Лекция 12

Общая схема процедуры нормализации

Семантическое моделирование

Общая схема процедуры нормализации

Пусть дана некоторая переменная-отношение R, представленная в 1НФ, и определен набор ФЗ, МЗЗ и ЗС.

Задача нормализации заключается в систематическом разбиении исходной переменной-отношения R на набор переменных-отношений, имеющих меньшую степень, который эквивалентен переменной-отношению R.

Схема процедуры нормализации

1. Переменную-отношение в 1НФ следует разбить на проекции, позволяющие исключить все ФЗ, не являющиеся неприводимыми.
   Результат— набор переменных-отношений в 2НФ.
2. Переменные-отношения в 2НФ следует разбить на проекции, позволяющие исключить все транзитивные ФЗ.
   Результат— набор переменных-отношений в 3НФ.
3. Переменные-отношения в 3НФ следует разбить на проекции, позволяющие исключить любые оставшиеся ФЗ, в которых детерминанты не являются ПК.
   Результат— набор переменных-отношений в НФБК.
4. Переменные-отношения в НФБК следует разбить на проекции, позволяющие исключить любые МЗЗ, которые не являются ФЗ.
   Результат— переменных-отношений в 4НФ.
5. Переменные-отношения в 4НФ следует разбить на проекции, позволяющие исключить любые ЗС, которые не определяются ПК.
   Результат— набор переменных-отношений в 5НФ.

- ✓ Переменная-отношения находится в 1НФ, если каждый её атрибут атомарен и все кортежи различны.
- ✓ Переменная-отношения находится 2НФ тогда и только тогда, когда она находится в 1НФ, и каждый неключевой атрибут неприводимо функционально зависит от ПК.
- ✓ Переменная-отношения находится в 3НФ тогда и только тогда, когда она находится во 2НФ, и каждый неключевой атрибут нетранзитивно функционально зависит от ПК.
- ✓ Переменная-отношение R находится в НФБК тогда и только тогда, когда каждая ФЗ определяется ее ПК.
- ✓ Переменная-отношение R находится в 4НФ тогда и только тогда, когда каждая МЗЗ определяется ее ПК.
- ✓ Переменная-отношение R находится в 5НФ тогда и только тогда, когда каждая ЗС определяется ее ПК.

Аномалии обновления были вызваны именно теми ФЗ, МЗЗ или ЗС, которые не определялись потенциальными ключами.

Денормализация

1. Полная нормализация означает наличие большого количества логически независимых базовых переменных-отношений.
2. Большое количество логически независимых переменных-отношений означает наличие большого количества физически отдельно хранимых файлов.
3. Большое количество физически отдельно хранимых файлов означает большое количество операций ввода-вывода.#

Нормализацией переменной-отношения ***R*** называется такая замена ***R*** множеством проекций R_1, ..., R_n для всех возможных значений ***r*** переменной- отношения ***R***, что результатом обратного соединения значений r_1, ..., r_n проекций R_1, ..., R_n обязательно будет значение ***r***.

Конечная цель нормализации : *сокращение уровня избыточности данных* за счет приведения проекций к максимально высокому уровню (к 5НФ).

Денормализацией множества переменных-отношений R_1, ..., R_n называется такая замена переменных-отношений их соединением ***R***, что для всех возможных значений r_1, ..., r_n переменных-отношений R_1, ..., R_n операция проекции значения ***r*** переменной-отношения ***R*** по атрибутам R_i обязательно снова приводит к созданию значений r_i (где i=1, ..., n).

Конечная цель денормализации: *сокращение* количества соединений, выполняемых во время работы приложений.

Пример. Денормализация переменных-отношений деталей и поставок для получения переменной-отношения PSQ.

Переменная-отношение PSQ находится в 1НФ (не в 2НФ).

