Московский государственный университет экономики, статистики и информатики (МЭСИ)

«Базы данных»

Лекция № 3

Начальник отдела НИЧ, к.э.н., доцент Д.Г. Корнеев

2009 год

Этапы проектирования БД

- Широкое распространение реляционных СУБД и их использование в самых разнообразных приложениях показывает, что реляционная модель данных достаточна для моделирования предметных областей. Однако проектирование реляционной базы данных в терминах отношений на основе кратко рассмотренного нами механизма нормализации часто представляет собой очень сложный и неудобный для проектировщика процесс.

Этапы проектирования БД

- При этом проявляется ограниченность реляционной модели данных в следующих аспектах:

1. Модель не предоставляет достаточных средств для представления смысла данных. Семантика реальной предметной области должна независимым от модели способом представляться в голове проектировщика. В частности, это относится к упоминавшейся нами проблеме представления ограничений целостности.
2. Несмотря на то, что процесс проектирования начинается с выделения некоторых существенных для приложения объектов предметной области ("сущностей") и выявления связей между этими сущностями, реляционная модель данных не предлагает какого-либо аппарата для разделения сущностей и связей.

Этапы проектирования БД

- Потребности проектировщиков баз данных в более удобных и мощных средствах моделирования предметной области вызвали к жизни направление *семантических моделей данных.* При том, что любая развитая семантическая модель данных, как и реляционная модель, включает структурную, манипуляционную и целостную части, главным назначением семантических моделей является *обеспечение возможности выражения семантики данных.*

Этапы проектирования БД

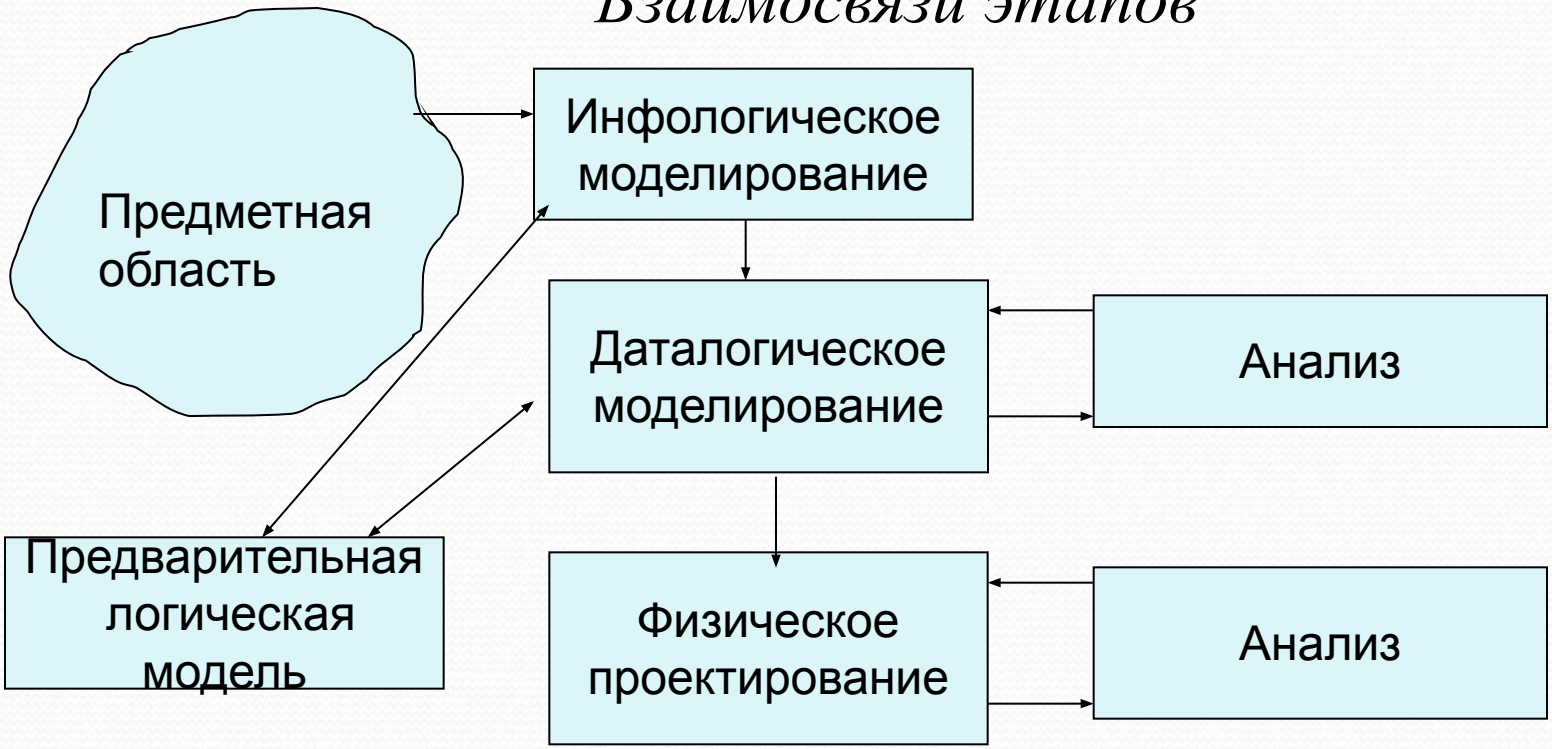
- Наиболее часто на практике ***семантическое моделирование используется на первой стадии проектирования базы данных.*** При этом в терминах семантической модели производится концептуальная схема базы данных, которая затем автоматически преобразуется к реляционной (или какой-либо другой) схеме. Этот процесс выполняется под управлением методик, в которых достаточно четко оговорены все этапы.

