Are foreign keys really necessary in a database design?

Asked 16 years, 4 months ago Modified 2 years, 8 months ago

Viewed 51k times

139

As far as I know, foreign keys (FK) are used to aid the programmer to manipulate data in the correct way. Suppose a programmer is actually doing this in the right manner already, then do we really need the concept of foreign keys?

Are there any other uses for foreign keys? Am I missing something here?

database oracle-database foreign-keys

Share

Improve this question

Follow

edited Sep 30, 2016 at 4:34

Mark Harrison

304k ● 131 ● 350 ● 489

asked Aug 20, 2008 at 20:18

Niyaz

54.8k ● 56 ● 152 ● 183

3 This comes up often around here. I blame Joel Spolsky:-). There are many good answers here; rather than retype mine, I'll just give you a link:

stackoverflow.com/questions/83147/whats-wrong-with-foreign-keys – SquareCog Apr 19, 2009 at 15:02

85 "Suppose a programmer is actually doing this in the right manner already" - I can't even imagine such a scenario. – recursive Apr 19, 2009 at 15:05

12 "Foreign Key" is an idea, not a technology. It's a relational rule. Your question is really about whether you should attempt to enforce the rule in your code or let the database help you. When concurrency is involved, it's best to let the database engine enforce the rule, since it's aware of EVERYTHING that happens in the database, while your code can not possibly be aware. – Triynko Aug 25, 2009 at 18:20

5 @lubos & cdeszaq. Actually, it IS a relational rule... it's a subset of rule 10 of Codd's "Twleve Commandments"... "Integrity Independence", which basically says that the RDBMS's relational integrity must be maintained independently of any application that accesses it, which is exactly what I was explaining in an easy-to-understand way. This rule is implemented by, among other things, foreign key constraints. So yes, the idea of a foreign key is "a" relational rule. – Triynko Feb 24, 2010 at 22:01

1 @lubos: To clarify, you're talking about whether or not you're going to use a particular feature, but I'm talking about whether that feature's presence is necessary to have a complete, fully functional RDBMS. Referential constraints, if and when you choose to use them, is something that should be enforced within the RDBMS (rather than the application), so it's a feature that should be there, and in that sense it is a requirement of the relational model if you're going to develop a complete RDBMS. – Triynko Mar 1, 2010 at 18:27

24 Answers

Sorted by: Highest score (default)

111

Foreign keys help enforce referential integrity at the data level. They also improve performance because they're normally indexed by default.

Share Improve this answer

Follow

answered Aug 20, 2008 at 20:19

John Topley

115k ● 47 ● 199 ● 240

60 If you need an index create one, this should not be a primary reason for FKs. (In fact in certain circumstances (More inserts than selects for example) maintaining a FK might be slower.) – Robert Mar 21, 2010 at 17:28

13 That's a horrible answer FKs genaerally can add extra overhead not improve performance. – Agile Jedi Feb 13, 2015 at 18:42

In SQL-Server, they are not indexed by default on either the referee or the referrer. sqlskills.com/blogs/kimberly/… – user420667 Jun 27, 2016 at 20:26

2 Nor in Oracle; you have to create indexes (on the FK columns) yourself. – Littlefoot Feb 10, 2018 at 20:17

"because they're normally indexed by default." in postgres not – vi0 Oct 26 at 9:33

66

Foreign keys can also help the programmer write less code using things like `ON DELETE CASCADE`. This means that if you have one table containing users and another containing orders or something, then deleting a user could automatically delete all orders that point to that user.