PSQ						
P#	**PNAME**	**COLOR**	**WEIGHT**	**CITY**	**S#**	**QTY**
P_1	Nut	Red	12.0	London	S_1	300
P_1	Nut	Red	12.0	London	S_2	300
P_2	Bol	Green	17.0	Paris	S_2	200
. . . .						
P_6	Cog	Red	19.0	London	S_1	100

Таблица 9.1. Денормализованные данные о товарах и поставках

Проблемы денормализации

1. Проблема завершения .
2. Проблемы, связанные с избыточностью хранения данных и аномалиями обновления.
3. Проблема *извлечения* данных. Денормализация может существенно усложнить выполнение некоторых запросов.
4. Проблема эффективности для широкого спектра приложений.

Пример

Запрос: "Найти средний вес всех деталей определенного цвета".

Для нормализованной БД .

SUMMARIZE P PER P PROJECT [COLOR] ADD AVG (WEIGHT) AS AVWT

При использовании переменной-отношения PSQ

SUMMARIZE PSQ PROJECT [P#, COLOR, WEIGHT] PER PSQ PROJECT [COLOR] ADD AVG (WEIGHT) AS AVWT

Ортогональное проектирование

Принцип ортогонального проектирования (principle of orthogonal design) декларирует полную взаимную независимость базовых переменных-отношений (отсутствие перекрытия их смысловых значений).

SA

/* Поставщики в Париже */

S#	SNAME	STATUS	CITY
S2	Jones	10	Paris
S3	Blake	30	Paris

/* Поставщики не из Парижа или со статусом, равным 30 */

SB

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

избыточность

Таблица 9.2. *Структура представления данных о поставщиках*

Принцип ортогонального проектирования *(исходная версия).*

Никакие две переменные-отношения в БД не должны иметь перекрывающихся смысловых значений.

Замечания.

1. Две переменные-отношения могут иметь перекрывающееся смысловое значение только в том случае, если они имеют одинаковые типы (т.е. одинаковые заголовки).
2. Вставка кортежа рассматривается как операция вставки кортежа *в БД,* а не в какую-то конкретную переменную-отношение, поскольку существует не более одной переменной-отношения, предикату которой этот кортеж удовлетворяет.

При вставке кортежа требуется указывать имя переменной-отношения R в которую вставляется кортеж. Это не противоречит п.2., т. к. имя R является *сокращением для соответствующего предиката* (например PR).

INSERT *кортеж t, где t удовлетворяет предикату PR*.

SX		
S#	SNAME	STATUS
S_1	Smith	20
S_2	Jones	10
S_3	Blake	30
S_4	Clarck	20
S_5	Adams	30

SY		
S#	SNAME	CITY
S_1	Smith	London
S_2	Jones	Paris
S_3	Blake	Paris
S_4	Clarck	London
S_5	Adams	Athens

Таблица 9.3. Неудачный вариант представления данных о поставщиках

Две переменные-отношения не имеют перекрывающегося смыслового значения, имеют их проекции по атрибутам {S#, SNAME}.

В результате попытка вставки некоторого кортежа (например, ('S_6', 'Lopez') в представление, определенное как объединение этих двух проекций, приведет к вставке кортежа ('S_6', 'Lopez', t) в переменную-отношение SX кортежа ('S_6', 'Lopez', c) — в переменную-отношение SY (где t и c — используемые по умолчанию значения).

Для устранения подобных проблем принцип ортогонального проектирования необходимо несколько расширить.

Принцип ортогонального проектирования *(окончательная версия).*
Пусть A и B являются двумя базовыми переменными-отношениями в некоторой базе данных. Тогда для переменных-отношений A и B не должно существовать декомпозиций без потерь на такие проекции A_1, ..., A_nи B_1, ..., B_m соответственно, что некоторая проекция A_i в множестве проекций A_1, ..., A_n и некоторая проекция B_j в множестве проекций B_1, ..., B_m будут обладать перекрывающимися смысловыми значениями.

термин "декомпозиция без потерь" означает декомпозицию на множество таких проекций, которые обладают следующими свойствами:

1. исходная переменная-отношение может быть восстановлена за счет обратной операции соединения проекций;
2. ни одна из проекций не является избыточной в процессе восстановления.

