**Foreign Key Advantage And Disadvantage:**

The basic concept about foreign key:

A foreign key is a DBMS concept for ensuring database integrity. i.e., it ensures your data is consistent. Suppose, You have a table named Person and you have the concept that a person can have many phonenumbers as he/she want. So, another table is required named PersonPhoneNumberMap. Now, suppose the table looks like following:

**Person:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **NULL** | **Key** | **Default** | **Extra** |
| ID | INT(11) | NO | PRI | NULL | AUTO\_  INCREMENT |
| NAME | VARCHAR(80) | NO |  | NULL |  |
| AGE | TINYINT | NO |  | NULL |  |

And **PersonPhoneNumberMap**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Field** | **Type** | **NULL** | **Key** | **Default** | **Extra** |
| PersonID | INT(11) | NO | PRI | NULL |  |
| PhoneNumber | VARCHAR(20) | NO | PRI | NULL |  |

Now, when, a person is removed, if his/her records is not deleted from PersonPhoneNumberMap, PersonPhoneNumberMap contains invalid data. Also, PersonPhoneNumberMap might contain PersonID which does not exist in Person table.

Now, if instead of making PersonID and PhoneNumber as primary key, PersonID is created as a foreign key which refers to the Person table and set Option CASCADE on deleting, it will make database more consistent.

Any performance implications/improvements will be specific to the database technology being used and are secondary to the purpose of a foreign key.  
  
It is good practice in SQL Server to ensure that all foreign keys have at least a non clustered index on them.

The Reasons to use Foreign Keys and the reasons not to use Foreign keys are listed below: (taken from stackoverflow)

**Reasons to use Foreign Keys:**

* you won't get Orphaned Rows
* you can get nice "on delete cascade" behavior, automatically cleaning up tables
* knowing about the relationships between tables in the database helps the Optimizer plan your queries for most efficient execution, since it is able to get better estimates on join cardinality.
* Foreign Keys give a pretty big hint on what statistics are most important to collect on the database, which in turn leads to better performance
* They enable all kinds of auto-generated support -- ORMs can generate themselves, visualization tools will be able to create nice schema layouts for you, etc
* someone new to the project will get into the flow of things faster since otherwise implicit relationships are explicitly documented

**Reasons not to use Foreign Keys:**

* You are making the DB work extra on every CRUD operation because it has to check FK consistency. This can be a big cost if you have a lot of churn.
* By enforcing relationships, Foreign Keys specify an order in which you have to add/delete things, which can lead to refusal by the DB to do what you want. (Granted, in such cases, what you are trying to do is create an Orphaned Row, and that's not usually a good thing). This is especially painful when you are doing large batch updates, and you load up one table before another, with the second table creating consistent state (but should you be doing that sort of thing if there is a possibility that the second load fails and your database is now inconsistent?).
* Sometimes you know beforehand your data is going to be dirty, you accept that, and you want the DB to accept it.
* You are just being lazy.

However, actually lot of things are related to this topic.

Like, Foreign key use can boost fetch data type query. But, it gives burden during insertion,deletion and update on parent table.

Here is another example:

create table Employee(EmployeeID int primary key);

create table EmployeeOrder(OrderID int primary key, EmployeeID int not null constraint fkOrderCust references Employee(EmployeeID));

You can see this time optimizer executes the EXISTS operator and Employee table is shown in execution plan. This is because no foreign key constraint was found and SQL Server could not be sure that all orders actually have valid employee references. Therefore it had to execute the EXISTS operator.   
  
I have found subtreecost of the first query was 0.0376 and the second one is 0.0443. Remember these are empty tables. For a large table, this can make a huge difference in performance.   
  
But for any DML operation i.e. Insert, Update & Delete, foreign key constraint degrades performance as SQL Server needs to validate data with primary table's column. However, the data is referentially correct and that can save some additional time with queries being able to rely on data integrity.