Раздел 3. Проектирование баз данных

Тема 3.1.

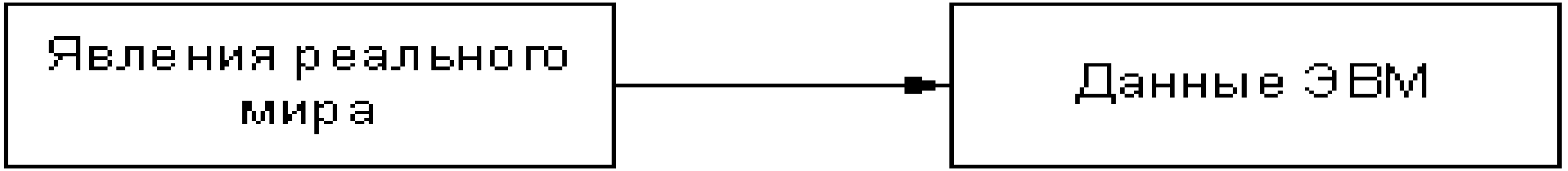
Логическое проектирование баз данных.

Вопросы лекции:

1. **Этапы проектирования БД**.
2. Создание логической модели БД.
3. CASE средства проектирования модели БД.

Этапы жизненного цикла БД

Проектирование данных (базы данных) представляет собой процесс последовательного отображения исследуемых явлений реального мира в виде данных в памяти ЭВМ.



Конкретные явления реального мира, **представляющие интерес** для проводимого исследования, будем называть **предметной областью**.

1. Предметная область

2. Информационные потребности пользователя

3. Восприятие, абстрагирование и описание предметной области

4. Изучение и описание информационных потребностей пользователя

Системный анализ – Инфологическое проектирование

5. Проектирование концептуальной инфологической модели и внешних инфологических моделей предметной области

6. Выбор СУБД

7. Проектирование концептуальной **даталогической** модели и **внешних** даталогических моделей (логическое проектирование схемы БД)

Даталогическое проектирование

8. Проектирование **внутренней** даталогической модели (физическое проектирование схемы БД)

Взаимосвязь этапов моделирования данных в процессе проектирования схемы БД

С точки зрения **инфологического** проектирования БД в рамках системного анализа, необходимо осуществить первый этап, то есть провести подробное **словесное описание** объектов предметной области и реальных связей, которые присутствуют между описываемыми объектами.

Желательно, чтобы данное описание позволяло корректно определить все **взаимосвязи** между объектами предметной области.

В общем случае существуют два подхода к выбору состава и структуры предметной области:

- Функциональный подход
- Предметный подход

Функциональный подход — он реализует принцип движения "**от задач**" и применяется тогда, когда заранее известны функции некоторой группы лиц и комплексов задач, для обслуживания информационных потребностей которых создается рассматриваемая БД. В этом случае мы можем четко выделить **минимальный необходимый** набор объектов предметной области, которые должны быть описаны.

Предметный подход — когда информационные потребности будущих пользователей БД **жестко не фиксируются**. Они могут быть многоаспектными и весьма динамичными. Мы не можем точно выделить минимальный набор объектов предметной области, которые необходимо описывать.

В описание предметной области в этом случае включаются такие объекты и взаимосвязи, которые наиболее характерны и наиболее существенны для нее.

БД, конструируемая при этом, называется предметной, то есть она может быть использована при решении множества разнообразных, заранее не определенных задач.

Конструирование предметной БД в некотором смысле кажется гораздо более заманчивым, однако трудность всеобщего охвата предметной области с невозможностью конкретизации потребностей пользователей может привести к избыточно сложной схеме БД, которая для конкретных задач будет неэффективной.

Чаще всего на практике рекомендуется использовать некоторый компромиссный вариант, который,

- ✓ с одной стороны, ориентирован на **конкретные задачи** или **функциональные** потребности пользователей, а
- ✓ с другой стороны, учитывает возможность **наращивания** новых приложений.

Системный анализ должен заканчиваться

- подробным описанием информации об объектах предметной области, которая требуется для решения конкретных задач и которая должна храниться в БД,
- формулировкой конкретных задач, которые будут решаться с использованием данной БД **с кратким описанием алгоритмов** их решения,
- описанием выходных документов, которые должны генерироваться в системе, и на их основе -
- описанием входных документов, которые служат основанием для заполнения данными БД.

Пример описания предметной области

Пусть **требуется разработать информационную систему** для автоматизации учета получения и выдачи книг в библиотеке.

