



Bob_1

12 фев 2020 в 15:13

Связи между таблицами базы данных



9 мин



564K

SQL*, Microsoft SQL Server*

Из песочницы

1. Введение

Связи — это довольно важная тема, которую следует понимать при проектировании баз данных. По своему личному опыту скажу, что осознав связи, мне намного легче далось понимание нормализации базы данных.

1.1. Для кого эта статья?

Эта статья будет полезна тем, кто хочет разобраться со связями между таблицами базы данных. В ней я постарался рассказать на понятном языке, что это такое. Для лучшего понимания темы, я чередую теоретический материал с практическими примерами, представленными в виде диаграммы и запроса, создающего нужные нам таблицы. Я использую СУБД Microsoft SQL Server и запросы пишу на T-SQL. Написанный мною код должен работать и на других СУБД, поскольку запросы являются универсальными и не используют специфических конструкций языка T-SQL.

1.2. Как вы можете применить эти знания?

1. Процесс создания баз данных станет для вас легче и понятнее.
2. Понимание связей между таблицами поможет вам легче освоить нормализацию, что является очень важным при проектировании базы данных.
3. Разобраться с чужой базой данных будет значительно проще.
4. На собеседовании это будет очень хорошим плюсом.

2. Благодарности

Учтены были советы и критика авторов @jobgemws, @unfilled, @firnind, @Hamaruba. Спасибо!

3.1. Как организовываются связи?

Связи создаются с помощью внешних ключей (foreign key).

Внешний ключ — это атрибут или набор атрибутов, которые ссылаются на primary key или unique другой таблицы. Другими словами, это что-то вроде указателя на строку другой таблицы.

3.2. Виды связей

Связи делятся на:

1. Многие ко многим.
2. Один ко многим.
 - с обязательной связью;
 - с необязательной связью;
3. Один к одному.
 - с обязательной связью;
 - с необязательной связью;

Рассмотрим подробно каждый из них.

4. Многие ко многим

Представим, что нам нужно написать БД, которая будет хранить работников IT-компаний. При этом существует некий стандартный набор должностей. При этом:

- Работник может иметь одну и более должностей. Например, некий работник может быть и админом, и программистом.
- Должность может «владеть» одним и более работников. Например, админами является определенный набор работников. Другими словами, к админам относятся некие работники.

Работников представляет таблица «Employee» (id, имя, возраст), должности представляет таблица «Position» (id и название должности). Как видно, обе эти таблицы связаны между собой по правилу многие ко многим: каждому работнику соответствует одна и больше должностей (многие должности), каждой должности соответствует один и больше

работников (многие работники).

4.1. Как построить такие таблицы?

Мы уже имеем две таблицы, описывающие работника и профессию. Теперь нам нужно установить между ними связь многие ко многим. Для реализации такой связи нам нужен некий посредник между таблицами «Employee» и «Position». В нашем случае это будет некая таблица «EmployeesPositions» (работники и должности). Эта таблица-посредник связывает между собой работника и должность следующим образом:

EmployeeId	PositionId
1	1
1	2
2	3
3	3

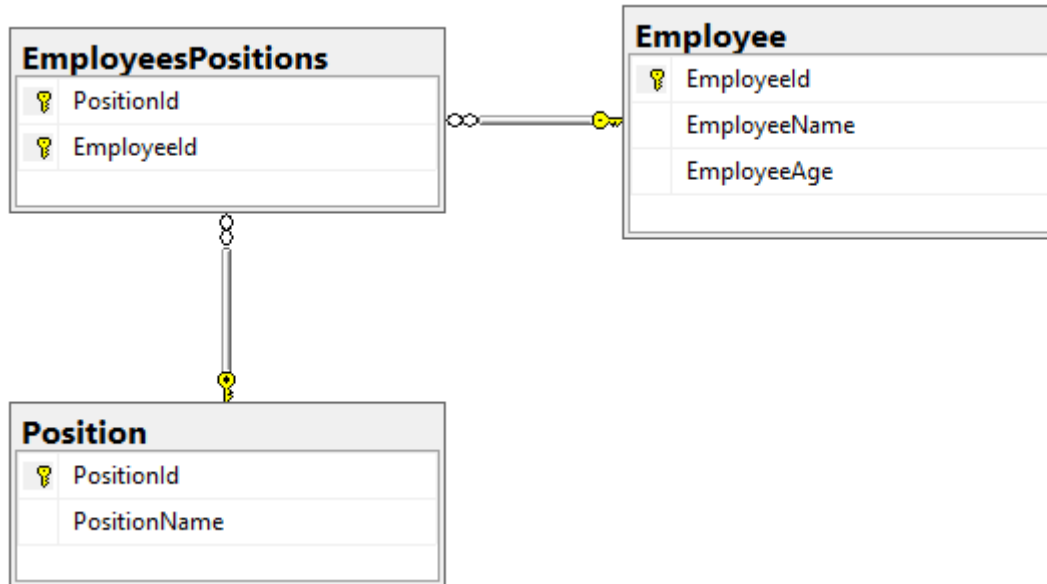
Слева указаны работники (их id), справа — должности (их id). Работники и должности на этой таблице указываются с помощью id'шников.

