

Настоящее изобретение, в основном, относится к определению доступа пользователя к вычислительным системам, и, в частности, для выборочного и гибкого определения границы доступа каждого из множества пользователей к функциям одной или нескольких прикладных программ или приложений, которые могут выполняться в вычислительной системе.

Предпосылки создания изобретения

В среде типа бюро обслуживания с общедоступным ресурсом, где многие служащие и/или клиенты имеют доступ к вычислительной системе, способной выполнять многочисленные прикладные программы или приложения, часто желательно иметь возможность ограничить доступ некоторых пользователей или классов пользователей к одной или нескольким функциям таких прикладных программ. В используемом здесь смысле термин "функция" включает любую из почти бесконечного числа возможных прикладных задач, например, доступ к данным из таблиц базы данных, создание, просмотр и печать отчетов, и получение и/или посылку электронной почты.

В настоящее время такая гибкость в ограничении доступа пользователей к вычислительным ресурсам пока недоступна. В отношении ограничения доступа к данным, корпорацией «Oracle Corporation» ныне используется один способ, по которому пользователи базы данных ограничены в доступе к конкретным таблицам данных. Корпорация «Oracle Corporation» осуществляет такое ограничение, обеспечивая присваивание "ролей" пользователям, которые ограничивает доступ не конкретно к самим данным, а к таблицам, содержащим эти данные. Необходимость в большей гибкости при ограничении доступа к функциям прикладных программ, включая возможность доступа к данным корпорации «Oracle Corporation» посредством распределения ролей, может быть проиллюстрирована простым примером. Ниже представлена гипотетическая таблица данных по конфиденциальным финансовым сделкам, заключенным клиентами А, В и С утром 15 июня 1998 года, где WDRWL обозначает изъятие, DPST обозначает депозит и PYMNT обозначает оплату.

Таблица 1

	Клиент	Время	Тип сделки	Сумма
1	А	9:15	WDRWL	1000,00
2	В	9:17	DPST	2500,00
3	В	9:24	DPST	1750,00
4	А	9:35	PYMNT	5000,00
5	С	10:02	WDRWL	50,46
6	А	10:41	DPST	106,08
7	С	10:47	PYMNT	530,06

Чтобы подготовить отчет по конфиденциальным сделкам только для клиента А в течение июня месяца, пользователю нужно обратиться к данным в строках 1, 4 и 6, а не в строках 2, 3, 5 и 7. Так как эти данные предназначены исключительно для служебного пользования, крайне

желательно ограничить доступ к данным по сделкам (то есть, к отчету о сделках клиента А).

Кроме того, прикладная программа, используемая для подготовки отчета о прошлых сделках клиента А, может составить несколько различных отчетов, включая отчеты, прогнозирующие будущую деятельность в дополнение к отражению прошлой деятельности. В зависимости от того, кому выдано разрешение на доступ к данным, может оказаться нежелательным разрешение доступа к обоим типам отчетов. Может также оказаться нежелательным разрешение на печать созданных отчетов.

В качестве наиболее близкого технического решения может быть указан патент США № 6,085,191 G01 6F 017/30 от 04.07.2001. Согласно этому патенту известен способ и система для управления доступом пользователей в базы данных. Хотя в описании патента (колонка 3, строка 24-30, колонка 5, строки 15-23 и колонка 16, строки 3-7) декларируется, что заявленные способ и система обеспечивают ограничение доступа к данным в базах данных, однако, ни дальнейшее описание, ни формула изобретения не раскрывают чем это обеспечивается. При этом, независимые пункты формулы изобретения раскрывают лишь то, как уточняется доступ пользователей к дальнейшим приложениям.

Краткое описание изобретения

В соответствии с настоящим изобретением, пользователям вычислительной системы, способной выполнять большое количество прикладных программ, могут быть присвоены определенные "атрибуты". Каждый атрибут представляет собой пару «имя-значение», в которой имя определяет функцию или функции приложения, к которому разрешен доступ (например, доступ к данным, создание отчетов), а значение устанавливает границы доступа (например, ко всем или к некоторым данным). Атрибуты могут быть назначены группам, чтобы облегчить бремя подготовки присваивания отдельных атрибутов для каждого пользователя.

В соответствии с изобретением, предлагаются система и способ для определения возможности пользователя выполнять, по меньшей мере, одну функцию приложения. В соответствии с этой системой и способом, пользователю присваивается, по меньшей мере, один атрибут. Атрибуты хранятся в табулированной базе данных. При выполнении прикладной программы пользователь получает присвоенные ему атрибуты. Эти атрибуты используются в приложении таким образом, что доступ пользователя к функциям приложения определяется в соответствии с полученными атрибутами.

В соответствии с другой целью изобретения, предлагаются система и способ для определения способности пользователя выполнять, по меньшей мере, одну функцию приложения, в которых, по меньшей мере, один атрибут присваивается группе, и эта группа присваивается

пользователю. Группа хранится в таблице в базе данных. При выполнении пользователем прикладной программы он находит присвоенную ему группу. Атрибуты, присвоенные группе, используются при решении прикладной задачи, и доступ пользователя к функциям приложения определяется в соответствии с найденными атрибутами.

Таким образом, целью настоящего изобретения является обеспечение способности выборочно определять доступ к функциям приложения, доступным данному пользователю или группе пользователей вычислительной системы.

Еще одной целью настоящего изобретения является обеспечение большей гибкости по сравнению с той, которая доступна в настоящее время при ограничении пользователя в доступе к данным, содержащимся в базах данных, ориентированных на таблицы.

Эти и другие цели и особенности настоящего изобретения станут более понятны при чтении следующего описания некоторых вариантов изобретения со ссылками на приложенные чертежи; объем изобретения определен приложенной формулой изобретения.

Краткое описание чертежей

Фиг. 1 - блок-схема, изображающая примерную систему в соответствии с настоящим изобретением;

фиг. 2 - блок-схема, изображающая схему группирования системы атрибутов пользователя в соответствии с настоящим изобретением; и

фиг. 3 - блок-схема, изображающая один вариант способа по настоящему изобретению.

Подробное описание изобретения

Атрибуты пользователя

Фиг. 1 представляет собой блок-схему, изображающую примерную систему в соответствии с настоящим изобретением. Компьютер 5 служит для выполнения программ из базы данных 8, которая поддерживает таблицы данных 11, 13 и 15. Различные прикладные программы, ABC, DEF и GHI, также выполняются на компьютере 5, причем некоторые из них обеспечивают средства для поиска и управления данными в таблицах 11, 13 и 15. Каждый из пользователей системы, X, Y и Z, имеет доступ к главному компьютеру 5 через терминалы, представленные персональными компьютерами 18, 21 и 24, соответственно. Согласно настоящему изобретению, пользователям K, Y и Z присваивается один или несколько "атрибутов". Каждый атрибут имеет имя, которое определяет функцию приложения, к которому разрешается доступ (например, функцию доступа к данным в пределах базы данных) и значение, определяющее границы доступа, как будет подробно обсуждено ниже. Если не оговорено особо, используемый здесь термин «атрибут» будет обозначать пару «имя-значение».

Атрибуты поддерживаются в таблице программным обеспечением базы данных 8 и опре-

деляют соответствующие функции пользователей по выполнению прикладных программ ABC, DEF и GHI. Например, двумя полезными атрибутами являются DATA_SCOPE (объем данных) и USER_LEVEL (уровень пользователя). DATA_SCOPE определяет данные, к которым пользователю разрешен доступ, например, в приведенной здесь табл. 1 в этот объем включены возможные значения A, B, C или ALL (все), в который входят данные, приложения, обеспечивая, таким образом, способность назначать отдельные атрибуты и значения для отдельных приложений. В альтернативном варианте, группы атрибутов могут быть созданы независимо от конкретных приложений так, что одна группа может содержать атрибуты для всех приложений. Хотя такая система является менее гибкой при определении доступа к функциям отдельных приложений, она проще для практической реализации. В предпочтительном варианте, таблица APPS (таблица приложений), которая содержит, по меньшей мере, один столбец, именуемый APP_CODE (код приложения), определяет перечень действительных приложений, которые, возможно, имеют соответствующие атрибуты. Снова обращаясь к фигуре 1, мы видим, что действительные значения APP_CODE в этом примере представляют собой приложения ABC, DEF и GHI. Другие столбцы в таблице APPS могут содержать любую информацию, которая необходима для каждого из приложений. Например, в системе меню, которая обеспечивает пиктограммы, по которым пользователь выбирает для выполнения какую-либо прикладную программу, столбец APP_NAME (наименование приложения) может содержать строку символов, которая используется как видимая метка пиктограммы, связанной с прикладной программой в столбце APP_CODE. Каждая группа атрибутов определяет установку нуля или некоторого количества атрибутов для конкретного приложения, идентифицированного как APP_CODE (код приложения). Группа нулевых атрибутов может быть использована, чтобы указать значение атрибута по умолчанию для атрибутов, которые должны присваиваться для этого приложения. Как правило, значения по умолчанию будут наиболее ограничены. Альтернативно, группа нулевых атрибутов может использоваться для того, чтобы указать, что нет никаких функций приложения, идентифицированного кодом APP_CODE, к которому определяется иной доступ, чем функция выполнения прикладной программы.

