

Grupo Handbook

Handbook de Questões de TI

comentadas para CONCURSOS

Além do gabarito

Volume 7

*Processos de Negócio
Parte 1
Fundação Cesgranrio*

Prefácio

Este é o volume 7 da série *Handbook de Questões de TI Comentadas para Concursos – Além do Gabarito*, que traz para você uma coletânea especial de questões da área de processos selecionadas de provas organizadas pela Fundação Cesgranrio.

Entre as provas das quais as questões foram selecionadas, estão a prova para Analista de Sistema Pleno – Processos para a Petrobras em 2006, e, também, a prova para Analista de Sistemas Júnior – Processos de Negócio para a Petrobras aplicada em 2008.

Notoriamente, a Fundação Cesgranrio vem sendo a responsável pela organização de concursos de TI de alta concorrência e visibilidade, como o da Petrobras e suas subsidiárias. Mais recentemente, a Fundação Cesgranrio também foi a escolhida para realizar os concursos do Banco Central e de outras instituições do ciclo de gestão do Governo Federal.

Uma interessante característica das provas aplicadas pela Fundação Cesgranrio é o fato das questões possuírem pesos diferentes, o que exige dos candidatos ainda mais sagacidade. Além disso, é comum que sejam cobrados temas recentes de TI, como forma de selecionar profissionais atualizados e dinâmicos.

Por isso, o Grupo Handbook de TI preparou para você uma edição especial com questões da área de processos, abordando temas clássicos e modernos da área de TI. Este volume é especialmente útil para todos que desejam prestar concursos na área de processos, em especial, aqueles organizados pela Fundação Cesgranrio.

Bons estudos,

Grupo Handbook de TI

Direitos Autorais

Este material é registrado no Escritório de Direitos Autorais (EDA) da Fundação Biblioteca Nacional. Todos os direitos autorais referentes a esta obra são reservados exclusivamente aos seus autores.

Os autores deste material não proíbem seu compartilhamento entre amigos e colegas próximos de estudo. Contudo, a reprodução, parcial ou integral, e a disseminação deste material de forma indiscriminada através de qualquer meio, inclusive na Internet, extrapolam os limites da colaboração. Essa prática desincentiva o lançamento de novos produtos e enfraquece a comunidade concurseira Handbook de TI.

A série *Handbook de Questões de TI Comentadas para Concursos – Além do Gabarito* é uma produção independente e contamos com você para mantê-la sempre viva.

Grupo Handbook de TI

Canais de Comunicação

O Grupo Handbook de TI disponibiliza diversos canais de comunicação para os concurseiros de TI.

Loja Handbook de TI

Acesse a nossa loja virtual em <http://www.handbookdeti.com.br>

Serviço de Atendimento

Comunique-se diretamente conosco através do e-mail faleconosco@handbookdeti.com.br

Twitter do Handbook de TI

Acompanhe de perto promoções e lançamentos de produtos pelo nosso Twitter <http://twitter.com/handbookdeti>

1. **Assuntos relacionados:** *Banco de Dados, ACID, Atomicidade em Transação, Consistência em Transação, Isolamento entre Transações, Durabilidade em Transação,*
Banca: *Cesgranrio*
Instituição: *Petrobras*
Cargo: *Analista de Sistemas Pleno - Processos*
Ano: *2006*
Questão: *22*

T1	
1	Ler(A);
2	A = A - 30;
3	Escrever(A);
4	Ler(B);
5	B = B + 30;
6	Escrever(B);

A transação T1, pertencente a um sistema bancário e definida pelas operações listadas acima, é responsável pela transferência de R\$ 30,00 da conta A para a conta B. Considere também uma transação T2 que esteja sendo executada simultaneamente a T1. Caso a transação T2 realize uma operação Escrever(B) após a execução da operação 4 e antes da execução da operação 6 por T1, qual das propriedades das transações estará sendo violada no banco de dados do sistema bancário?

- (a). Atomicidade.
- (b). Distributividade.
- (c). Consistência.
- (d). Durabilidade.
- (e). Isolamento.

Solução:

O cerne desta questão está relacionado às propriedades de um transação em banco de dados.

