\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
|  | ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ  Кафедра №42  «Криптология и кибербезопасность» |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Федеральное государственное автономное образовательное учреждение высшего образования*

**«Национальный исследовательский ядерный университет «МИФИ»»**

ЛАБОРАТОРНАЯ РАБОТА №2-2:

**«Пользователи. Роли. Привилегии. Профили.»**

Аверин Владислав

Группа: Б19-505

Октябрь, 2022

*Содержание*

[*1.* *Формирование “ТЗ”* 4](#_Toc115885625)

[*2.* *Создание ролей* 6](#_Toc115885626)

[*3.* *Создание схем и профилей* 11](#_Toc115885627)

[*4.* *Функциональное тестирование* 14](#_Toc115885628)

*Цель работы*

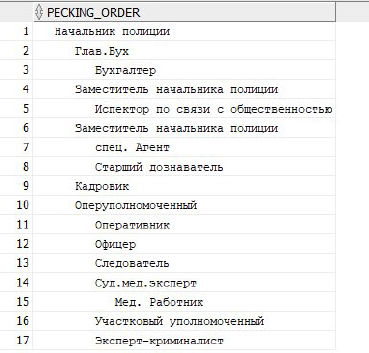
Изучение механизмов базы данных, связанных с управлением возможностями пользователей. Приобретение навыков разработки многопользовательской базы данных.

*Ход работы*

1. Продумать список пользователей и ролей, необходимых для эксплуатации базы данных (администратор БД, рядовые пользователи, менеджеры и пр.). Для каждой категории пользователей определить список необходимых привилегий. Решить, в какой схеме (схемах) следует создавать таблицы базы данных. Для решения таких задач, как ограничение доступа к некоторым столбцам таблицы, предоставление доступа только к статистике, но не к самим данным, — можно использовать представления (VIEW);
2. Внести соответствующие изменения в базу данных: добавить новых пользователей, предоставить им соответствующие привилегии и роли, при необходимости повторно создать таблицы схемы данных от лица выбранного пользователя (пользователей), при необходимости создать публичные синонимы для таблиц (CREATE PUBLIC SYNONYM);
3. Проконтролировать корректную работу базы данных от лица каждой категории пользователей. В частности, убедиться, что рядовые пользователи не могут выполнять административные функции и получать доступ к данным, которые по легенде должны быть для них закрыты;
4. Рассмотреть необходимость создания одного или нескольких профилей для рядовых пользователей базы данных; при необходимости — создать и применить их;
5. Оформить отчёт.
6. *Формирование “ТЗ”*

Для начала определимся со схемой, где будут создаваться таблицы. Так как изначально для данной цели использовался самый первый созданный пользователь (Infernal), который фактически владеет всеми созданными таблицами, то пусть его схема и будет основной (сделаем из него в последствии администратора БД). Синонимы для таблиц можно создать для сокращения записи в запросах, но т.к. фантазия (а точнее, ее отсутствие) придумать синонимичные названия для таблиц не позволяет, просто добавим к их названию постфикс Ps (pseudonym), а-ля CasesPs. А для представлений будем использовать постфикс V.

Что касается ролей. Не зря я все таки создавал в лабораторной 1-3 иерархический запрос с введенными в БД должностями, хех :) Возьмем несколько должностей оттуда (которые гипотетически должны иметь доступ к каким-либо данным):