Пример группирования приведен на фиг. 2, которая изображает более сложную структуру пользователя, чем в обсужденном выше примере в связи с фиг. 1 и табл. 1. Группы атрибутов 106, 115, 130, 145 и 148 из фиг. 2 имеют атрибуты, присвоенные следующим образом:

Таблица 2

Группа атрибутов атрибута	Прикладная программа	Имя атрибута	Значение
106	GHI	DATA_SCOPE USER_LEVEL	ALL ADMIN
115	DEF	DATA_SCOPE USER_LEVEL	ALL REGULAR
130	ABC	DATA_SCOPE DATA_SCOPE USER_LEVEL	A B REGULAR
145	ABC	DATA_SCOPE DATA_SCOPE USER_LEVEL	A B RESTRICTED
148	DEF	DATA_SCOPE DATA_SCOPE USER_LEVEL	B C RESTRICTED

DATA_SCOPE = ОБЪЕМ ДАННЫХ
 USER_LEVEL = УРОВЕНЬ ПОЛЬЗОВАТЕЛЯ
 ALL = ВСЕ ДАННЫЕ
 REGULAR = ОГРАНИЧЕННЫЙ ДОСТУП
 ADMIN = АДМИНИСТРАТИВНЫЙ ДОСТУП

В этом предпочтительном варианте, приложение, к которому применима каждая группа атрибутов, определяется как время создания группы и, в данном обсуждаемом примере, обозначена во втором столбце фиг. 2.

Возвращаясь к фиг. 2, мы видим, что пользователь, принадлежащий к группе атрибутов 106, будет иметь атрибуты DATA_SCOPE 107 и USER_LEVEL 108 со значениями ALL и ADMIN, соответственно, как показано в табл. 2. В соответствии с этими атрибутами, пользователю должен быть разрешен административный доступ к приложению GHI и ему будет предоставлен доступ к данным, относящимся к каждому из клиентов А, В и С при выполнении этого приложения. Пользователь, принадлежащий к группе атрибутов 130, будет иметь три атрибута DATA_SCOPE 133, DATA_SCOPE 136 и USER_LEVEL 139 с значениями А, В и REGULAR, соответственно. В соответствии с этими атрибутами, этот пользователь будет иметь доступ к данным, относящимся к клиентам А или В, и ему будет обеспечен доступ к функциям приложению ABC, которое предопределено для пользователя со статусом REGULAR. Пользователь, которому присвоена группа атрибутов 145, будет иметь ограниченный доступ к приложению ABC и будет иметь доступ к данным, относящимся к клиентам А и В. Пользователь, которому присвоена группа атрибутов 148, будет иметь ограниченный доступ к приложению ABC и ему будет гарантирован доступ к данным, относящимся к клиентам В и С.

Одна или несколько групп атрибутов могут быть присвоены группам получателей атрибутов. На фиг. 2 группа получателей атрибутов 142, например, может состоять из групп атрибутов 145 и 148 и может включать все пары имя-значение атрибута, перечисленные в последних двух столбцах табл. 3:

Таблица 3

Группа получателей атрибутов	Группа атрибутов	Приложение	Имя атрибута	Значение атрибута
142	145	ABC	DATA_SCOPE DATA_SCOPE USER_LEVEL	A B RESTRICTED
	148	DEF	DATA_SCOPE DATA_SCOPE USER_LEVEL	B C RESTRICTED

Группы получателей атрибутов могут входить в другие группы получателей. Это показано на фигуре 2 группой 103 получателей, которая включает все атрибуты в группе получателей 142, также как все атрибуты в группах атрибутов 130 и 115. Группа получателей 100 в верхней части этой фигуры состоит из группы получателей 103 и группы атрибутов 106. Поэтому группа получателей 100 содержит все атрибуты в каждой из пяти групп атрибутов 130, 145, 148, 115 и 106.

Система группирования атрибутов особенно хорошо подходит для присваивания атрибутов служащим с разными уровнями ответственности. Например, группы атрибутов 115, 130, 145 и 148 могут быть присвоены служащим низкого или среднего уровня, в то время как группы получателей 100, 103 и 142 и группа атрибутов 106 могут быть присвоены управляющему персоналу, в обязанности которого входит контроль деятельности служащих более низкого уровня и который использует свои собственные приложения по отношению к группе атрибутов 106.

В этой предпочтительной системе группирования атрибуты, присвоенные группам атрибутов, группам получателей и пользователей, хранятся в таблице атрибутов ATTRIBUTES. Таблица атрибутов имеет три столбца: ASSIGNEE (получатель атрибута), ATTRIBUTE_NAME (имя атрибута) и ATTRIBUTE_VALUE (значение атрибута). В столбце ASSIGNEE могут быть имя группы атрибутов, имя группы получателя или пользователя. ATTRIBUTE_NAME - имя атрибута (например, DATA_SCOPE). ATTRIBUTE_VALUE - конкретное значение для названного атрибута (например, ALL (все)).

Таблица атрибутов поддерживается, используя семь основных команд. Эти типовые команды сформулированы ниже как процедуры Oracle® («Оракул») для использования в среде базы данных «Оракул». Специалисты в данной области понимают, что для других сред могут быть использованы аналогичные команды. В последующем описании параметры заключены в одиночные кавычки, а строки символов заключены в двойные кавычки.

Команда No. 1

```
Attr_utils.create_group ('group_name', 'group_type', 'app_code')
```

'Group_name': имя группы

'Group_type': "ATTRIBUTE" или "ASSIGNEE"

'App_code': если 'group_type' - "ATTRIBUTE", требуется это поле; иначе, команда будет

проигнорирована.

Примечание: 'Group_type' - 'тип группы'

'App_code' - 'код приложения'

"ATTRIBUTE"- "АТТРИБУТ"

"ASSIGNEE" - "ПОЛУЧАТЕЛЬ"

Эта процедура создает группу указанного типа. Команда будет выполнена с сигналом ошибки, если 'Group_name' (имя группы) уже существует как группа или как пользователь «Оракул».

В соответствии с данной процедурой, значения 'group_name' и 'group_type' переводятся в верхний регистр. В результате, в таблицу атрибутов записывается следующий столбец значений:

Установить ASSIGNEE = 'group_name'

Установить ATTRIBUTE_NAME = "ASSIGNEE_TYPE"

Установить АТТРИБУТ VALUE="ATTRIBUTE_GROUP" или группа "ASSIGNEE_GROUP", основанная на значении 'group_type'

Кроме того, если 'group_type' (тип группы) является атрибутом, то в таблицу атрибутов будет внесена другая запись со следующими значениями столбца:

Установить ASSIGNEE = 'group_name'

Установить ATTRIBUTE_NAME = "APP_CODE"

Установить ATTRIBUTE_VALUE = 'app_code'.

Команда No. 2

attr_utils.assign_group ('assignee', 'group_name')
'assignee' ('получатель': пользователь или группа пользователей, которой присваивается имя группы пользователей 'Group_name' («имя группы»))
'Group_name': группа, присваиваемая пользователю.

По этой процедуре одна группа назначается другой группе или пользователю. Команда будет выполнена с сигналом ошибки, если 'assignee' («получатель») не существует как группа получателей или как пользователь. Будет также отмечена ошибка, если не существует имени 'group_name'.

Эта процедура сначала переведет надписи значения 'assignee' и 'group_name' в верхний регистр. Далее в таблицу атрибутов записываются следующие столбцы значений:

Установить ASSIGNEE = 'assignee'

Установить ATTRIBUTE_NAME = "ASSIGNED_GROUP"

Установить ATTRIBUTE-VALUE = 'group_name'.

Команда No. 3

attr_utils.assign_attribute ('assignee', 'attribute_name', 'attribute_value')

'assignee': имя группы атрибутов. Это имя должно принадлежать группе атрибутов.

'attribute_name': имя атрибута

'attribute_value': значения определенного атрибута

По этой процедуре получателю присваивается атрибут с идентифицированным значением. Команда будет выполнена с сигналом ошибки,

если получателя не существует в виде группы атрибутов или если имя атрибута зарезервировано.

Эта процедура сначала переведет надписи значения 'assignee' и 'attribute_name' в верхний регистр и затем делает запись со следующими значениями столбца в таблице атрибутов:

Установить ASSIGNEE = 'assignee'

Установить ATTRIBUTE_NAME =

'attribute_name'

Установить ATTRIBUTE_VALUE = 'attribute_value'.

Команда No. 4

attr_utils.drop_group ('group_name')

'group_name': имя группы или пользователей, которая будет удалена вместе со всеми ссылками к этой группе или к пользователю.

