بسم الله الرحمن الرحيم

Denormalization

By:-
1- Amna Magzoub
2- Selma Yahyia
3- Ala Eltayeb

Objectives:-

- ❖ Introduction
- ❖ Definition
- ❖ Why and when to denormalize data
- ❖ Method of denormalization
- ❖ Manage denormalization data
- ❖ Advantages and disadvantages of denormalization
- ❖ References

Introduction

- Result of normalization is a design that is structurally consistent with minimal redundancy.
- However, sometimes a normalized database does not provide maximum processing efficiency.
- May be necessary to accept loss of some benefits of a fully normalized design in favor of performance.

Definition :-

- ❖Denormalization is a process of combine two relation into one new relation.
- ❖ Denormalization is the process of taking a normalized database and modifying table structures to allow controlled redundancy for increased database performance.

Cont..

- The argument in favor of denormalization is basically that it makes retrievals esaier to express and makes the perform better.

- It sometimes claimed to make the database easier to understand.

When and why to denormailize

Some issues need to be considered before denormalization:

1- Is the system’s performance unacceptable with fully normalized data? Meet a client and do some testing.

2- If the performance is unacceptable, will denormalizing make it acceptable?

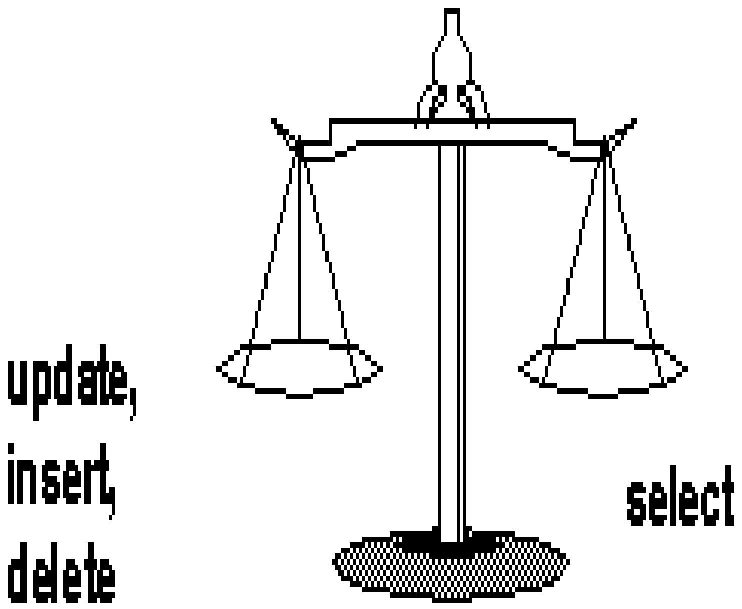
3- If you denormalize to clear those bottlenecks, will the system and its data still be reliable?

Cont…

- Speed up retrievals.
- A strict performance is required.
- It is not heavily updated.

- So, denormalize only when there is a very clear advantage to doing.

Balancing denormalization issues



Low number of updates +
Large number of queries =
Denormalization

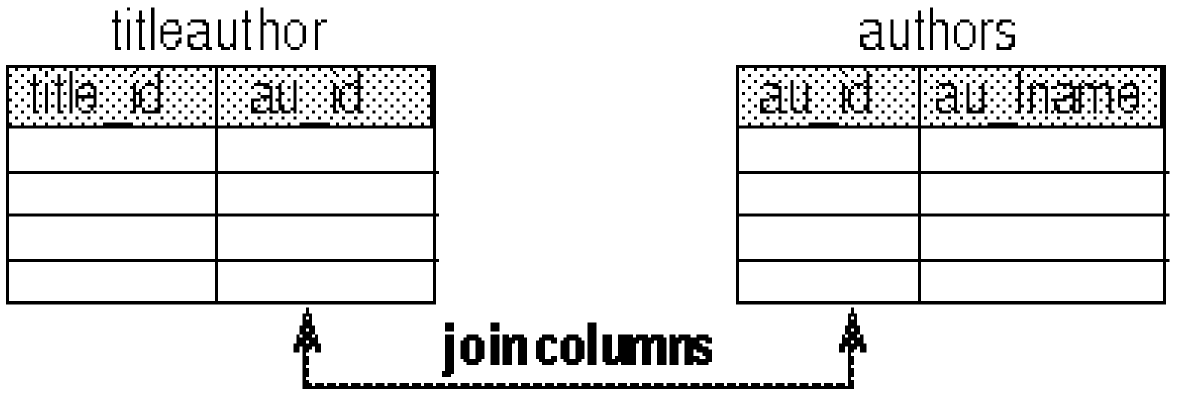
Method of denormalization:-

1) **Adding Redundant Columns**.
2) **Adding Derived Columns**
3) **Combining Tables**
4) **Repeating Groups**
5) **Creating extract tables**
6) **Partitioning Relations**