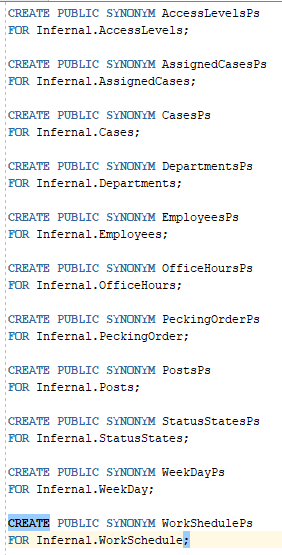


* Администратор БД (его схема является той, в которой будут создаваться таблицы) – имеет полный доступ к БД (хотя, насколько я знаю, централизированное абсолютное управление не есть хороший паттерн администрирования, но в моем случае БД просто не настолько большая, чтобы разделить эти обязанности. Поэтому для контроля корректной работы будет использоваться одна роль);
* Бухгалтерия: имеет доступ и к делам, и к данным о персонале, но не может модифицировать таблицы (у меня вообще никто не будет иметь права на ALTER TABLE, вот такое преступное управление… или халатничество, эт как посмотреть). Хотя, если считать, что именно там будет вестись различного рода статистика, то рациональнее дать доступ на чтение ко всем таблицам (т.к. там находится дополнительная информация, и фиг знает, какую именно статистику им захочется получить, на все представлений не оберешься). Кроме того, могут добавлять (без права удаления) строки в Cases. Удалять строки с делами пока в функционал данной БД не входит, оставим это на волю схемы администратора :) Таким образом, разные теть Гали и Тамары могут анализировать все данные, и, если необходимо завести дело, это также проходит через них, как через формальный (в данном упрощенном случае) орган. Ну, и чтобы они совсем бесполезными не были, добавим для них еще дополнительный атрибут “salary” в Employees, который они смогут изменять.
* Кадровый отдел (кадровики): имеют read-only к связанным с таблицей Employees таблицам (т.е. ко всем, кроме Cases и AssignedCases), могут менять уже существующую информацию и добавлять новую в Employees (в т.ч. атрибут salary, т.к. он NOT NULL и они при запрете просто не смогут добавлять новый персонал (аж в рифму)). Удалять сотрудников из базы напрямую они не cмогут, т.к. это влечет за собой конфликты в смежных таблицах, где на месте ключей могут фигурировать удаляемые id. Решение: либо дать кадровому отделу также доступ и к смежным таблицам (что не очень хорошо), либо оставить как есть, и для увольнения сотрудника дополнительно обращаться к главе его отдела (см. далее) для удаления связанной с этим сотрудником информации, например, в таблице AssignedCases. Второй вариант мне больше по душе (бо-о-ольше бумажной волокиты богу волокиты).
* Глава отдела: они должны уметь модифицировать таблицы, связанные с сотрудниками (график работы, уровни доступа дела, назначать кого-то на работу с делом, etc.), но не саму таблицу Employees, для этого есть другие отделы, изменять состояния дел, и иметь возможность так же, как и бухгалтерия, просматривать остальную статистику о сотрудниках (правда, как сделать так, чтобы глава каждого отдела имел доступ только к данным о сотрудниках только своего отдела, кроме как максимального костыля в виде нескольких одинаковых ролей на каждый отдел с отдельным представлением для каждой, я не придумал);
* Сотрудники отделов (оперативники, участковые, оперуполномоченные, etc.): имеют доступ (на чтение) только к делам (и по-хорошему только связанными с собой, т.е., чтобы каждая новая схема пользователя в роли зависела от, например, ID сотрудника). Если решить эту проблему с отображением индивидуальный строк для каждой схемы, то сюда можно добавить представление с его собственной информацией.
* Мед. персонал: введем в таблицу Employees еще один обязательный атрибут “номер мед.карты”, а также необязательный атрибут “заключение мед.экспертизы” в таблицу Cases (чтобы хоть какая-то информация для тех же судмедэкспертов была). Таким образом, мед. работники будут иметь представления (всм, View) на чтение с этими атрибутами, и иметь возможность менять эти два нововведённых столбца (типом может быть либо BLOB, либо проще мне кажется просто добавлять ссылки на файлы с какого-нибудь локального FTP-сервера, так что будет просто CHAR).

Таким образом, мы выделили 6 ролей (первая, правда, уже организована, но все же), которые в достаточной роли описывают взаимодействие различных частей структуры объекта БД с данными. Все по канонам жанра бюрократии: при появлении нового дела, главы отделов посылают на него своих сотрудников, параллельно делая запрос бухгалтерии, чтобы те создали данную запись в таблице. В течение работы над делом сотрудники отчитываются своих главным, те в свою очередь меняют статус дела, добавляют новые фигурирующие лица и т.п. Все ради того, чтобы побесить :) Правда, получается так, что суд.мед.эксперт или криминалист может напрямую изменять состояния атрибутов (и нужна еще одна роль, специально для начальника их отдела), но спишем это на формальности, и что данная информация зависит исключительно и персонально от самих работников, а в других атрибутах дел данные общие, поэтому изменять их может только вышестоящее начальство.