Uma transação é um conjunto de operações em banco de dados, possivelmente de leitura e/ou alteração, que devem ser executadas como uma única unidade de trabalho. Elas devem ser processadas de forma confiável, de tal forma que nenhum dado se perca, ou seja indevidamente alterado, em decorrência de múltiplos usuários e possíveis falhas de sistema. Um exemplo típico de transação é a uma transferência bancária entre contas. Ela exige várias operações (débito na conta de origem, crédito na conta de destino, cobrança de taxas, etc.), mas todas são processadas como uma única unidade. É dessa forma que se elimina a possibilidade de situações indesejadas no que se refere a banco de dados, como: débito na conta de origem sem o devido crédito na conta de destino; crédito na conta de destino sem o devido débito na conta de origem; cobrança de taxas sem a devida efetivação da transferência.

São os Sistemas de Gerenciamento de Banco de Dados (SGBD) que devem assegurar que transações obedeçam certas propriedades. Afinal de contas, são eles que são responsáveis pelo gerenciamento do banco de dados (como o seu próprio nome já diz). Dessa forma, garante-se que sempre os dados do banco estarão íntegros.

As propriedades mais importantes e mais difundidas entre os SGBDs são as que formam o acrônimo ACID:

- **atomicidade:** ela assegura que a transação é indivisível. Ou todas as operações que a compõem são concluídas ou nada é realizado. Isso implica que quando uma transação tem que ser abortada, por qualquer motivo, as suas operações que eventualmente forma concluídas devem ser desfeitas. Essa forma de “desfazer” uma transação recebe o nome de rollback. No nosso exemplo acima, é esta a propriedade que garante que nenhuma situação não desejada citada acima ocorra;
- **consistência:** ela garante que as regras de integridade do banco de dados, definidas pelo seu projetista, serão sempre preservadas. Caso uma transação tente levar o banco a um estado não-integro, o SGBD abortará essa transação. Vejamos um exemplo no caso de uma transferência bancária. Caso haja hipoteticamente uma restrição de integridade que diga que qualquer conta bancária não possa ficar com saldo negativo e uma transação tente transferir R\$500,00 a partir de uma conta com apenas R\$200,00, o SGBD não permitirá a conclusão dessa transação;
- **isolamento:** ela garante que transações não interferem umas com as outras, exceto de forma permitida. De maneira geral, uma transação nunca deve sobrescrever alterações feitas por outra transação. Além disso, outras restrições podem ser estabelecidas, como por exemplo habilitar/desabilitar leitura de mudanças temporárias realizadas por outras transações. Esta propriedade é assegurada pelo componente chamado de Controle de Concorrência. Este é justamente o caso descrito no enunciado. Como a transação T2 escreve na conta B após a leitura dessa conta pela transação T1, quanto T1 for escrever na conta B (linha 6), a escrita feita por T2 será simplesmente sobrescrita. Portanto, é a alternativa E que o candidato deve marcar;
- **durabilidade:** ela assegura que qualquer alteração resultante de uma transação será permanente. E isso deve ser verdade mesmo em caso de falha de sistema após a conclusão da transação. Imagine que uma transferência bancária seja feita em um caixa eletrônico e que 30 segundos depois da sua conclusão o equipamento falhe. É a propriedade de durabilidade que garante que a tal transferência permanece válida na sua conta.

Já a alternativa B nos traz a propriedade de distributividade, que não tem um relacionamento direto com transações de banco de dados. Essa propriedade está relacionada a operações algébricas (da Matemática ou Relacional). Um exemplo bem simples de distributividade é visto entre operações de soma: $(3+6)+5 = 3+(6+5) = 14$. Ou seja, como a ordem entre as execuções dessas operações não interfere no resultado, se diz que esse tipo de operação respeita a propriedade de distributividade.

2. **Assuntos relacionados:** *Banco de Dados, Modelo Entidade-Relacionamento, Modelo Relacional, Projeto Lógico de Banco de Dados,*

Banca: *Cesgranrio*

Instituição: *Petrobras*

Cargo: *Analista de Sistemas Pleno - Processos*

Ano: *2006*

Questão: *23*

Considere o modelo entidade-relacionamento representado abaixo.