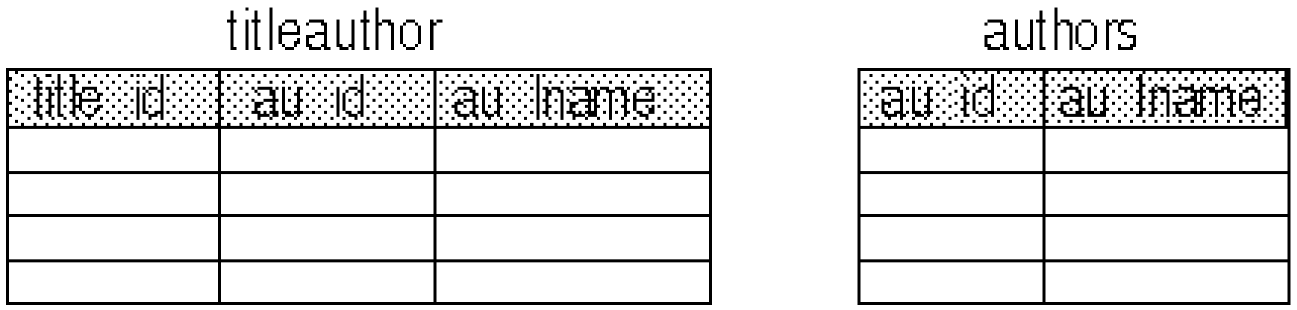
Adding Redundant Columns

- You can add redundant columns to eliminate frequent joins. For example, if frequent joins are performed on the *titleauthor* and *authors* tables in order to retrieve the author's last name, you can add the *au_lname* column to *titleauthor*.

```
select ta.title_id, a.au_id, a.au_lname
from titleauthor ta, authors a
where ta.au_id = a.au_id
```



```
select title_id, au_id, au_lname
from titleauthor
```

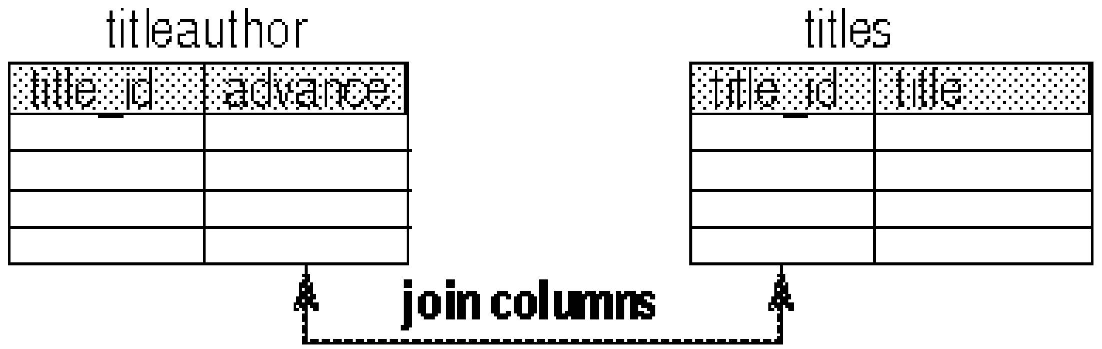


- Adding redundant columns eliminates joins for many queries. The problems with this solution are that it:
- Requires maintenance of new column. All changes must be made to two tables, and possibly to many rows in one of the tables.
- Requires more disk space, since *au_lname* is duplicated.

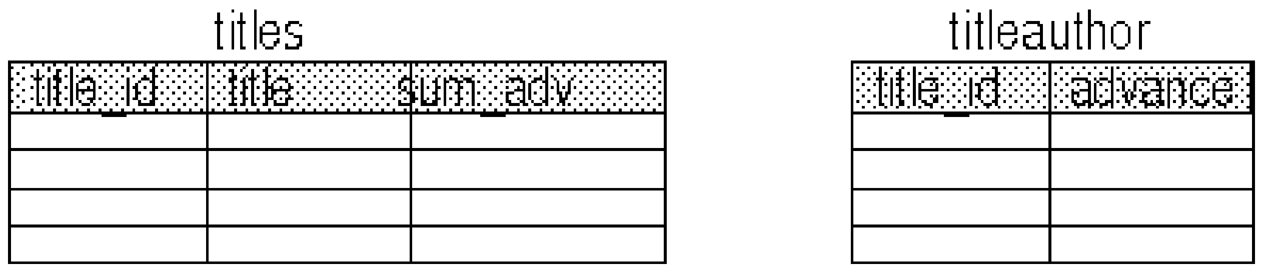
Adding Derived Columns

- Adding derived columns can help eliminate joins and reduce the time needed to produce aggregate values.
- The example shows both benefits. Frequent joins are needed between the *titleauthor* and *titles* tables to provide the total advance for a particular book title.

```
select title, sum(advance)
from titleauthor ta, titles t
where ta.title_id = t.title_id
group by title_id
```



```
select title, sum_adv
from titles
```

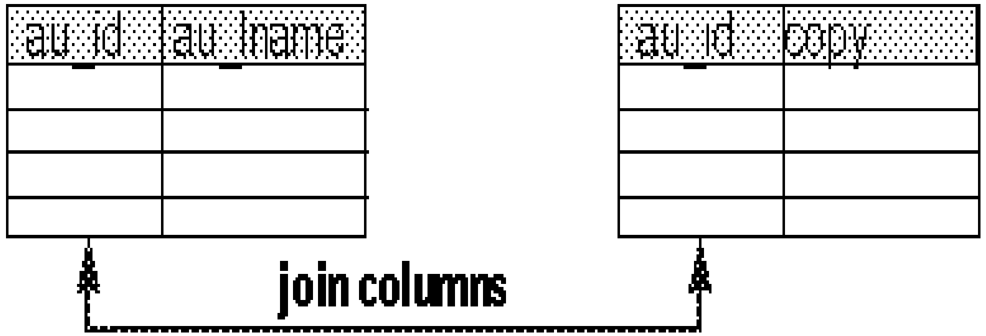


- You can create and maintain a derived data column in the *titles* table, eliminating both the join and the aggregate at run time. This increases storage needs, and requires maintenance of the derived column whenever changes are made to the *titles* table.

