR)
        curs = conn.cursor(); curs.execute(query)
        rows = curs.fetchall(); result = []
        for rec in rows:
            geom = rec[0]
            if 'MULTIPOLYGON' in geom:
                dlt = 3; strgeom = geom[13:]; startpos = 2
                while 1:
                    endpos = strgeom.find('))')
                    if endpos < 0: break
                    coords = strgeom[startpos:endpos]
                    lstCoords = coords.split(',')
                    lstXYZ=[]
                    for pnt in lstCoords:
                        pntXYZ=[]; xyz=pnt.split(' ')
                        for val in xyz:
                            pntXYZ = pntXYZ + [float(val)]
                        lstXYZ = lstXYZ + [pntXYZ]
                    result = result + [lstXYZ]
                    strgeom = strgeom[endpos+3:]; startpos = 2
            elif 'POLYGON' in geom:
                dlt = 2; plgn = parsGeometry(geom,dlt)
                result = result + [plgn]
        curs.close()
        conn.close()
        return result
```

**Редактирование вершин полилиний**

Редактирование вершин полилиний связано со вставкой новой, удалением или переносом существующей. Учитывая необходимость контроля правильности задания координат такое преобразование следует выполнять при вводе положения редактируемой вершины.

Фрагмент обработчика кнопки «Принять», выполняющий добавление, удаление и перемещения вершины полилинии, задаваемой выбранным пользователем объектом self.canva.selShape:

```
        if (self.canva.Operation in [«InsPnt»,»DelPnt»,»MovePnt»]):
            # self.canva.lstPnt[-1] – Координаты точки
            # self.canva.selShape  – Объект для редактирования
            resPnt = lstXY                  # Координаты в окне X,Y,Z
            if (self.canva.Operation in [«MovePnt»]) and
           (len(self.canva.lstPnt)<2):
```

```
        self.SetStatusText(«Нет точки», 2)
        return
    pnts = getPoints(self.canva.selShape)
    p1 = pnts[0]; newPnts = [p1]
    for p2 in pnts[1:]:
        if (self.canva.Operation in [«InsPnt»]):
            if inline(resPnt, p1, p2, eps=0.5):
                newPnts.append(resPnt)
            newPnts.append(p2)
        elif (self.canva.Operation in [«DelPnt»]):
            if not near(self.canva.lstPnt[-1],p2):
                newPnts.append(p2)
        elif (self.canva.Operation in [«MovePnt»]):
            if near(self.canva.lstPnt[-2],p2, eps=0.5):
                newPnts.append(self.canva.lstPnt[-1])
            else: newPnts.append(p2)
        p1 = p2
```

Булевские функции inline и near производят проверку принадлежности точки отрезку и ее близости к точке.

**Заключение**

Материал статьи иллюстрирует основные понятия и требования при создании гео-информационных приложений, согласованных с рекомендациями OGC и поддерживающих их программных продуктов для хранения и отображения геоданных. Приведенные примеры реализации основных функций хранения и манипулирования геоданными позволяют самостоятельно вести разработку ГГИС без привлечения дорогостоящих коммерческих продуктов, гибко адаптируя к требованиям предметной области их использования.

***СПИСОК ЛИТЕРАТУРЫ***


1. *Абатурова И.В.* Научно-методические основы изучения, оценки и прогноза инженерно-геологических условий месторождений полезных ископаемых в скальных породах, Автореф. докт. диссер. – Екатеринбург. – 2012. – С. 41

2. *Аленичев В.М., Суханов В.И., Хохряков В.С.* Моделирование природно-сырьевых технологических комплексов (горное производство). – Екатеринбург: УрО РАН, 1998. – 256 с.

