Passar a BD – 2021/2022

Autores: José Cutileiro e Guilherme Pascoal

Aulas

Exercicios (notas adicionais)

Semana 1 Hora 1: Apresentação

Nada de jeito aqui

Semana 1 Hora 2: Intrudução SGBD

Definições importantes

Dados: Factos de algo (ainda não interpretados)

Informação: Resultado de interpertar os dados

Conhecimento: Combinar informação com experiência

Podemos perder: Dados e conhecimento

Não podemos perder: Informação (se os dados não forem perdidos)

Sistemas de informação

Adquirir, guardar e processar dados em informação para criar valor para algo ou alguém

Dados, Aplicações, Processos e Pessoas (muito mais que dados)

DBMS: Pacote de software para aceder, manipular e alterar dados mais facilmente (e com mais consitência)

Como são guardados os dados lá dentro?

Tabelas

Tabelas: Coluna, linhas, (células)

Propriedades:

1. Ordem das colunas/linhas não adiciona informação
2. Células apenas teem valores atómicos
3. PK (a ver mais à frente)

Relação entre tabelas: FK (a ver mais à frente)

Base de dados: Organizar uma coleção de dados relacionados entre si (facila acessos, alterações …)

Esquema da BD:

1. Coleção de tabelas individuais
2. Relação entre tabelas
3. Ics

Cenas fixes das DBMS:

1. Intelligent I/O
2. Cache
3. Otimização de queries
4. Concurrência
5. Undo e sistemas para recuperar dados

Semana 3 Hora 4: Normalização Parte 1

Redundância: Dados repetidos na base de dados

account_number	balance	customer_name	customer_city	branch_name	branch_city
A-101	500.00	Johnson	Palo Alto	Downtown	Brooklyn
A-215	700.00	Smith	Rye	Mianus	Horseneck
A-102	400.00	Hayes	Harrison	Perryridge	Horseneck
A-101	500.00	Hayes	Harrison	Downtown	Brooklyn
A-305	350.00	Turner	Stamford	Round Hill	Horseneck
A-201	900.00	Johnson	Palo Alto	Perryridge	Horseneck
A-217	750.00	Jones	Harrison	Brighton	Brooklyn
A-222	700.00	Lindsay	Pittsfield	Redwood	Palo Alto
A-333	850.00	Majeris	Rye	Central	Rye
A-444	625.00	Smith	Rye	North Town	Rye

Nota: A Redundância pode aparecer desde logo no EA

Anomalias:

1. Inserção: Ex: Para adicionar uma conta também tenho que inserir uma cidade
2. Atualização: Ex: Atualizar um dos registos afeta multiplos registos
3. Eliminação: Ex: Apagar uma das contas também remove a cidade em si
4. Querie: O processo de interrugação à BD fica mais lento sem utilidade acrescentada

É suposto haver uma independência entre os dados, factos independentes devem-se manter independentes mesmo na tabela (record aliasing)

De onde surgem estas anomalias?

Surgem pois o esquema da BD não está normalizado

Teoria da normalização

<u>Objetivos:</u>

1. Reduzir redundância
2. Independência de factos é mantida
3. Interrugações à BD fáceis e rápidas

Dependência Funcional

Y é funcionalmente dependente de X se para cada r[X] temos um e apenas um r[y]

Propriedades (x->y):

1. Reflexivity: Se Y ⊆ X, then X→ Y
2. Augmentation: Se X → Y, then X Z → YZ, para qualquer Z
3. Transitivity: Se X → Y e Y→ Z, então X→ Z
4. Decomposição: Se X → Y Z, then X → Y e X → Z
5. Composition: Se X →Y and A → B then XA → YB
6. Union: Se X→ Y and X → Z then X → YZ

Fecho de atributos (a+)

a é super chave se o seu fecho inclui TODOS os atributos

exemplo:

Esquema: r(A, B, C, G, H, I)

{A → B , A → C , CG → H , CG → I , B → H}

(AG)+ {AG é super chave}

1. Com A temos B
2. Com A temos C
3. Com CG temos H
4. Com CG temos I
5. Já temos tudo 😊

Não confundir CHAVES

Super chave: Identifica a relação (pode ter mais atributos do que necessário)