Share Improve this answer

Follow

edited Sep 18, 2018 at 6:47

Nae

15.3k ● 8 ● 61 ● 85

answered Aug 20, 2008 at 20:22

Greg Hewgill

990k ● 191 ● 1.2k ● 1.3k

5 @Greg Hewgill This could pontentially lead to a lot of problems. You should be very careful with thinks like DELETE CASCADE, as in many cases, you would want to keep the orders created by a user when deleting the user. – Kibbee Aug 20, 2008 at 20:28

11 Although this should probably be handled in business logic layer. Deciding whether or not to keep related child records, is not quite the same as ensuring that no values violate foreign key relationships. – Codewerks Oct 6, 2008 at 21:38

5 The other issue is auditing, if auditing is not done at the db level, cascading updates or deletes will invalidate your audit trail. – si618 Sep 1, 2009 at 4:52

@Codewerks: Business logic can be in the DB. – Fantius Feb 1, 2011 at 19:31

@Kibbee I disagree. If you want to keep the orders of client X, then it makes no sense to delete client X to begin with -it contains important data relevant to the order. You would only

delete client X if you intend to also delete everything that references it -in which case ON DELETE CASCADE does exactly that. If you don't want a cascade, then you shouldn't delete at all. – Juan Perez Sep 10, 2023 at 14:20

50

I can't imagine designing a database without foreign keys. Without them, eventually you are bound to make a mistake and corrupt the integrity of your data.

They are not *required*, strictly speaking, but the benefits are huge.

I'm fairly certain that FogBugz does not have foreign key constraints in the database. I would be interested to hear how the Fog Creek Software team structures their code to guarantee that they will never introduce an inconsistency.

Share Improve this answer

Follow

edited Sep 18, 2018 at 7:04

Nae

15.3k ● 8 ● 61 ● 85

answered Aug 20, 2008 at 20:24

Eric Z Beard

38.4k ● 27 ● 101 ● 147

52 Joel: "So far we've never had a problem." So far, I've never driven into a lamp-post. But I still think it's a good idea to wear seat belts ;-) – Tony Andrews Nov 4, 2008 at 10:36

2 May be you never have SEEN the problem, but may be it's there... The most of databases use a convention like id_xxx

that is exactly the same that ixXXX – FerranB Feb 5, 2009 at 21:54

2 @Joel: Naming conventions in place of enforcement of rules? Might as well do away with type while you're at it. – jcollum Sep 17, 2009 at 20:54

9 @Eric: You're holding Fog Creek up as some sort of avatar of software development here. If you said "A company in New York City does not have foreign keys in their db ..." we'd all say "And?" – jcollum Sep 17, 2009 at 20:56

3 Eric: FogBugz uses a naming convention for foreign keys. For example ixBug is understood to be an index into the primary key of the table Bug. So far we've never had a problem. -- Joel Spolsky – Sam Saffron Feb 13, 2012 at 1:48

48

A database schema without FK constraints is like driving without a seat belt.

One day, you'll regret it. Not spending that little extra time on the design fundamentals and data integrity is a sure fire way of assuring headaches later.

Would you accept code in your application that was that sloppy? That directly accessed the member objects and modified the data structures directly.

Why do you think this has been made hard and *even unacceptable* within modern languages?

Share Improve this answer

Follow

edited Apr 5, 2022 at 6:44

philipxy

15.1k • 6 • 42 • 94

answered Aug 20, 2008 at 22:29

Guy
9,826 • 7 • 39 • 43

4 +1 for a good analogy between encapsulation and FK/PK relationships. – jcollum Sep 17, 2009 at 20:58

@jumping_monkey Your edit was not a clarification of what the author said, it added something new, which is inappropriate. I rolled it back. It should be a comment suggestion to the author. Also it was not grammatically correct & it left out a space & it had unnecessary boldface, it was a bad edit for that content. Help center – philipxy Apr 5, 2022 at 6:48

Hey @philipxy, thanks for checking that. I would argue that it is, as referential integrity(which is the whole point of foreign keys) is a subset of data integrity. That's the reason i added it, with the link to read more about it. Anyway, you can remove it, your call. I like Guy's answer, that's what is more important. Cheers. – jumping_monkey Apr 5, 2022 at 8:49

23

Yes.