Na transformação deste modelo conceitual Entidade-Relacionamento em um modelo lógico relacional, as cardinalidades do relacionamento entre as entidades exercem papel importante. Dado que se deseja gerar um modelo relacional que atenda à terceira forma normal, pode-se afirmar que sempre darão origem a uma tabela para cada uma das entidades relacionadas os relacionamentos do tipo:

- $(0,n) \times (0,n)$, podendo ou não gerar uma tabela para o relacionamento.
- $(0,1) \times (0,n)$, podendo ou não gerar uma tabela para o relacionamento.
- $(0,1) \times (1,1)$, gerando uma tabela para o relacionamento.
- $(1,n) \times (1,n)$, podendo ou não gerar uma tabela para o relacionamento.
- $(1,1) \times (1,n)$, devendo gerar uma tabela para o relacionamento.

Solução:

Uma entidade corresponde à representação de todo e qualquer substantivo, concreto ou abstrato, sobre o qual precisa-se armazenar ou recuperar informações. No modelo Entidade-Relacionamento é representado por um retângulo.

Já o relacionamento é a forma como os objetos que compõem a realidade se relacionam. É representado por um losango, mas há um conceito importante a ser entendido que é a cardinalidade do relacionamento. Consiste de números cardinais colocados ao lado do nome do relacionamento e dimensiona o número de ocorrências de uma entidade que pode estar envolvido em um relacionamento, sendo útil para extrair daí regras de consistência e integridade dos dados.

Existem 3 (três) tipos básicos de relacionamento entre as entidades de acordo com a cardinalidade:

- Um-para-um (1:1): representa que uma única ocorrência de uma entidade pode se relacionar com apenas uma única ocorrência de outra entidade;
- um-para-muitos (1:N): representa que uma ocorrência de uma entidade pode se relacionar com muitas ocorrências de outra entidade. No entanto, a recíproca não é verdadeira;
- muitos-para-muitos (N:M): representa que várias ocorrências de uma entidade pode se relacionar com muitas ocorrências de outra entidade.

Há ainda os relacionamentos recursivos onde uma entidade se relaciona com si própria e são relacionamentos mais raros.

Para que o modelo E-R pudesse representar melhor os conceitos foi observado que cardinalidades genéricas do tipo 1:N (um-para-muitos) e N:M (muitos-para-muitos) não são suficientes. Isto porque o conceito de “muitos” é um conceito vago, podendo ser um ou qualquer número acima de um, existindo, ainda, o valor zero, pois, em alguns casos, nem todas as ocorrências das entidades participam do relacionamento. Para que isso seja resolvido, o modelo E-R propõe que seja utilizado o conceito de *Cardinalidade Mínima* e de *Cardinalidade Máxima*.

Para um melhor entendimento das Cardinalidades Mínimas e Máximas, vamos analisar os tipos de relacionamento de cada uma das alternativas aplicadas ao problema do enunciado. Na letra (A), a cardinalidade do tipo (0,n) x (0,n) quer dizer o seguinte: uma pessoa pode lavar um, vários carros ou nenhum e um carro pode não ser lavado ou ser lavado por uma ou mais pessoas. Já na letra (B), uma pessoa pode lavar nenhum, um ou vários carros, mas um carro pode não ser lavado ou ser lavado por, no máximo, uma pessoa. A letra (C) indica que um carro deve ser lavado por exatamente uma pessoa, mas que uma pessoa pode não ter lavado nenhum carro. Na letra (D), uma pessoa deve ter lavado, no mínimo, um carro e todo carro deve ter sido lavado por uma ou mais pessoas. E, para finalizar, na letra (E): um carro deve ter sido lavado por exatamente uma pessoa e uma pessoa deve ter lavado um ou mais carros.

Ao passarmos um relacionamento para o modelo relacional, temos três opções:

1. Entidades relacionadas podem ser fundidas em uma única tabela;
2. tabelas podem ser criadas para o relacionamento;
3. chaves estrangeiras podem ser criadas em tabelas a fim de representar adequadamente o relacionamento.