Эта процедура пропустит группу или пользователя вместе со всеми ссылками к группе или пользователю. Команда будет выполнена с сигналом ошибки, если 'group_name' (имя группы) не существует. Эта процедура сначала переведет надпись 'group_name' в верхний регистр и затем удалит все записи в таблице атрибутов, в которой значение в столбце ASSIGNEE соответствует 'group_name'. Кроме того, все записи в таблице атрибутов, которые соответствуют обоим следующим критериям, будут удалены:

a. Значение в столбце ATTRIBUTE_NAME равно значению "ASSIGNED_GROUP"

b. Значение в столбце ATTRIBUTE_VALUE соответствует 'group_name'.

Команда No. 5

attr_utils.rescind_group ('assignee', 'group_name')

'assignee': пользователь или группа получателей, которая удалит атрибут group_name

'group_name': группа, которая будет удалена из атрибута получателя.

Эта процедура удалит имя группы из имени получателя.

Команда будет выполнена с сигналом ошибки, если 'group_name' или 'assignee' не существуют. Эта процедура сначала переведет надписи 'assignee' и 'group_name' в верхний регистр и затем удалит все записи из таблицы ATTRIBUTES, которые соответствуют следующим трем критериям:

a. значение в столбце ATTRIBUTE_NAME равно значению "ASSIGNED_GROUP".

b. значение в столбце ATTRIBUTE_VALUE соответствует значению имени 'group_name'.

c. значение в столбце ASSIGNEE соответствует значению 'assignee'.

Команда No. 6

attr_utils.rescind_attribute ('assignee', 'attribute_name')

'assignee': имя группы атрибутов. Оно должно соответствовать группе атрибутов.

'attribute_name': имя атрибута.

Эта процедура удалит данное имя атрибута 'attribute_name' из 'assignee'. Команда будет выполнена с сигналом ошибки, если имени атрибута attribute_name или получателя не существует или если имя атрибута attribute_name зарезервировано.

Эта процедура сначала переведет надписи 'assignee' и 'attribute_name' в верхний регистр и затем удалит все записи в таблице атрибутов, которые соответствуют обоим следующим критериям:

a. значение в столбце ASSIGNEE соответствует 'assignee'.

b. значение в столбце ATTRIBUTE_NAME соответствует 'attribute_name'.

Команда No. 7

attr_utils.update_attribute ('assignee', 'attribute_name', 'attribute_value')

'assignee': имя группы атрибутов. Это должна быть группа атрибутов.

'attribute_name': имя атрибута.

'attribute_value': новое значение заданного атрибута.

Эта процедура обновит определенное значение атрибута 'attribute_value' для имени атрибута идентифицированного как 'assignee' и 'attribute_name'. Команда будет выполнена с сигналом ошибки, если имени attribute_name или получателя assignee не существует или если имя attribute_name зарезервировано.

Эта процедура сначала переведет надписи 'assignee' и 'attribute_name' в верхний регистр и затем обновляет таблицу атрибутов, переводя столбец ATTRIBUTE_VALUE в 'attribute_value' для всех записей, которые соответствуют обоим следующим критериям:

значение в столбце ASSIGNEE соответствует 'assignee'.

значение в столбце ATTRIBUTE_NAME соответствует 'attribute_name'.

Вышеупомянутые команды используются для поддержки таблицы атрибутов. Как показано в командах 1 и 2 в предпочтительном варианте, в таблице атрибутов используются несколько зарезервированных имен ATTRIBUTE_NAMES, чтобы отыскать конкретную информацию, которую нужно использовать в системе. Имя ATTRIBUTE_NAME кода "APP_CODE" автоматически присваивается группе атрибутов для идентификации приложения, связанного с той группой. Имя ATTRIBUTE_NAME группы "ASSIGNED_GROUP" используется для присваивания атрибутов группе атрибутов и группам получателей и для присваивания групп атрибутов и групп получателей пользователям. Признаки ATTRIBUTE_NAME типа "ASSIGNEE_TYPE" используются, чтобы выявить, является ли эта группа группой атрибутов или группой получателей. Эта процедура проверяет правильность любых параметров ATTRIBUTE_NAME, чтобы убедиться, что они не зарезервированы, и выдаст ошибку, если сделана

попытка использовать зарезервированное имя ATTRIBUTE_NAME.

Для примера часть таблицы атрибутов, относящейся к группе получателей 142 на фиг. 2 представлена в виде следующей табл. 4.

Таблица 4

ПОЛУЧАТЕЛЬ	ИМЯ АТТРИБУТА	ЗНАЧЕНИЕ АТТРИБУТА
ASSIGNEE GROUP 142	ASSIGNEE TYPE	ASSIGNEE GROUP
ATTRIBUTE GROUP 145	ASSIGNEE TYPE	ATTRIBUTE GROUP
ATTRIBUTE GROUP 145	APP CODE	ABC
ATTRIBUTE GROUP 148	ASSIGNEE TYPE	ATTRIBUTE GROUP
ATTRIBUTE GROUP 148	APP CODE	ABC
ATTRIBUTE GROUP 145	DATA SCOPE	A
ATTRIBUTE GROUP 145	DATA SCOPE	B
ATTRIBUTE GROUP 145	USER LEVEL	RESTRICTED
ATTRIBUTE GROUP 148	DATA SCOPE	B
ATTRIBUTE GROUP 148	DATA SCOPE	C
ATTRIBUTE GROUP 148	USER LEVEL	RESTRICTED
ASSIGNEE GROUP 142	ASSIGNED GROUP	ATTRIBUTE GROUP 142
ASSIGNEE GROUP 142	ASSIGNED GROUP	ATTRIBUTE GROUP 145

В обсужденном выше альтернативном варианте изобретения, где группы атрибутов не ограничены конкретными приложениями, процедура создания группы (create_group) не требует ввода кода 'app_code', и атрибуты APP_CODE не будут поддерживаться в таблице атрибутов. Тем не менее, можно управлять общим доступом к приложениям, используя дополнительную таблицу, в которой коды APP_CODE присваиваются непосредственно пользователям. Если пользователю не был присвоен конкретный код APP_CODE, соответствующее приложение будет ему недоступно.

Иерархия «родитель-дочь»

Способность присваивать одной группе получателей множественные группы атрибутов и/или группы получателей часто приводит к присваиванию группе или пользователю перекрывающихся друг друга, повторяющихся и даже противоречивых значений одного и того же атрибута. Например, группа получателей 100, показанная в верхней части фигуры 2, включает каждый атрибут в эту фигуру и поэтому, как показано в вышеприведенной таблице 2, включает для того же самого приложения DEF отдельные значения B, C и ALL для атрибута DATA_SCOPE и отдельные значения REGULAR и RESTRICTED для атрибута USER_LEVEL. По этой причине, в предпочтительном варианте, установлены правила иерархии атрибута, в которых для каждого значения атрибута присваивается "parent value" (родительское значение), в результате чего, например, значение B для атрибута DATA_SCOPE присваивается как родительское значение ALL. В тех практических случаях, когда вызываются атрибуты, присвоенные пользователю, а родительское значение и дочернее значение выданы для одного и того же атрибута и приложения, родительское значение будет поддерживаться, а дочернее значение игнорироваться. Кроме того, будут аннулированы все повторные значения.

Родительские и дочерние значения поддерживаются в таблице уровней атрибутов ATTRIBUTE_LEVELS, которая имеет три столбца: ATTRIBUTE_NAME, CHILD_VALUE и PARENT_VALUE. ATTRIBUTE_NAME - имя атрибута (например, DATA_SCOPE). CHILD_VALUE -

фактическое значение атрибута (например, C). PARENT_VALUE - значение, у которого фактическое значение представляет собой подмножество (например, ALL). Для случая, где фактическое значение атрибута является самым высоким в иерархии, например ALL, присвоенное родительское значение представляет собой NULL (нуль). Таблица уровней атрибутов поддерживается, используя четыре основные команды. Эти примерные команды ниже сформированы как процедуры «Оракул» для использования в среде базы данных «Оракул». Специалисты в данной области понимают, что аналогичные команды могут быть сформированы и для других баз данных. Отметим, что в последующем описании, параметры заключены в одиночные кавычки, и строки символов - в двойные кавычки.

Команда No. 1

```
attr_utils.add_attr_level      ('attribute_name',
'child_value', 'parent_value')
'attribute_name': имя атрибута
'child_value': дочернее значение для заданного имени 'attribute_name'
'parent_value': родительское значение заданного значения 'child_value' для заданного имени 'attribute_name'
```

Эта процедура добавит новый уровень атрибута для заданных параметров. Значение 'parent_value' может быть нулевым, если значение 'child_value' имеет самый высокий уровень полномочий указанного имени 'attribute_name'. Команда будет выполнена с сигналом ошибки, если родительское значение не нулевое или не существует.

Эта процедура сначала переведет 'attribute_name' в верхний регистр и затем сделает запись следующих значений столбца в таблице ATTRIBUTE_LEVELS:
Установить ATTRIBUTE_NAME= 'attribute_name'
Установить CHILD_VALUE = 'child_value'
Установить PARENT_VALUE = 'parent_value'.