На эту таблицу можно посмотреть с двух сторон:

1. Таким образом, мы говорим, что работник с id 1 находится на должность с id 1. При этом обратите внимание на то, что в этой таблице работник с id 1 имеет две должности: 1 и 2. Т.е., каждому работнику слева соответствует некая должность справа.
2. Мы также можем сказать, что должности с id 3 принадлежат пользователи с id 2 и 3. Т.е., каждой роли справа принадлежит некий работник слева.

4.2. Реализация

▼ Диаграмма



▼ Код на T-SQL

```
create table dbo.Employee
(
    EmployeeId int primary key,
    EmployeeName nvarchar(128) not null,
    EmployeeAge int not null
)

-- Заполним таблицу Employee данными.
insert into dbo.Employee(EmployeeId, EmployeeName, EmployeeAge) values (1, N'
insert into dbo.Employee(EmployeeId, EmployeeName, EmployeeAge) values (2, N'
insert into dbo.Employee(EmployeeId, EmployeeName, EmployeeAge) values (3, N'

create table dbo.Position
(
    PositionId int primary key,
    PositionName nvarchar(64) not null
)

-- Заполним таблицу Position данными.
insert into dbo.Position(PositionId, PositionName) values(1, N'IT-director')
insert into dbo.Position(PositionId, PositionName) values(2, N'Programmer')
insert into dbo.Position(PositionId, PositionName) values(3, N'Engineer')

-- Заполним таблицу EmployeesPositions данными.
create table dbo.EmployeesPositions
(
    PositionId int foreign key references dbo.Position(PositionId),
```

```
EmployeeId int foreign key references dbo.Employee(EmployeeId),
primary key(PositionId, EmployeeId)
)

insert into dbo.EmployeesPositions(EmployeeId, PositionId) values (1, 1)
insert into dbo.EmployeesPositions(EmployeeId, PositionId) values (1, 2)
insert into dbo.EmployeesPositions(EmployeeId, PositionId) values (2, 3)
insert into dbo.EmployeesPositions(EmployeeId, PositionId) values (3, 3)
```

▼ Объяснения

С помощью ограничения foreign key мы можем ссылаться на primary key или unique другой таблицы. В этом примере мы

- ссылаемся атрибутом PositionId таблицы EmployeesPositions на атрибут PositionId таблицы Position;
- атрибутом EmployeeId таблицы EmployeesPositions — на атрибут EmployeeId таблицы Employee;

4.3. Вывод

Для реализации связи многие ко многим нам нужен некий посредник между двумя рассматриваемыми таблицами. Он должен хранить два внешних ключа, первый из которых ссылается на первую таблицу, а второй — на вторую.

5. Один ко многим

Эта самая распространенная связь между базами данных. Мы рассматриваем ее после связи многие ко многим для сравнения.

Предположим, нам нужно реализовать некую БД, которая ведет учет данных о пользователях. У пользователя есть: имя, фамилия, возраст, номера телефонов. При этом у каждого пользователя может быть от одного и больше номеров телефонов (многие номера телефонов).

В этом случае мы наблюдаем следующее: пользователь может иметь многие номера телефонов, но нельзя сказать, что номеру телефона принадлежит определенный пользователь.

Другими словами, телефон принадлежит только одному пользователю. А пользователю могут принадлежать 1 и более телефонов (многие).

Как мы видим, это отношение один ко многим.

5.1. Как построить такие таблицы?

Пользователей будет представлять некая таблица «Person» (id, имя, фамилия, возраст), номера телефонов будет представлять таблица «Phone». Она будет выглядеть так:

PhoneId	PersonId	PhoneNumber
1	5	11 091-10
2	5	19 124-66
3	17	21 972-02

Данная таблица представляет три номера телефона. При этом номера телефона с id 1 и 2 принадлежат пользователю с id 5. А вот номер с id 3 принадлежит пользователю с id 17.

Заметка. Если бы у таблицы «Phones» было бы больше атрибутов, то мы смело бы их добавляли в эту таблицу.

5.2. Почему мы не делаем тут таблицу-посредника?