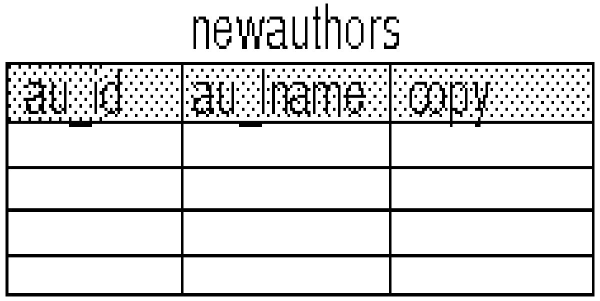
Combining Tables

- If most users need to see the full set of joined data from two tables, collapsing the two tables into one can improve performance by eliminating the join.
- For example, users frequently need to see the author name, author ID, and the *blurbs* copy data at the same time. The solution is to collapse the two tables into one. The data from the two tables must be in a one-to-one relationship to collapse tables.

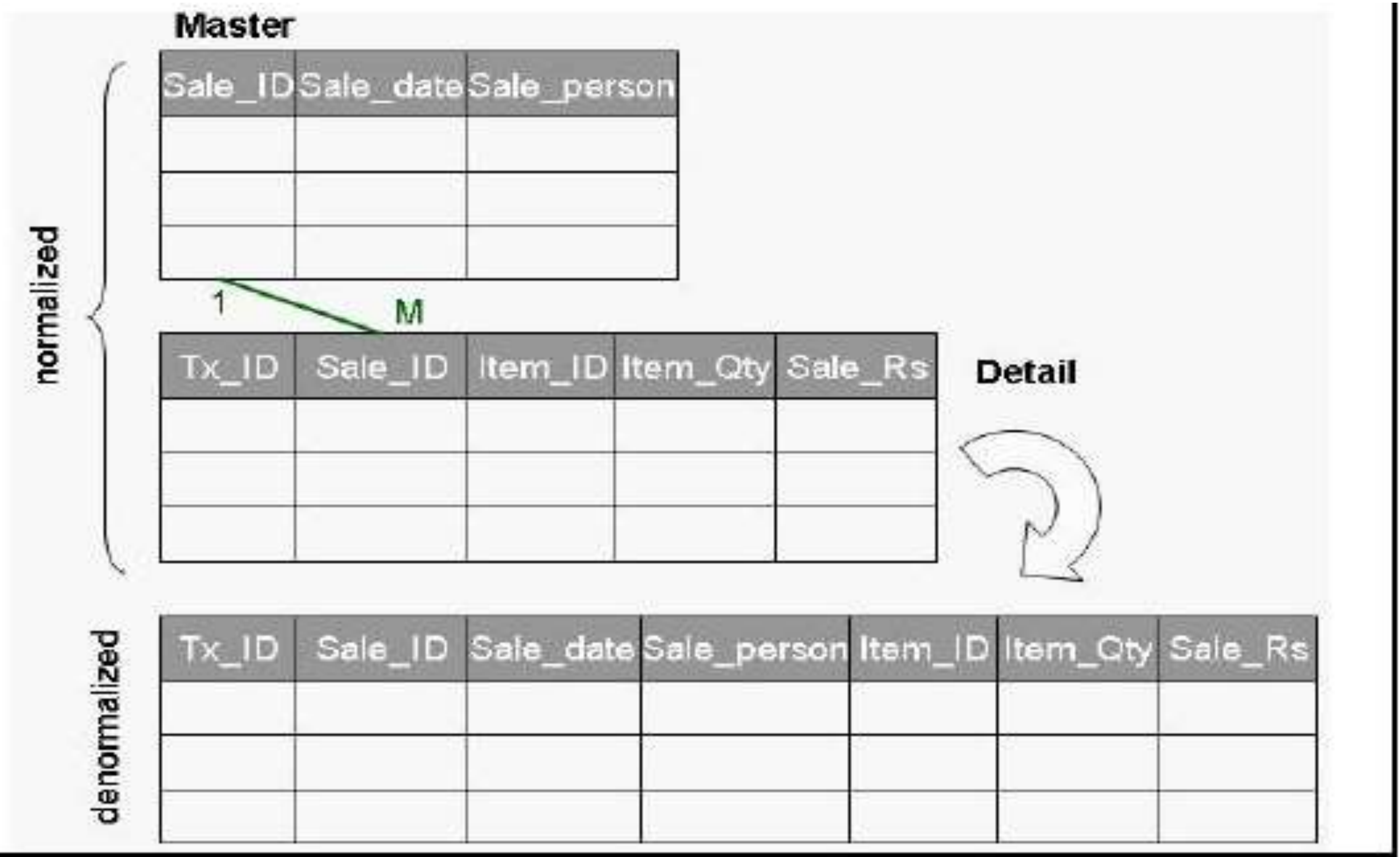
```
select a.au_id, a.au_lname,
b.copy
from authors a, blurbs b
where a.au_id = b.au_id
```



```
select * from newauthors
```