Na letra (C), que é o único caso de mapeamento um-para-um, a melhor alternativa para atender a terceira forma normal é incluir a chave primária de Pessoa como chave estrangeira na tabela Carro (opção 3). Sendo assim, todo registro da tabela Carro terá representado uma pessoa, que será a pessoa que lava o carro. Caso o relacionamento fosse do tipo (1,1) x (1,1), uma tabela poderia representar as duas entidades sem problemas (opção 1). Neste caso, não haveria o problema de redundância indesejado para a terceira forma normal e nenhum dado seria perdido. Gerar uma tabela definitivamente não é uma opção para a letra (C), que está incorreta.

Os relacionamentos muitos-para-muitos estão representados nas letras (A) e (D). Nestes casos, a única solução possível é utilizar uma tabela para o relacionamento (opção 2). Não é possível representar esse tipo de relacionamento através de uma chave-estrangeira em uma tabela que representa uma entidade e nem mesmo fundir tabelas de entidade sem que haja problemas de redundância. As alternativas (A) e (D) estão erradas, pois a tabela “deve” ser criada para o relacionamento e não é uma opção não tê-la.

Na letra (E), a melhor alternativa é criar chave estrangeira para representar a pessoa que lavou o carro na tabela Carro (opção 3). Entretanto, ainda é necessário garantir que uma pessoa só exista na tabela Pessoa se já houver lavado um carro. Isso pode ser feito com inclusão de regra adicional de integridade. A opção 1 não é aceitável, pois representaria dados redundantes. A opção (2) também tem a possibilidade de representar um modelo relacional

que atenda o que a questão está pedindo e também deve ter adicionada a restrição de que toda pessoa da tabela Pessoa tenha um registro na tabela que representa o relacionamento “Lava”. Sem dúvida, é uma opção, mas não “deve gerar uma tabela” conforme diz o enunciado. A letra (E) está errada.

Na alternativa (B), só o fato de não ser um relacionamento muitos-para-muitos, sabemos que o uso de tabela para representar o relacionamento “Lava” não é obrigatório. Há a possibilidade de se criar uma chave estrangeira que represente uma pessoa na tabela Carro (opção 3). Neste caso, a tabela Carro deve permitir que o valor da chave estrangeira também possa ser nulo, pois um carro não necessariamente é lavado por alguma pessoa. Há controvérsias se, neste caso, a terceira forma normal é obedecida. Se optássemos por criar uma tabela para o relacionamento, não haveria problemas, pois, para garantir que um carro seja lavado por no máximo uma pessoa, basta adotar a chave primária de Carro para garantir a unicidade do mesmo. A opção 1 não é adequada para o caso, pois as informações das pessoas iriam ser redundantes na nova tabela “fundida”. Embora haja a controvérsia de a permissão do uso de valor nulo não garantir a terceira forma normal e nem mesmo a primeira, vamos adotar que o uso de chave estrangeira é factível para o problema e que o uso de tabela para o relacionamento também é possível. Sendo assim, a alternativa (B) é a alternativa correta.

3. **Assuntos relacionados:** *Protocolos de Redes, HTTP, HTTPS, Cookies, Variáveis de Sessão,*

Banca: *Cesgranrio*

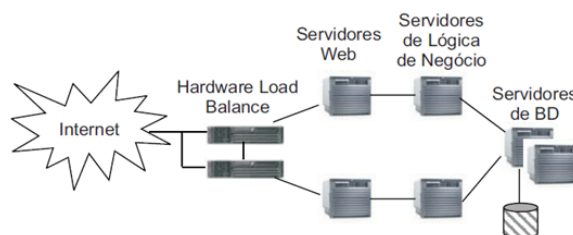
Instituição: *Petrobras*

Cargo: *Analista de Sistemas Pleno - Processos*

Ano: *2006*

Questão: *25*

A figura abaixo apresenta uma típica arquitetura utilizada para disponibilizar sites na Internet.



Sobre essa arquitetura, são feitas as afirmativas abaixo.