Chave candidata: Identifica a relação (tem o número minimo de atrivutos)

Chave primária: Uma das chaves candidatas

Semana 3 Hora 5: Normalização Parte 2

Formas normais

Usadas para detetar redundância e coerência

Primeira forma normal

Está na primeira forma normal se todas as células tiverem apenas um valor atribuido

Lisboa	01-02-92 01-02-93

<não

sim>

Lisboa	01-02-92
Lisboa	01-01-93

Problemas: Existem anomalias de inserção, atualização e remoção

Eliminar anomalias: Separar em várias relações

Segunda forma normal

Todos os atributos não chave dependem COMPLETAMENTE de atributos chave. Nota: Todas as relações que estão em 1NF que tem apenas uma chave estão também na 2NF

Ex:

R(A,B,C,D) e A->D: **Não** está na 2NF

Como resolver?

R1(A,B,C) e R2(A,D): Esta já está na 2NF

Problemas: Não garante ausência de redundância

Terceira forma normal

Todos os atributos não chave são independentes entre si

Ex:

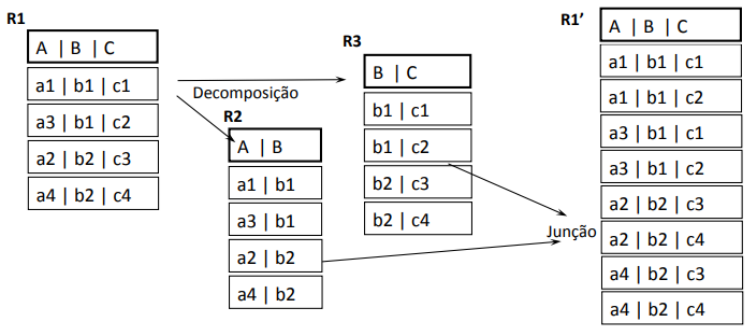
R(A,B,C) e B->C: Não está na 3NF

Como resolver?

R1(B,C) e R2(A,B): Já está na 3NF

Problemas: Surgem por exemplo em R(A,B,C,D) e C->A, Como resolver sem haver perda de info (Lossy Decomposition)?

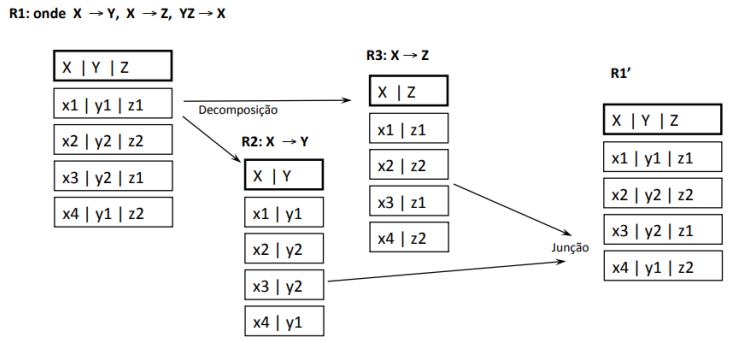
Lossy Decomposition



Repara que houve perda de informação R1 é diferente de R1`

Garantir que não perdemos informação:

Obtemos quando fazemos NATURAL JOIN de relações que forma decompostas



Teorema de Heath:

r(XYZ) é decomposto em r1(XY) e r2(XZ) é lossless se X->Y ou X->Z

Exemplo de decomposição: R(A,B,C,D,E,F,G) e SD->P,J->S,JP->C

R1(S,D,P)

R2(J,S)

R3(C,J,D,Q,V)

Forma normal de Boyce-Codd

Todos os atributos são dependentes de uma chave candidata

Uma FNBC não tem redundâncias

Dada uma relacao que não está em FNBC é sempre possivel decompo-la de modo a que fique deste modo