Дополнительные замечания.

1. Пусть переменную-отношение разделили на несколько фрагментов. В соответствии с принципом ортогонального проектирования полученные фрагменты не должны пересекаться.
   Назовем такое разбиение *ортогональной декомпозицией.*
2. Общее назначение ортогонального проектирования заключается в сокращении избыточности и в исключении аномалий обновления.
   Ортогональное проектирование дополняет нормализацию.
   Нормализация сокращает избыточность данных внутри переменных-отношений.
   Ортогональное проектирование сокращает избыточность данных между переменными-отношениями.
3. Если A и B являются базовыми отношениями одного типа, то :

$$A \text{ INTERSECT } B = \varnothing$$

4. Принципы ортогональности часто игнорируются на практике, причем иногда их даже рекомендуется игнорировать.

   Пример структуры данных, распространенной в финансовых базах данных.

ACTIVITIES_**2003** {ENTRY#, DESCRIPTION, AMOUNT, NEW_BAL}
ACTIVITIES_**2004** {ENTRY#, DESCRIPTION, AMOUNT, NEW_BAL}
ACTIVITIES_**2005** {ENTRY#, DESCRIPTION, AMOUNT, NEW_BAL}
ACTIVITIES_**2006** {ENTRY#, DESCRIPTION, AMOUNT, NEW_BAL}
ACTIVITIES_**2007** {ENTRY#, DESCRIPTION, AMOUNT, NEW_BAL}

Другие нормальные формы

Нормальные формы, не основанные на проекционно-соединительной теории нормализации.

1. Доменно-ключевая нормальная форма (ДКНФ).

Предложена Фейджином. Не определяется в терминах ФЗ, МЗЗ или ЗС.

Переменная-отношение R находится в **ДКНФ** тогда и только тогда, когда каждое наложенное на нее ограничение является логическим следствием *ограничений доменов* и *ограничений ключей,* наложенных на данную переменную-отношение R.

- *Ограничение домена* —ограничение, предписывающее использование для определенного атрибута значений только из некоторого заданного домена.
- *Ограничение ключа* —ограничение, утверждающее, что некоторый свойство или комбинация атрибутов представляет собой ПК.

Фейджин показал, что любая переменная-отношение, находящаяся в ДКНФ, находится в 5НФ (а значит, в 4НФ и т.д.).

2. Нормальная форма типа "выборка — объединение".

Применение декомпозиции на основе операций, отличных от проекции.

«Ограничительно-объединительную» теория нормализации, независимая от проекционно-соединительной теории нормализации.

Операцией декомпозиции является *непересекающееся ограничение (выборка),* а соответствующей операцией композиции — *непересекающееся объединение.*

Переменная-отношение S с данными о поставщиках в 5НФ, не характеризуется аномалиями, не нуждается в дальнейшем разбиении на проекции.

Можно произвести декомпозицию на основе некоторого ограничения, а не на основе проекции.
Например, разместить данные о поставщиках из Лондона в одном отношении (LS), из Парижа — в другом (PS).

Семантическое моделирование

Основные этапы семантического моделирования характеризуется .

1. Выявление некоторого множества *семантических* концепций (понятий), полезных при неформальном обсуждении реального мира.

   - Мир построен из сущностей.
   - Сущности могут быть классифицированы по разным типам сущностей. Все сущности определенного типа будут обладать некоторыми общими свойствами. В терминах реляционной теории выявленная общность зафиксирована в заголовке переменной-отношения.
   - Каждая сущность обладает неким особым свойством, предназначенным для ее *идентификации.*
   - Каждая сущность может быть связана с другими сущностями посредством некоторых связей.

2. Определение набора соответствующих формальных объектов, которые могут использоваться для представления семантических концепций.
   Пример. Расширенная реляционная модель (RM/T). E-отношения ("entity-relations")— сущности, P-отношения ("property-relations") — свойства.