Команда No. 2

```
attr_utils.update_attr_level  ('attribute_name',
'child_value', 'parent_value')
'attribute_name': имя атрибута
'child_value': дочернее значение для заданного имени attribute_name
'parent_value': родительское значение заданного дочернего значения для заданного имени атрибута.
```

Эта процедура обновит родительское значение для указанных параметров. Значение 'parent_value' может быть нулевым, если значение 'child_value' имеет самый высокий уровень полномочий для заданного имени 'attribute_name'. Команда будет выполнена с сигналом ошибки, если родительское значение не является нулевым или не существует и если сочетания имени

attribute_name и значения child_value также не существует.

Эта процедура сначала переведет 'attribute_name' в верхний регистр и затем обновит таблицу ATTRIBUTE_LEVELS, устанавливая значение столбца PARENT_VALUE в столбец 'parent_value' для всех записей, которые соответствуют обоим следующим критериям:

- Значение в столбце ATTRIBUTE_NAME соответствует значению 'attribute_name'.
- Значение в столбце CHILD_VALUE соответствует значению 'child_value'.

Команда No. 3

```
attr_utils.delete_attr_levels ('attribute_name',
child_value')
'attribute_name': имя атрибута
'child_value': дочернее значение для заданного имени attribute_name.
```

Эта процедура удалит уровень атрибута и все уровни дочернего атрибута для заданных параметров. Возникнут ошибки, если этой комбинации не существует.

Эта процедура сначала переведет термин 'attribute_name' в верхний регистр и затем удалит все записи в таблице уровней атрибутов, которые являются «дочерними» в отношении указанной пары 'attribute_name' и 'child_value'. В качестве примера можно использовать следующий оператор на языке структурированных запросов SQL, чтобы выполнить эти первые два этапа:

```
DELETE ATTRIBUTE_LEVELS
WHERE (attribute_name, CHILD_VALUE) IN
(SELECT attribute_name, CHILD_VALUE
FROM ATTRIBUTE_LEVELS
START WITH PARENT_VALUE=P.CHILD_VALUE
AND ATTRIBUTE_NAME=UPPER(P.ATTRIBUTE_NAME)
CONNECT BY PARENT_VALUE=PRIOR CHILD_VALUE
AND ATTRIBUTE_NAME=PRIOR ATTRIBUTE_NAME).
This procedure will delete all records in the ATTRIBUTE_LEVELS
(УДАЛИТЬ ATTRIBUTE_LEVELS,
ГДЕ (attribute_name, CHILD_VALUE) В
КОМАНДЕ (ВЫБРАТЬ attribute_name, CHILD_VALUE
ИЗ УРОВНЕЙ АТТРИБУТОВ
НАЧАТЬ С РАВЕНСТВА PARENT_VALUE=P.CHILD_VALUE
И ATTRIBUTE_NAME = UPPER (P.ATTRIBUTE_NAME)
СОЕДИНИТЬ С ПОМОЩЬЮ РАВЕНСТВ PARENT_VALUE =
PRIOR CHILD_VALUE И
ATTRIBUTE_NAME=PRIOR ATTRIBUTE_NAME).
```

Эта процедура удалит все записи в таблице уровней атрибутов, которые соответствуют обоим соответствующим критериям:

- Значение в столбце ATTRIBUTE_NAME соответствует 'attribute_name'.
- Значение в столбце CHILD VALUE соответствует 'child_value'.

Команда No. 4

```
attr_utils.delete_all_levels ('attribute_name')
'attribute_name': имя атрибута
```

Эта процедура удалит все уровни атрибутов для заданного имени attribute_name. Команда бу-

дет выполнена с сигналом ошибки, если имени attribute_name не существует.

Эта процедура сначала переведет надпись 'attribute_name' в верхний регистр и затем удалит все записи в таблице уровней атрибутов, в которых значение в столбце ATTRIBUTE_NAME соответствует 'attribute_name'.

Таблица уровней атрибутов поддерживается, используя вышеупомянутые команды.

Примерная таблица уровней атрибутов для выборки упомянутых выше атрибутов может быть представлена следующим образом.

Таблица 5

ИМЯ АТТРИБУТА	ДОЧЕРНЕЕ ЗНАЧЕНИЕ	РОДИТЕЛЬСКОЕ ЗНАЧЕНИЕ
DATA_SCOPE	A	ALL
DATA_SCOPE	B	ALL
DATA_SCOPE	C	ALL
DATA_SCOPE	ALL	NULL
USER_LEVEL	RESTRICTED	REGULAR
USER_LEVEL	REGULAR	ADMIN
USER_LEVEL	ADMIN	NULL

DATA_SCOPE - объем данных

USER_LEVEL - уровень пользователя

RESTRICTED - ограниченный

ALL- все данные

REGULAR - обычный объем данных

ADMIN- административный объем

NULL- данные отсутствуют

В соответствии с табл. 5, любые присваивания атрибута DATA_SCOPE, имеющего значения A, B или C, будут отброшены, если доступ к данным в полном объеме обеспечивается одному и тому же пользователю для одного и того же приложения. Точно так же присваивания более низких уровней пользователя будут отброшены в пользу самого высокого присвоенного значения.

Предпочтительно, чтобы значение атрибута было отражено в таблице атрибутов, но не было определено в таблице уровней атрибутов, в результате чего оно будет обработано, как если бы оно было определено в таблице уровней атрибутов с нулевым родительским значением и без другого значения, имеющего присвоенное родительское значение. Дальнейшая минимизация может быть достигнута, признавая, что полный набор присвоенных значений на более низком уровне может быть заменен более высоким значением уровня. Например, если присвоены значения A, B и C в столбце DATA_SCOPE, система может вернуть значение ALL. Эта минимизация должна быть выполнена только в том случае, когда полный набор более низких значений уровня фактически представляет то же самое, что и более высокое значение уровня, поскольку существует вероятность, что более высокое значение уровня может быть больше суммы более низких значений уровня.

Работа системы атрибутов пользователя

При последующем обсуждении примерного использования системы атрибутов пользователя, в соответствии с настоящим изобретением, мы будем ссылаться на фиг. 3. В соответствии с предпочтительным вариантом изобретения, дос-

туп к используемым приложениям в среде базы данных осуществляется под управлением исходного графического интерфейса пользователя (IGUI). В качестве примеров с помощью интерфейса IGUI можно обрабатывать сайты домашних страниц в Интернете и страницы запуска локальной сети. Однако перед тем, как получить доступ к приложениям, пользователь, как правило, входит в вычислительную систему, показанную в виде блока 201 и, если вход в систему выполнен правильно, вычислительная система идентифицирует пользователя. В блоке 204 исходный графический интерфейс пользователя (IGUI) отыскивает приложения, доступные данному пользователю. В предпочтительном варианте IGUI делает это, обращаясь к таблице атрибутов, по стрелке 205, которая содержит группы, присваиваемые пользователю. Как обсуждено выше, в предпочтительном варианте группы включают присвоенные группы атрибутов, что, в свою очередь, включает спецификацию доступных приложений. В системе базы данных «Оракул», вместо поиска по всей таблице атрибутов, которая может быть весьма объемной, можно подготовить «обзоры», содержат результаты часто используемых поисков (например, многократно используемые атрибуты для конкретного пользователя).

В альтернативном варианте, где атрибуты не ограничены конкретными приложениями, приложения, доступные конкретному пользователю, могут быть сохранены в отдельной таблице и извлекаться из нее по мере надобности.

По стрелке 205 интерфейс IGUI также имеет доступ к таблице APPS действительных приложений, которая содержит столбец кода APP_CODE, чтобы сообщить интерфейсу, какая строка символов будет изображена для пользователя в блоке 207. В этой строке интерфейс IGUI отображает для пользователя доступные приложения. В блоке 210, пользователь выбирает одно из доступных приложений. В блоке 213 интерфейс IGUI обращается к таблице атрибутов по стрелке 214 и отыскивает атрибуты пользователя для выбранного приложения. Снова можно использовать систему базы данных «Оракул» для извлечения атрибутов. Кроме того, интерфейс IGUI может обращаться к таблице уровней атрибутов, чтобы уменьшить число атрибутов.

По стрелке 215 интерфейс IGUI передает нужные атрибуты к приложению и, в блоке 216, выполняется прикладная программа с приложением, предписывающим атрибуты. Если пользователь пытается превысить пределы доступа, определенные атрибутами, появляется сигнал ошибки или предупреждение. Специалисты в данной области понимают, что в альтернативном варианте приложение может извлекать атрибуты непосредственно, без помощи графического интерфейса.

Обзоры

Как обсуждено выше, в настоящем изобретении можно использовать обзоры «Оракул», чтобы отыскивать нужные данные и делать записи в таблицах.

Ниже приведен листинг двенадцати обзоров, которые могут быть полезны при реализации изобретения. Предусмотрены объяснения каждого обзора и примеры оператора SQL.