Semana 4 Hora 3: Views

Vista : É tipo uma tabela virtual, definida por uma query

Notação: CREATE VIEW vista AS SELECT …

DROP VIEW vista

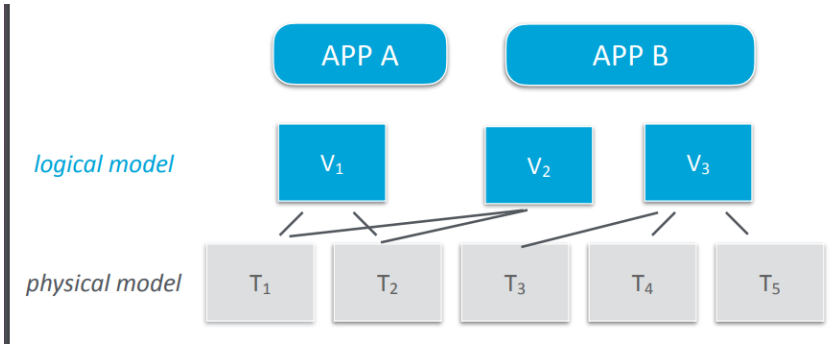
Uma vista pode ser queried (tal como uma tabela normal)

SELECT *

FROM vista;

Independência lógica

Criar vistas separa a lógica do modelo do modelo físico



Cada app cria/consegue a sua forma de acessar à BD, deste modo restringimos o acesso (+segurança).

Podemos limitar as Apps a certas views (+segurança)

Controlo de users

As BDs suportam um sistema de utilizadores

Notação: CREATE USER 'scott' WITH PASSWORD 'tiger'

DROP USER [IF EXISTS] 'scott'

Dar acesso:

GRANT <privilégios> | ALL

ON <obj_name>

TO <user_or_role>

Tirar acesso:

(em vez de GRANT usar REVOKE)

<privilégios>: SELECT, INSERT, DELETE

<obj_name>: Vista

<user_or_role>: Utilizador/Role na BD

Updatable Views

Podemos alterar as linhas

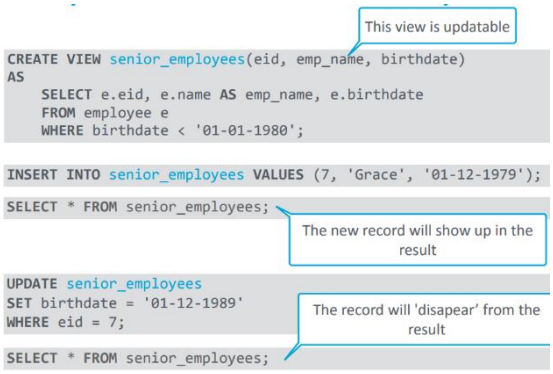
Podemos alterar as colunas, se:

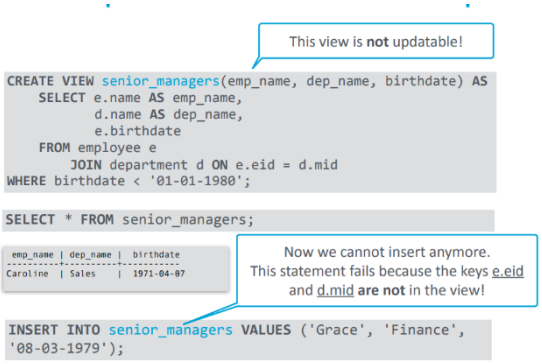
1: Obtemos diretamente da BD

2: Temos a PK incluida na vista

Insertable Views

Podemos adicionar novas linhas





Vistas não atualizáveis:

Se tiverem

1. Aggregate queries
2. Joins
3. Project out attributes~

Vistas materializáveis (postgres):

Para vistas mais complexas

Notação: CREATE MATERIALIZED VIEW vista_mat AS SELECT …

Vista normal: não ocupa espaço físico

Vista materializada: Ocupa

Vista materializada: Pode ser guardade numa outra DB

Vista materializada: Serve para replicar ou guardar abuso de dados (data warehouse)

Semana 4 Hora 5: Transações

Notas:

commit ou abort (apenas um deles)

ex SQL:

```
start transaction;
update account set balance = balance – 350 where     account_number = 'A-101';
update account set balance = balance + 350 where     account_number = 'A-102';
commit;
```

Propriedades **ACID**:

Atomicidade

Concorrência: Garante integridade

Isolamento: Garante serializabilidade

Durabilidade

Modelos de storage:

Volatil: Não sobrevive ao crash e boot

Ñ Volatil: Sobrevive a crashes

Estável: Sobrevive a tudo

Transações SQL:

Iniciar >>> start transaction;

Terminar >>> Commit -> Resultados permanentes

RollBack -> Desfaz tudo (aborta)

Niveis de isolamento:

Phantom read: O número de registos pode alterar a meio da transação(caso outra faça commit)

Non-repeatable reads: Os dados em si podem alterar (caso outra faça commit)

Dirty reads: Pode ler dados que não foram commited

Nível de isolamento	*dirty reads*	*non-repeatable reads*	*phantom reads*
SERIALIZABLE	não	não	não
REPEATABLE READ	não	não	possível
READ COMMITTED	não	possível	possível
READ UNCOMMITTED	possível	possível	possível

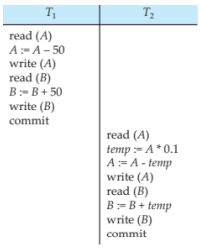
Semana 6 Hora 2: Transações – Concorrência

Componentes SGBD

Recovery manager: Amoticidade e Durabilidade

Transaction e Concurrency manager: Isolamento e Coerência

Concurrência solução óbvia:



Locks por tudo o que é transação (em série)

Problema: Não existe concurrência verdadeiramente
(mais lentidão)

Escalanomento:

O escalomento é a ordem pela qual as instruções são executadas

1. Em série: Separar as transações
2. Equivalente: Mesmo resultado que (1) mas com a ordem das intruções alternadas (caso necessário).
3. Não preserva: Pode não ter os memos resultados que (1)

Serializabilidade:

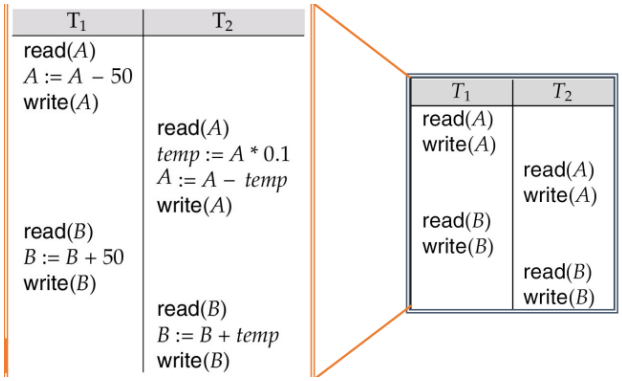
É serializável se é equivalente ao escalonamento em série 😊

Exercicio: 3 transações, quantos escalonamentos **em série** possiveis?

Resposta: Nenhum + Apenas_1 + Grupos_de_2 + Grupos_de_3

1 + 3 + 4 + 6 = 14

Transações (ver facilmente):



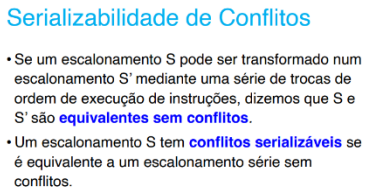
Conflitos:

- Existe conflito se T1 e T2 tiverem uma certa ordem de execução

Exemplos:

1. Leitura com Leitura: Não gera conflito
2. Leitura com escrita: Gera conflitos
3. Escrita com escrita: Gera conflitos – Afeta a próxima instrução read

Que porra é esta?



Conflito serializável: Existe uma ordem nas operações que fica equivalente a um escalonamento em série sem conflitos

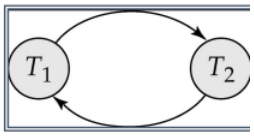
Conflito não serializável: (contrário do de cima)

É serializável?