Таблица-посредник нужна только в том случае, если мы имеем связь многие-ко-многим. По той простой причине, что мы можем рассматривать ее с двух сторон. Как, например, таблицу EmployeesPositions ранее:

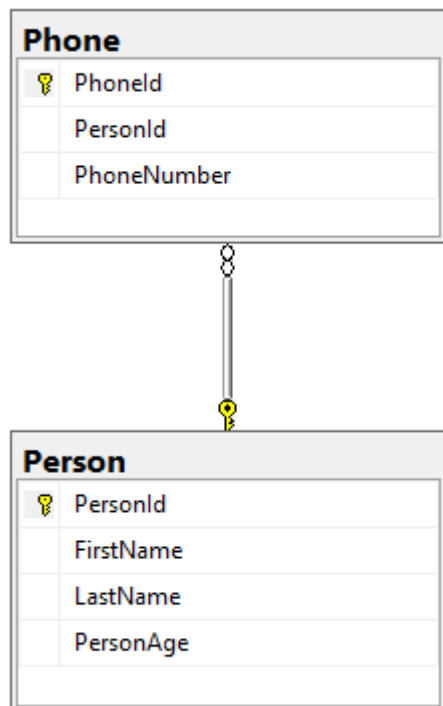
1. Каждому работнику принадлежат несколько должностей (многие).
2. Каждой должности принадлежит несколько работников (многие).

Но в нашем случае мы не можем сказать, что каждому телефону принадлежат несколько пользователей — номеру телефона может принадлежать только один пользователь. Теперь прочтите еще раз заметку в конце пункта 5.1. — она станет для вас более

ПОНЯТНОЙ.

5.3. Реализация

▼ [Диаграмма](#)



▼ [Код на T-SQL](#)

```
create table dbo.Person
(
    PersonId int primary key,
    FirstName nvarchar(64) not null,
    LastName nvarchar(64) not null,
    PersonAge int not null
)

insert into dbo.Person(PersonId, FirstName, LastName, PersonAge) values (5, N
insert into dbo.Person(PersonId, FirstName, LastName, PersonAge) values (17,

create table dbo.Phone
(
    PhoneId int primary key,
    PersonId int foreign key references dbo.Person(PersonId),
    PhoneNumber varchar(64) not null
```

)

```
insert into dbo.Phone(PHONEID, PersonId, PhoneNumber) values (1, 5, '11 091-1  
insert into dbo.Phone(PHONEID, PersonId, PhoneNumber) values (2, 5, '19 124-6  
insert into dbo.Phone(PHONEID, PersonId, PhoneNumber) values (3, 17, '21 972-
```

▼ Объяснения

Наша таблица Phone хранит всего один внешний ключ. Он ссылается на некоего пользователя (на строку из таблицы Person). Таким образом, мы как бы говорим: «этот пользователь является владельцем данного телефона». Другими словами, телефон знает id своего владельца.

6. Один к одному

Представим, что на работе вам дали задание написать БД для учета всех работников для HR. Начальник уверял, что компании нужно знать только об имени, возрасте и телефоне работника. Вы разработали такую БД и поместили в нее всю 1000 работников компании. И тут начальник говорит, что им зачем-то нужно знать о том, является ли работник инвалидом или нет. Наиболее простое, что приходит в голову — это добавить новый столбец типа bool в вашу таблицу. Но это слишком долго вписывать 1000 значений и ведь true вы будете вписывать намного реже, чем false (2% будут true, например).

Более простым решением будет создать новую таблицу, назовем ее «DisabledEmployee». Она будет выглядеть так:

DisabledPersonId	EmployeeId
1	159
2	722
3	937

Но это еще не связь один к одному. Дело в том, что в такую таблицу работник может быть вписан более одного раза, соответственно, мы получили отношение один ко многим: работник может быть несколько раз инвалидом. Нужно сделать так, чтобы работник мог

быть вписан в таблицу только один раз, соответственно, мог быть инвалидом только один раз. Для этого нам нужно указать, что столбец EmployeeId может хранить только уникальные значения. Нам нужно просто наложить на столбец EmployeeId ограничение unique. Это ограничение сообщает, что атрибут может принимать только уникальные значения.

Выполнив это мы получили связь один к одному.

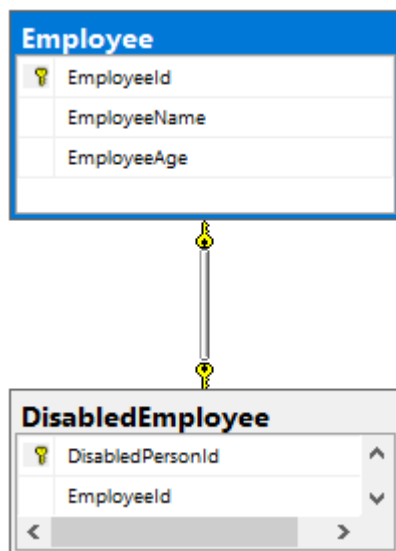
Заметка. Обратите внимание на то, что мы могли также наложить на атрибут EmployeeId ограничение primary key. Оно отличается от ограничения unique лишь тем, что не может принимать значения null.

6.1. Вывод

Можно сказать, что отношение один к одному — это разделение одной и той же таблицы на две.