More examples:-



Sale_ID	Sale_date	Sale_person

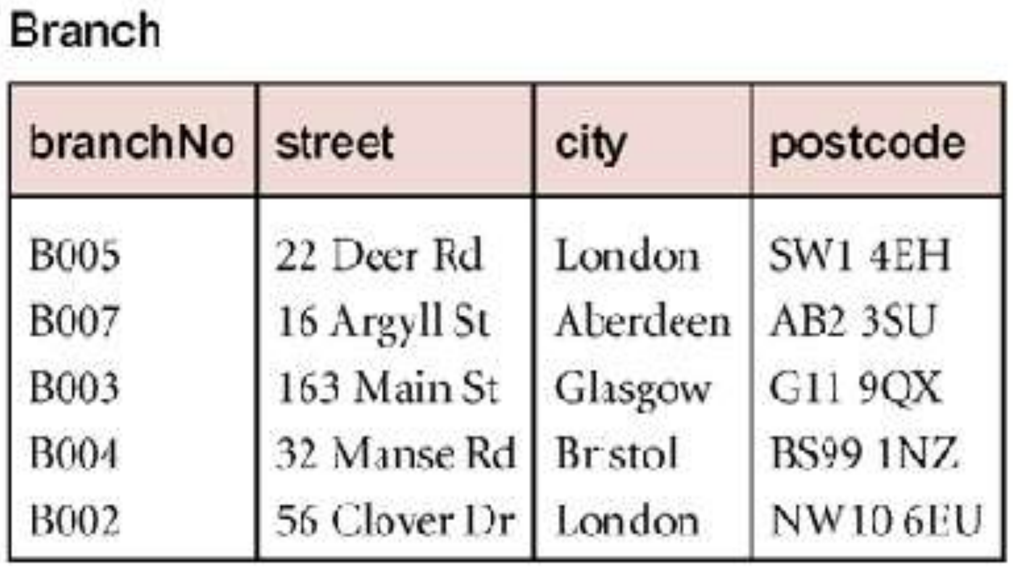
Tx_ID	Sale_ID	Item_ID	Item_Qty	Sale_Rs

Tx_ID	Sale_ID	Sale_date	Sale_person	Item_ID	Item_Qty	Sale_Rs

Repeating Groups

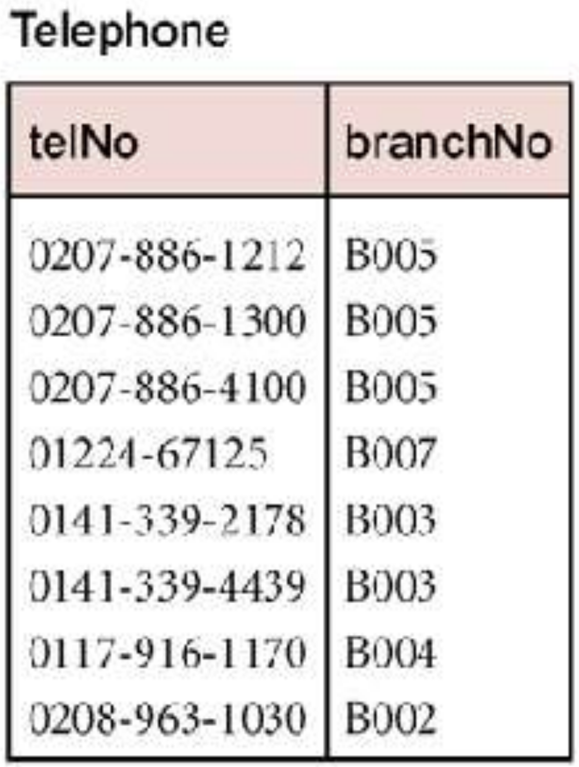
These repeating groups can be stored as a nested table within the original table.

example



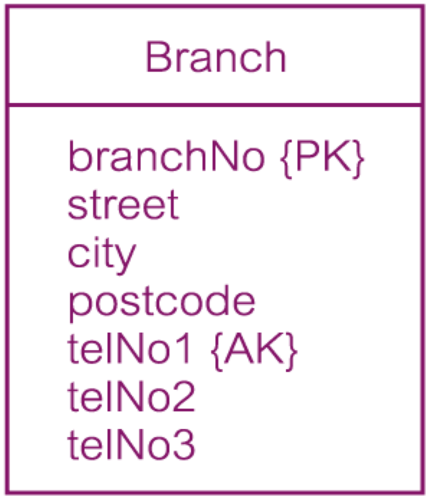
Branch

branchNo	street	city	postcode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU



Telephone

telNo	branchNo
0207-886-1212	B005
0207-886-1300	B005
0207-886-4100	B005
01224-67125	B007
0141-339-2178	B003
0141-339-4439	B003
0117-916-1170	B004
0208-963-1030	B002



(a)

Branch

branchNo	street	city	postcode	telNo1	telNo2	telNo3
B005	22 Deer Rd	London	SW1 4EH	0207-886-1212	0207-886-1300	0207-886-4100
B007	16 Argyll St	Aberdeen	AB2 3SU	01224-67125		
B003	163 Main St	Glasgow	G11 9QX	0141-339-2178	0141-339-4439	
B004	32 Manse Rd	Bristol	BS99 1NZ	0117-916-1170		
B002	56 Clover Dr	London	NW10 6EU	0208-963-1030		

(b)

Creating extract tables

- Reports can access derived data and perform multi-relation joins on same set of base relations. However, data the report is based on may be relatively static or may not have to be current.
- Possible to create a single, highly denormalized extract table based on relations required by reports, and allow users to access extract table directly instead of base relations.