Этапы проектирования БД

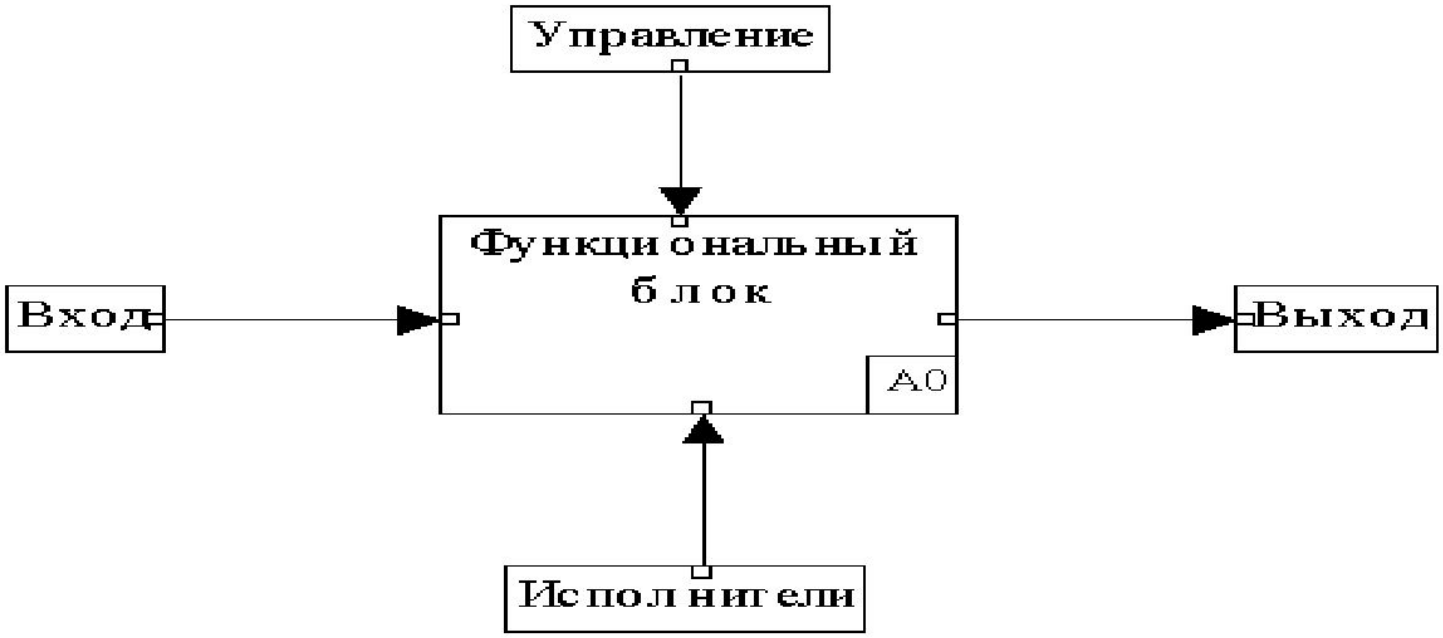
- Инфологическая модель (или семантическая или концептуальная модель) – формализованное представление предметной области (без привязки к СУБД, типам данных, пограммным средствам и т.п.);
- Даталогическая модель – привязка к конкретной СУБД (типы данных, длины, синонимы и др.); Конечная цель – описание структуры БД на языке описания данных СУБД.
- Физический уровень проектирования – проектирование физической структуры БД (выборы носителей, определение размеров физических блоков , буферизация и др.)