1. *Создание ролей*

Каждая из вышеописанных ролей должна включать только одну системную привилегию CREATE SESSION, остальные будут объектные. Единственное, что я опустил, так это роль администратора БД, т.к. по идее это и есть почти sysdba с теми привилегиями, которые мы давали нашему самому пользователю в лабораторных 1-\*. Так что такая схема уже присутствовала по дефолту. Для начала от имени уже настоящего sysdba дадим псевдонимы всем существующим таблицам:



Уверен, что давать псевдонимы поголовно это излишние действия, ибо многие таблицы, использованные для нормализации, будут вызываться очень редко, но теоретически от этого не должно быть хуже. Плюс, мы в какой-то мере скрываем от пользователей схему, которая владеет таблицами, им куда проще запомнить просто название псевдонима, чем еще и владельца.

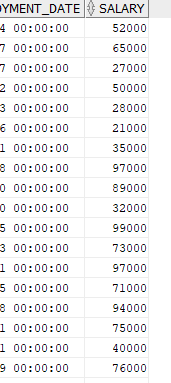
Теперь идем по ролям по порядку:

Бухгалтера:

Создаем им необходимый атрибут (чтобы сделать его NOT NULL, сначала создаем без constraint, заполняем существующие строки рандомными значениями зарплаты, а потом модифицируем до NOT NULL)

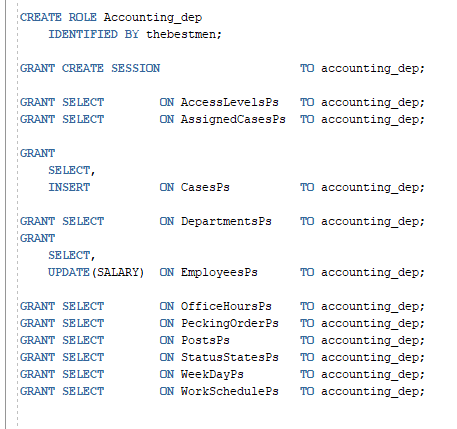




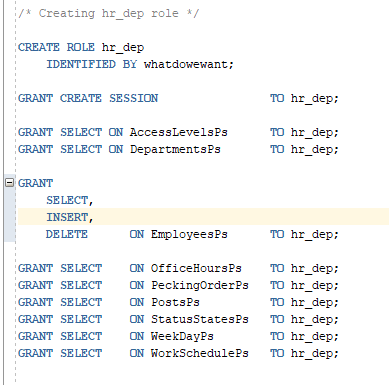




Теперь, собственно, описываем саму роль в соответствии с ТЗ:



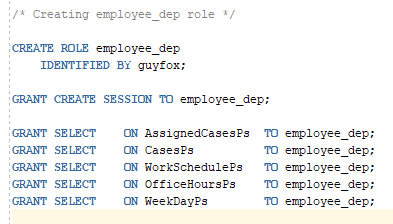
Кадровый отдел:



Отдел руководства (судя по получившемуся конфигу, он самый детально проработанный…):

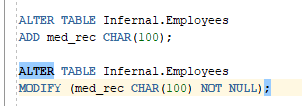


Сотрудники отделов:

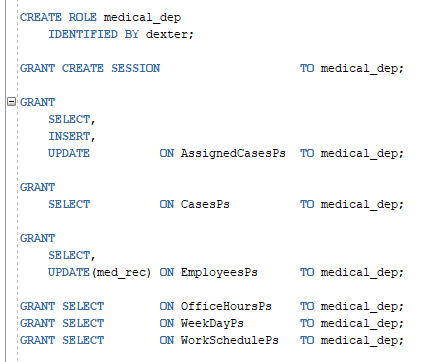


Мед. отдел:

Для создания обязательного атрибута поступаем так же, как и с Salary:



Роль:



1. *Создание схем и профилей*

Создадим для каждой из пяти (не считаем администратора БД) ролей по одной схеме. Из функционала профилей я позаимствовал только кол-во неверных попыток ввода пароля, чтобы дать администратору время на определение причины, но не сутки, как стоит по умолчанию (все-таки, БД является частью гос. учреждения, и хотя бы какие-никакие, а методы защиты да должны быть)

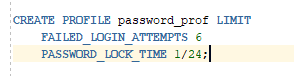
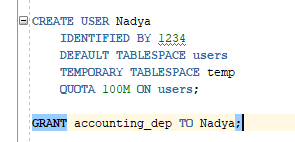


Схема бухгалтера:



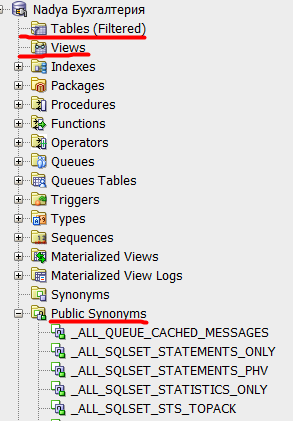
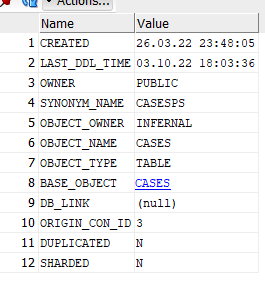
(Судя по всему, роли это тоже объекты, и без предоставления пользователю привилегии использовать роль он ее не увидит)

После создания и подключения к данному пользователю он сможет взять соответствующую роль по паролю:

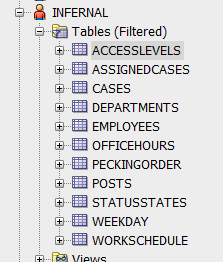




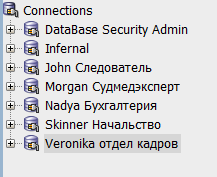
В самом SQL developer (я работаю с ним, т.к. здесь визуально легче представлять результаты) ни в Tables, ни Views, мы ничего не найдем. Необходимо либо искать псевдонимы таблиц в Public Synonyms (но их там куча и больше):

Либо найти предоставленные в доступ таблицы у самого владеющего пользователя:



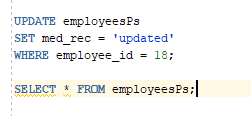
Аналогично (с одной и той же схемой) создаем остальных пользователей и выдаем им соответствующие роли.

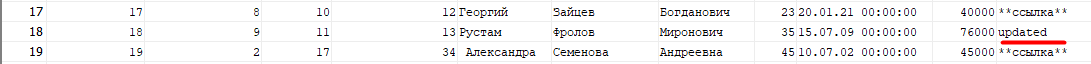


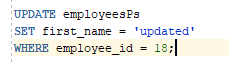
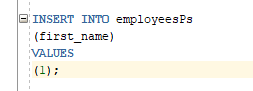
(По поводу квот: оценить реальное кол-во по крайней мере необходимого дискового пространства, лично для меня не просто, не имея какого-то практического опыта. Но можно оценить относительное потребление ресурсов среди ролей. Так, для мед.персона, который может вносить изменения только в мед. данные сотрудников (даже связанные с делами файлы передаются ими через начальника), достаточно минимальных ресурсов. То же касается и обычных сотрудников, чьего участия в изменении самих данных нет. Далее идет кадровый отдел, т.к. они заведуют данными о сотрудниках, но изменения в этих данных происходят не так часто. А вот бухгалтерия и отдел руководства используют БД практически регулярно (не считая самого администратора). Для них необходимо выделить самые большие квоты по памяти и процессорным ресурсам.)

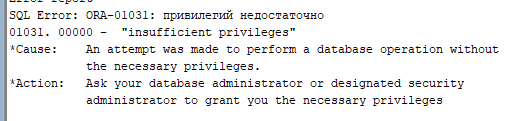
1. *Функциональное тестирование*

Вообще говоря, проверить все роли даже в такой простой БД задача сама по себе достаточно рутинная. Можно, учитывая тот факт, что все, что мы явно не разрешили в роли, запрещено, проверить эти разрешения, но по описанию роли и так понятно, что это все разрешено. Поэтому просто убедимся, что правила сработали, и все не разрешенные действия для пользователей закрыты. Например, тот же медработник может изменять данные о медицинской карте и только их:







(Жаль, а я надеялся, что вдруг сначала проверяется валидность данных, а потом разрешения, была бы потенциальная утечка информации)

*Выводы:*

В результате данной лабораторной работы была реализована базовая сегментация пользовательской работы с рассматриваемой базой данных. Спроектированы и созданы роли для потенциальных пользователей БД, а также добавлены схемы для каждой роли.

Возможно, стоило бы сделать отдельно большие представления из всех таблиц измерений для каждого пользователя, и работать уже с ними, чтобы скрыть ненужную информацию и избавиться от соединений. Однако такие представления наверняка требуют очень больших ресурсов (например, для получения таблицы со всеми данными по сотрудникам, нужно минимум три соединения). И даже для какого-нибудь простого select из двух атрибутов необходимо будет выполнять все представление.

Кроме того, я так и не придумал метода, по которому можно было бы задавать переменные условия для каждой определенной схемы в каждой роли (например, зависимость отображаемых строк в таблице от id сотрудника).