Partitioning Relations

- Rather than combining relations together, alternative approach is to decompose them into a number of smaller and more manageable partitions.
- Two main types of partitioning:-

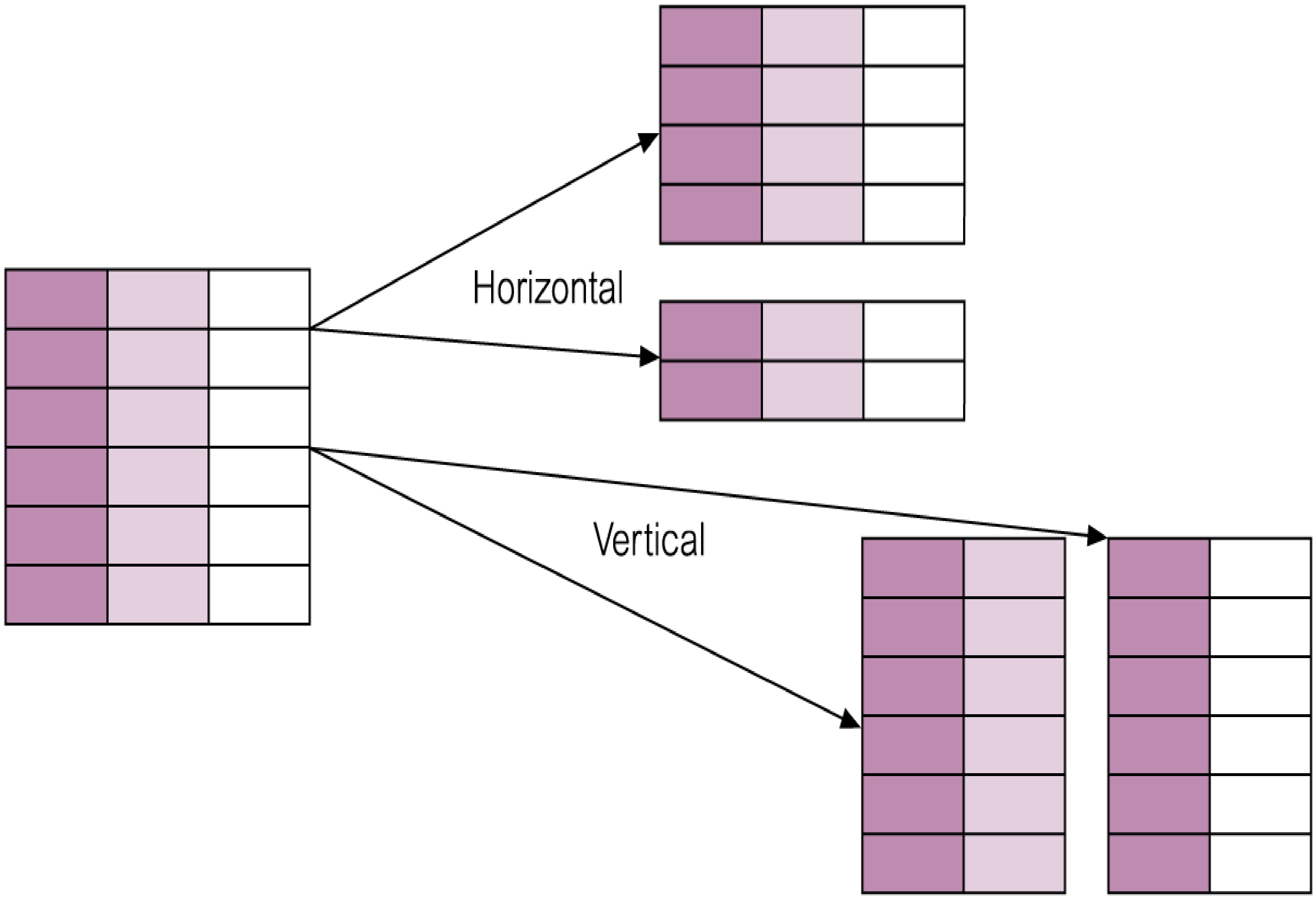
  horizontal and vertical.

Horizontal :-

Distributing the tuples of relation across a number of (smaller) partitioning relation.