6.2. Реализация

▼ Диаграмма



▼ Код на T-SQL

```
create table dbo.Employee
(
    EmployeeId int primary key,
```

```

EmployeeName nvarchar(128) not null,
EmployeeAge int not null
)

insert into dbo.Employee(EmployeeId, EmployeeName, EmployeeAge) values (159,
insert into dbo.Employee(EmployeeId, EmployeeName, EmployeeAge) values (722,
insert into dbo.Employee(EmployeeId, EmployeeName, EmployeeAge) values (937,
insert into dbo.Employee(EmployeeId, EmployeeName, EmployeeAge) values (100,
insert into dbo.Employee(EmployeeId, EmployeeName, EmployeeAge) values (99, N
insert into dbo.Employee(EmployeeId, EmployeeName, EmployeeAge) values (189,

create table dbo.DisabledEmployee
(
    DisabledPersonId int primary key,
    EmployeeId int unique foreign key references dbo.Employee(EmployeeId)
)

insert into dbo.DisabledEmployee(DisabledPersonId, EmployeeId) values (1, 159
insert into dbo.DisabledEmployee(DisabledPersonId, EmployeeId) values (2, 722
insert into dbo.DisabledEmployee(DisabledPersonId, EmployeeId) values (3, 937

```



▼ Объяснения

Таблица DisabledEmployee имеет атрибут EmployeeId, что является внешним ключом. Он ссылается на атрибут EmployeeId таблицы Employee. Кроме того, этот атрибут имеет ограничение unique, что говорит о том, что в него могут быть записаны только уникальные значения. Соответственно, работник может быть записан в эту таблицу не более одного раза.

7. Обязательные и необязательные связи

Связи можно поделить на обязательные и необязательные.

7.1. Один ко многим

1. Один ко многим с обязательной связью:

К одному полку относятся многие бойцы. Один боец относится только к одному полку.

Обратите внимание, что любой солдат обязательно принадлежит к одному полку, а полк не может существовать без солдат.

2. Один ко многим с необязательной связью:

На планете Земля живут все люди. Каждый человек живет только на Земле. При этом планета может существовать и без человечества. Соответственно, нахождение нас на Земле не является обязательным

Одну и ту же связь можно рассматривать как обязательную и как необязательную. Рассмотрим вот такой пример:

У одной биологической матери может быть много детей. У ребенка есть только одна биологическая мать.

А) У женщины необязательно есть свои дети. Соответственно, связь необязательна.

Б) У ребенка обязательно есть только одна биологическая мать – в таком случае, связь обязательна.

7.2. Один к одному

1. Один к одному с обязательной связью:

У одного гражданина определенной страны обязательно есть только один паспорт этой страны. У одного паспорта есть только один владелец.

2. Один к одному с необязательной связью:

У одной страны может быть только одна конституция. Одна конституция принадлежит только одной стране. Но конституция не является обязательной. У страны она может быть, а может и не быть, как, например, у Израиля и Великобритании.

Одну и ту же связь можно рассматривать как обязательную и как необязательную:

У одного человека может быть только один загранпаспорт. У одного загранпаспорта есть только один владелец.

А) Наличие загранпаспорта необязательно – его может и не быть у гражданина. Это необязательная связь.

Б) У загранпаспорта обязательно есть только один владелец. В этом случае, это уже обязательная связь.

7.3. Многие ко многим

Любая связь многие ко многим является необязательной. Например:

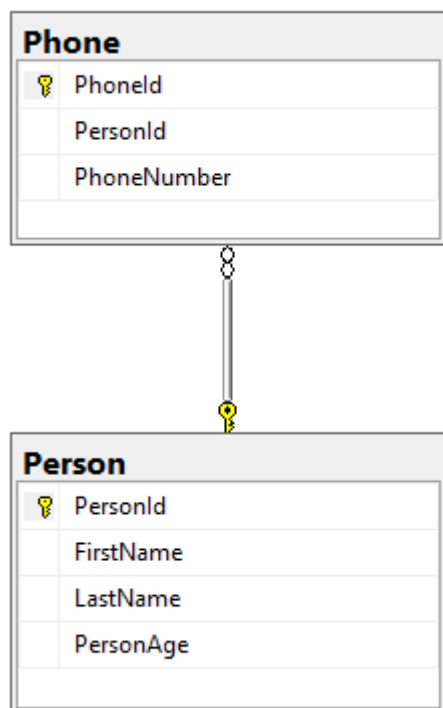
Человек может инвестировать в акции разных компаний (многих). Инвесторами какой-то компании являются определенные люди (многие).

А) Человек может вообще не инвестировать свои деньги в акции.

Б) Акции компании мог никто не купить.

8. Как читать диаграммы?

Выше я приводил диаграммы созданных нами таблиц. Но для того, чтобы их понимать, нужно знать, как их «читать». Разберемся в этом на примере диаграммы из пункта 5.3.



Мы видим отношение один ко многим. Одной персоне принадлежит много телефонов.

1. Возле таблицы **Person** находится золотой ключик. Он обозначает слово «один».
2. Возле таблицы **Phone** находится знак бесконечности. Он обозначает слово «многие».

9. Итоги

1. Связи бывают:
 - Многие ко многим.
 - Один ко многим.
 - 1) с обязательной связью;
 - 2) с необязательной связью.