3. Определение набора формальных общих правил целостности (или "метаограничений"), предназначенных для работы с такими формальными объектами.

4. Определение набора формальных операторов, предназначенных для манипулирования этими формальными объектами. Например, оператор PROPERTY, который можно использовать для соединения E-отношения со всеми соответствующими ему P-отношениями.

Понятие	Неформальное определение	Примеры
СУЩНОСТЬ Entity	Некоторый отличимый объект	Поставщик, деталь, поставка Служащий, отдел, человек Произведение, концерт, оркестр, дирижер Заказ на поставку, серия заказов
СВОЙСТВО Property	Элемент информации, описывающий сущность	Номер поставщика, поставляемое количество Отдел служащего, рост фотография человека Тип концерта Дата заказа
СВЯЗЬ Relationship	Сущность, которая служит для обеспечения взаимо-действия между двумя или более другими сущностями	Поставка (поставщик — деталь) Должность (служащий — отдел) Запись (произведение — оркестр — дирижер)
ПОДТИП Subtype	Сущность типа Y является подтипом сущности типа X тогда и только тогда, когда каждый экземпляр	"Служащий" является подтипом сущности "Человек" "Концерт" является подтипом сущности "Произведение"

Таблица 9.4. Некоторые семантические концепции

Модель "сущность/связь"

(Entity-Relationship или ER.-модели). Этот подход строится на использовании модели "сущность/связь", предложенной Питером Ченом (Peter Chen) в 1976 году и с тех пор неоднократно усовершенствовавшейся Ченом и многими другими исследователями.

ER-модель широко распространена в CASE-системах.

CASE-система (Computer Aided Software Engineering) это средства автоматизированной поддержки проектирования баз данных, они обычно включают инструменты для поддержки проектирования баз данных и инструменты для разработки приложений баз данных.