Обзор No. 1

V_ATTRIBUTE_APP_CODES

Этот обзор выдает перечень всех выбранных кодов приложений, установленных в системе атрибутов пользователя. Этот обзор выполняет условие SELECT (выбрать) с индивидуальным оператором DISTINCT для записей, где зарезервировано имя ATTRIBUTE_NAME кода 'APP_CODE'

Пример оператора SQL:

```
CREATE OR REPLACE VIEW V_ATTRIBUTE_APP_CODES AS
SELECT DISTINCT ASSIGNEE
ATTRIBUTE_VALUE APP_CODE
FROM ATTRIBUTES
WHERE ATTRIBUTE NAME = 'APP_CODE'
```

ВЫБРАТЬ ИЛИ ЗАМЕНИТЬ ОБЗОР V_ATTRIBUTE_APP_CODES КАК ВЫБОР ОТДЕЛЬНОГО ПОЛУЧАТЕЛЯ,
ЗНАЧЕНИЕ АТТРИБУТА КОДА ПРИЛОЖЕНИЯ
ИЗ АТТРИБУТОВ,
ГДЕ ИМЯ АТТРИБУТА = КОД АТТРИБУТА 'APP_CODE'

Обзор No. 2

V_ATTRIBUTE_GROUPS_ATTR

Этот обзор выдает перечень всех заданных групп атрибутов. Этот обзор выполняет условие выбора SELECT с индивидуальным оператором DISTINCT для записей, где ATTRIBUTE_NAME зарезервировано, как ATTRIBUTE_NAME типа получателя 'ASSIGNEE_TYPE' и значение ATTRIBUTE_VALUE представляет собой группу 'ATTRIBUTE_GROUP'.

Пример оператора SQL:

```
CREATE OR REPLACE VIEW V_ATTRIBUTE_GROUPS_ATTR
AS SELECT DISTINCT ASSIGNEE GROUP_NAME
FROM ATTRIBUTES
WHERE ATTRIBUTE NAME = 'ASSIGNEE_TYPE'
AND ATTRIBUTE_VALUE = 'ATTRIBUTE_GROUP'
(СОЗДАТЬ ИЛИ ЗАМЕНИТЬ ОБЗОР V_ATTRIBUTE_GROUPS_ATTR
КАК ВЫБОР ИМЕНИ ОТДЕЛЬНОЙ ГРУППЫ ПОЛУЧАТЕЛЕЙ ИЗ АТТРИБУТОВ,
ГДЕ ИМЯ АТТРИБУТА = 'ASSIGNEE_TYPE'
И ЗНАЧЕНИЕ АТТРИБУТА = 'ATTRIBUTE_GROUP'.
```

Обзор No. 3

V_ATTRIBUTE_GROUPS_ASSIGN

Этот обзор выдает перечень всех отдельных групп получателей. Этот обзор выполняет команду SELECT («выбрать») с оператором DISTINCT для записей, где зарезервировано имя ATTRIBUTE_NAME типа 'ASSIGNEE_TYPE', и значение ATTRIBUTE_VALUE является группой 'ASSIGNEE_GROUP'.

Пример оператора SQL:

```
CREATE OR REPLACE VIEW
V_ATTRIBUTE_GROUPS_ASSIGN AS
```

```
SELECT DISTINCT ASSIGNEE GROUP_NAME
FROM ATTRIBUTES
WHERE ATTRIBUTE_NAME = 'ASSIGNEE_TYPE'
AND ATTRIBUTE_VALUE = 'ASSIGNEE_GROUP';
```

(СОЗДАТЬ ИЛИ ЗАМЕНИТЬ ОБЗОР

```
V_ATTRIBUTE_GROUPS_ASSIGN КАК
ВЫБОР ИМЕНИ ГРУППЫ DISTINCT ASSIGNEE
GROUP_NAME ИЗ АТТРИБУТОВ,
ГДЕ ИМЯ АТТРИБУТА ATTRIBUTE_NAME = 'ASSIGNEE_TYPE' И
ЗНАЧЕНИЕ АТТРИБУТА ATTRIBUTE_VALUE = 'ASSIGNEE_GROUP')
```

Обзор No. 4

V_ATTRIBUTE_USERS

Этот обзор выдает перечень всех отдельных групп получателей. Этот обзор выполняет команду SELECT (выбрать) с оператором DISTINCT для записей, где получатель ASSIGNEE равен имени пользователя USER_NAME из словаря «Оракул» таблицы данных всех пользователей ALL_USERS.

Пример оператора SQL:

```
CREATE OR REPLACE VIEW V_ATTRIBUTE_USERS AS
SELECT DISTINCT ASSIGNEE USERID
FROM ATTRIBUTES,
ALLUSERS
WHERE ASSIGNEE = USERNAME;
```

СОЗДАТЬ ИЛИ ЗАМЕНИТЬ ОБЗОР V_ATTRIBUTE_USERS
ВЫБРАТЬ ОТДЕЛЬНОГО ПОЛУЧАТЕЛЯ ASSIGNEE USERID
ИЗ
АТТРИБУТОВ ВСЕ ПОЛЬЗОВАТЕЛИ ALL_USERS,
ГДЕ ASSIGNEE = USER_NAME (имя пользователя);

Обзор No.5

V_ATTRIBUTE_GROUPS_ALL

Этот обзор выдает перечень всех отдельных групп в системе. Этот обзор включает группы атрибутов и получателей. Этот обзор выполняет команду SELECT (выбрать) с оператором DISTINCT для записей, когда получатель не является пользователем атрибута.

Пример оператора SQL:

```
CREATE OR REPLACE VIEW V_ATTRIBUTE_GROUPS_ALL AS
SELECT DISTINCT ASSIGNEE GROUP_NAME
FROM ATTRIBUTES,
V_ATTRIBUTE_USERS
WHERE ASSIGNEE = USERID(+)
AND USERID IS NULL;
```

(СОЗДАТЬ ИЛИ ЗАМЕНИТЬ ОБЗОР
V_ATTRIBUTE_GROUPS_ALL КАК ВЫБОР ИМЕНИ
ГРУППЫ GROUP_NAME ОТДЕЛЬНОГО
ПОЛУЧАТЕЛЯ ИЗ АТТРИБУТОВ
V_ATTRIBUTE_USERS,
ГДЕ ASSIGNEE = ASSIGNEE = USERID(+) (идентификатор пользователя) И
ИДЕНТИФИКАТОР ПОЛЬЗОВАТЕЛЯ USERID РАВЕН
НУЛЮ).

Обзор No. 6

V_USER_GROUPS

Этот обзор выдает перечень всех групп, присвоенных пользователю, который в данный момент подключен к базе данных «Оракул». Результаты включают группы, присвоенные непосредственно пользователю, а также группы, присвоенные пользователю косвенным образом. Иными словами, группы, которые присвоены группе получателей ASSIGNEE_GROUP, присвоены и пользователю. Этот обзор выполняет запрос типа генеалогического дерева, используя оператор CONNECT BY.

Пример оператора SQL:

```
CREATE OR REPLACE VIEW V_USER_ATTR_APPS AS
SELECT ATTR.ATTRIBUTE_NAME,
ATTR.ATTRIBUTE_VALUE,
APPS.APP_CODE
FROM V_USER_GROUPS GROUPS,
ATTRIBUTES ATTR,
V_ATTRIBUTE_APP_CODES APPS
WHERE GROUPS.ATTRIBUTE_VALUE = ATTR.ASSIGNEE
AND ATTR.ASSIGNEE = APPS.ASSIGNEE
AND ATTR.ATTRIBUTE_NAME NOT IN
```

```
(СОЗДАТЬ ИЛИ ЗАМЕНИТЬ ОБЗОР VIEW V_USER_
GROUPS КАК
ВЫБОР ИМЕНИ ATTRIBUTE_NAME,
ЗНАЧЕНИЯ ATTRIBUTE_VALUE,
КОДА ПРИЛОЖЕНИЯ APPS.APP_CODE
(ВЫБРАТЬ ИЛИ ЗАМЕНИТЬ ОБЗОР V_USER_GROUPS
КАК
ВЫБОР ИМЕНИ ATTRIBUTE_NAME = 'ASSIGNEE TYPE'
НАЧАТЬ С АТРИБУТА ПОЛУЧАТЕЛЬ=ПОЛЬЗОВАТЕЛЬ
СОЕДИНИТЬ ЧЕРЕЗ ПОЛУЧАТЕЛЯ = ДО ЗНАЧЕНИЯ
АТРИБУТА
ИМЯ АТРИБУТА = 'ASSIGNED_GROUP'
ИЗ ГРУПП V_USER_GROUPS,
АТРИБУТОВ ATTRIBUTES ATTR,
КОДОВ ПРИЛОЖЕНИЯ V_ATTRIBUTE_APP_CODES
APPS,
ГДЕ GROUPS.ATTRIBUTE_VALUE = ATTR.ASSIGNEE
И ATTR.ASSIGNEE = APPS.ASSIGNEE
И ATTR.ATTRIBUTE_NAME НЕ НАХОДИТСЯ В ПАРА-
МЕТРАХ
('ASSIGNED_GROUP', 'APP_CODE', ASSIGNEE_TYPE)).
```

Обзор No. 7

V_USER_GROUPS

Этот обзор выдает перечень всех групп, присвоенных пользователю, который в данный момент подключен к «Оракулу» вместе с соответствующим кодом приложения. Этот обзор объединяет перечень групп, присвоенных пользователю (V_USER_GROUPS), таблицу атрибутов и перечень ATTRIBUTE_GROUPS с соответствующими значениями кодов приложений APP_CODE (V_ATTRIBUTE_APP_CODES).