- Один к одному.
 - 1) с обязательной связью;
 - 2) с необязательной связью.

2. Связи организовываются с помощью внешних ключей.

3. Foreign key (внешний ключ) — это атрибут или набор атрибутов, которые ссылаются на primary key или unique другой таблицы. Другими словами, это что-то вроде указателя на строку другой таблицы.

10. Задачи

Для лучшего усвоения материала предлагаю вам решить следующие задачи:

1. Описать таблицу фильм: id, название, длительность, режиссер, жанр фильма.
Обратите внимание на то, что у фильма может быть более одного жанра, а к одному жанру может относиться более, чем один фильм.
2. Описать таблицу песня: id, название, длительность, певец. При этом у песни может быть более одного певца, а певец мог записать более одной песни.



332



26

- Описать отдельную таблицу производитель: id, название, рейтинг.
- Описать отдельную таблицу цвета: id, название.

У одной машины может быть только один производитель, а у производителя — много машин. У одной машины может быть много цветов, а у одного цвета может быть много машин.

4. Добавить в БД из пункта 6.2. таблицу военно-обязанных по типу того, как мы описали отдельную таблицу DisabledEmployee.

Теги: связи таблиц, sql, базы данных

Хабы: SQL, Microsoft SQL Server

Редакторский дайджест

Присылаем лучшие статьи раз в месяц

Электронная почта



**11****0**

Карма Рейтинг

Борис Махлин @Bob_1

Пользователь

Комментарии 26**playerro**

12 фев 2020 в 16:56

Мне кажется, или это статья о том, что преподается на 1 курсе любого вуза, есть в 1 главе любой книги по бд и тонне статей даже для самых маленьких?



Ответить

**Bob_1**

12 фев 2020 в 17:47

Здравствуйте! Статья оригинальная и не была взята из какой-то методички или книжки. В ней я хотел рассказать свое виденье темы и максимально просто ее объяснить, без всяких заумных терминов. На Хабре я не нашел похожих статей, потому решил написать свою, поскольку я считаю, что понимание связей между таблицами очень важно при проектировании бд.



Ответить

**vdem**

12 фев 2020 в 18:54

На Хабре я не нашел похожих статей

Азбуку здесь тоже не найдешь.



Ответить

**milomory**

13 мар 2020 в 14:50

А может стоило бы



Ответить

**CherryPah**

13 фев 2020 в 17:16

Действительно. Ведь куча статей вида «я распаковываю новую %железконейм%» или ставшее мемом «как я ремонтирую балкон» куда больше относятся к IT-тематике Хабра.



Ответить





Kirill192

28 апр 2021 в 20:26

Прекрасная статья, лишние усложнения для понимания — я считаю понты. Еще раз. я считаю. Человек сразу обозначил для кого статья, и может я тупой, но именно это я и искал. Спасибо автору!



Ответить



osipov_dv

12 фев 2020 в 19:23

Foreign key (вторичный ключ) — really?



Ответить



IonianWind

12 фев 2020 в 22:38

Тут бы хоть про нормальные формы написать, чтобы помяснее было, но нет...



Ответить



Bob_1

12 фев 2020 в 22:44

Планирую написать отдельную статью по ним. Мне кажется, что впихнуть их сюда это будет лишним, потому что статья именно про связи.



Ответить



unfilled

13 фев 2020 в 07:57

Спасибо за работу, возможно, кому-то пост поможет.

Написанный мною код должен работать и на других СУБД, поскольку запросы являются универсальными и не используют специфических конструкций языка T-SQL.

На самом деле это не так — как минимум `nvarchar` и `identity` будут не во всех СУБД. Выбран очень странный стиль написания имён объектов — где-то используется схема, где-то не используется. ИМХО, в «справочном» посте стиль должен быть унифицирован. Плюс, вы много где объявляете поле как `nvarchar`, а вставляете туда `varchar` — не надо так):

```
create table dbo.Person
(
    ...
    FirstName nvarchar(64) not null,
    LastName nvarchar(64) not null,
    ...)
```

```
insert into dbo.Person(PersonId, FirstName, LastName, PersonAge) values (5, 'John', 'D
```

Ещё хотелось бы добавить, что у явно объявленных связей есть определённые сайд-эффекты, о которых тоже хотелось бы получить информацию в таком «справочном» посте. Связи требуют определённой стратегии индексирования, которая зависит от предполагаемого использования данных.

Просто, в качестве примера, рассмотрим вот эту таблицу:

```
create table dbo.Phone
(
    PhoneId int identity primary key,
    PersonId int foreign key references Person(PersonId),
    PhoneNumber varchar(64) not null
)
```

При удалении записи из таблицы Person будет происходить полное сканирование таблицы Phone — а это прямой путь к избыточным блокировкам и/или дедлокам.

↑  ↓ Ответить  

 **Bob_1**
13 фев 2020 в 21:23 

Большое спасибо за критику!