Основные требования к системе и информации в ней:

1. Система должна предусматривать режимы ведения системного каталога, отражающего перечень областей знаний, по которым имеются книги в библиотеке.

2. Внутри библиотеки области знаний в систематическом каталоге могут иметь уникальный внутренний номер и полное наименование.

3. Каждая книга может содержать сведения из нескольких областей знаний.

4. Каждая книга в библиотеке может присутствовать в нескольких экземплярах.

5. Книги могут иметь одинаковые названия, но они различаются по своему уникальному шифру (ISBN).

Пример описания предметной области **для пользователя**

В библиотеке ведется картотека читателей.

На каждого читателя в картотеку заносятся следующие сведения:

- фамилия, имя, отчество;
- домашний адрес;
- телефон (будем считать, что у нас два телефона — рабочий и домашний);
- дата рождения.

Каждому читателю присваивается уникальный номер читательского билета. Каждый читатель может одновременно держать на руках не более 5 книг.

Читатель не должен одновременно держать более одного экземпляра книги одного названия.

Каждая книга в библиотеке может присутствовать в нескольких экземплярах.

Каждый экземпляр имеет следующие характеристики:

- уникальный инвентарный номер;
- шифр книги, который совпадает с уникальным шифром из описания книг;
- место размещения в библиотеке, автора, год издания, издательство и др..

Предусмотреть следующие ограничения на информацию в системе:

--------- ---------- --------- и т.д.

Раздел 3. Проектирование баз данных

Тема 3.1.

Логическое проектирование баз данных.

Вопросы лекции:

1. Этапы проектирования БД.
2. **Создание логической модели БД.**
3. CASE средства проектирования модели БД.

Итак, этап **даталогического** проектирования делится на логическое и физическое проектирование модели БД.

*Оба они **зависят от типа выбранной СУБД**.*

Задача логического проектирования БД – организация данных в форме модели данных, принятой для использования в выбранной СУБД – построение концептуальной даталогической (**логической**) модели БД.

Задача физического проектирования БД – выбор рациональной структуры хранения данных, их типов и методов доступа к ним в рамках возможностей выбранной СУБД - построение внутренней даталогической (физической) модели БД.

Логическое проектирование модели БД

Этап начинается с выбора модели данных:

- Сетевая;
- Иерархическая;
- Реляционная;
- Объектно-реляционная *(или пост-реляционная);*
- Объектная
- *Многомерная (для специальных аналитических задач)*.

Зачастую этот выбор определяется успехом (или наличием) той или иной СУБД.

На этом этапе концептуальная инфологическая модель преобразуется в концептуальную **даталогическую модель базы данных данных, поддерживаемую выбранной СУБД**.

Версия концептуальной даталогической модели, которая может быть обеспечена конкретной **СУБД**, обычно и называется логической моделью или схемой БД. *(схема БД – более информативна)*

Схема БД – совокупность отношений, которые **адекватно моделируют** абстрактные объекты предметной области и семантические связи между этими объектами.

Иногда в литературе процесс определения концептуальных инфологической и даталогической моделей называется определением структуры данных.

Итак, если на этапе даталогического проектирования БД (**создания логической модели БД**) решаются следующие вопросы:

1. представление объектов.
2. представление связей.
3. идентификация объектов и связей,

то **проект (схема) БД** – это набор взаимосвязанных отношений, в которых:

- определены все атрибуты,
- заданы первичные ключи отношений,
- заданы дополнительные свойства отношений для поддержки целостности данных в БД.

Схема БД должна быть корректной, ориентированной на выбранную модель данных и обеспечивающей минимизацию избыточности за счет исключения лишних атрибутов, дублирования и т.п.

Корректной также назовем и схему БД, в которой отсутствуют нежелательные зависимости между атрибутами отношений (нежелательные связи между таблицами) .

Как правило, в большинстве БД поддерживаются связи четырех типов:

- «один к одному»;
- «один ко многим» = «многие к одному»;
- «многие ко многим» (возможно через промежуточные объекты).

Проектирование **корректной** схемы БД (устранение нежелательных зависимостей) может быть выполнено двумя путями:

- *путем **декомпозиции** (разбиения)*, когда исходное множество отношений, входящих в схему БД заменяется другим множеством отношений, являющихся **проекциями исходных** отношений (число их при этом возрастает) – проектирование на основе процесса **последовательной нормализации схем отношений**;

- *путем **синтеза**,* то есть путем компоновки из заданных исходных элементарных зависимостей между объектами предметной области схемы БД (построение схемы БД) на основе **модели отношений «сущность-связь»**.

