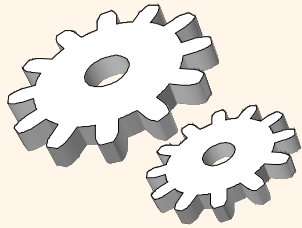
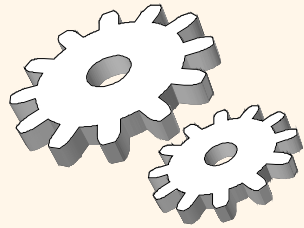


Projeto Físico de Banco de Dados

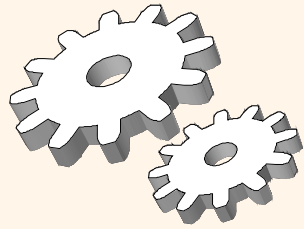
Capítulo 20

Visão Geral



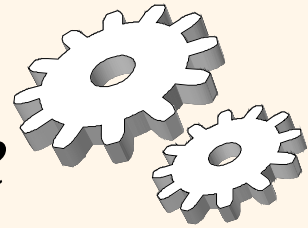
- Depois do projeto ER, refinamento do esquema e a definição de visões, nós temos os esquemas *conceitual* e *externo* para nosso banco de dados.
- O próximo passo é escolher índices, fazer decisões de agrupamento, e (se necessário) refinar os esquemas conceitual e externo para atingir metas de desempenho.
- Precisamos começar pela compreensão da *carga de trabalho*:
  - As consultas mais importantes e quão freqüentemente elas surgem.
  - As atualizações mais importantes e quão freqüentemente elas surgem.
  - O desempenho desejado para estas consultas e atualizações.

Decisões a Tomar



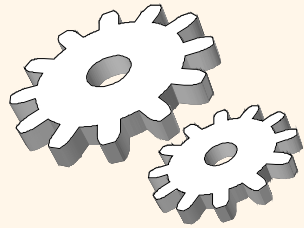
- Que índices devemos criar?
  - Quais relações devem ter índices? Qual(is) campo(s) devem estar na chave de pesquisa? Devemos construir vários índices?
- Para cada índice, que tipo de índice deve ser utilizado?
  - Agrupado? Hash/árvore?
- Devemos fazer mudanças no esquema conceitual?
  - Considerar esquemas normalizados alternativos? (Lembre-se, há várias escolhas na decomposição em BCNF etc.)
  - Devemos "desfazer" alguns passos da decomposição e ficar com uma forma normal mais baixa? (*Desnormalização.*)
  - Partição horizontal, replicação, visões...

Considerações para Seleção de Índice



- É necessária a criação do índice?
- Escolha da chave de busca.
- Utilizar múltiplos atributos na chave de busca.
- É necessário o agrupamento?
- Devo usar hash ou árvore?
- Balanceamento do custo de manutenção do índice.

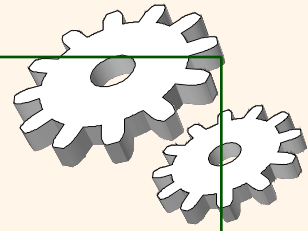
Seleção de Índices para Junções



- Ao considerar uma condição de junção:
  - Índice hash na mais interna é muito bom para Laços Aninhados com Índice (LAI).
    - Deve ser agrupado se a coluna de junção não é chave da mais interna e as tuplas internas precisam ser lidas.
  - Árvore B+ agrupada na(s) coluna(s) de junção é boa para Ordenação-Intercalação.