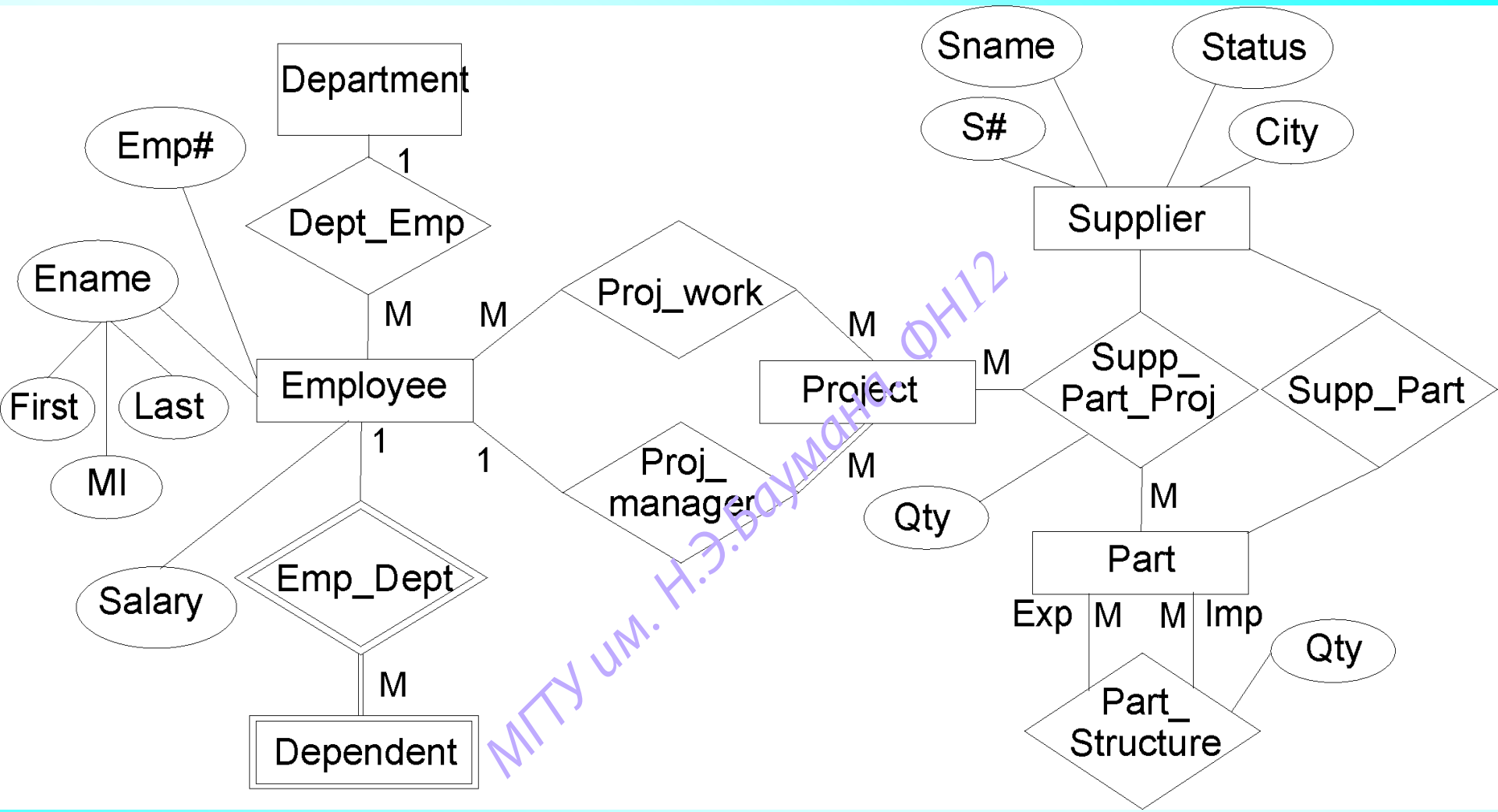

Рис. 9.4. Диаграмма модели "сущность/связь"

Сущности

Сущность—это реальный или представляемый объект, информация о котором должна сохраняться и быть доступной.

Сущность (entity)—предмет, который может быть четко идентифицирован.

Каждый тип сущности на ER-диаграмме представляется в виде прямоугольника, содержащего имя сущности.

Сущности подразделяются на сильные и слабые.

Слабой называется такая сущность, существование которой зависит от другой сущности, т.е. она не может существовать, если этой другой сущности не существует.

Прямоугольники сущностей слабых типов рисуются двойной линией.

Сильной называется сущность, которая не является слабой.
Иногда вместо термина "сильная сущность" используют термин "нормальная (regular) сущность".

Свойства

Свойством сущности является любой признак, который служит для уточнения, идентификации, классификации, числовой характеристики или выражения состояния *сущности*.

На ER-диаграмме отображаются в виде эллипсов, содержащих имена этих свойств. Эллипсы соединяются с сущностью (или связью) сплошной линией.

1. **Простое** или **составное** свойство.
2. **Ключевое** свойство (т.е. уникальное, возможно, в определенном контексте). Имена ключевых свойств подчеркиваются.
3. **Однозначное** или **многозначное** свойство (т.е. в этой модели допускаются повторяющиеся группы). Контур рисуется двойной линией.
4. **Опущенное** свойство (т.е. "неизвестное" или "непредставленное").
5. **Базовое** или **производное** свойство.

Связи

П. Чен определяет **связь** (relationship) как "ассоциацию, объединяющую несколько сущностей".

Связь – это графически изображаемая ассоциация, устанавливаемая между двумя *типами сущностей*.

Связь – это типовое понятие, все *экземпляры* своих связываемых *типов сущностей* подчиняются устанавливаемым правилам связывания.

Сущности, включенные в связь, называются ее **участниками**, а количество участников называется **степенью связи**.

Типы связи

- **“один к одному”**;
- **“один ко многим”** (**“многие к одному”**) ;
- **“многие ко многим”**.