Chapter 5:
Database Performance

Introduction

- How are tables stored on the disk?
- Main (only?) issue is *performance*
- **Minimize disk access**
- Logical design should NOT change!
- Many decisions should be deferred till implementation
- Highly DBMS dependent

Hard Disks

- Most databases reside in hard disks
- Hard disk issues
  - Info is transferred in ***pages*** **(blocks)**
  - Slight difference in access time to reach each page, but considered the same
  - Speed: seek time, transfer rate (about 100 times slower than RAM)
- The fewer the pages I read/write, the faster my database

Fields: Choosing Data Types

- CHAR–fixed-length character
- VARCHAR–variable-length character (memo)
- LONG–large number
- NUMBER–positive/negative number
- INEGER–positive/negative whole number
- DATE–actual date
- BLOB–binary large object (good for graphics, sound clips, etc.)

Fields: Coding

- Uses a reference instead of an actual value, and puts the actual value in another table
- **Saves space but needs additional lookup**
- Can improve integrity

ID	Name	Gender
1	Alex	Male
2	Ela	Female
3	Jason	Male

ID	Name	Gender
1	Alex	M
2	Ela	F
3	Jason	M

Code	Name
M	Male
F	Female

Physical Records

- Physical Record: A group of fields stored in adjacent memory locations and retrieved together as a unit (~row)
- *Page:* The amount of data read or written in one I/O operation (device dependent)
- *Blocking Factor*: The number of physical records per page. May vary!

Example Pages (Disk Blocks)

Staff#	Position	Branch#
A1	Assistant	1
M1	Manager	1
A2	Assistant	2
M2	Manager	2

Staff#	Position	Branch#	
A1	Assistant	1	page 1
M1	Manager	1	
A2	Assistant	2	page 2
M2	Manager	2	

Denormalization

- Transforming normalized relations into denormalized physical record specifications
- Benefits:
  - Can improve performance (speed) by reducing number of table lookups (i.e. reduce number of necessary join queries)
- Costs (due to data duplication)
  - Wasted storage space
  - Data integrity/consistency threats
- Common denormalization opportunities
  - One-to-one relationship (Fig. 5-3)
  - Associative entity (Fig. 5-4)
  - Reference data (1:N relationship where 1-side has data not used in any other relationship) (Fig. 5-5)

Figure 5-3 A possible denormalization situation: one-to-one relationship

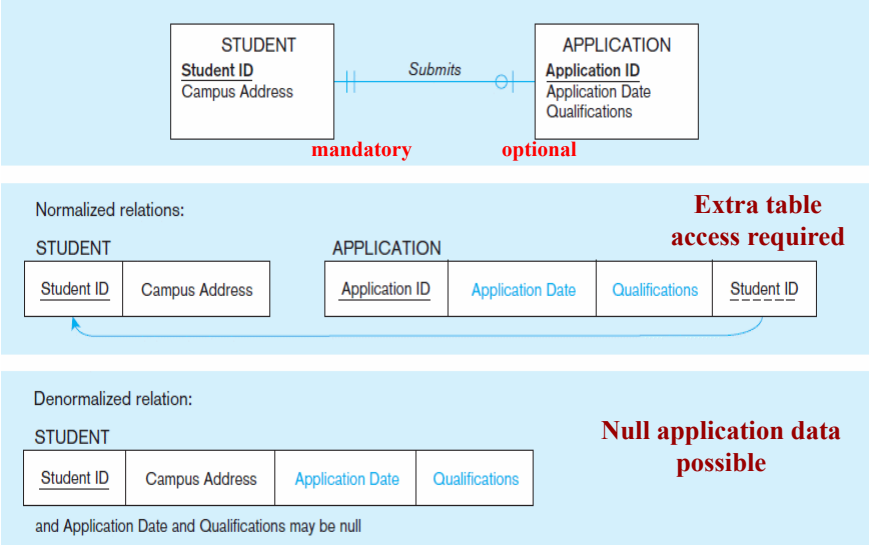


Figure 5-4 A possible denormalization situation: a many-to-many relationship with nonkey attributes

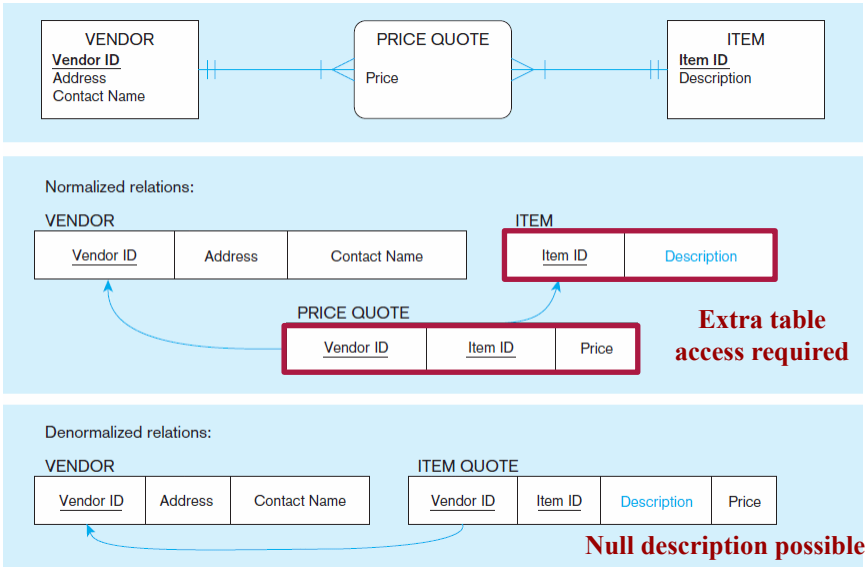
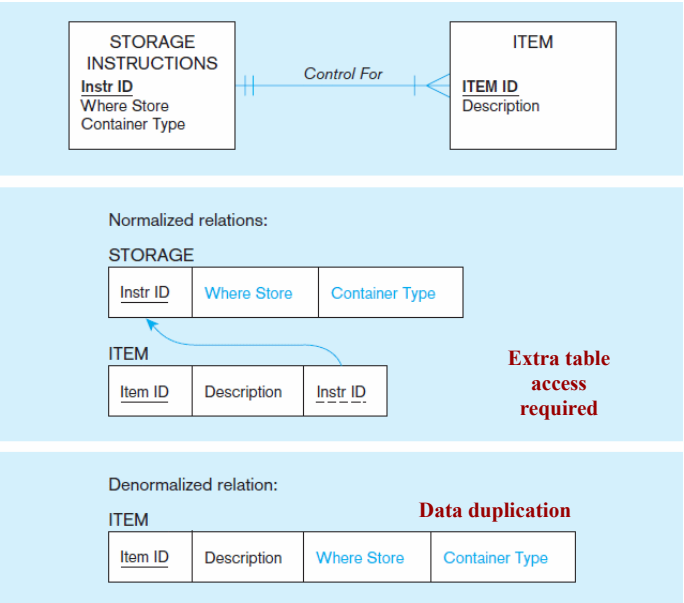