Exemplo 1

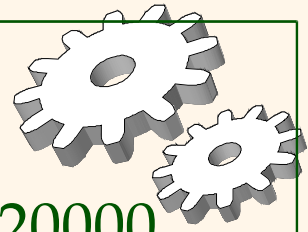
```
SELECT  E.ename, D.mgr
FROM  Emp E, Dept D
WHERE  D.dname='Toy' AND E.dno=D.dno
```



- Índice hash em *D.dname* suporta a seleção 'Toy'
  - Portanto, índice em D.dno não é necessário.
- Índice hash em *E.dno* permite obter as tuplas casadas de Emp (mais interna) para cada tupla de Dept (mais externa) selecionada.
- E se o WHERE incluísse: `` ... AND E.age=25'' ?
  - Poderia ler as tuplas de Emp usando um índice em *E.age* e então juntar com as tuplas de Dept satisfazendo a seleção por *dname*. Comparável à estratégia que usou o índice em *E.dno*.
  - Portanto, se um índice em *E.age* já está criado, esta consulta provê muito menos motivação para adicionar um índice em *E.dno*.

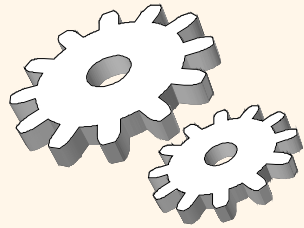
Exemplo 2

```
SELECT  E.ename, D.mgr
FROM  Emp E, Dept D
WHERE  E.sal BETWEEN 10000 AND 20000
  AND E.hobby='Stamps' AND E.dno=D.dno
```



- Claramente, Emp deveria ser a relação mais externa.
  - Sugere que devemos construir um índice hash em *D.dno.*
- Qual índice devemos construir em Emp?
  - Árvore B+ em *E.sal* poderia ser usada, OU um índice em *E.hobby* poderia ser usado. Somente um destes é necessário, e qual é melhor depende da seletividade das condições.
    - Como regra geral, seleções de igualdade são mais seletivas do que seleções de faixa.
- Como ambos os exemplos indicam, nossa escolha de índices é guiada pelo(s) plano(s) que esperamos que o otimizador considere para uma consulta. *Tem-se que entender os otimizadores!*

Agrupamento e Junções

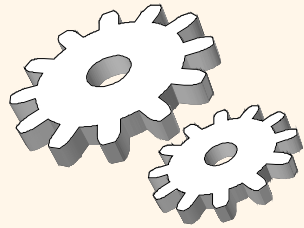


```
SELECT  E.ename, D.mgr
FROM  Emp E, Dept D
WHERE  D.dname='Toy' AND E.dno=D.dno
```

- Agrupamento é especialmente importante quando o acesso de tuplas mais internas em LAI.
  - O índice em *E.dno* deve ser agrupado.
- Suponha que a cláusula WHERE fosse:

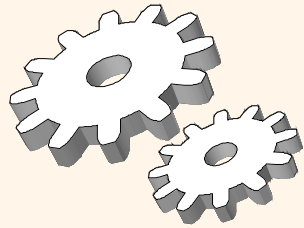
  WHERE E.hobby='Stamps' AND E.dno=D.dno
  - Se vários empregados colecionam selos, vale a pena considerar uma junção por Ordenação-Intercalação. Um índice *agrupado* em D.dno ajudaria.
- *Sumário*: Agrupamento é útil sempre que várias tuplas precisam ser obtidas.

Ajustando o Esquema Conceitual



- A escolha do esquema conceitual deveria ser guiada pela carga de trabalho, além de considerações sobre redundância:
  - Podemos ficar com um esquema 3NF no lugar de BCNF.
  - A carga de trabalho pode influenciar na escolha que fazemos na decomposição de uma relação em 3NF ou BCNF.
  - Podemos decompor ainda mais um esquema BCNF!
  - Podemos *desnormalizá-lo* (i.e., desfazer uma etapa de decomposição) ou podemos adicionar campos a uma relação.
  - Podemos considerar *decomposições horizontais.*
- Se tais escolhas são feitas depois de um banco de dados estar em uso, são chamadas de *evolução de esquema*; algumas destas mudanças podem ser mascaradas para as aplicações através da definição de *visões*.