- I - Os protocolos HTTP e HTTPS são protocolos inerentemente com informação de estado, o que facilita o gerenciamento dos estados por parte das aplicações e dos servidores Web, permitindo o balanceamento de carga através da distribuição das chamadas entre os servidores Web.
- II - Os cookies podem ser utilizados em alguns casos como alternativa para realizar o controle de estados através do armazenamento de informações no lado do cliente Web (Browser), sendo que uma desvantagem desta abordagem é que ela adiciona tráfego extra na rede, além de ser menos segura que o gerenciamento de estado feita no servidor.
- III - Se forem utilizadas variáveis de sessão para o gerenciamento de estado em servidores que não permitam o compartilhamento das variáveis entre eles, métodos que implementam afinidade de sessões podem ser utilizados de forma a fazer com que um browser, ao se conectar com um servidor, tenha seus pedidos subsequentes sempre direcionados para o mesmo servidor. A afinidade de sessão pode prejudicar o balanceamento de carga utilizado em configurações como a apresentada na figura.

Está(ão) correta(s) a(s) afirmativa(s):

- (a). I, apenas.
- (b). II, apenas.
- (c). III, apenas.
- (d). II e III, apenas.
- (e). I, II e III.

Solução:

Abordaremos cada item a fim de solucionarmos a presente questão.

Sobre o item I:

HTTP (*Hyper Text Transfer Protocol*): trata-se de um protocolo da Camada de Aplicação

que define tanto a estrutura da mensagem como o modo que o cliente e o servidor as trocam. Podemos classificá-lo como uma aplicação cliente-servidor.

Lembre-se que uma página Web nada mais é que um conjunto de objetos, em que um objeto pode ser um arquivo HTML, uma imagem, um *applet* JAVA, etc. Cada objeto é referenciado por uma única URL, a qual possui dois componentes: o nome do hospedeiro e o nome do caminho do objeto. Por exemplo, na URL `http://www.universidade.br/departamento/figura.gif`, `www.universidade.br` é o nome do hospedeiro e `/departamento/figura.gif` é o nome do caminho do objeto.

Vejam, agora, como o HTTP realiza a comunicação entre o cliente (representado pelo Browser, que apresenta páginas Web e fornece várias características de navegação) e o servidor (por exemplo, Apache ou *Microsoft Internet Information Server* — IIS, que é responsável por abrigar objetos Web).

O HTTP usa o TCP como seu protocolo de transporte, isto é, as trocas de mensagens são realizadas orientada a conexão. Além disso, tais conexões podem ser persistentes ou não-persistentes. Abaixo, apresentamos o fluxo de uma conexão não-persistente:

1. o processo cliente abre uma conexão TCP executando o *three way handshake*;
2. o processo cliente envia uma mensagem de requisição HTTP ao servidor;
3. o processo servidor recebe a mensagem de requisição, encapsula o objeto Web requisitado em uma nova mensagem de resposta HTTP, e a envia ao cliente;
4. o processo servidor solicita o encerramento da conexão TCP;
5. o processo cliente recebe a mensagem de resposta, extrai o objeto da mensagem de resposta e o apresenta.

Podemos observar, pelo fluxo acima, que para cada objeto transferido há a exigência de abertura de uma nova conexão TCP, o que se mostra um tanto quanto ineficiente, pois as páginas Web são, geralmente, constituídas de um arquivo-base HTML e de diversos objetos referenciados. Esta ineficiência é sanada pela conexão persistente, pois o servidor “deixa” a conexão TCP aberta após enviar a resposta, ou seja, requisições e respostas subsequentes entre os mesmos cliente e servidor podem ser enviadas por meio da mesma conexão.

É importante salientar que o servidor envia as respostas HTTP ao cliente sem armazenar nenhuma informação de estado sobre este, isto é, se um determinado cliente solicita o mesmo objeto duas vezes em um período de poucos segundos, o servidor não responde dizendo que acabou de enviar o objeto. Em vez disso, ele envia novamente o objeto ao cliente. Em outras palavras, dizemos que o HTTP é um **protocolo sem estado**. Portanto, já temos condições de dizer que o item I está errado.