Vertical :-

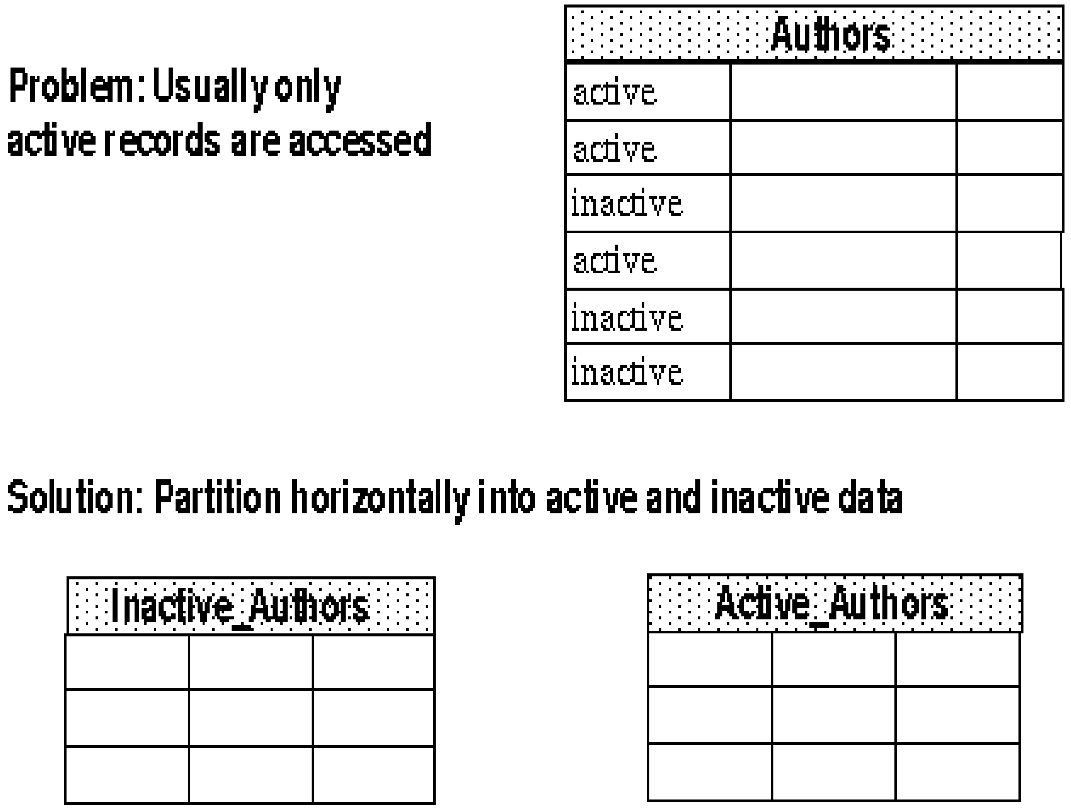
Distributing the attributes of a relation a cross a number of (smaller) partitioning relation(the primary key duplicated to allow the original relation to be reconstucted.


Horizontal
Vertical

Horizanteial:-

- Separate data into partitions so that queries do not need to examine all data in a table when WHERE clause filters specify only a subset of the partitions.
- Horizontal splitting can also be more secure since file level of security can be used to prohibit users from seeing certain rows of data

The example shows how the *authors* table might be split to separate active and inactive authors:



Problem: Usually only active records are accessed

Solution: Partition horizontally into active and inactive data

Vertical Splitting:

- Vertical splitting can be used when some columns are rarely accessed rather than other columns

The example shows how the *authors* table can be partitioned.