Этапы проектирования БД

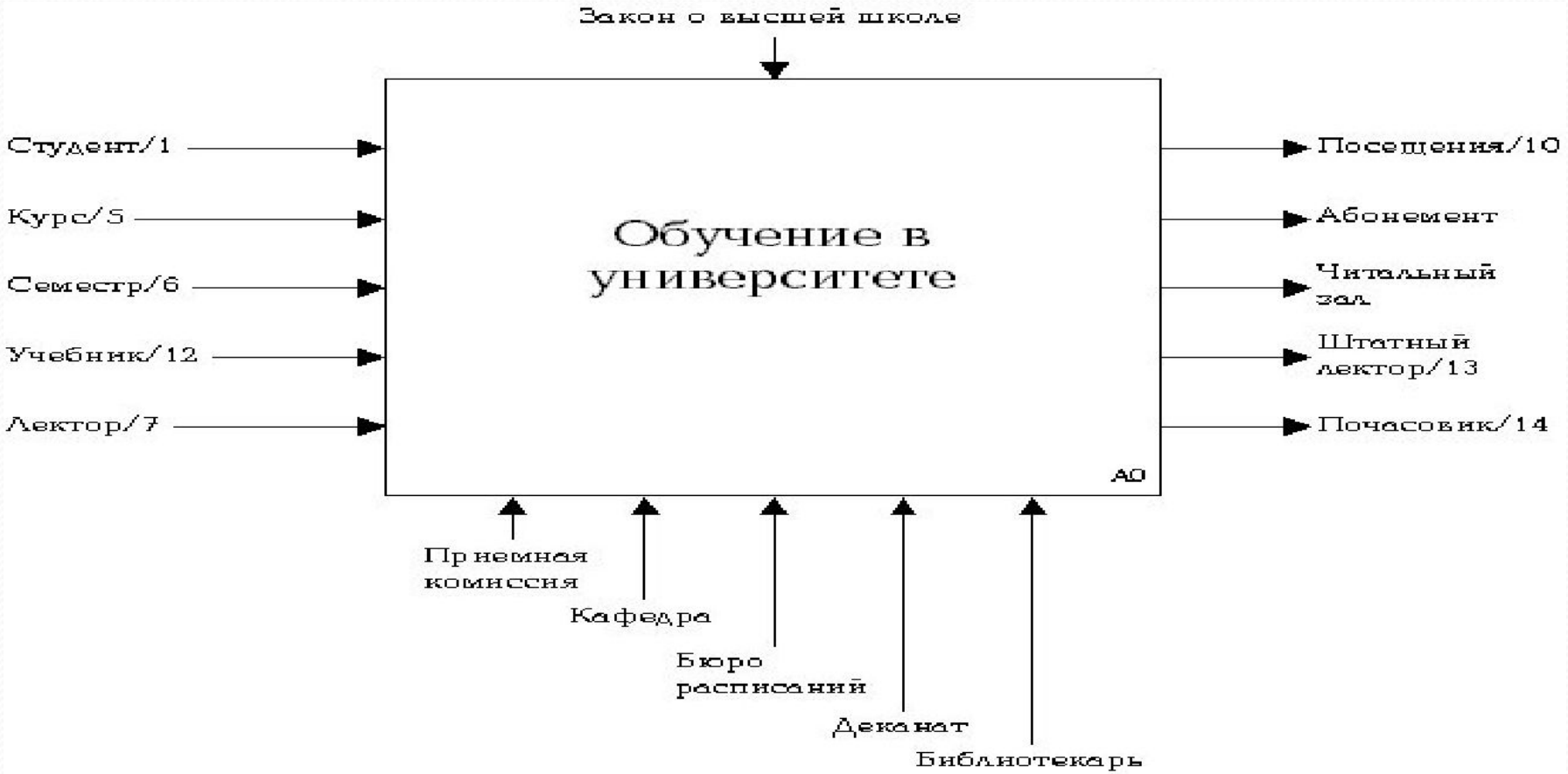
Взаимосвязи этапов



Описание предметной области IDEF0



Описание предметной области IDEF0



Цель : Составить структуру функциональных связей организации учебного процесса (1-го года обучения)

Точка зрения : общая точка зрения студента и преподавателя

Основные понятия модели

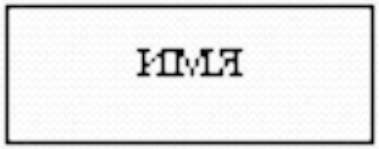
Entity-Relationship (Сущность-Связи)

- Далее кратко рассмотрим некоторые черты одной из наиболее популярных семантических моделей данных - модель "Сущность-Связи" ***(часто ее называют кратко ER-моделью).***
- На использовании разновидностей ER-модели основано большинство современных подходов к проектированию баз данных (главным образом, реляционных). Модель была предложена Ченом (Chen) в 1976 г. Моделирование предметной области базируется на использовании графических диаграмм, включающих небольшое число разнородных компонентов. В связи с наглядностью представления концептуальных схем баз данных ER-модели получили широкое распространение в системах CASE, поддерживающих автоматизированное проектирование реляционных баз данных, ***в частности DESIGN/IDEF (1x).***

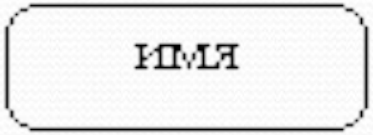
Основные понятия модели Entity-Relationship (Сущность-Связи)

- Основными понятиями ER-модели являются **сущность, связь и атрибут.**
- Сущность - это реальный или представляемый объект, информация о котором должна сохраняться и быть доступна. В диаграммах ER-модели сущность представляется в виде прямоугольника, содержащего имя сущности. ***При этом имя сущности - это имя типа, а не некоторого конкретного экземпляра этого типа.***