Пример оператора SQL:

```
CREATE OR REPLACE VIEW V_USER_ATTR_APPS AS
SELECT ATTR.ATTRIBUTE_NAME,
ATTR.ATTRIBUTE_VALUE,
APPS.APP_CODE
FROM V_USER_GROUPS GROUPS,
ATTRIBUTES ATTR,
V_ATTRIBUTE_APP_CODES APPS
WHERE GROUPS.ATTRIBUTE_VALUE=ATTR.ASSIGNEE
AND ATTR.ASSIGNEE = APPS.ASSIGNEE
```

```
AND ATTR.ATTRIBUTE_NAME NOT IN
('ASSIGNED_GROUP','APP_CODE','ASSIGNEE_TYPE')
```

```
(СОЗДАТЬ ИЛИ ЗАМЕНИТЬ VIEW V_USER_ATTR_APPS
КАК
ВЫБОР ИМЕНИ ATTR.ATTRIBUTE_NAME,
ВЕЛИЧИНЫ ATTR.ATTRIBUTE_VALUE,
КОДА APPS.APP_CODE
ИЗ ГРУПП V_USER_GROUPS GROUPS,
ATTRIBUTES ATTR,
V_ATTRIBUTE_APP_CODES APPS
ГДЕ GROUPS.ATTRIBUTE_VALUE = ATTR.ASSIGNEE
И ATTR.ASSIGNEE = APPS.ASSIGNEE
И ИМЯ ATTR.ATTRIBUTE_NAME НЕ НАХОДИТСЯ В
ПАРАМЕТРАХ
('ASSIGNED_GROUP','APP_CODE','ASSIGNEE_TYPE'))
```

Обзор No. 8

V_USER_ATTR_HIGHEST_VALUES

Этот обзор выдает перечень значений высокого уровня атрибутов для соответствующих имен атрибутов ATTRIBUTE_NAME. Этот обзор может содержать двойные входные параметры, при этом обсуждаемый ниже обзор V_USER_ATTRIBUTES, извлекает перечень этих отдельных значений. Это обзор передает код, имя и значение APP_CODE, ATTRIBUTE_NAME и ATTRIBUTE_VALUE для каждого атрибута, присвоенного данному пользователю, в функции ATTR_UTILS.HIGHEST_VALUE. Функция - это то же самое, что и процедура за исключением того, что она может быть выполнена как часть запроса и выдает нужное значение.

Здесь функция выдает самое высокое родительское значение, присвоенное в данный момент пользователю. Описание этой функции следует за примером оператора SQL для этого обзора.

Пример оператора SQL:

```
CREATE OR REPLACE VIEW
V_USER_ATTR_HIGHEST_VALUES AS
SELECT APP_CODE,
ATTRIBUTE_NAME,
SUBSTR(ATTR_UTILS.HIGHEST_VALUE(APP_CODE,
ATTRIBUTE_NAME, ATTRIBUTE_VALUE), 1, 30)
ATTRIBUTE_VALUE
FROM V_USER_ATTR_APPS;
(СОЗДАТЬ ИЛИ ЗАМЕНИТЬ ОБЗОР
V_USER_ATTR_HIGHEST_VALUES КАК
ВЫБОР КОДА APP_CODE,
ИМЕНИ АТРИБУТА ATTRIBUTE_NAME,
КОДА ПРИЛОЖЕНИЯ, ИМЕНИ АТРИБУТА, ЗНАЧЕНИЯ
АТРИБУТА
SUBSTR(ATTR_UTILS.HIGHEST_VALUE(APP_CODE,
ATTRIBUTE_NAME, ATTRIBUTE_VALUE), 1, 30)
ЗНАЧЕНИЯ АТРИБУТА ATTRIBUTE_VALUE
ИЗ ПРИЛОЖЕНИЙ V_USER_ATTR_APPS;
```

Функция ATTR_UTILS.HIGHEST_VALUE требует ввода трех параметров: кода приложения APP_CODE, имени атрибута ATTRIBUTE_NAME и значения атрибута ATTRIBUTE_VALUE.

Эта функция сначала загрузит внутреннюю таблицу значений атрибутов, включающую все значения атрибутов, присвоенных данному пользователю, код атрибута APP_CODE и имя атрибута ATTRIBUTE_NAME. Эта функция

может использовать один из существующих обзоров для выполнения этой операции (т.е. обзор V_USER_ATTR_VALUE_LEVELS). Затем функция использует значение текущего атрибута и найдет все родительские записи для этого значения. Функция будет использовать оператор SQL, который выглядит следующим образом:

```
SELECT PARENT_VALUE
FROM ATTRIBUTE_LEVELS
WHERE ATTRIBUTE_NAME = P_ATTRIBUTE_NAME
START WITH CHILD_VALUE = P_ATTRIBUTE_VALUE
CONNECT BY
CHILD_VALUE = PRIOR PARENT_VALUE
ORDER BY LEVEL;
(ВЫБРАТЬ РОДИТЕЛЬСКОЕ ЗНАЧЕНИЕ
ИЗ УРОВНЕЙ АТТРИБУТА, ГДЕ ИМЯ АТТРИБУТА =
P_ATTRIBUTE_NAME НАЧАТЬ С ДОЧЕРНЕГО ЗНАЧЕНИЯ =
P_ATTRIBUTE_VALUE СОЕДИНИТЬ С ДОЧЕРНИМ ЗНАЧЕНИЕМ =
ПРЕДЫДУЩЕЕ РОДИТЕЛЬСКОЕ ЗНАЧЕНИЕ, ВЫДАННОЕ ПО УРОВНЮ)
```

Затем функция сравнивает каждое родительское значение с записями в таблице значений атрибутов, чтобы определить, является ли одно из других значений атрибутов родительским отчетом, т.е. родителем, дедушкой и т.д. значения текущего атрибута. Если это так, возвращается значение атрибута более высокого уровня. В противном случае, возвращается значение текущего атрибута.

Обзор No. 9 V_ATTR_VALUE_LEVELS

Этот обзор выдает перечень имен ATTRIBUTE_NAME, соответствующих значений ATTRIBUTE_VALUE и соответствующий уровень значения. Например, самый высокий уровень значения ATTRIBUTE_VALUE будет обозначен, как уровень ATTRIBUTE_LEVEL 1, в то время как дочерний уровень имеет уровень 2. Этот обзор выполняет запрос типа генеалогического дерева, использующего оператор CONNECT BY.

Пример SQL:

```
CREATE OR REPLACE VIEW V_ATTR_VALUE_LEVELS AS
SELECT ATTRIBUTE_NAME,
CHILD_VALUE ATTRIBUTE_VALUE,
LEVEL VALUE_LEVEL FROM ATTRIBUTE_LEVELS
START WITH PARENT_VALUE IS NULL
CONNECT BY PARENT_VALUE = PRIOR CHILD_VALUE
AND ATTRIBUTE_NAME = PRIOR ATTRIBUTE_NAME;
(СОЗДАТЬ ИЛИ ЗАМЕНИТЬ ОБЗОР V_ATTR_VALUE_LEVELS
КАК ВЫБОР ИМЕНИ АТТРИБУТА ATTRIBUTE_NAME,
ДОЧЕРНЕГО ЗНАЧЕНИЯ АТТРИБУТА ATTRIBUTE_VALUE,
ВЕЛИЧИНЫ УРОВНЯ VALUE_LEVEL
ИЗ УРОВНЕЙ АТТРИБУТА ATTRIBUTE_LEVELS
НАЧАТЬ С НУЛЕВОГО РОДИТЕЛЬСКОГО ЗНАЧЕНИЯ
СОЕДИНИТЬ РОДИТЕЛЬСКОЕ ЗНАЧЕНИЕ, РАВНОЕ
ПРЕДЫДУЩЕМУ ДОЧЕРНЕМУ ЗНАЧЕНИЮ, С ИМЕНЕМ
АТТРИБУТА ATTRIBUTE_NAME = PRIOR ATTRIBUTE_NAME).
```

Обзор No. 10 V_USER_ATTR_VALUE_LEVELS

Этот обзор выдает перечень всех атрибутов, присвоенных пользователю, и их соответствующий уровень. Это обзор объединит перечень атрибутов, присвоенных пользователю, который в данный момент соединен с базой данных «Оракул» (V_USER_ATTR_APPS) и перечень атрибутов и их соответствующих уровней (V_ATTR_VALUE_LEVELS).