HTTPS: nada mais é que o uso do protocolo SSL (Secure Sockets Layer) com HTTP, por isso o nome HTTPS (**Secure HTTP**). O protocolo SSL, introduzido pela Netscape em 1995, está posicionado entre a camada de transporte e a camada de aplicação da pilha TCP/IP, provendo serviços de autenticação do servidor, comunicação secreta e integridade de dados.

Sobre o item II:

Como vimos no item I, basicamente a Web não tem estados. O navegador envia uma solicitação a um servidor e recebe de volta um arquivo. Depois, o servidor esquece que já viu esse

cliente específico. Este modelo era bem adequado no início, quando a Web era usada apenas para recuperar documentos publicamente disponíveis. Porém, à medida que a Web começou a adquirir outras funções, o modelo causou problemas. Um segundo exemplo é o e-commerce (comércio eletrônico). Se um usuário percorrer uma loja eletrônica colocando itens em seu carrinho de compras de vez em quando, como o servidor irá controlar o conteúdo do carrinho?

Para resolver esse problema, a Netscape criou uma técnica muito criticada, denominada cookies. O nome deriva de uma antiga gíria dos programadores, na qual um programa chama um procedimento e recebe de volta algo que talvez precise apresentar mais tarde para conseguir a realização de algum trabalho.

Quando um cliente solicita uma página da Web, o servidor pode fornecer informações adicionais junto com a página solicitada. Essas informações podem incluir um cookie, um pequeno arquivo ou string (com 4 KB no máximo), o que, obviamente, adiciona tráfego extra à rede. Os navegadores armazenam os cookies oferecidos em um diretório de cookies no disco rígido do cliente, o que, diga-se de passagem, é um tanto quanto vulnerável. Como estes cookies podem conter informações sobre número de cartões de crédito, números de CPF e outras informações confidenciais, é sempre possível que alguém mal-intencionado invada o computador do usuário a fim de obter tais informações. Portanto, o item II está correto.

Sobre o item III:

O problema que as variáveis de sessão têm que superar é o fato do protocolo HTTP que você usa para navegar na web ser sem estado, como já dissemos no item I.

A primeira vez que um novo usuário visita uma página, uma nova variável de sessão é automaticamente criada. Como? Por meio de cookies. Então, na primeira vez que você visita um site, não há cookie e o servidor cria uma nova sessão e “seta” um cookie com um valor único (ID). Posteriormente, quando o usuário navega pelo site, o cookie é enviado de volta ao servidor, permitindo que este reconheça o usuário. A sessão termina quando o usuário não abre outra página na aplicação por certo período de tempo, ou quando o usuário termina a sessão (clicando em *log-off*, por exemplo). Além disso, enquanto existir, a sessão é específica para um usuário individual, e cada usuário tem uma sessão separada.

Uma desvantagem da variável de sessão é que cada cliente exige afinidade com o servidor onde começou a sessão, uma vez que é somente aí que estão as variáveis de sessão (neste servidor local). Isto requer aderência de sessão, o que elimina a redundância *on-the-fly* (caso o servidor caia ou seja desligado, as sessões dos usuários serão destruídas), e prejudica o balanceamento dinâmico de carga mesmo em momentos em que há servidores ociosos (isto faz os usuários experimentarem situações irregulares, onde alguns experimentam um site extremamente lento enquanto outros navegam com rapidez e facilidade). Assim, concluímos que o item III também está correto.

Portanto, a alternativa **D** está correta.

Questao	Resposta
1	E
2	B
3	D
4	B
5	D
6	A
7	E
8	C
9	C
10	E
11	D
12	A
13	E
14	D
15	A
16	B
17	E
18	E
19	A
20	C
21	C
22	A
23	C
24	E
25	C
26	B
27	B
28	E
29	A
30	D
31	A
32	B
33	B
34	C
35	C
36	D
37	C
38	A
39	E
40	E
41	B
42	C
43	B
44	B
45	D
46	D
47	A
48	C
49	E
50	A

Índice Remissivo

- Árvore Binária de Busca, 73
- Árvores Binárias, 73
- Índice de Performance de Processo, 135