1. Grafo de precedências
   a. Nós: Transações
   b. Arcos: Conflitos
2. Reconhecer arcos (T1 para T2)?
   a. T1: write antes de T2: Read
   b. T1: read antes de T2: Write
   c. T1: writes antes de T2: Write

3. Se não tiver ciclos é serializável

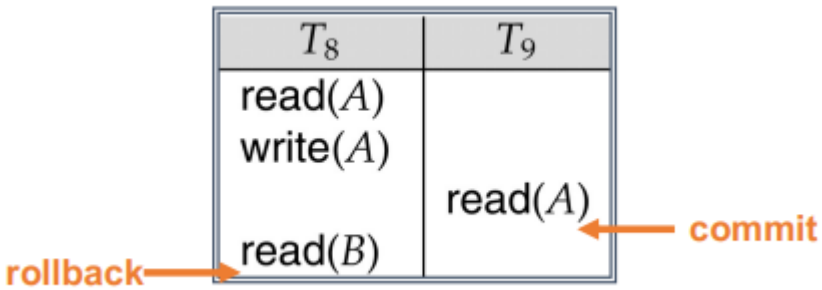
T_1	T_2
read(A)	
$A := A - 50$	
	read(A)
	$temp := A * 0.1$
	$A := A - temp$
	write(A)
	read(B)
write(A)	
read(B)	
$B := B + 50$	
write(B)	
	$B := B + temp$
	write(B)



Relembra: Transação teve sucesso -> commit

Escalonamento recuperável

Exemplo de escalonamento não recuperável:



Explicação: Alterámos o valor de A (T8) depois lemos o valor de A (T9) e dá commit dado que a transação foi concluída com sucesso. T8 falha dando rollback, A volta ao valor original. O commit de 9 está inconsictente com os dados atuais.

Rollbacks em cadeia:

Várias transações brincam com os mesmos registos (uncommited) em simultâneo, uma delas dá ROLLBACK, as outras devem abortar (ROLLBACK).

Recuperável e sem ROLLBACK em cadeia:

Tk lê dados escritos por Ti então Ti faz commit antes de Tk

Problemas com Atomicidade:

Não é funcional para transações longas e com grande probabilidade de falhar

Para existem existem estas extensões:

1. Savepoints
2. Chained transactions
3. Nested Transactions
4. …

SavePoints:

Gravamos um ponto que reconhecemos os valores (S1) e o próximos ROLLBACK vai para esse valor.

Controlo de concorrência

Sincronização com locks

1. Leitura (partilhados): P
2. Escrita (exclusivos): E

Matriz de Compatibilidade

	E	P	--
E	N	N	S
P	N	S	S
--	S	S	S

Protocolo (semelhante a SO)

Para ler devemos ter um lock partilhado

Para escrever/alterar devemos ter lock exclusivo

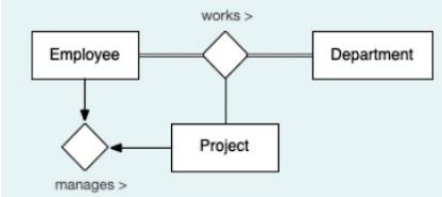
Nota: Locks são libertados no commit

Em falta: Two-Phase-Locking, Strict Two-Phase-Locking, Conservative Two-Phase-Locking, Handing Deadlock, Outros protocolos de gestão da concorrência, Políticas de TimeStamp

Semana 6 Hora 3: Transações – Recuperação

Exercicios

Teste modelo (2021/2022)

1. X tem Y habitantes e uma poluição anual de Z; X é a capital de W
    a. Relativamente a este esquema relacional:
        i. Todas as relações devem estar na FNBC para assegurar que a inserção ou remoção de dados é feita de modo independente
2. 

    a. Relativamente a este diagrama
        i. Departamentos têm empregados a trabalhar neles e um projeto tem no máximo um empregado que o gere
3. Trip

person	country
Mary	Portugal
Bill	Portugal
John	France
Mary	France
Bill	France
John	Australia
Bill	Australia

Country

name
Portugal
France
Australia

```
SELECT DISTINCT person
FROM trip t1
WHERE NOT EXISTS(
  SELECT name
  FROM country
    EXCEPT
  SELECT country
  FROM trip t2
  WHERE t1.person = t2.person)
```

Resultado da consulta: Bill

Explicação: Quais as pessoas que se eu retirar à tabela países todos os países que eles visitaram fica a tabela vazia?