Пример оператора SQL:

```
CREATE OR REPLACE VIEW V_USER_ATTR_VALUE_LEVELS
AS SELECT APP_CODE,
ATTR.ATTRIBUTE_NAME,
ATTR.ATTRIBUTE_VALUE,
NVL(VALUE_LEVEL, 1) VALUE_LEVEL FROM
V_USER_ATTR_APPS ATTR,
V_ATTR_VALUE_LEVELS LVL
WHERE
ATTR.ATTRIBUTE_NAME = LVL.ATTRIBUTE_NAME(+)
AND ATTR.ATTRIBUTE_VALUE = LVL.ATTRIBUTE_VALUE(+);
```

```
(СОЗДАТЬ ИЛИ ЗАМЕНИТЬ ОБЗОР
V_USER_ATTR_VALUE_LEVELS
КАК ВЫБОР КОДА APP_CODE,
ИМЕНИ ATTR.ATTRIBUTE_NAME,
ЗНАЧЕНИЯ ATTR.ATTRIBUTE_VALUE,
УРОВНЯ NVL(VALUE_LEVEL,1) VALUE_LEVEL
ИЗ V_USER_ATTR_APPS ATTR,
V_ATTR_VALUE_LEVELS LVL
ГДЕ
ИМЯ АТТРИБУТА ATTR.ATTRIBUTE_NAME =
LVL.ATTRIBUTE_NAME(+) И ЗНАЧЕНИЕ АТТРИБУТА
ATTR.ATTRIBUTE_VALUE =
LVL.ATTRIBUTE_VALUE(+))
```

Обзор No. 11

V_USER_APP_CODES

Этот обзор выдает перечень всех отдельных кодов APP_CODES, присвоенных текущему пользователю. Этот обзор выполняет запрос типа генеалогического дерева, использующего оператор CONNECT BY.

Пример оператора SQL:

```
CREATE OR REPLACE VIEW V_USER_APP_CODES AS
SELECT DISTINCT ATTR.ATTRIBUTE_VALUE APP_CODE
FROM (SELECT ATTRIBUTE_NAME,
ATTRIBUTE_VALUE
FROM ATTRIBUTES
START WITH ASSIGNEE = USER
CONNECT BY ASSIGNEE = PRIOR ATTRIBUTE_VALUE
AND ATTRIBUTE_NAME='ASSIGNED_GROUP') GROUPS,
ATTRIBUTES ATTR
WHERE GROUPS.ATTRIBUTE_VALUE=ATTR.ASSIGNEE
AND ATTR.ATTRIBUTE_NAME='APP_CODE';
(СОЗДАТЬ ИЛИ ЗАМЕНИТЬ ОБЗОР V_USER_APP_CODES
КАК
ВЫБРАТЬ ОТДЕЛЬНОЕ ЗНАЧЕНИЕ
ATTR.ATTRIBUTE_VALUE APP_CODE
ИЗ (ВЫБРАТЬ ИМЯ АТТРИБУТА,
ЗНАЧЕНИЕ АТТРИБУТА
ИЗ ТАБЛИЦЫ ATTRIBUTES
НАЧАТЬ С ПОЛУЧАТЕЛЯ = ПОЛЬЗОВАТЕЛЬ
СОЕДИНИТЬ ЧЕРЕЗ ПРЕДСТАВИТЕЛЯ = ПРЕДЫДУЩЕЕ
ЗНАЧЕНИЕ
АТТРИБУТА И ATTRIBUTE_NAME = 'ASSIGNED_GROUP')
GROUPS,
ATTRIBUTES ATTR, ГДЕ ЗНАЧЕНИЕ
GROUPS.ATTRIBUTE_VALUE = ATTR.ASSIGNEE
И ИМЯ ATTR.ATTRIBUTE_NAME = 'APP_CODE')
```

Обзор No. 12 V_USER_ATTRIBUTES

Этот обзор выдает перечень всех атрибутов, присвоенных пользователю вместе со значением атрибута самого высокого уровня для соответствующих имен атрибутов. Это обзор выполняет функцию SELECT DISTINCT по самым высоким значениям уровня атрибута пользователя

V_USER_ATTR_HIGHEST_VALUES(V_USER_ATTR_VALUE_LEVELS).

Пример оператора SQL:

```
CREATE OR REPLACE VIEW V_USER_ATTRIBUTES AS
SELECT DISTINCT
APP_CODE
ATTRIBUTE_NAME,
FROM V_USER_ATTR_HIGHEST_VALUES
```

(СОЗДАТЬ ИЛИ ЗАМЕНИТЬ ОБЗОР
V_USER_ATTRIBUTES, КАК-ТО
ВЫБРАТЬ DISTINCT
КОД APP_CODE,
ИМЯ АТТРИБУТА ATTRIBUTE_NAME,
ЗНАЧЕНИЯ АТТРИБУТА ATTRIBUTE_VALUE
ИЗ ЗНАЧЕНИЙ V_USER_ATTR_HIGHEST_VALUES)

Для того, чтобы гарантировать, что данные в таблицах ATTRIBUTES и ATTRIBUTE_LEVELS поддерживаются правильно, предпочтительно использовать процедуры только в среде «Оракул» для поддержки атрибутов пользователей. Это можно сделать, ограничивая доступ к отдельным объектам, например, таблицам, процедурам и функциям в системе атрибутов пользователя. Таблицам АТТРИБУТАМ и ATTRIBUTE_LEVELS и всем обзорам должен быть придан статус «только чтение». Право выполнения процедур в среде «Оракул» и поддержки этих таблиц должен иметь только администратор, распоряжающийся атрибутами пользователя. При такой системе несанкционированные пользователи не смогут манипулировать уровнями атрибутов.

Хотя описаны и показаны на чертежах только некоторые конкретные варианты настоящего изобретения, они не должны рассматриваться, как ограничивающие объем и дух изобретения. На практике специалистами в данной области легко могут быть сделаны различные модификации, не выходящие из объема настоящего изобретения, как это определено формулой изобретения.

ФОРМУЛА ИЗОБРЕТЕНИЯ

1. В вычислительной системе, способной выполнять, по меньшей мере, одну прикладную программу и поддерживать базы данных, причем каждая прикладная программа имеет, по меньшей мере, одну функцию, способ для определения доступа пользователя, по меньшей мере, к одной указанной функции, включает следующие стадии:

присвоение, по меньшей мере, двух атрибутов указанному пользователю, причем

один из атрибутов относится к способности доступа к данным, а другой атрибут относится к способности манипулировать полученными данными;

сохранение, по меньшей мере, двух атрибутов в первой таблице в указанной базе данных;

выполнение прикладной программы в указанной вычислительной системе, обладающей, по меньшей мере, двумя функциями в указанной вычислительной системе, причем одна из этих функций относится к способности доступа к данным, а вторая из этих функций относится к способности манипулирования полученными данными;

поиск в указанной первой таблице, двух или большего количества указанных атрибутов, присвоенных пользователю, причем один из найденных атрибутов относится к способности доступа к данным, а второй атрибут относится к способности манипулирования полученными данными и

использование найденных атрибутов для осуществления доступа пользователя к данным и возможность манипулирования полученными данными в соответствии с найденными атрибутами.

2. Способ по п.1, отличающийся тем, что больше чем одна, по меньшей мере, две функции относятся к способности доступа к данным.

3. Способ по п.1, отличающийся тем, что содержит до выполнения операции следующие дополнительные стадии:

присваивание родительского значения для каждого фактического значения, по меньшей мере, для двух атрибутов, посредством чего создаются одно или несколько родительско-дочерних взаимоотношений, и

сохранение указанного одного или нескольких значений родительско-дочерних взаимоотношений во второй таблице в указанной базе данных;

а до этапа выполнения следующие дополнительные стадии:

поиск указанного одного или нескольких значений родительско-дочерних взаимоотношений в указанной второй таблице и

определение, может ли какой-либо из найденных атрибутов быть отброшен в соответствии с найденными значениями родительско-дочерних взаимоотношений.

4. В вычислительной системе, способной к выполнению, по меньшей мере, одной прикладной программы и поддержке базы данных, причем каждая прикладная программа имеет, по меньшей мере, одну функцию, способ для доступа пользователя к указанной, по меньшей мере, одной функции, включает

присваивание, по меньшей мере, двух атрибутов указанному пользователю, причем один из атрибутов относится к способности доступа к данным, а второй относится к спо-

способности манипулирования полученными данными;

сохранение, по меньшей мере, двух атрибутов в первой таблице в указанной базе данных;

выполнение прикладной программы, имеющей, по меньшей мере, две функции в указанной вычислительной системе, причем одна из этих функций относится к способности доступа к данным, а вторая из этих функций относится к способности манипулирования полученными данными;

поиск в указанной первой таблице в базе данных, по меньшей мере, двух или больше, чем двух атрибутов, присвоенных пользователю, причем один из атрибутов относится к способности доступа к данным, а второй относится к способности манипулирования полученными данными;

ввод найденных атрибутов в указанную выполняемую прикладную программу и

использование найденных атрибутов, посредством которых осуществляется доступ пользователя к указанным данным и к способности манипулирования полученными данными, определяемой в соответствии с найденными атрибутами.