Процесс проектирования с использованием **декомпозиции**, представляющий собой **процесс последовательной нормализации схем отношений**, **основан на том**, что каждая последующая итерация соответствует нормальной форме более высокого уровня и обладает лучшими свойствами по сравнению с предыдущей.

Этот подход к проектированию схем БД **принято называть восходящим**, когда работа начинается с нижнего уровня – уровня определения атрибутов, которые на основе анализа группируются в отношения (объекты и связи между ними) и далее нормализуются.

Хорошо подходит для проектирования **относительно небольших** БД (до 100-200 атрибутов).

При увеличении числа атрибутов до нескольких сотен и даже тысяч **восходящий подход является весьма сложным и неудобным** для проектировщика.

Процесс проектирования с использованием **синтеза** является примером **нисходящего** **подхода** к проектированию схемы БД.

Начинается этот подход с определения нескольких высокоуровневых сущностей и связей между ними.

Затем эти объекты детализируются до необходимого уровня.

Основой этого подхода является использование семантических моделей данных типа **сущность-связь** (ER-моделей – Entity-Relationship).

Иногда применяется комбинированный (смешанный) подход.

Пример фрагмента ER-диаграммы

Однако этап концептуального логического проектирования **не заканчивается** проектированием схемы отношений.

В общем случае в результате выполнения этого этапа **должны быть получены следующие результирующие документы**:

- ✓ Описание концептуальной схемы БД в терминах выбранной СУБД.
- ✓ Описание внешних моделей в терминах выбранной СУБД.
- ✓ Описание декларативных правил поддержки целостности базы данных.
- ✓ Разработка процедур (триггеров) поддержки семантической целостности базы данных.

Построение **внутренней даталогической** (**физической**) модели БД

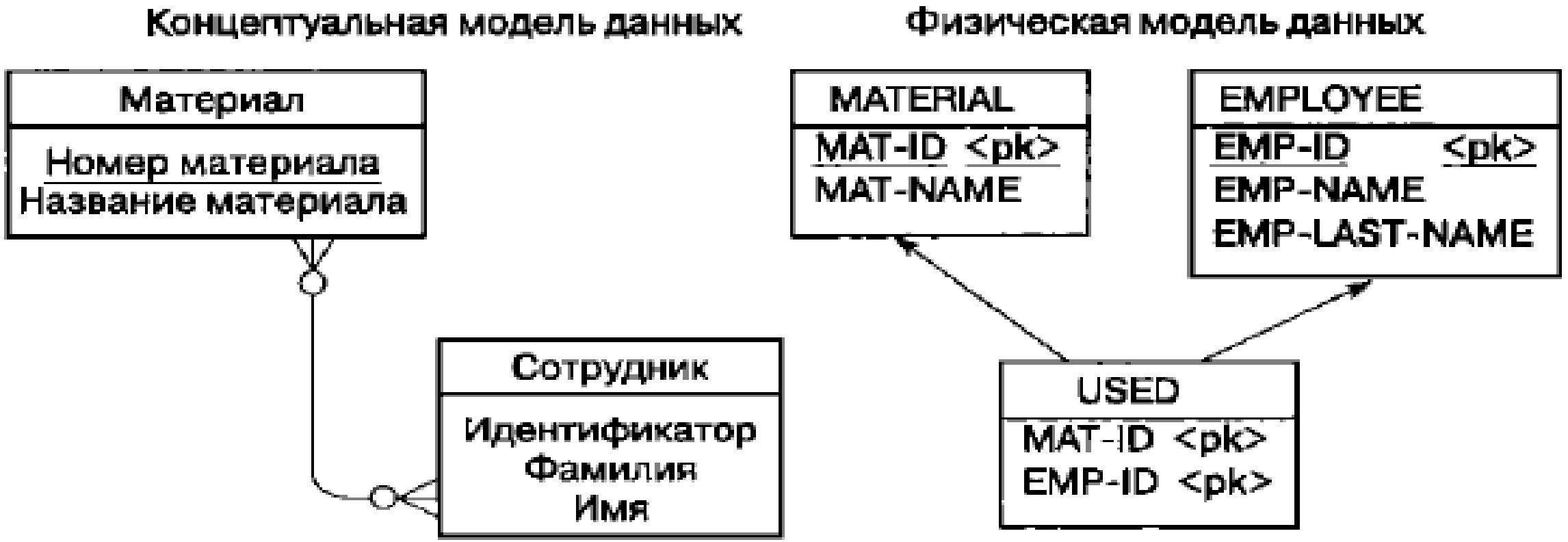
Физическая модель БД определяет способы индексирования, варианты и форматы размещения данных, методы доступа и т.д.