- Каждый экземпляр сущности должен быть отличим от любого другого экземпляра той же сущности ***(это требование в некотором роде аналогично требованию отсутствия кортежей-дубликатов в реляционных таблицах).***

Нотация DESIGN/IDEF1X



Независимая сущность



Зависимая сущность

Сущность является ***"независимой"***, если каждый экземпляр сущности может быть однозначно идентифицирован без определения его отношений с другими сущностями.

Сущность называется ***"зависимой"***, если однозначная идентификация экземпляра сущности зависит от его отношения к другой сущности.

Нотация DESIGN/IDEF1X

- Связь - это графически ***изображаемая ассоциация, устанавливаемая между двумя сущностями***. Эта ассоциация всегда является бинарной и может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь).

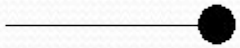
Элемент диаграммы	Обозначает
- - - - - - - - - - - -	неидентифицирующая связь
————————	идентифицирующая связь

Нотация DESIGN/IDEF1X

- Сущность обладает одним или несколькими атрибутами, которые являются либо собственными для сущности, либо наследуются через другое отношение *(от PK «родителя» передается FK в «сущность-потомок»)*.
- Атрибуты однозначно идентифицируют каждый экземпляр сущности.
- Каждый атрибут идентифицируется уникальным именем.
- Атрибуты изображаются в виде списка их имен внутри блока ассоциированной сущности, причем каждый атрибут занимают отдельную строку.
- Определяющие первичный ключ атрибуты размещаются наверху списка и отделяются от других атрибутов горизонтальной чертой.