5. Способ по п.4, отличающийся тем, что включает до стадии выполнения дополнительную стадию обеспечения указанному пользователю выбора одной или нескольких прикладных программ, выполняемых, по меньшей мере, в соответствии с двумя атрибутами, присвоенными указанному пользователю.

6. Способ по п.4, отличающийся тем, что больше чем одна, по меньшей мере, две функции относятся к способности доступа к данным.

7. Способ по п.4, отличающийся тем, что до этапа выполнения включает следующие дополнительные стадии:

присваивание родительского значения для каждого фактического значения, по меньшей мере, двух атрибутов и

сохранение значения родительско-дочерних взаимоотношений во второй таблице в указанной базе данных;

а до стадии его осуществления включает следующие дополнительные стадии:

поиск указанного значения родительско-дочерних взаимоотношений в указанной второй таблице и

определение, может ли какой-либо из найденных атрибутов быть отброшен в соответствии с найденными значениями родительско-дочерних взаимоотношений.

8. В вычислительной системе, способной к выполнению, по меньшей мере, одной прикладной программы и поддержке базы данных, причем каждая прикладная программа имеет, по меньшей мере, одну функцию, способ для определения доступа пользователя к указанной, по меньшей мере, одной функции содержит

присваивание, по меньшей мере, двух атрибутов группе, причем один из указанных атрибутов относится к способности доступа к данным, а второй из указанных атрибутов относится к способности манипулирования полученными данными;

присваивание указанной группы, по меньшей мере, одному пользователю,

сохранение указанной группы в таблице в указанной базе данных;

выполнение прикладной программы в указанной вычислительной системе, имеющей, по меньшей мере, две функции в указанной вычислительной системе, причем одна из этих функций относится к способности доступа к данным, а вторая из этих функций относится к способности манипулирования полученными данными;

поиск указанной группы, присвоенной указанному пользователю в указанной таблице данных, и

использование найденных атрибутов, посредством которых определяется доступ пользователя к (указанной, по меньшей мере, одной функции) указанным данным и возможность манипулирования полученными данными в указанной прикладной программе в соответствии, по меньшей мере, с (одним присвоенным указанной группе атрибутом) двумя атрибутами, найденными в указанной таблице данных.

9. Способ по п.8, в котором указанные, по меньшей мере, два атрибута, присвоенные указанной группе, определяют доступ только к указанной прикладной программе.

10. Вычислительная система, содержащая средство для выполнения прикладной программы, причем указанная прикладная программа имеет, по меньшей мере, две функции в указанной вычислительной системе, причем одна из этих функций относится к способности доступа к данным, а вторая из этих функций относится к способности манипулирования полученными данными;

средство для поддержки базы данных;

средство для присваивания, по меньшей мере, двух атрибутов пользователю, причем один из атрибутов относится к способности доступа к данным, а второй атрибут относится к способности манипулирования полученными данными;

средство для сохранения, по меньшей мере, двух указанных атрибутов в первой таблице в указанной базе данных;

средство для поиска в указанной первой таблице, по меньшей мере, двух атрибутов, присвоенных пользователю;

средство для ввода полученных атрибутов в указанную прикладную программу и

средство для использования найденных атрибутов, обеспечивающих доступ пользователя к указанным данным и способность манипу-

лирования полученными данными указанной прикладной программы, определяемые в соответствии с указанным, по меньшей мере, (одним атрибутом), двумя атрибутами, присвоенными пользователю.

11. Вычислительная система по п.10, отличающаяся тем, что больше чем одна, по меньшей мере, две функции относятся к способности доступа к данным.

12. Вычислительная система по п.10, отличающаяся тем, что дополнительно содержит средство для присваивания родительского значения для каждого фактического значения, по меньшей мере, двум атрибутам, благодаря чему создаются родительско-дочерние взаимоотношения;

средство для хранения указанного одного или нескольких значений родительско-дочерних взаимоотношений во второй таблице в указанной базе данных;

средство для поиска указанного одного или нескольких значений родительско-дочерних взаимоотношений в указанной второй таблице и

средство для определения, может ли какой-либо из найденных атрибутов быть отброшен в соответствии с указанными значениями родительско-дочерних взаимоотношений.

13. Вычислительная система, содержащая средство для выполнения прикладной программы, причем указанная прикладная программа имеет, по меньшей мере, две функции в указанной вычислительной системе, причем одна из этих функций относится к способности доступа к данным, а вторая из этих функций относится к способности манипулирования полученными данными;

средство для поддержки базы данных;

средство для присваивания, по меньшей мере, двух атрибутов пользователю, причем один из атрибутов относится к способности доступа к данным, а второй относится к способности манипулирования полученными данными;

средство для сохранения, по меньшей мере, одного указанного атрибута в первой таблице в указанной базе данных;

средство для поиска в указанной первой таблице, по меньшей мере, двух атрибутов, присвоенных пользователю;

средство для ввода полученных атрибутов в указанную прикладную программу и

средство для использования найденных атрибутов, обеспечивающих доступ пользователя к указанным данным и способность манипулирования полученными данными указанной прикладной программы, определяемые в соответствии с указанными, по меньшей мере, двумя атрибутами, присвоенными пользователю.

14. Вычислительная система по п.13, отличающаяся тем, что дополнительно содержит

средство для обеспечения указанному пользователю выбора одной или нескольких прикладных программ, чтобы работать в соответствии с указанными, по меньшей мере, двумя атрибутами, присвоенными указанному пользователю.

15. Вычислительная система по п.13, отличающаяся тем, что больше чем одна, по меньшей мере, две функции относятся к способности доступа к данным.

16. Вычислительная система по п.13, отличающаяся тем, что дополнительно содержит средство для присваивания родительского значения для каждого фактического значения, по меньшей мере, для двух атрибутов;

средство для сохранения значения родительско-дочерних взаимоотношений во второй таблице в указанной базе данных;

средство для поиска указанного значения родительско-дочерних взаимоотношений в указанной второй таблице и

средство для определения, может ли какой-либо из найденных атрибутов быть отброшен в соответствии с найденным значением родительско-дочерних взаимоотношений.

17. Вычислительная система, содержащая средство для выполнения прикладной программы, причем указанная прикладная программа имеет, по меньшей мере, две функции, причем одна из этих функций относится к способности доступа к данным, а вторая из этих функций относится к способности манипулирования полученными данными;

средство для поддержки базы данных;

средство для присваивания, по меньшей мере, двух атрибутов группе, причем один из атрибутов относится к способности доступа к данным, а второй относится к способности манипулирования полученными данными;

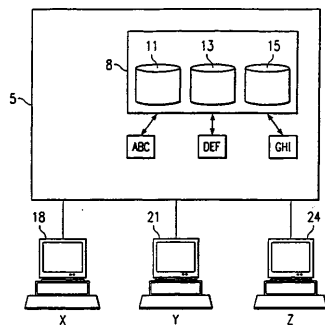
средство для присваивания указанной группы пользователю,

средство для сохранения указанной группы в таблице в указанной базе данных;

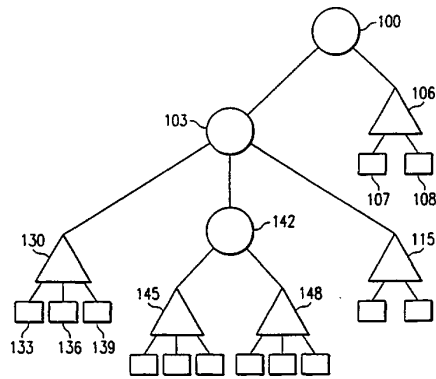
средство для поиска в указанной таблице указанной группы, присвоенной указанному пользователю, и

средство для использования найденных атрибутов, благодаря которым определяется доступ пользователя, при этом данные и способность манипулировать полученными данными определяются в соответствии с указанными, по меньшей мере, двумя присвоенными указанной группе атрибутами, найденными в указанной таблице данных.

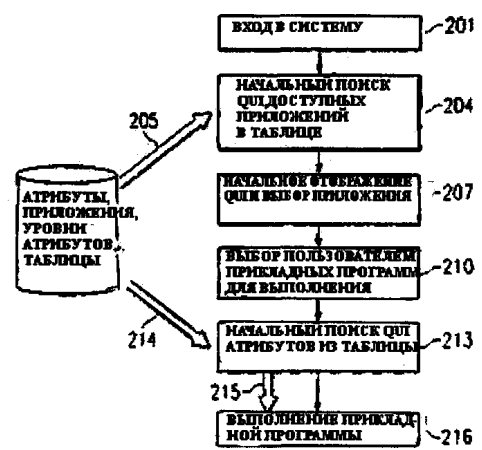
18. Вычислительная система, по п.11, отличающаяся тем, что, по меньшей мере, (один атрибут) два атрибута, присвоенные указанной группе, определяют доступ только к указанной прикладной программе.



Фиг. 1



Фиг. 2



Фиг. 3