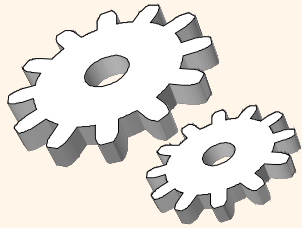
Esquemas de Exemplo



Contracts (Cid, Sid, Jid, Did, Pid, Qty, Val)
Depts (Did, Budget, Report)
Suppliers (Sid, Address)
Parts (Pid, Cost)
Projects (Jid, Mgr)

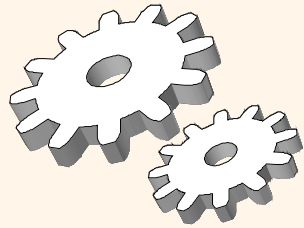
- Iremos nos concentrar em Contracts, denotada por CSJDPQV. As seguintes RIs são válidas:
  - Um projeto compra um dado item usando um único contrato; logo, não pode haver dois contratos distintos nos quais o mesmo projeto compra o mesmo item (JP $\rightarrow$ C);
  - Um departamento compra no máximo um item de um dado fornecedor (SD $\rightarrow$ P);
  - C é a chave primária.
- Quais as chaves candidatas para CSJDPQV?
- Este esquema está em qual forma normal?

Escolha de 3NF vs BCNF



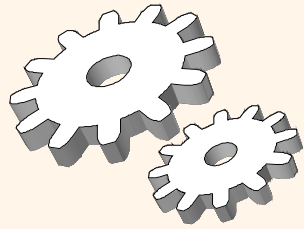
- CSJDPQV pode ser decomposta em SDP e CSJDQV e ambas as relações estão em BCNF. (Qual DF sugere que o façamos?)
  - Decomposição sem perdas, mas não preservadora de dependências.
  - Adicionando CJP torna-a preservadora de dependências também.
- Suponha que esta consulta seja muito importante:
  - *Encontrar o número de cópias Q do item P ordenado por contrato C.*
  - Requer uma junção no esquema decomposto, mas pode ser respondida por uma varredura na relação CSJDPQV original.
  - Poderia levar-nos a ficar com o esquema CSJDPQV na 3NF.

Desnormalização



- Suponha que a seguinte consulta seja importante:
  - *O valor de um contrato é menor do que o orçamento do departamento?*
- Para acelerar esta consulta, poderíamos adicionar um campo *orçamento* B a Contracts.
  - Isto introduz a DF D $\rightarrow$ B em relação a Contracts.
  - Portanto, Contracts não está mais na 3NF.
- Poderíamos escolher modificar Contracts se a consulta é suficientemente importante e não podemos obter desempenho adequado de outra forma (i.e., pela adição de índices ou pela escolha de um esquema 3NF alternativo).

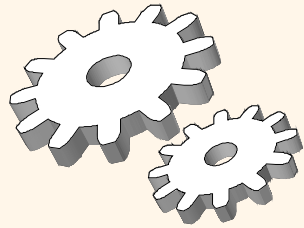
Escolha de Decomposições



- Há duas formas de decompor CSJDPQV em BCNF:
  - SDP e CSJDQV; junção sem perda mas não preserv. de dep.
  - SDP, CSJDQV e CJP; é também preserv. de dep.
- A diferença entre elas é somente o custo de garantir a DF JP $\rightarrow$ C.
  - 2ª decomposição: Índice em JP na relação CJP.
  - 1ª:

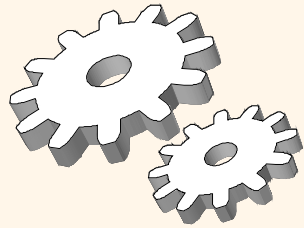
```
CREATE ASSERTION CheckDep
CHECK ( NOT EXISTS ( SELECT *
FROM PartInfo P, ContractInfo C
WHERE P.sid=C.sid AND P.did=C.did
GROUP BY C.jid, P.pid
HAVING COUNT (C.cid) > 1 ))
```

Escolha de Decomposições (Cont.)