1. They keep you honest
2. They keep new developers honest
3. You can do `ON DELETE CASCADE`
4. They help you to generate nice diagrams that self explain the links between tables

Share Improve this answer edited Aug 10, 2013 at 19:30

Follow

Peter Mortensen
31.6k ● 22 ● 109 ● 133

answered Aug 20, 2008 at 20:37

csmba
4,083 ● 3 ● 33 ● 42

2 what do you mean by honesty? – Daniel Sep 6, 2015 at 2:33

3 Honest with the conception I guess. It prevent you from cheating with the data by doing quick and lame programming. – Oreste Viron Jan 8, 2017 at 21:09

16

> Suppose a programmer is actually doing this in the right manner already

Making such a supposition seems to me to be an extremely bad idea; in general software is phenomenally buggy.

And that's the point, really. Developers can't get things right, so ensuring the database can't be filled with bad data is a Good Thing.

Although in an ideal world, natural joins would use relationships (i.e. FK constraints) rather than matching column names. This would make FKs even more useful.

Share Improve this answer

Follow

answered Aug 20, 2008 at 20:35

DrPizza
18.3k ● 7 ● 42 ● 53

2 Good point, it would be nice to join two tables with "ON [Relationship]" or some other keyword and let the db figure out what columns are involved. Seems pretty reasonable really. – jcollum Sep 17, 2009 at 20:58

15

Personally, I am in favor of foreign keys because it formalizes the relationship between the tables. I realize that your question presupposes that the programmer is not introducing data that would violate referential integrity, but I have seen way too many instances where data referential integrity is violated, despite best intentions!

Pre-foreign key constraints (aka declarative referential integrity or DRI) lots of time was spent implementing these relationships using triggers. The fact that we can formalize the relationship by a declarative constraint is very powerful.

@John - Other databases may automatically create indexes for foreign keys, but SQL Server does not. In SQL Server, foreign key relationships are only constraints. You must defined your index on foreign keys separately (which can be of benefit.)

Edit: I'd like to add that, IMO, the use of foreign keys in support of ON DELETE or ON UPDATE CASCADE is not necessarily a good thing. In practice, I have found that cascade on delete should be carefully considered based on the relationship of the data -- e.g. do you have a natural parent-child where this may be OK or is the

related table a set of lookup values. Using cascaded updates implies you are allowing the primary key of one table to be modified. In that case, I have a general philosophical disagreement in that the primary key of a table should not change. Keys should be inherently constant.

Share Improve this answer Follow

edited Aug 21, 2008 at 12:12

answered Aug 20, 2008 at 20:35

Peter Meyer
26.1k ● 1 ● 35 ● 53

10

Without a foreign key how do you tell that two records in different tables are related?

I think what you are referring to is referential integrity, where the child record is not allowed to be created without an existing parent record etc. These are often known as foreign key constraints - but are not to be confused with the existence of foreign keys in the first place.

Share Improve this answer Follow

answered Aug 20, 2008 at 20:24

samjudson
56.8k ● 7 ● 60 ● 69

I suppose you are talking about *foreign key constraints enforced by the database*. You probably already are using

10

foreign keys, you just haven't told the database about it.

> Suppose a programmer is actually doing this in the right manner already, then do we really need the concept of foreign keys?

Theoretically, no. However, there have never been a piece of software without bugs.

Bugs in application code are typically not that dangerous - you identify the bug and fix it, and after that the application runs smoothly again. But if a bug allows currupt data to enter the database, then you are stuck with it! It's very hard to recover from corrupt data in the database.

Consider if a subtle bug in FogBugz allowed a corrupt foreign key to be written in the database. It might be easy to fix the bug and quickly push the fix to customers in a bugfix release. However, how should the corrupt data in dozens of databases be fixed? *Correct* code might now suddenly break because the assumptions about the integrity of foreign keys dont hold anymore.

In web applications you typically only have one program speaking to the database, so there is only one place where bugs can corrupt the data. In an enterprise application there might be several independent applications speaking to the same database (not to mention people working directly with the database shell).