Problem:
Frequently access lname and fname, infrequently access phone and city

Solution: Partition data vertically

Authors

au_id	lname	fname	phone	city

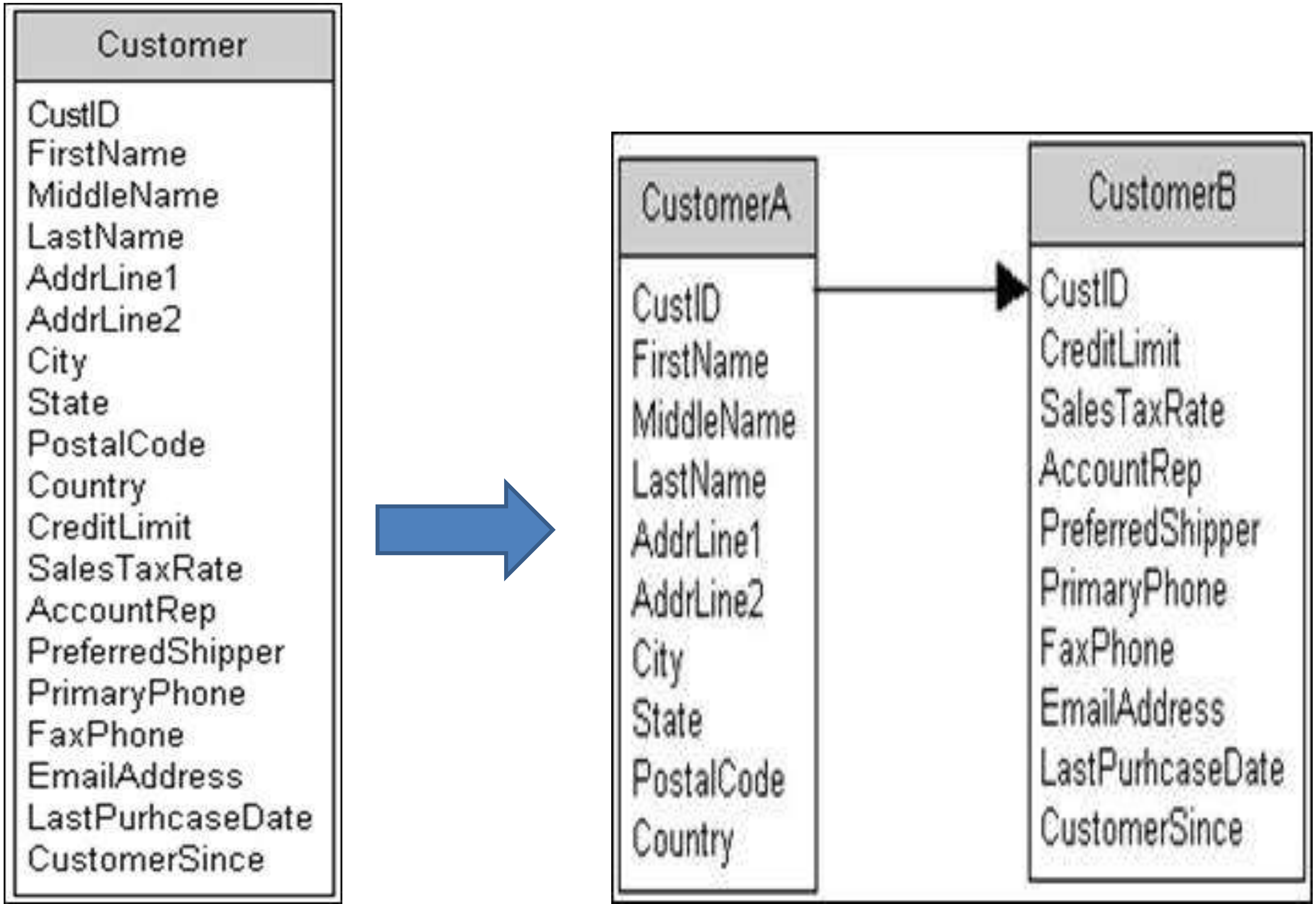
Authors_Frequent

au_id	lname	fname

Authors_Infrequent

au_id	phone	city

Vertical Examples



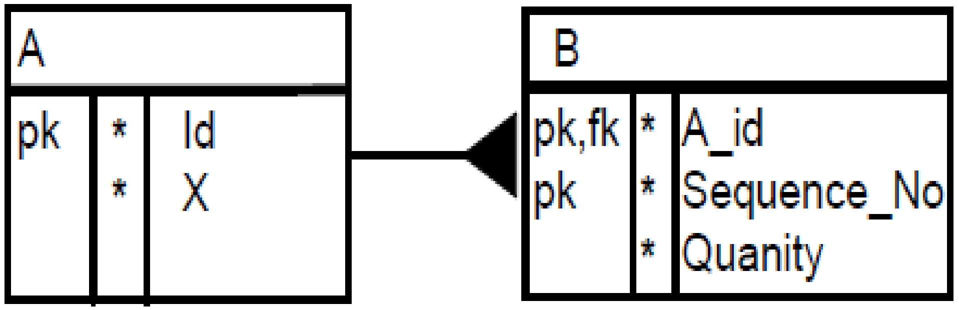
Managing Denormalized Data

Whatever denormalization techniques you use, you need to develop management techniques to ensure data integrity. Choices include:

- Triggers, which can update derived or duplicated data anytime the base data changes.

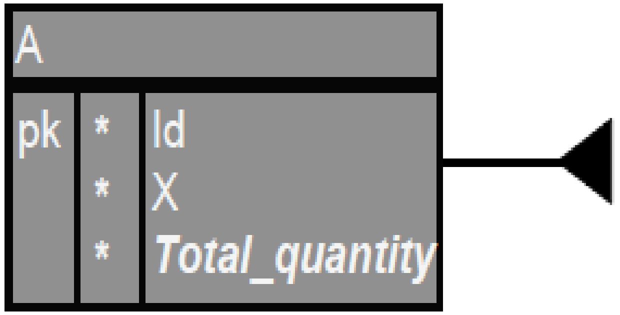
Storing Derivable Values

Before

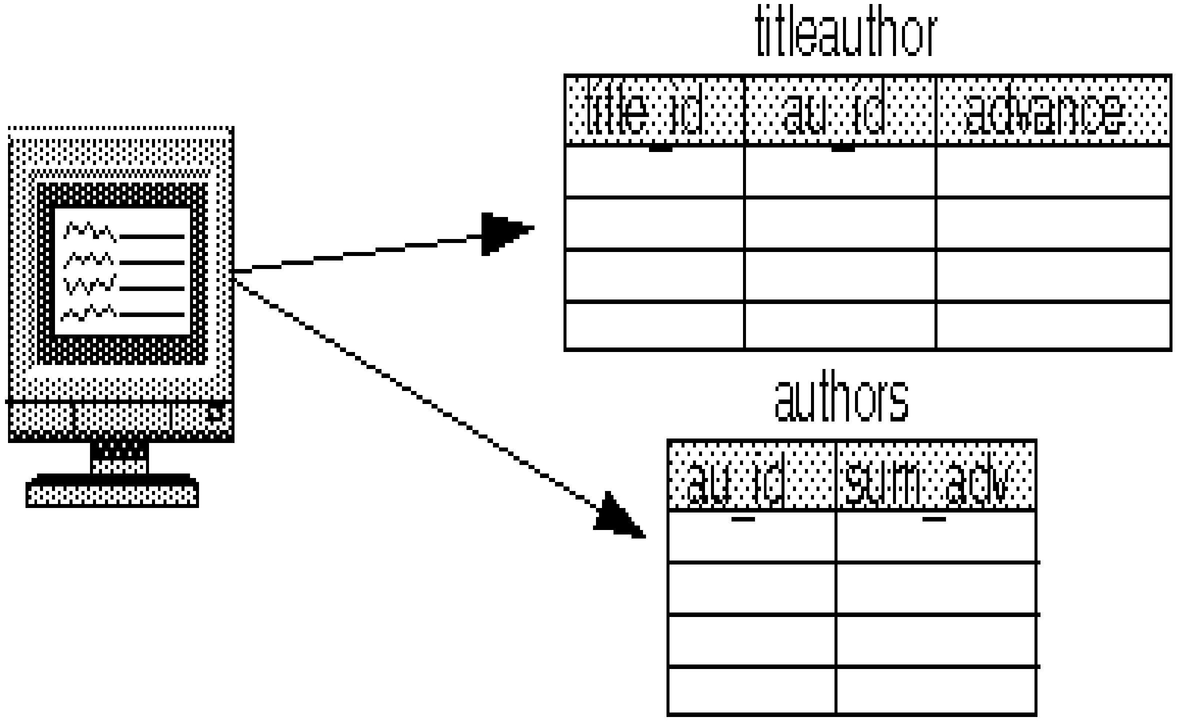


Add a column to store derivable data in the "referenced" end of the foreign key.