3. *Standards OGC* [electronic resource]. Mode of access: http://gis-lab.info/wiki/Стандарты_OGC. 12/05/2014

4. *PostGIS* 2.0 Manual. [Электронный ресурс]. Режим доступа: http://postgis.org. 20.09.2013

5. *GeoServer* User Manual . Release 2.0.1 [Электронный ресурс]. Режим доступа: http://docs.geoserver.org/stable/en/user/. 20.09.2013

6. *Open* CASCADE Technology [Электронный ресурс]. Режим доступа: http://www.opencascade.org/. 20.09.2013

7. *Суханов В.И., Зобнин Б.Б., Тимошенко С.И., Ажипа И.А., Рыжков Д.С.* Обзор возможностей геометрических ядер с открытым кодом для задач построения геоинформационного обеспечения горного производства // Известия вузов. Горный журнал. – 2010. – № 8. – С. 87–102.

8. Psyc №8, 2010, с. 87–102opg [Электронный ресурс]. Режим доступа: http://initd.org/psycopg/. 20.09.2013. ГИАБ

***КОРОТКО ОБ АВТОРАХ***

*Суханов Владимир Иванович* – доктор технических наук, доцент,
зав. кафедрой, *e*-mail: cyx-fat@mail.ru,
Уральский федеральный университет (УрФУ),
*Аленичев Виктор Михайлович* – доктор технических наук, профессор,
главный научный сотрудник, ИГД УрО РАН,
*e*-mail: alenichev@igduran.ru.

UDC 622:528.9:912:004

**PROSPECTS FOR THE INTRODUCTION OF GEOLOGICAL INFORMATION SYSTEMS ON THE DOMESTIC MINING ENTERPRISES**

*Sukhanov V.I.*, Doctor of Technical Sciences, Assistant Professor, Head of Chair,
*e*-mail: cyx-fat@mail.ru, Ural Federal University, Ekaterinburg, Russia,
Alenichev V.M., Doctor of Technical Sciences, Professor, Chief Researcher,
Institute of Mining of Ural Branch of Russian Academy of Sciences,
620219, Ekaterinburg, Russia, *e*-mail: alenichev@igduran.ru.

*Provides basic information about technology development mining information system based on open systems standards and geometric kernel, open source. Detailed code samples allow developers to implement basic functions GGIS storage and manipulation of geometric data of the elements and the career field model. Used the recommendations of the consortium OGC and software products with open source: PostGIS, Open CASCADE, PythonOCC, wxPython.*

*Key words: career, GIS, engineering, programming, open source.*

**REFERENCES**

1. Abaturova I.V. *Nauchno-metodicheskie osnovy izucheniya, otsenki i prognoza inzhenerno-geologicheskikh usloviy mestorozhdeniy poleznykh iskopaemykh v skal'nykh porodakh* (Scientific and methodological basis of the study, assessment and prediction of engineering-geological conditions of mineral deposits in the rocks), Doctor's thesis, Ekaterinburg, 2012, p. 41.

2. Alenichev V.M., Sukhanov V.I., Khokhryakov V.S. *Modelirovanie prirodno-syr'evykh tekhnologicheskikh kompleksov (gornoe proizvodstvo)* (Simulation of natural raw technological complexes (mining industry)), Ekaterinburg, UrO RAN, 1998, 256 p.

3. *Standards OGC*. Mode of access: http://gis-lab.info/wiki/Стандарты_OGC. 12/05/2014

4. *PostGIS 2.0 Manual*, available at: http://postgis.org. 20.09.2013

5. *GeoServer User Manual. Release 2.0.1*, available at: http://docs.geoserver.org/stable/en/user/. 20.09.2013

6. *Open CASCADE Technology*, available at: http://www.opencascade.org/. 20.09.2013

7. Sukhanov V.I., Zobnin B.B., Timoshenko S.I., Azhipa I.A., Ryzhkov D.S. *Izvestiya vuzov. Gornyy zhurnal*. 2010, no 8, pp. 87–102.

8. *Psycopg*, available at: http://initd.org/psycopg/. 20.09.2013.

***Бутафорское реформирование делает его бессмысленным.***