There is no way to be sure that all applications follow the same assumptions without bugs, always and forever.

If constraints are encoded in the database, then the worst that can happen with bugs is that the user is shown an ugly error message about some SQL constraint not satisfied. This is *much* prefereable to letting currupt data into your enterprise database, where it in turn will break all your applications or just lead to all kinds of wrong or misleading output.

Oh, and foreign key constraints also improves performance because they are indexed by default. I can't think of any reason *not* to use foreign key constraints.

Share Improve this answer

Follow

edited Aug 10, 2013 at 19:42

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Sep 17, 2008 at 13:24

JacquesB

42.6k ● 13 ● 75 ● 88

8

Is there a benefit to not having foreign keys? Unless you are using a crappy database, FKs aren't that hard to set up. So why would you have a policy of avoiding them? It's one thing to have a naming convention that says a column references another, it's another to know the database is actually verifying that relationship for you.

Share Improve this answer

answered Aug 22, 2008 at 15:40

Follow

Tundey
2,965 ● 1 ● 24 ● 28

2 The benefit is performance. I'm not saying you should not have FK's, just strictly answering your question. Suppose you have a huge (100GB) table with a FK to another table. If you delete a record from "another table" - the engine will scan the entire 100GB table to make sure you're not deleting anything useful. Unless you have that FK column indexed (FK are not indexed by default in SQL Server) – Alex from Jitbit Jun 26, 2021 at 18:58

I'm not a db expert but I don't think that should be how you address performance problems. Like you said, you can index the FK column (which you'll realize pretty quickly that SQL doesn't do by default) and you do want the database to enforce that the record you're deleting isn't in use in your 100GB table. – Tundey Jun 28, 2021 at 14:40

1 I (mostly) agree. Just wanted to mention that when you're managing databases of size of tens of terabytes, dropping FKs is an an unspoken common practice among DBAs. In essence, at this scale you're moving to "NoSQL land", where you have to drop one of the "A", "C", "I" or "D" out of the "ACID" principle. – Alex from Jitbit Jun 28, 2021 at 16:46

8

FKs are very important and should always exist in your schema, unless you are eBay.

Share Improve this answer

Follow

answered Apr 19, 2009 at 14:53

cherouvim
31.9k ● 15 ● 105 ● 155

2 that link is actually extremely fascinating... i'd truly like to know more details and i'm somewhat scared to use ebay now. for other people: click on the 4th question to see what he says about their db structure. the whole interview is worth watching, though. also... `unibrow` – gloomy.penguin Aug 23, 2013 at 18:52

6

Foreign keys allow someone who has not seen your database before to determine the relationship between tables.

Everything may be fine now, but think what will happen when your programmer leaves and someone else has to take over.

Foreign keys will allow them to understand the database structure without trawling through thousand of lines of code.

Share Improve this answer

Follow

answered Sep 17, 2008 at 4:44

Craig

105 ● 1 ● 3

6

I think **some single thing** at some point must be responsible for ensuring valid relationships.

For example, Ruby on Rails does not use foreign keys, but it validates all the relationships itself. If you only ever access your database from that Ruby on Rails application, this is fine.

However, if you have other clients which are writing to the database, then without foreign keys they need to implement their own validation. You then have two copies of the validation code which are most likely different, which any programmer should be able to tell is a cardinal sin.

At that point, foreign keys really are neccessary, as they allow you to move the responsibility to a single point again.

Share Improve this answer

Follow

edited Aug 10, 2013 at 19:29

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Aug 20, 2008 at 20:28

Orion Edwards

123k ● 66 ● 245 ● 339

2 It's like an onion. FKs are the last layer of defense. Unless it's an embedded local database, apps trying to do referential integrity is always an bad idea. – Fabricio Araujo Oct 4, 2013 at 19:23

5

> As far as I know, foreign keys are used to aid the programmer to manipulate data in the correct way.