Виды связей

Отношение дополнительно определяется с помощью указания **мощности**: какое количество экземпляров сущности-потомка может существовать для сущности-родителя.

	1, 1
	**0, M**
	**0, 1**
	**1, M**
	**точно *N* (*N*-произвольное число)**

Виды связей

- Связи "many-to-many". Иногда бывает необходимо связывать сущности таким образом, что с обоих концов связи могут присутствовать несколько экземпляров сущности (например, все члены кооператива сообща владеют имуществом кооператива). Для этого вводится разновидность связи "многие-со-многими".
- Оформляются через «развязочные таблицы», например: «преподаватель-дисциплина» (сущность из двух атрибутов: код преподавателя (FK), код дисциплины (FK)).

Виды связей

Уточняемые степени связи. Иногда бывает полезно определить возможное количество экземпляров сущности, участвующих в данной связи (например, служащему разрешается участвовать не более, чем в трех проектах одновременно). Для выражения этого семантического ограничения разрешается указывать на конце связи ее максимальную или обязательную степень.

Нормальные формы ER-схем

- Как и в реляционных схемах баз данных, в ER-схемах вводится понятие ***нормальных форм,*** причем их смысл соответствует смыслу реляционных нормальных форм.
- Заметим, что формулировки нормальных форм ER-схем делают более понятным смысл нормализации реляционных схем. Мы приведем только очень краткие и неформальные определения трех первых нормальных форм.
- ***В первой нормальной форме ER-схемы устраняются повторяющиеся атрибуты или группы атрибутов, т.е. производится выявление неявных сущностей, "замаскированных" под атрибуты.***

Нормальные формы ER-схем

- Во второй нормальной форме устраняются атрибуты, зависящие только от части уникального идентификатора. *Эти атрибуты должны определять отдельную сущность.*
- В третьей нормальной форме устраняются атрибуты, зависящие от атрибутов, не входящих в уникальный идентификатор. *Эти атрибуты являются основой отдельной сущности.*

При правильном проектировании все СУЩНОСТИ должны быть по крайней мере в третьей нормальной форме.

Получение реляционной схемы из ER-схемы

- **Шаг 1.** Каждая простая сущность превращается в таблицу. Имя сущности становится именем таблицы.
- **Шаг 2.** Каждый атрибут становится возможным столбцом с тем же именем; может выбираться более точный формат. Столбцы, соответствующие необязательным атрибутам, могут содержать неопределенные значения; столбцы, соответствующие обязательным атрибутам, - не могут.
- **Шаг 3.** Компоненты уникального идентификатора сущности превращаются в первичный ключ таблицы.

Получение реляционной схемы из ER-схемы

- **Шаг 4.** Связи многие-к-одному (и один-к-одному) становятся внешними ключами. Т.е. делается копия уникального идентификатора с конца связи "один", и соответствующие столбцы составляют внешний ключ. Необязательные связи соответствуют столбцам, допускающим неопределенные значения; обязательные связи - столбцам, не допускающим неопределенные значения.
- **Шаг 5.** Индексы создаются для первичного ключа (уникальный индекс) и внешних ключей.

Команды SQL для создания объектов

SQL включает в себя:

- ЯМД (англ. DML) - Язык манипулирования данными (SELECT, INSERT, UPDATE, DELETE);
- ЯОД (англ. DDL) - Язык описания данных – (CREATE);

Основные команды:

CREATE TABLE – создание таблиц

CREATE INDEX – создание индексов

1. DESIGN/IDEF генерирует скрипт на DDL SQL.

2. Заполнение БД – команды INSERT.

Многопользовательский режим:

CREATE TABLESPASE – создание табличного пространства;

CREATE DATABASE – создание базы данных.

Обеспечение безопасности в современных СУБД

Как только данные структурированы и сведены в базу данных, возникает проблема организации доступа к ним множества пользователей. Очевидно, что нельзя позволить всем без исключения пользователям беспрепятственный доступ ко всем элементам базы данных.