На самом деле это не так — как минимум nvarchar и identity будут не во всех СУБД.

Убрал identity с глаз долой :)

nvarchar решил оставить, на сколько я знаю, в больших СУБД он есть. Даже если это не так, то думаю, что заменить тип не составит труда.

Выбран очень странный стиль написания имён объектов — где-то используется схема, где-то не используется.

Исправил — обращаюсь ко всем объектам через схему.

Плюс, вы много где объявляете поле как nvarchar, а вставляете туда varchar — не надо так)

Тоже исправил.

Ещё хотелось бы добавить, что у явно объявленных связей есть определённые сайд-эффекты, о которых тоже хотелось бы получить информацию в таком «справочном» посте. Связи требуют определённой стратегии индексирования, которая зависит от предполагаемого использования данных.

В ближайшее время немного подробнее опишу foreign key.



Ответить



firnind

13 фев 2020 в 14:01

“Invalide” — не очень хорошее название для таблицы. Лучше “disability” или “employee_disability”. Звучит несколько более этично.



Ответить



Bob_1

13 фев 2020 в 20:48

Согласен.

Спасибо, внес правки в статью.



Ответить



vassabi

24 окт 2020 в 12:19

там не только «звучит», там коннотация достаточно негативная.

Это для славян легко написать invalid\invalide для людей (или как в одной истории про дизайнера — он сокращал button до butt, и в проекте у него были цвета с размерами, так что даже дошло до black_big_butt), но для англоязычных это слова и у них есть свой смысл.

Интересно, что бы сказали, если бы в коде на 1C (кстати, а где еще в продакшене пишут по русски?) встретили «ВЫБЕРИТЕ ИМЯ ИЗ НЕВЕРНЫХ ГДЕ ВОЗРАСТ > 30» и т.д. (или если про дизайнера — увидев переменную «большая_черная_жопа» %))?

PS: это я пытаюсь усилить вашу позицию, что оно не просто «звучит неэтично», на него все хорошо-англоговорящие сразу делают внутренний «эээээ, вы точно это имели в виду?» (так же как и хорошо-русскоговорящие на фразу иностранца «этот чурка»).



Ответить



Hamaruba

13 фев 2020 в 15:53

Заявка на создание своего введения в реляционные БД для новичков имеет право на жизнь. Как говорится пока кому-нибудь не объяснишь, сам не поймешь. Написано конечно топорно, но не у каждого найдется сил и на это. Дальше только полировать.

Связи один-к-одному не такие уж и редкие. На больших объемах данных возникает необходимость в вертикальном секционировании для ускорения работы запросов.

Если некий объект имеет множество атрибутов, часть из которых используется очень часто, а часть очень редко, целесообразно хранить их отдельно друг от друга, чтобы уменьшить расходы на чтение диска.

«вторичный ключ» переключается с «альтернативный ключ» — по этому это название плохо применимо для «foreign key» = «внешний ключ».



Ответить



Bob_1

13 фев 2020 в 20:53



Связи один-к-одному не такие уж и редкие. На больших объемах данных возникает необходимость в вертикальном секционировании для ускорения работы запросов. Если некий объект имеет множество атрибутов, часть из которых используется очень часто, а часть очень редко, целесообразно хранить их отдельно друг от друга, чтобы уменьшить расходы на чтение диска.

Довольно-таки логично, спасибо за дополнение!

«вторичный ключ» переключается с «альтернативный ключ» — по этому это название плохо применимо для «foreign key» = «внешний ключ».

Благодарю, заменил на «внешний ключ».



Ответить



jobgemws

14 фев 2020 в 15:19

Статья явно не будет лишним для тех, кто только начинает проектировать схему БД, а также чтобы вспомнить некоторые моменты.

Однако, мне в ВУЗе долго не доходило какую связь использовать между сущностями (3 курс). В итоге, пришло озарение, что все 7 формальных правил отношений между сущностями есть в жизни между людьми, предметами и т д и т п.

Т е по сути эти отношения были просто формализованы, а не придуманы человеком.

Ну в принципе как и многое мы берем из окружающего мира уже готовое и адаптируем под наши нужды. Вон тот же самолет похож на птицу. Ну грубо, но думаю идея понятна.

Так что если не удастся понять технические детали-лучше посмотреть именно суть-что мы хотим сделать, а затем оглянуться вокруг и найти решение уже в этом реальном мире. После чего формализовать этот существующий процесс и адаптировать для достижения нужной цели. На сколько хорошо были проведены формализация и адаптация покажет время, а также необходимость как можно дольше кардинально не изменять созданное решение.



Ответить



Bob_1

14 фев 2020 в 17:38



В итоге, пришло озарение, что все 7 формальных правил отношений между сущностями есть в жизни между людьми, предметами и т д и т п.

Т е по сути эти отношения были просто формализованы, а не придуманы человеком.

Ну в принципе как и многое мы берем из окружающего мира уже готовое и адаптируем под наши нужды. Вон тот же самолет похож на птицу. Ну грубо, но думаю идея понятна.