FKs allow the DBA to protect data integrity from the fumbling of users when the programmer *fails* to do so,

and sometimes to protect against the fumbling of programmers.

> Suppose a programmer is actually doing this in the right manner already, then do we really need the concept of foreign keys?

Programmers are mortal and fallible. FKs are *declarative* which makes them harder to screw up.

> Are there any other uses for foreign keys? Am I missing something here?

Although this is not why they were created, FKs provide strong reliable hinting to diagramming tools and to query builders. This is passed on to end users, who desperately need strong reliable hints.

Share Improve this answer

Follow

answered Oct 6, 2008 at 21:26

Peter Wone
18.7k • 14 • 94 • 147

4

They are not strictly necessary, in the way that seatbelts are not strictly necessary. But they can really save you from doing something stupid that messes up your database.

It's so much nicer to debug a FK constraint error than have to reconstruct a delete that broke your application.

Share Improve this answer

Follow

answered Aug 20, 2008 at 20:57

Mark Harrison

304k ● 131 ● 350 ● 489

4

They are important, because your application is not the only way data can be manipulated in the database. Your application may handle referential integrity as honestly as it wants, but all it takes is one bozo with the right privileges to come along and issue an insert, delete or update command at the database level, and all your application referential integrity enforcement is bypassed. Putting FK constraints in at the database level means that, barring this bozo choosing to disable the FK constraint before issuing their command, the FK constraint will cause a bad insert/update/delete statement to fail with a referential integrity violation.

Share Improve this answer

Follow

answered Sep 17, 2008 at 19:09

Mike McAllister

1,549 ● 2 ● 12 ● 15

3

I think about it in terms of cost/benefit... In MySQL, adding a constraint is a single additional line of DDL. It's just a handful of key words and a couple of seconds of thought. That's the only "cost" in my opinion...

Tools love foreign keys. Foreign keys prevent bad data (that is, orphaned rows) that may not affect business logic or functionality and therefor go unnoticed, and build up. It also prevents developers who are unfamiliar with the

schema from implementing entire chunks of work without realizing they're missing a relationship. Perhaps everything is great within the scope of your current application, but if you missed something and someday something unexpected is added (think fancy reporting), you might be in a spot where you have to manually clean up bad data that's been accumulating since the inception of the schema without a database enforced check.

The little time it takes to codify what's already in your head when you're putting things together could save you or someone else a bunch of grief months or years down the road.

The question:

> Are there any other uses for foreign keys? Am I missing something here?

It is a bit loaded. Insert comments, indentation or variable naming in place of "foreign keys"... If you already understand the thing in question perfectly, it's "no use" to you.

edited Aug 10, 2013 at 19:35

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Aug 21, 2008 at 3:30

danb

10.4k ● 15 ● 61 ● 77

2

Entropy reduction. Reduce the potential for chaotic scenarios to occur in the database. We have a hard time as it is considering all the possiblilites so, in my opinion, entropy reduction is key to the maintenance of any system.

When we make an assumption for example: each order has a customer that assumption should be enforced by *something*. In databases that "something" is foreign keys.

I think this is worth the tradeoff in development speed. Sure, you can code quicker with them off and this is probably why some people don't use them. Personally I have killed a number of hours with NHibernate and some foreign key constraint that gets angry when I perform some operation. HOWEVER, I know what the problem is so it's less of a problem. I'm using normal tools and there are resources to help me work around this, possibly even people to help!

The alternative is allow a bug to creep into the system (and given enough time, it will) where a foreign key isn't set and your data becomes inconsistent. Then, you get an unusual bug report, investigate and "OH". The database is screwed. Now how long is that going to take to fix?

Share Improve this answer

Follow

edited Aug 10, 2013 at 19:45

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Dec 1, 2009 at 23:22

Quibblesome

25.4k ● 10 ● 62 ● 104

1

You can view foreign keys as a constraint that,

- Help maintain data integrity
- Show how data is related to each other (which can help in enforcing business logic and rules)
- If used correctly, can help increase the efficiency with which the data is fetched from the tables.