Пользователи создаются командой:

CREATE USER

Роли пользователей создаются командой:

CREATE ROLE

Обеспечение безопасности в современных СУБД

В общем виде основные требования к безопасности реляционных СУБД формулируются таким образом:

- Любые данные из СУБД должны быть доступны только после прохождения авторизации;
- Данные в любой таблице должны быть доступны не всем пользователям, а лишь некоторым из них.

Обеспечение безопасности в современных СУБД

- Некоторым пользователям должно быть разрешено обновлять данные в таблицах, в то время как для других допускается лишь выбор данных из этих же таблиц.
- Для некоторых полей вводимые данные должны быть ограничены определенным набором значений.
- Для некоторых таблиц необходимо обеспечить выборочный доступ к ее столбцам.
- Некоторым пользователям должен быть запрещен непосредственный (через запросы) доступ к таблицам, но разрешен доступ к этим же таблицам в диалоге с прикладной программой.

Идентификация пользователей и управление доступом

- Обычно в СУБД для идентификации и проверки подлинности пользователей применяются либо соответствующие механизмы операционной системы, либо SQL-оператор CONNECT. Например, в случае СУБД Oracle оператор CONNECT имеет следующий вид:
- CONNECT пользователь[/пароль] [@база_данных];
- Так или иначе, в момент начала сеанса работы с сервером баз данных, пользователь идентифицируется своим именем, а средством аутентификации служит пароль. Детали этого процесса определяются реализацией клиентской части приложения.

Пользователи СУБД

- Пользователей СУБД можно разбить на три основные категории:
- **Администратор баз данных**. Он ведает установкой, конфигурированием БД, регистрацией пользователей, групп, ролей и т.п. В обязанности администратора, кроме того, входит обеспечение пользователям доступа к необходимым им данным, а также написание (или оказание помощи в определении) необходимых пользователю внешних представлений данных. Администратор определяет правила безопасности и целостности данных.

Пользователи СУБД

- **Прикладные программисты** - отвечают за создание программ, использующих базу данных. В смысле защиты данных программист может быть как администраторы базы данных, имеющим привилегии создания объектов данных и манипулирования ими, так и пользователем, имеющим привилегии только манипулирования данными
- **Конечные пользователи базы данных** - работают с БД непосредственно через терминал или рабочую станцию. Как правило, конечные пользователи имеют строго ограниченный набор привилегий манипулирования данными. Этот набор может определяться при конфигурировании интерфейса конечного пользователя и не изменяться. Политику безопасности в данном случае определяет администратор безопасности или администратор базы данных.

Привилегии доступа

Привилегия - это право выполнять определенный тип предложений SQL. Некоторые примеры привилегий включают:

- право соединяться с базой данных (создавать сессию)
- право создавать таблицу в вашей схеме
- право выбирать строки из чьей-либо таблицы
- право выполнять чью-либо хранимую процедуру

Привилегии доступа

Привилегии в базе данных ORACLE могут быть разделены на две различные категории: системные привилегии и объектные привилегии.

- 1. Системные привилегии:

Системные привилегии позволяют пользователям выполнять конкретное действие на уровне системы, или конкретное действие над конкретным типом объектов. Таковы, например, привилегия создавать табличное пространство или привилегия удалять строки любой таблицы в базе данных. *Большинство системных привилегий доступны только администраторам и разработчикам приложений, поскольку такие привилегии весьма мощны.*

Привилегии доступа

- 2. Объектные привилегии

Объектные привилегии позволяют пользователям выполнять конкретные действия на конкретном объекте. Такова, например, привилегия удалять строки в указанной таблице. Объектные привилегии назначаются конечным пользователям, так что они могут использовать приложения базы данных для выполнения конкретных задач.

Привилегии доступа

Предложения управления привилегиями:

- Назначение привилегии:

  GRANT привилегия [ON объект] TO субъект [WITH GRANT OPTION]

- отмена привилегии:

  REVOKE привилегия [ON объект] FROM субъект

- Если субъект=пользователь, то привилегия назначается ему явно.
- Если субъект=роль, то для управления привилегиями используются соответственно:

  GRANT ROLE имя_роли [ON объект] TO субъект [WITH GRANT OPTION]

  REVOKE ROLE имя_роли [ON объект] FROM субъект

Привилегии доступа

- Назначение привилегии всем пользователям системы осуществляется следующим образом:

  GRANT привилегия [ON объект] TO PUBLIC.