Довольно интересный подход!

Чем-то напоминает способ запоминания, когда изученный материал рассказываешь так, чтобы понял 8-летний ребенок.



Ответить



jobgemws

14 фев 2020 в 18:01



На самом деле можно изучить что-то на следующих уровнях:

- 1) чтобы понять, но не смочь сделать
- 2) чтобы понять и сделать (обычно здесь останавливаются, т к дальше нет необходимости в работе)
- 3) чтобы обучить и передать знания и хорошо сделать
- 4) чтобы быстро и легко обучить и также передать знания, а также быстро и долговечно сделать



Ответить



jobgemws

14 фев 2020 в 16:04



В статье не хватает акцента с необязательной и обязательной связью.

Всего в жизни 7 формальных правил, а именно:

- 1) один к одному с обязательной связью (у каждого гражданина РФ есть паспорт гражданина РФ и он только один)
- 2) один к одному с необязательной связью (у каждого человека может быть заграничный паспорт и если да, то только один)
- 3) один ко многим с обязательной связью (у каждого гражданина есть несколько документов, подтверждающих его личность)
- 4) один ко многим с необязательной связью (у каждого человека могут быть дети)
- 5) многие к одному с обязательной связью (у детей есть биологический отец/мать)
- 6) многие к одному с необязательной связью (у детей может быть отец/мать и если да, то только один/одна или сирота)
- 7) многие ко многим (женщины и мужчины-могут быть любовницами и любовниками причем не только относительно одного человека, а могут и не быть)

Причем отношения могут меняться-смотря как формализовать и для чего.

В этом весь фокус.

Т е можно взять детей и родитель (мама/папа)-если рассматривать без учета сирот, то связь будет обязательна, а если с сиротами-то не обязательна (многие к одному)

А вот если перевернуть: родитель (мама/папа) и дети, то необязательна-если допустить, что детей может не быть и обязательна, если дети точно есть (хотя бы один) (один ко многим).

Если же допустить в предыдущих вариантах, что ребенок может быть только один, то отношение станет 1:1 (с обязательной или необязательной связью соответственно).


А если рассмотреть вообще попечителей и детей, то получаем отношение многие ко многим. Т е в принципе все 7 формальных правил действуют по отношению между одними и теми же сущностями. Важно формализовать ту, которая нужна для реализации (для достижения конкретной цели в предметной области).

И т д и т п

↑  ↓ Ответить  

◉  **Bob_1**
14 фев 2020 в 17:43 

Логично. На выходных допишу это в статью.
Большое спасибо!

↑  ↓ Ответить  

◉  **Bob_1**
16 фев 2020 в 20:37

Добавил отдельный пункт по обязательным и необязательным связям, как и обещал. Еще раз спасибо.

↑  ↓ Ответить  

◉  **jobgemws**
20 июн 2023 в 13:24

И Вам)

↑  ↓ Ответить  

◉  **SaturnUpiter**
15 июн 2023 в 14:41 


jobgemws Извините, а разве пункты 3 и 5, 4 и 6 это не одно и тоже?

↑  ↓ Ответить  

◉  **jobgemws**
20 июн 2023 в 13:23 

Нет, это не одно и тоже. Постарался привести примеры выше, хотя соглашусь, что не все примеры прям идеальны.

↑  ↓ Ответить  

◉  **jobgemws**
16 авг 2020 в 11:35

Уважаемый @Bob_1, по проектированию БД опубликовал 7 формальных правил:
Основы правил проектирования базы данных

Только полноправные пользователи могут оставлять комментарии. [Войдите](#), пожалуйста.

Публикации

ЛУЧШИЕ ЗА СУТКИ

ПОХОЖИЕ



Holmogorov

22 часа назад

Советский фрикинг. Были ли телефонные фриеры в СССР?



Простой



9 мин



9.9K

Ретроспектива



+114



37



80



MadRinger_42

21 час назад

«Поздравляем с терабитом». Та самая статья про DDoS-2023 — без цензуры



Средний



9 мин



17K

Кейс



+72



60



28



EI_Gato_Grande

22 часа назад

В поисках самой мощной видеокарты! Тестируем A100 и A6000 Ada на большой языковой модели



11 мин



2.8K



+26



18



8



IgnatChuker

21 час назад

Обзор часов HUAWEI WATCH GT 4

 Простой  6 мин  7.2K

Обзор

 +24

 12

 34



skovalev

1 час назад

От идеи до сервера на Mac mini M2 Pro: как мы запускаем новые продукты

 5 мин  1.2K

Кейс

 +21

 2

 1



NSlyadneva

22 часа назад

Фронтенд-апгрейд для Jira. Как и зачем мы модернизировали сервисный портал КРОК

 9 мин  1.1K

Кейс

 +18

 14

 1



vkugay

21 час назад

Как в Node.js контролировать потребление памяти при обработке сетевых запросов

 Средний  10 мин  1.9K

Тutorial

Recovery Mode

 +17

 51

 0



divolko3

18 часов назад

Как распространялся open-source-софт в 1992 году: Walnut Creek Software

 5 мин  1.2K

+15

9

1



lis355

14 часов назад

В помощь музыканту: меняем тональность (и не только) звука с компьютера в Windows