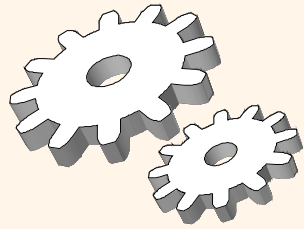
- As seguintes RIs são válidas:
  JP $\rightarrow$ C, SD $\rightarrow$ P, C é a chave primária.
- Suponha que, além disso, um determinado fornecedor sempre cobre o mesmo preço para um certo item: SPQ $\rightarrow$ V.
- Se nós decidirmos que queremos decompor CSJDPQV na BCNF, agora temos uma terceira escolha:
  - Começar decompondo-a em SPQV e CSJDPQ.
  - Então, decompor CSJDPQ (não está na 3NF) em SDP, CSJDQ.
  - Isto nos dá a decomposição junção-sem-perda: SPQV, SDP, CSJDQ.
  - Para preservar JP $\rightarrow$ C, podemos adicionar CJP, como antes.
- Escolha: { SPQV, SDP, CSJDQ } ou { SDP, CSJDQV } ?

Decomposição de uma Relação BCNF



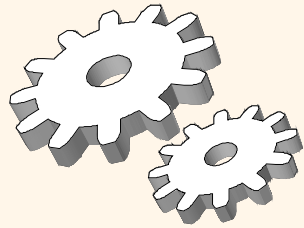
- Suponha que escolhemos { SDP, CSJDQV }. Está na BCNF e não há razão para decompor ainda mais (assumindo que todas as RIs conhecidas são DFs).
- Entretanto, suponha que estas consultas sejam importantes:
  - *Encontrar os contratos do fornecedor S.*
  - *Encontrar os contratos em que o departamento D está envolvido.*
- Decompondo ainda mais CSJDQV em CS, CD e CJQV poderia acelerar estas consultas. (Por quê?)
- Por outro lado, a seguinte consulta é mais lenta:
  - *Encontrar o valor total de todos os contratos do fornecedor S.*

Decomposições Horizontais



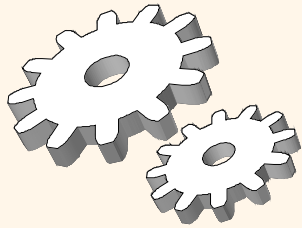
- Nossa definição de decomposição: a relação é substituída por uma coleção de relações que são *projeções*. Caso mais importante.
- Às vezes, poderíamos querer trocar a relação por uma coleção de relações que são *seleções*.
  - Cada nova relação tem o mesmo esquema do original, mas é um subconjunto de linhas.
  - Coletivamente, novas relações contêm todas as linhas da original. Tipicamente, as novas relações são disjuntas.

Decomposições Horizontais (Cont.)



- Suponha que contratos com valor > 10000 são sujeitos a regras diferentes. Isto significa que consultas em Contracts geralmente irão conter a condição *val>10000*.
- Uma forma de lidar com isto é construir um índice agrupado de árvore B+ no campo *val* de Contracts.
- Um segundo enfoque é trocar Contracts por duas novas relações: LargeContracts e SmallContracts, com os mesmos atributos (CSJDPQV).
  - Executa como índice nestas consultas, mas sem sobrecarga de gerenciamento de índice.
  - Além disso, pode-se construir índices agrupados em outros atributos!

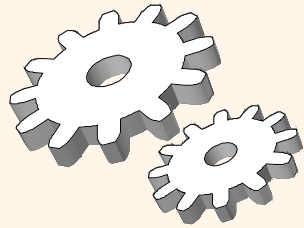
Mascarando Mudanças no Esquema Conceitual



```
CREATE VIEW  Contracts(cid, sid, jid, did, pid, qty, val)
        AS  SELECT  *
        FROM  LargeContracts
        UNION
        SELECT  *
        FROM  SmallContracts
```

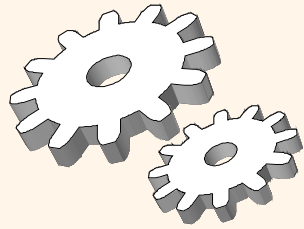
- A troca de Contracts por LargeContracts e SmallContracts pode ser mascarada pela visão.
- Entretanto, consultas com a condição *val>10000* devem ser feitas em LargeContracts para uma execução eficiente: portanto usuários preocupados com desempenho precisam estar cientes da mudança.