- Acessibilidade de Aplicações Web, 105
- ACID, 4
- Algoritmos, 80, 90
- Algoritmos de Criptografia, 35
- Algoritmos de Ordenação, 82
- Análise Combinatória, 27, 31
- Análise de Ponto de Teste, 76
- Análise de Valor Agregado, 12
- Análise de Viabilidade de Projetos, 25
- Análise e Especificação de Requisitos, 68, 70
- Arranjo com Repetição, 27
- Arranjo Simples, 27
- ATOM, 102
- Atomicidade em Transação, 4

- Backup, 35
- Balanced Scorecard, 45
- Banco de Dados, 4, 6, 58, 60, 115
- BPM, 133, 137, 139
- BPMN, 139
- BPSS, 139
- BS 7799, 113
- BSC, 45
- Bubble Sort, 82
- Business Process Modeling, 133

- Caminho Crítico, 21
- Classificação ABCD, 130
- Combinação com Repetição, 27
- Combinação Simples, 27
- Complexidade de Algoritmos, 80, 90
- Consistência em Transação, 4
- Cookies, 9
- CPM, 21
- Criptografia Simétrica, 35
- CSS, 102

- DAS, 92
- Data Warehouse, 60, 98, 115, 119
- Desenvolvimento Web, 14, 105
- Diagramas UML, 40
- DSDM, 92
- DTD, 14
- Durabilidade em Transação, 4

- Engenharia de Requisitos, 121
- Engenharia de Software, 70, 76, 84, 92, 121, 125
- Erlang, 139
- ERP, 54, 130
- Estruturas de Dados, 73
- EVA, 12

- Família ISO 27000, 111
- FDD, 92
- Fechamento de Projetos, 108
- Firewall, 35

- Gerência de Projetos, 21, 108
- Gerenciamento de Projetos, 12, 23, 25
- Gerenciamento de Requisitos, 70
- Gerenciamento de Riscos, 23
- Gerenciamento Financeiro de Projetos, 12
- GERT, 21

- HTML, 14
- HTTP, 9
- HTTPS, 9, 102

- Implantação da Função de Qualidade (IFQ), 68
- Indicador de Desempenho, 43
- ISO 27001, 111
- ISO 27002, 110
- ISO/IEC 17799-2005, 113
- ISO/IEC 27002, 113
- Isolamento entre Transações, 4

- Lógica, 33
- Lambda-Calculus, 139

- Método de Ordenação da Bolha, 82
- Métrica de Performance de Processo, 135
- Matriz de Probabilidade e Impacto de Riscos, 23

- Modelagem de Dados, 60
- Modelagem de Negócio, 63
- Modelagem de Processos de Negócio, 133
- Modelagem Multidimensional, 58, 60, 63, 98, 119
- Modelo Ágil de Software, 92
- Modelo Entidade-Relacionamento, 6
- Modelo Estrela, 60, 98
- Modelo Relacional, 6, 58

- Normas de Segurança da Informação, 111

- OCL - Object Constraint Language, 40
- OLAP, 115

Padrões W3C, 105
Permutação Circular, 27
Permutação com Repetição, 27, 31
Permutação Simples, 27
PERT, 21
Pi-Calculus, 139
PMBOK, 12
Política de Segurança da Informação, 38, 113
Processo Unificado, 65
Programação, 82
Projeto Lógico de Banco de Dados, 6
Protocolos de Redes, 9

Qualidade Total, 43

Raciocínio Lógico, 31, 33
Rede de Petri, 51
Requisito de Software, 121
Requisitos Esperados, 68
Requisitos Excitantes, 68
Requisitos Normais, 68
RSS, 102
RUP, 65

Segurança da Informação, 35, 38, 110, 111, 113
Sistemas de Informações Gerenciais, 60
SOA, 128

Técnica de Avaliação, 21
Termo de Confidencialidade, 110
Teste de Usabilidade, 125
Testes de Software, 84
TPA, 76

UML, 40, 87

Valor Presente Líquido, 25
Variáveis de Sessão, 9
VPL, 25

Web 2.0, 102
Workflow, 49, 137
Workflow Management System (WFMS), 49

XML, 14
XP, 92
XSLT, 14