Share Improve this answer

Follow

edited Aug 10, 2013 at 19:27

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Aug 20, 2008 at 20:26

Pascal

4,127 ● 8 ● 34 ● 29

1

We don't currently use foreign keys. And for the most part we don't regret it.

That said - we're likely to start using them a lot more in the near future for several reasons, both of them for similar reasons:

1. Diagramming. It's so much easier to produce a diagram of a database if there are foreign key relationships correctly used.

2. Tool support. It's a lot easier to build data models using Visual Studio 2008 that can be used for LINQ to SQL if there are proper foreign key relationships.

So I guess my point is that we've found that if we're doing a lot of manual SQL work (construct query, run query, blahblahblah) foreign keys aren't necessarily essential. Once you start getting into using tools, though, they become a lot more useful.

Share Improve this answer

Follow

edited Aug 10, 2013 at 19:32

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Aug 20, 2008 at 20:55

John Christensen

5,030 ● 1 ● 29 ● 26

1 I work on systems that don't use them. And I regret it regularly. I have seen more instances I can count of non-sensical data that would have been prevented by proper constraints. – recursive Apr 19, 2009 at 15:15

And having been working with foreign keys on our current project for nearly six months, I totally agree with this comment. – John Christensen Apr 20, 2009 at 13:17

1

The best thing about foreign key constraints (and constraints in general, really) are that you can rely on them when writing your queries. A lot of queries can

become a lot more complicated if you can't rely on the data model holding "true".

In code, we'll generally just get an exception thrown somewhere - but in SQL, we'll generally just get the "wrong" answers.

In theory, SQL Server could use constraints as part of a query plan - but except for check constraints for partitioning, I can't say that I've ever actually witnessed that.

Share Improve this answer

Follow

edited Aug 10, 2013 at 19:33

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Aug 20, 2008 at 22:10

Mark Brackett

85.6k ● 17 ● 111 ● 155

> Uniqueness constraints indicate high cardinality which is used by the optimiser in selecting a join mechanism.
> – Peter Wone Dec 22, 2009 at 22:14

1

Foreign keys had never been explicit (FOREIGN KEY REFERENCES table(column)) declared in projects (business applications and social networking websites) which I worked on.

But there always was a kind of convention of naming columns which were foreign keys.

It's like with database normalization -- you have to know what are you doing and what are consequence of that (mainly performance).

I am aware of advantages of foreign keys (data integrity, index for foreign key column, tools aware of database schema), but also I am afraid of using foreign keys as general rule.

Also various database engines could serve foreign keys in a different way, which could lead to subtle bugs during migration.

Removing all orders and invoices of deleted client with ON DELETE CASCADE is the perfect example of nice looking, but wrong designed, database schema.

Share Improve this answer

Follow

edited Aug 10, 2013 at 19:37

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Aug 21, 2008 at 9:06

Grzegorz Gierlik

11.2k ● 4 ● 48 ● 57

0

Yes. The ON DELETE [RESTRICT|CASCADE] keeps developers from stranding data, keeping the data clean. I recently joined a team of Rails developers who did not focus on database constraints such as foreign keys.

Luckily, I found these:
http://www.redhillonrails.org/foreign_key_associations.ht

ml -- RedHill on Ruby on Rails plug-ins generate foreign keys using the convention over configuration style. A migration with *product_id* will create a foreign key to the *id* in the *products* table.

Check out the other great plug-ins at RedHill, including migrations wrapped in transactions.

Share Improve this answer

Follow

edited Aug 10, 2013 at 19:38

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Sep 17, 2008 at 4:30

unknown (yahoo)

0

If you plan on generating your data access code, ie, Entity Framework or any other ORM you entirely lose the ability to generate a hierarchical model without Foreign Keys

Share Improve this answer

Follow

answered Feb 5, 2020 at 17:16

Mike Griffin

26 ● 2

Highly active question. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.