Ajustando Consultas e Visões



- Se uma consulta é executada mais lentamente do que o esperado, verificar se um índice precisa ser reconstruído, ou se estatísticas estão muito antigas.
- Às vezes o SGBD pode não estar executando o plano que você tinha em mente. Áreas comuns de fraqueza:
  - Seleções envolvendo valores nulos.
  - Seleções envolvendo expressões aritméticas ou de string.
  - Seleções envolvendo condições OR.
  - Inabilidade de reconhecer um plano sofisticado, como estratégias apenas-de-índice ou certos métodos de junção, ou estimativa pobre de tamanho.
- Verifique o plano que está sendo usado! Então ajuste a escolha de índices ou reescreva a consulta/visão.

Reescrevendo Consultas SQL



- Complicada pela interação de:
  - NULLs, duplicatas, agregados, subconsultas.
- ***Orientação: Use apenas um "bloco de consulta", se possível.***

```
SELECT DISTINCT *
  FROM Sailors S
 WHERE S.sname IN
       (SELECT Y.sname
          FROM YoungSailors Y)
```

=

```
SELECT DISTINCT S.*
  FROM Sailors S,
       YoungSailors Y
 WHERE S.sname = Y.sname
```

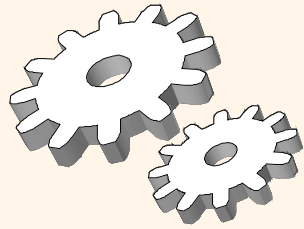
- *Nem sempre possível...*

```
SELECT *
  FROM Sailors S
 WHERE S.sname IN
       (SELECT DISTINCT Y.sname
          FROM YoungSailors Y)
```

≠

```
SELECT S.*
  FROM Sailors S,
       YoungSailors Y
 WHERE S.sname = Y.sname
```

O Notório Erro de COUNT



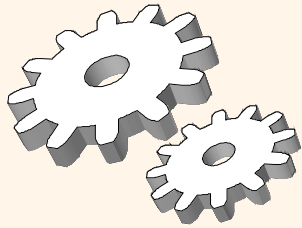
```
SELECT dname FROM Department D
 WHERE D.num_emps  >
          (SELECT COUNT(*) FROM Employee E
            WHERE D.building = E.building)
```

```
CREATE VIEW Temp (empcount, building) AS
       SELECT COUNT(*), E.building
         FROM Employee E
       GROUP BY E.building

SELECT dname
  FROM Department D,Temp
 WHERE D.building = Temp.building
   AND D.num_emps > Temp.empcount;
```

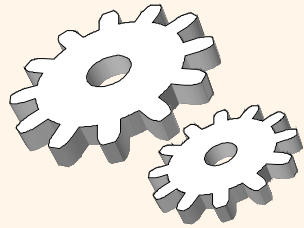
- O que acontece quando Employee está vazia??

Sumário sobre Desaninhamento de Consultas



- DISTINCT no nível superior: *Pode ignorar duplicatas.*
  - Às vezes pode inferir DISTINCT no nível superior! (e.g. cláusula da subconsulta casa no máximo uma tupla)
- DISTINCT na subconsulta sem DISTINCT no nível superior: *Difícil converter.*
- Subconsultas dentro de OR: *Difícil converter.*
- Subconsultas com ALL : *Difícil converter.*
  - EXISTS e ANY são exatamente como IN.
- Agregados em subconsultas: *Tricky.*
- Boas notícias: Atualmente alguns sistemas reescrevem as consultas por debaixo dos panos (e.g. DB2).

Orientações para Ajuste de Consultas



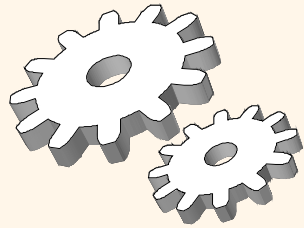
- Minimize o uso de DISTINCT: não é necessário se duplicatas são aceitáveis, ou se a resposta contém uma chave.
- Minimize o uso de GROUP BY e HAVING:

```
SELECT  MIN (E.age)
FROM  Employee E
GROUP BY  E.dno
HAVING  E.dno=102
```

```
SELECT  MIN (E.age)
FROM  Employee E
WHERE  E.dno=102
```

- Considere o uso de índice pelo SGBD na escrita de expressões aritméticas: *E.age=2*D.age* irá se beneficiar de um índice em *E.age*, mas pode não se beneficiar de um índice em *D.age*!