Простой



4 мин



1.3K

Тutorial

+14

20

1



GKM

19 часов назад

Радикальная асинхронщина



Средний



11 мин



5.1K

Мнение

+14

49

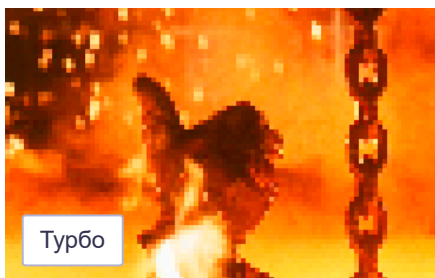
33

hh.ru проверил, к кому российские айтишники пойдут работать и почему

Турбо

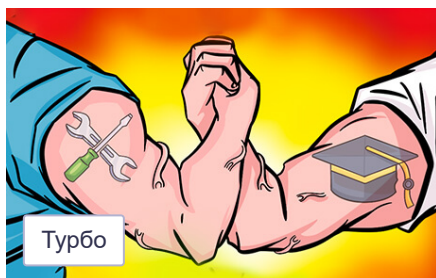
Показать еще

МИНУТОЧКУ ВНИМАНИЯ



Турбо

Из горнила конкуренции:
лучшие технобренды России



Турбо

Когда сотрудник — барьер для
киберпреступника



Интересно

Глупым вопросам и ошибкам —
быть! IT-менторство на ХК

ВОПРОСЫ И ОТВЕТЫ

Где у меня ошибка в sqlite3?

Python · Простой · 3 ответа

Как правильно решить задачу?

SQL · Простой · 2 ответа

Как правильно задать запрос UPDATE где название столбца переменная?

PHP · Простой · 2 ответа

Как составить запрос на экспорт таблицы БД?

SQL · Простой · 1 ответ

Как исправить "Input string was not in a correct format."?

SQL · Средний · 3 ответа

Больше вопросов на Хабр Q&A

ЧИТАЮТ СЕЙЧАС

Как я склеил картон и продал на маркетплейсах на 50 млн в год

 176K  188

Как я подарил своему старому ноутбуку вторую жизнь

 18K  56

Американец построил огромного боевого робота в лесу

 65K  80

Из-за ошибки «Локатора» iPhone полиция пришла с обыском к пожилой женщине и теперь должна выплатить \$3,76 млн

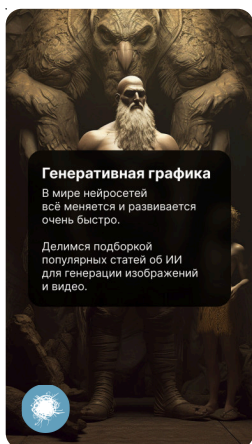
 3K  12

От идеи до сервера на Mac mini M2 Pro: как мы запускаем новые продукты

 1.2K  1

hh.ru проверил, к кому российские айтишники пойдут работать и почему

ИСТОРИИ



Нейросети:
интересное



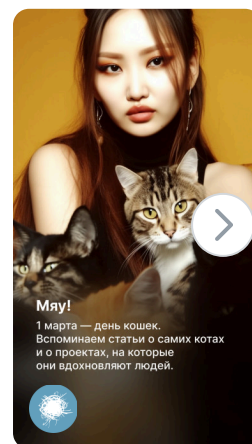
Что умеет
калькулятор зарплат
в IT



Яндекс 360
призывает героев
бэкэнда



Полезные книги для
библиотеки
айтишника



Букет котов

БЛИЖАЙШИЕ СОБЫТИЯ



Серия занятий
«Тренировки по
алгоритмам 5.0» от
Яндекса

1 марта – 19 апреля

19:00

Онлайн

[Подробнее в календаре](#)

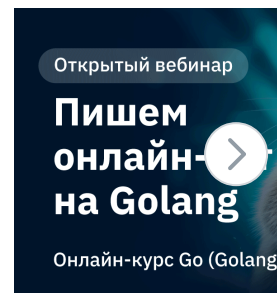


Вебинар «Взгляд на
FinOps со стороны
провайдера,
разработчика и
пользователя»

12 марта 11:00

Онлайн

[Подробнее в календаре](#)



Открытый уро
онлайн-чат на

12 марта

Онлайн

[Подробнее в календаре](#)

Ваш аккаунт

Войти

Регистрация

Разделы

Статьи

Новости

Хабы

Компании

Авторы

Песочница

Информация

Устройство сайта

Для авторов

Для компаний

Документы

Соглашение

Конфиденциальность

Услуги

Корпоративный блог

Медийная реклама

Нативные проекты

Образовательные

программы

Стартапам



Настройка языка

Техническая поддержка