Choosing PostgreSQL

Open Source Maturity Model and PostgreSQL

With a long development history, one of the strongest development communities in the world, and a global reputation for high quality software engineering, PostgreSQL gets the job done. Organizations from small online businesses to large corporations and government institutions trust PostgreSQL to handle their most valuable data and mission critical applications.

The **Open Source Maturity Model** provides specific guidelines for evaluating open source projects such as PostgreSQL. For businesses exploring the use of Open Source products, these guidelines provide valuable criteria for selecting and comparing products. Measured against these guidelines, PostgreSQL's product maturity makes it the best choice for your Open Source Relational Database Management System:

Product	Immature	Mature
Age	The project has just started. The stability of the developers group and need for the product are unclear.	The project is been active for some time. The project stability and need for the project are no longer issues.
	PostgreSQL has been an active open source project for almost 19 years. PostgreSQL's success shows a clear need for a stable and mature product in the rapidly-expanding open source relational database market.	
Licensing	Not fully described or clearly unsuitable for the product.	One of the standard licenses. Offers clear motives for choosing the license type, which is supported by the user community.
	PostgreSQL has always carried the clear and simple (modified) BSD license. Redistribution and use in source and binary forms, with or without modification, are permitted as long as the copyright is maintained.	
Human Hierarchies	Original founder is lead developer and solely responsible. Development depends on a single person.	Large community, multiple leaders who coordinate. Separation of development and maintenance.
	PostgreSQL is led by a small team of core members of the PostgreSQL Global Development Group and the larger international development community. Most people who develop also maintain their own code area. Core members have different areas of coordination.	
Developer community	Small tight knit group.	Very active developers community, several hand-offs have taken place. Documented procedures to becoming a member.
	PostgreSQL's active community is shown by the constant participation of both users and developers on all of the mailing lists and on the PostgreSQL IRC (#postgresql on freenode.net). Major hand-offs have taken place for PostgreSQL twice: in 1992 for a commercial venture, and in 1996 for the PostgreSQL Global Development Group. Minor hand-offs of responsibilities occur as necessary.	
Selling points	Only enthusiasm.	Commercial issues like security or maintainability.
	PostgreSQL is long past the fundamental building blocks of a large system. Code maintainability is addressed. Security, standards and data integrity call for prompt attention.	

For more information on the Open Source Maturity Model see:
www.seriouslyopen.org

Choosing PostgreSQL

Integrity

	Immature	Mature
Modularity	No modules, still one single product. Functionality is offered on a take all or nothing basis.	Product has been separated into smaller pieces of functionality. Users can select which parts are required. Allows tailoring of the product to a particular situation.
	PostgreSQL has a variety of client interfaces and administration GUIs that are written independently of the main server. The interfaces and languages are modularized by design so that new pl languages and indexing methods are easily added. Many extensions are available.	
Collaboration with other products	Not in focus yet. Product development is still firmly centered on core functionality.	Product is close to completion. Attention is shifting to linking the product to other products.
	PostgreSQL is still firmly centered on core functionality. This focus creates an imperative to "work well with others" as is shown by the wide variety of client interfaces and pl languages.	

Use

	Immature	Mature
Standards	Uses proprietary protocols or uses dead end technologies.	Uses current accepted protocols and models. Deals with issues surrounding standards, integration etc.
	PostgreSQL supports SQL standards in all new and improved features. And vice versa: standards like SQL2003 are officially adopting features long available in PostgreSQL. Packaging and distributions use accepted protocols.	
Support	Just within the own community and then only provided by a small minority within that community.	Besides community support, professional support can be purchased. The community itself is active and questions draw responses from a wide section of the community.
	PostgreSQL support is extremely effective, friendly and free via very active, multi-lingual mailing lists and IRC. All members of the community, including the core team contribute to the support effort. Commercial support is also available from Varlena, LLC, PgSQL Inc., CommandPrompt and others.	
Ease of deployment	Little to no training facilities or courses. Documentation is poor, particularly with regard to maintenance.	Training or courses available. In addition to well written documentation lots of HowTos of users detailing particular situations. Within the group knowledge about maintaining the product is readily available.
	PostgreSQL comes with full and complete documentation, translated into many languages. Additional technical documentation, training guides, tutorials, as well as internal documentation is available on the www.postgresql.org site.	
User Community	Small group, possible with a high proportion of lurkers.	Large group that often has divided itself into sub-groups. Each group has a specific focus. Traffic in general is high-volume. Multiple cases of successful implementation across a range of companies. Well known.
	PostgreSQL has divided itself organically into several mailing lists, general, hackers (developes), SQL, interfaces, packages and platforms. It has divided out the client and interface projects. PostgreSQL has a very active advocacy group.	

For more information on the Open Source Maturity Model see:
www.seriouslyopen.org