Orientações para Ajuste de Consultas (Cont.)



```
SELECT  *  INTO  Temp
FROM  Emp E, Dept D
WHERE  E.dno=D.dno
    AND  D.mgrname='Joe'

SELECT  T.dno, AVG(T.sal)
FROM  Temp T
GROUP BY  T.dno
```

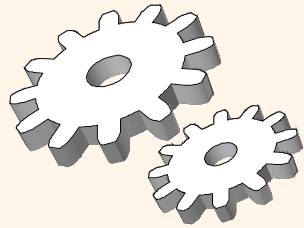
vs

- Evite o uso de relações intermediárias:

```
SELECT  E.dno, AVG(E.sal)
FROM  Emp E, Dept D
WHERE  E.dno=D.dno
          AND  D.mgrname='Joe'
GROUP BY  E.dno
```

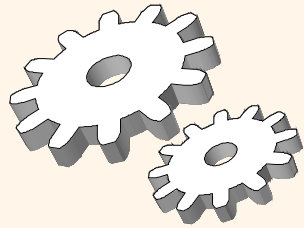
- Não materializa a relação intermediária Temp.
- Se há um índice de árvore B+ densa em <*dno, sal*>, um plano somente-de-índice pode ser usado para evitar obter as tuplas Emp na segunda consulta!

Sumário



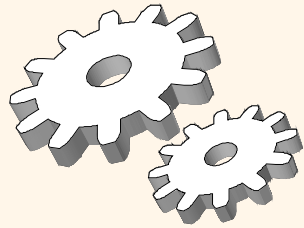
- Projeto de banco de dados consiste de várias tarefas: *análise de requisitos, projeto conceitual, refinamento do esquema, projeto físico e ajuste.*
  - Geralmente, tem-se que ir e voltar entre estas tarefas para refinar um projeto de banco de dados, e decisões em uma tarefa podem influenciar as escolhas em outra tarefa.
- Entender a natureza da *carga de trabalho* da aplicação e as metas de desempenho é essencial para desenvolver um bom projeto.
  - Quais são as consultas e atualizações importantes? Quais os atributos/relações envolvidos?

Sumário (Cont.)



- O esquema conceitual deve ser refinado considerando-se critérios de desempenho e carga de trabalho:
  - Pode-se escolher 3NF ou forma normal mais baixa em relação à BCNF.
  - Pode-se escolher entre decomposições alternativas em BCNF (ou 3NF) baseando-se na carga de trabalho.
  - Pode-se *desnormalizar*, ou desfazer algumas decomposições.
  - Pode-se decompor ainda mais uma relação BCNF!
  - Pode-se escolher uma *decomposição horizontal* de uma relação.
  - Importância de preservação-de-dependências baseada na dependência a ser preservada, e o custo da verificação de RI.
    - Pode-se adicionar uma relação para garantir preservação-de-dependência; ou então, pode-se verificar a dependência usando uma junção.

Sumário (Cont.)



- Com o passar do tempo, índices precisam ser ajustados mais precisamente (eliminados, criados, reconstruídos, ...) para melhorar o desempenho.
  - Deve-se determinar o plano usado pelo sistema e ajustar apropriadamente a escolha de índices.
- Sistema pode, mesmo assim, não encontrar um bom plano:
  - Somente planos de profundidade à esquerda são considerados!
  - Valores nulos, condições aritméticas, expressões de string, o uso de ORs etc. podem confundir o otimizador.
- Então, pode-se ter que reescrever a consulta/visão:
  - Evitar consultas aninhadas, relações temporárias, condições complexas e operações como DISTINCT e GROUP BY.