Proposta Velha

Esta palestra foi apresentada na Python Nordeste 2018. Slides disponíveis em: http://bit.ly/pyne-normal

Uma boa prática comumente ignorada é considerar o BD como uma representação de fatos sobre a realidade. Se houver mais de uma maneira de extrair um único fato do BD, então há uma redundância nele que pode causar anomalias nos dados e bugs na aplicação. A resposta clássica contra a redundância em BDs é normalização com formas normais 1NF, 2NF, etc. Desenvolvedores frequentemente ignoram a importância das formas normais devido ao seu excesso de formalismo. Porém, até as formas normais não são suficientes para evitar anomalias, visto que elas se preocupam com redundâncias apenas em uma única tabela. Por isso, devemos ir além das formas normais para evitar problemas.

Nesta palestra, apresentaremos regras de normalização em linguagem amigável e natural que vão além das formas normais. Vamos entender as dependências que os requisitos do sistema induzem aos dados, mesmo aqueles que se organizam em tabelas/models diferentes. Vamos usar exemplos reais com o ORM do Django, nos quais existem dependências não-triviais de implementar corretamente. Vamos mostrar como técnicas de normalização e desnormalização resolvem problemas de redundâncias, inconsistências, anomalias e bugs. Às vezes normalizar tabelas causa problemas de performance como consultas lentas, então discutiremos como desnormalizar e ganhar performance sem introduzir inconsistências usando funcionalidades do BD como indexes, materialized views e triggers, bem como ferramentas de caching e NoSQL.

Proposta Nova (muito melhor!)

Esta palestra foi apresentada na Python Nordeste 2018. Slides disponíveis em: http://bit.ly/pyne-normal

Será que você como programador Django realmente leva em consideração todas possíveis redundâncias de dados dos seus models? Tudo bem, talvez na primeira vez que você definiu seu esquema de banco de dados você tenha pensado nisso, mas e quando fez sua última migração? Será que o novo campo que você introduziu não é redundante pois é computável a partir de outros campos?

Muitos desenvolvedores, mesmo os mais experientes, ignoram a boa prática que é considerar o BD como uma representação de fatos sobre a realidade. Se houver mais de uma maneira de extrair um único fato do BD, então há uma redundância nele! No pior caso, uma redundância pode causar anomalias nos dados e bugs na aplicação. No melhor caso, uma redundância exige mais código de aplicação com testes para manter dados redundantes consistentes. Ou seja, redundância é uma dor de cabeça! Será que existe uma forma inteligente de descobrir e evitar redundância de dados? Ou quando for necessário ser redundante por questões de performance, fazer dados computados com o mínimo de código e o máximo de automação? A resposta é sim: com técnicas de Normalização e Desnormalização respectivamente.

Normalização é um processo que envolve seguir uma série de regras para reestruturar um banco de dados sem perder os fatos originais representados por ele. O conjunto tradicional dessas regras são as Formas Normais: 1NF, 2NF, etc... sim, aquela coisa chata e super formal que alguns já viram. Mas, na verdade, nem as formas normais são suficientes para evitar tipos comuns de redundância*! Por isso, nesta palestra você aprenderá uma regra mais geral sobre normalização em uma linguagem amigável. Mostraremos diferentes exemplos em Django sobre redundâncias não-triviais e como resolvê-las. Também veremos como query expressions, uma funcionalidade de Django que muitos não

conhecem e que permite evitar redundâncias ao computar dados em momento de consulta.

Sabendo que Normalização pode gerar problemas de performance, vamos discutir **Desnormalização**, que é diferente de não-normalizar. Na realidade, desnormalizar significa usar técnicas como cronjobs, indexes, caching, materialized views, triggers e NoSQL para agilizar consultas em uma tabela normalizada sem perder consistência.

Com os insights sobre Normalização e Desnormalização apresentados nesta palestra, programadores Django aprenderão novas técnicas para ter um BD que representa fatos sobre a realidade sem inconsistências.

** Talvez você duvide disso, mas realmente as Formas Normais só discutem dependências (e por sua vez redundâncias) em uma única tabela, ignorando dependências entre tabelas que são muito comuns em aplicações reais. Mais sobre isso neste artigo revisado por Codd, Fagin e Date, figuras chave do modelo relacional.*