  В этом случае каждый новый созданный пользователь автоматически получит такую привилегию.
- Отмена привилегии осуществляется:

  REVOKE привилегия [ON объект] FROM PUBLIC

Привилегии доступа

Привилегии выборки и модификации данных:

- SELECT - привилегия на выборку данных;
- INSERT - привилегия на добавление данных;
- DELETE - привилегия на удаление данных;
- UPDATE - привилегия на обновление данных (можно указать определенные столбцы, разрешенные для обновления).

Привилегии доступа

Привилегии изменения структуры таблиц:

- ALTER - изменение физической/логической структуры базовой таблицы (изменение размеров и числа файлов таблицы, введение дополнительного столбца и т.п.);
- INDEX - создание/удаление индексов на столбцы базовой таблицы;
- ALL - все возможные действия над таблицей.

По умолчанию пользователь не имеет никаких прав доступа к таблицам и представлениям - их необходимо передать с помощью операторов GRANT.

Привилегии доступа

- Привилегии назначаются пользователям с тем, чтобы эти пользователи могли обращаться к данным и модифицировать эти данные в базе данных. *Пользователь может получить привилегию двумя различными способами:*
- Привилегии могут назначаться пользователям явно. Например, привилегия вставлять записи в конкретную таблицу может быть явно назначена пользователю.
- Привилегии могут назначаться РОЛЯМ (именованным группам привилегий), **а затем эта роль может быть назначена одному или нескольким пользователям.**
- ***Поскольку роли облегчают и улучшают управление привилегиями, привилегии обычно назначаются ролям, а не конкретным пользователям.***

Привилегии доступа

- ORACLE предоставляет легкое и контролируемое управление привилегиями через использование ролей. Роли – это поименованные группы взаимосвязанных привилегий, которые назначаются пользователям или другим ролям. **Роли удобно использовать, когда тот или иной набор привилегий необходимо выдать (или отобрать) группе пользователей.** С одной стороны, это облегчает администратору управление привилегиями, с другой - вносит определенный порядок в случае необходимости изменить набор привилегий для группы пользователей сразу.

Аудит(инг)

- ORACLE позволяет осуществлять выборочный АУДИТ (ИНГ) (регистрируемое отслеживание) действий пользователей, чтобы помочь расследовать случаи подозрительного использования базы данных. Аудитинг может выполняться на трех различных уровнях:

- **Аудит(инг) предложений.** Отслеживание специфических предложений SQL безотносительно к конкретным объектам. Аудит(инг) предложений может быть широким, отслеживая всех пользователей в системе, или может быть сфокусирован лишь на выбранных пользователях. ***Например, можно отслеживать подключения и отключения от базы данных пользователей.***

Аудит(инг)

- **Аудит(инг) привилегий**. Отслеживание использования мощных системных привилегий безотносительно к конкретным объектам. Аудит (инг) привилегий может быть широким, отслеживая всех пользователей в системе, или может быть сфокусирован лишь на выбранных пользователях.
- **Аудит(инг) объектов**. Отслеживание обращений к конкретным объектам схем безотносительно к пользователям. Аудит(инг) объектов отслеживает предложения, позволяемые привилегиями объекта, такими как предложения SELECT или DELETE на данной таблице.

Аудит(инг)

- Для всех типов аудит(инг)а ORACLE допускает выборочное отслеживание успешных выполнений предложений, безуспешных выполнений, или и тех, и других. **Это позволяет отслеживать подозрительные предложения независимо от того, имел ли пользователь, выдавший их, соответствующие привилегии выдавать такие предложения.**
- Результаты аудит(инг)а записываются в специальную таблицу, известную как АУДИТОРСКИЙ ЖУРНАЛ (audit trail). Существуют предопределенные обзоры по этой таблице, так что записи аудиторского журнала могут быть легко извлечены.