Figure 5-5
A possible denormalization situation: reference data



Denormalize with CAUTION

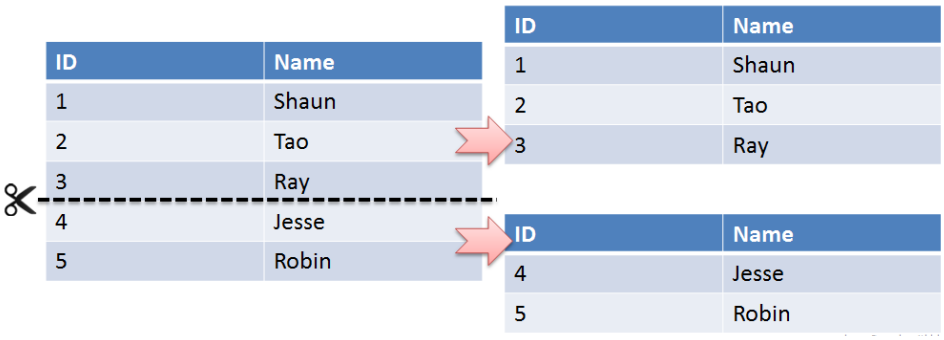
- Denormalization can
  - Increase chance of errors and inconsistencies
  - Reintroduce anomalies
  - Force reprogramming when business rules change
- Perhaps other methods could be used to improve performance
  - Organization of tables in the database (file organization and clustering)
  - Proper query design and optimization

Partitioning

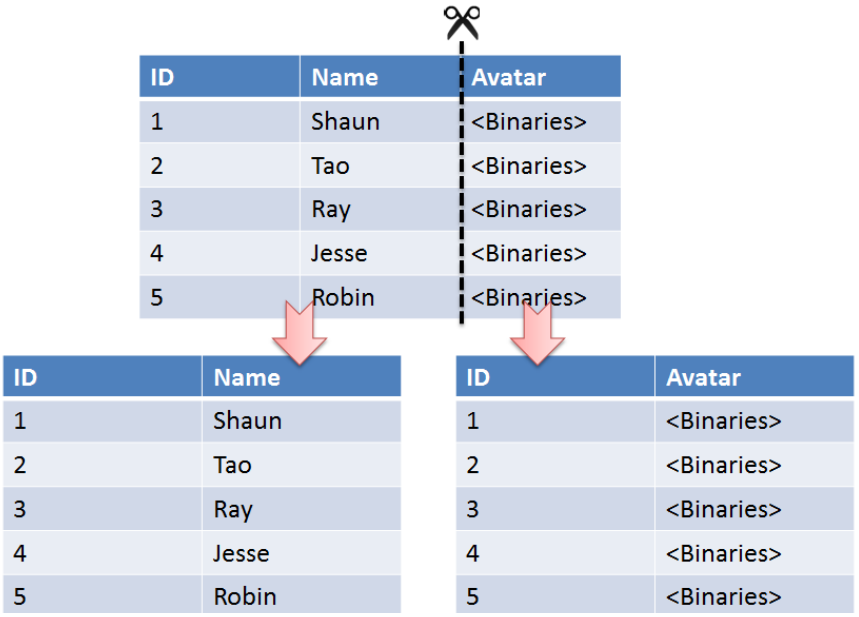
- **Horizontal**: Distribute rows into several files
  - Useful for situations where different users need access to different rows
- **Vertical**: Distribute columns into several relations
  - Useful for situations where different users need access to different columns
  - The primary key must be repeated in each file
- Combined Horizontal and Vertical
- Useful when working with large tables
- Enable parallel processing

Partitions often correspond with User Schemas (user views)

Horizontal Partitioning



Vertical Partitioning



ID	Name	Avatar
1	Shaun	<Binaries>
2	Tao	<Binaries>
3	Ray	<Binaries>
4	Jesse	<Binaries>
5	Robin	<Binaries>

ID	Name
1	Shaun
2	Tao
3	Ray
4	Jesse
5	Robin

ID	Avatar
1	<Binaries>
2	<Binaries>
3	<Binaries>
4	<Binaries>
5	<Binaries>