На этапе физического проектирования мы расписываем схему данных более детально, с **указанием типов, размеров полей и ограничений**.

Кроме разработки таблиц и индексов, на этом этапе производится также определение основных запросов и процедур обработки.

При построении физической модели **приходится решать две** взаимно противоположные по своей сути **задачи**:

- минимизация места хранения данных,
- достижение максимальной производительности, целостности и безопасности данных.



Пример перехода от ER-модели к физической модели

Т.о., при генерации физической модели БД, каждой сущности ставится в соответствие **именованная** таблица, атрибуты сущностей преобразуются в именованные колонки, а идентификаторы сущностей и связей становятся первичными и внешними **ключами**.

Раздел 3. Проектирование баз данных

Тема 3.1.

Логическое проектирование баз данных.

Вопросы лекции:

1. Этапы проектирования БД.
2. Создание логической модели БД.
3. **CASE средства проектирования модели БД.**

Проектирование БД можно проводить с помощью **автоматизированных систем разработки приложений**, так называемых CASE (Computer Aided Software Engineering) систем.

Автоматизированные системы разработки приложений представляют собой **программные средства, поддерживающие процессы создания и сопровождения информационных систем**, такие как

- анализ и формулировка требований,
- проектирование приложений,
- генерация кода,
- тестирование,
- управление конфигурацией и проектом.

Основная цель CASE-систем состоит в том, чтобы отделить процесс проектирования программного обеспечения от его кодирования и последующих этапов разработки (тестирование, документирование и т. д.), а также автоматизировать весь процесс создания программных систем.

Процесс разработки баз данных с помощью CASE-систем на этапе концептуального проектирования **обычно проводится с помощью модели «сущность — связь»**.

Результатом проектирования обычно является определённая **схема данных**.

Для автоматизации проектирования объектно-ориентированных баз данных **CASE-системы предоставляют специальный язык** Unified Modeling Language (**UML**), который можно определить как промышленный объектно-ориентированный стандарт моделирования.

Его составляющими можно назвать языки

- OMT (Object Modeling Technique) и
- OOSE (Object-Oriented Soft-ware Engineering).

Большую роль в создании этого языка сыграл консорциум OMG (Object Management Group), включающий ряд ведущих производителей программного обеспечения.

CASE-системы **различаются**:

- по ориентации,
- функциональной полноте и
- типу используемой модели
- по степени независимости от СУБД.

По ориентации можно выделить CASE-системы, предназначенные:

- только для анализа предметной области;
- для анализа и проектирования;
- для проектирования баз данных;
- для полной разработки приложений, содержащих БД.

К числу последних можно отнести **Borland Builder** (Delphi), **Jbuilder**, **MS Visual Studio** и аналогичные им продукты.

По функциональной полноте CASE-системы подразделяются:

- на системы, предназначенные для решения отдельных задач проектирования;
- на интегрированные системы, поддерживающие весь цикл разработки.

По типу используемой модели CASE-системы подразделяются на:

- Структурные (основаны на методах структурного и модульного программирования);
- объектно-ориентированные;
- комбинированные.

По степени независимости от СУБД CASE-системы бывают:

- независимые CASE-системы;
- CASE-системы, встроенные в СУБД (MS Access, mySQL).

Примерами CASE-систем могут служить:

- ERwin (Logic Works),
- BPwin (Computer Associates),
- Rational Rose (Rational Software),
- S-Designer (SPD),
- DataBase Designer,
- Developer/Designer 2000 (Oracle Corp.)
- mySQL WorkBench и т.п.

Анализ характеристик и возможностей большинства современных CASE-систем позволяет сделать следующие выводы.

1. CASE-системы позволяют ускорить и облегчить разработку, повысить качество создаваемых программ и информационных систем. Многие из CASE-систем имеют средства управления коллективной работой над проектом.

2. CASE-системы особенно полезными оказываются на начальных этапах разработки.

3. CASE-система полезна для решения задач совершенствования и переноса приложений из среды одной СУБД в другую.

4. Современные CASE-системы ориентированы на квалифицированного пользователя.

Спасибо за внимание !