Ограничение доступа с помощью представлений

- Представление (VIEW) – объект базы данных – виртуальное отношение, строящееся (наполняющееся данными) в момент обращения.
- В БД существует только описание (запрос, описывающий представление)

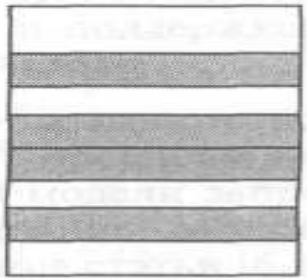
CREATE VIEW <имя представления> (столбцы)

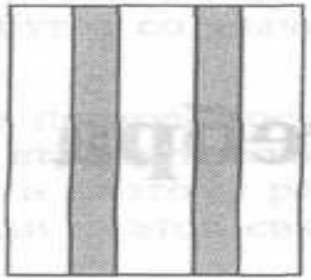
AS SELECT .... FROM .... WHERE .... ;

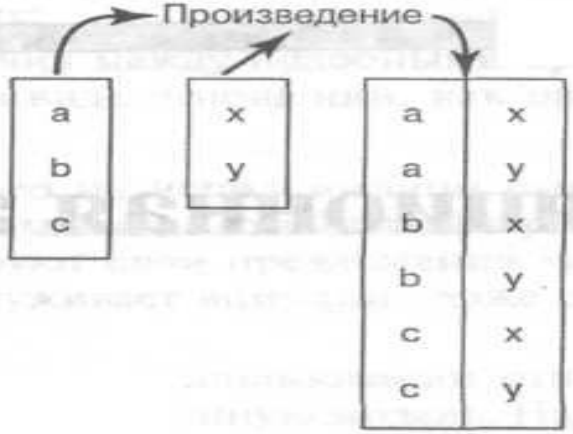
Ограничение доступа с помощью представлений

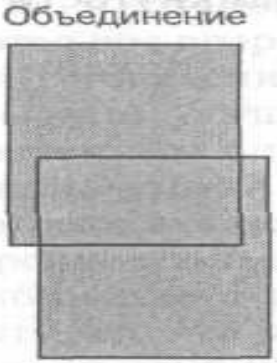
Позволяет разграничивать доступ на уровне данных (строк и столбцов определенных таблиц и других представлений). Например:

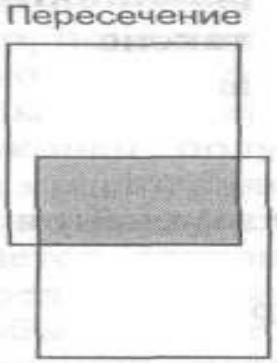
```
CREATE VIEW MOSC_STUD AS
SELECT *
FROM STUDENT
WHERE CITY='Москва';
```

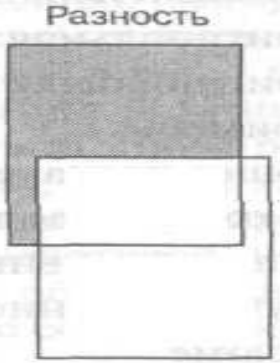

Сокращение


Проекция

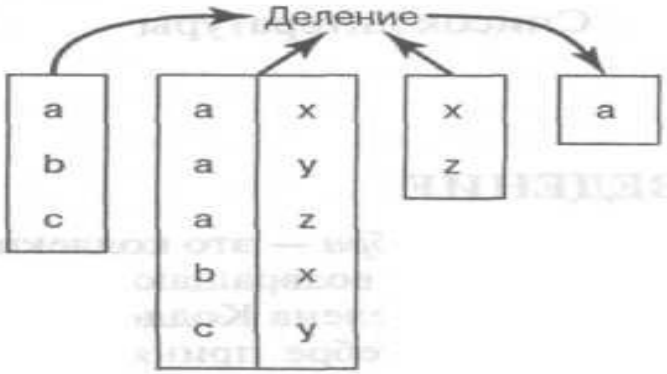

Произведение
a
b
c
x
y
a x
a y
b x
b y
c x
c y


Объединение


Пересечение


Разность


(Естественное)
соединение
a1 b1
a2 b1
a3 b2
b1 c1
b2 c2
b3 c3
a1 b1 c1
a2 b1 c1
a3 b2 c2


Деление
a
b
c
a x
a y
a z
b x
c y
x
z
a