After



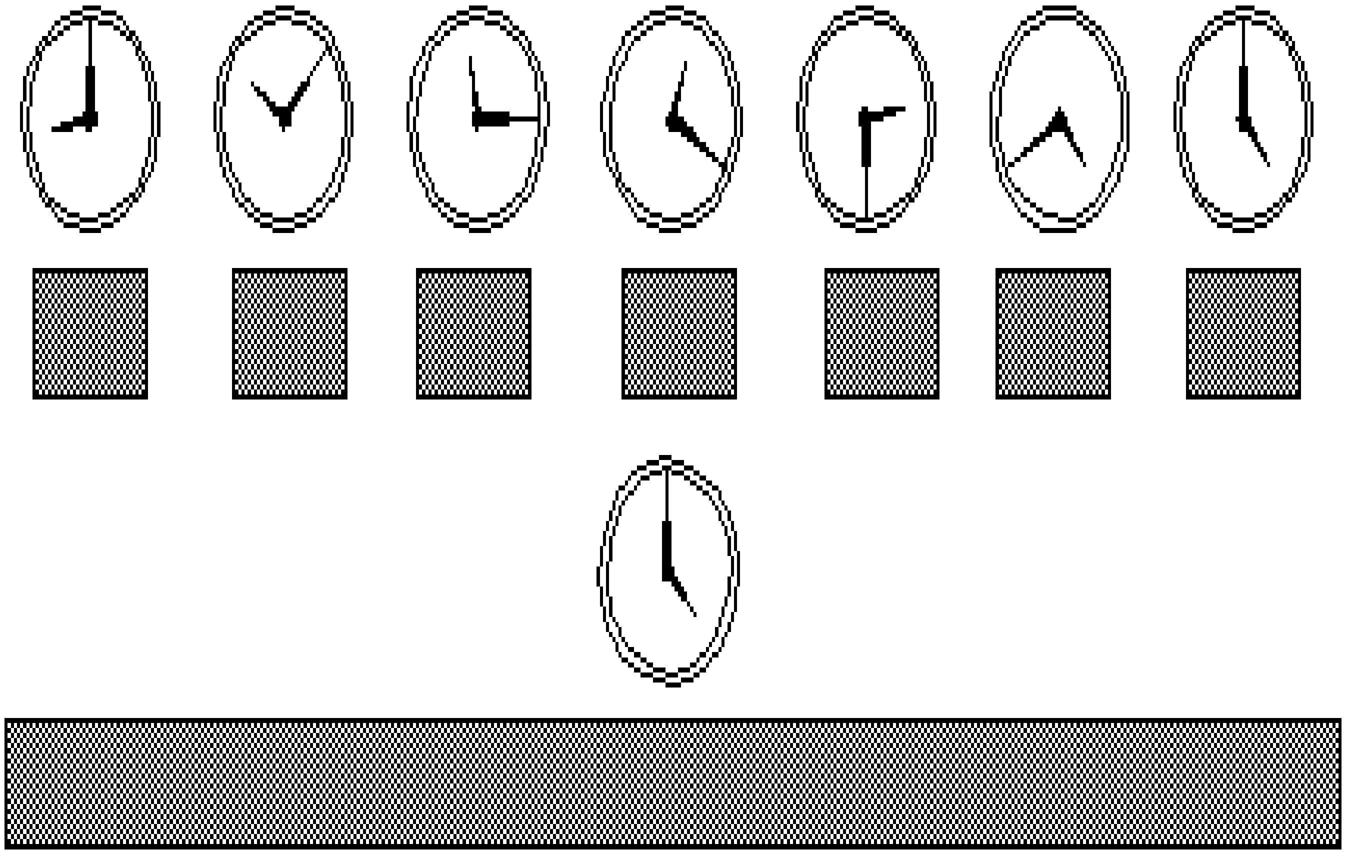
- Application logic, using transactions in each application that updates denormalized data to be sure that changes are atomic.



Cont…

- Batch reconciliation, run at appropriate intervals to bring the denormalized data back into agreement.
- If 100-percent consistency is not required at all times, you can run a batch job or stored procedure during off hours to reconcile duplicate or derived data.
- You can run short, frequent batches or longer, less frequent batches.

Cont…



Advantages vs. disadvantages

Advantages:-

- Precomputing derived data
- Minimizing the need for joins
- Reducing the number of foreign keys in relations
- Reducing the number of relations.

disadvantages :-

- May speed up retrievals but can slow down updates.
- Always application-specific and needs to be re-evaluated in the application changes.
- Can increase the size of relations.
- May simplify implementation in some cases but may make it more complex in other.
- reduce flexibility.

Summary

- Denormalization aids the process of adding redundancy to the database to improve performance .
- Denormalize can be done with tables or columns.
- Require aknowledge of how data is being used .
- There are costs of denormalization reduces the “integrity” of the design ,always slow DML (data manipulation language) , need more memory space to store redundant data and required additional programming to maintain the denormalized data.

References

- Database design & Relational theory, C.J.Date
- Database system .8th edition.
- Data Normalization, Denormalization,and the Forces of Darkness a white paper by Melissa Hollingsworth.
- www.icard.ru/~nail/sybase/perf/10.88.html

Q & A

Thanks you