DATA WAREHOUSING

ENGR. MADEHA MUSHTAQ
DEPARTMENT OF COMPUTER SCIENCE
IQRA NATIONAL UNIVERSITY

DENORMALIZATION

- Denormalization is a strategy used on a previously-normalized database to increase performance.
- Denormalization is used to improve query performance and speed up reporting.
- Normalization is a rule of thumb in DBMS but in DSS ease of use is achieved by way of Denormalization.

HOW DENORMALIZATION IMPROVES PERFORMANCE

- Denormalization specifically improves performance by either:
- Reducing the number of tables and hence the reliance on joins, which consequently speeds up performance.
- Reducing the number of joins required during query execution, or
- Reducing the number of rows to be retrieved from the Primary Data Table.

WHEN AND WHY TO USE DENORMALIZATION

- Improving query performance.
- Speeding up reporting.
- Computing commonly-needed values up front: We want to have some values ready-computed so we don’t have to generate them in real time.

DENORMALIZATION TECHNIQUES

- Storing Derivable Information
- Pre-Joining Tables
- Hard-Coded Values
- Keeping Detail with Master
- Repeating Single Detail with Master
- Short-Circuit Key

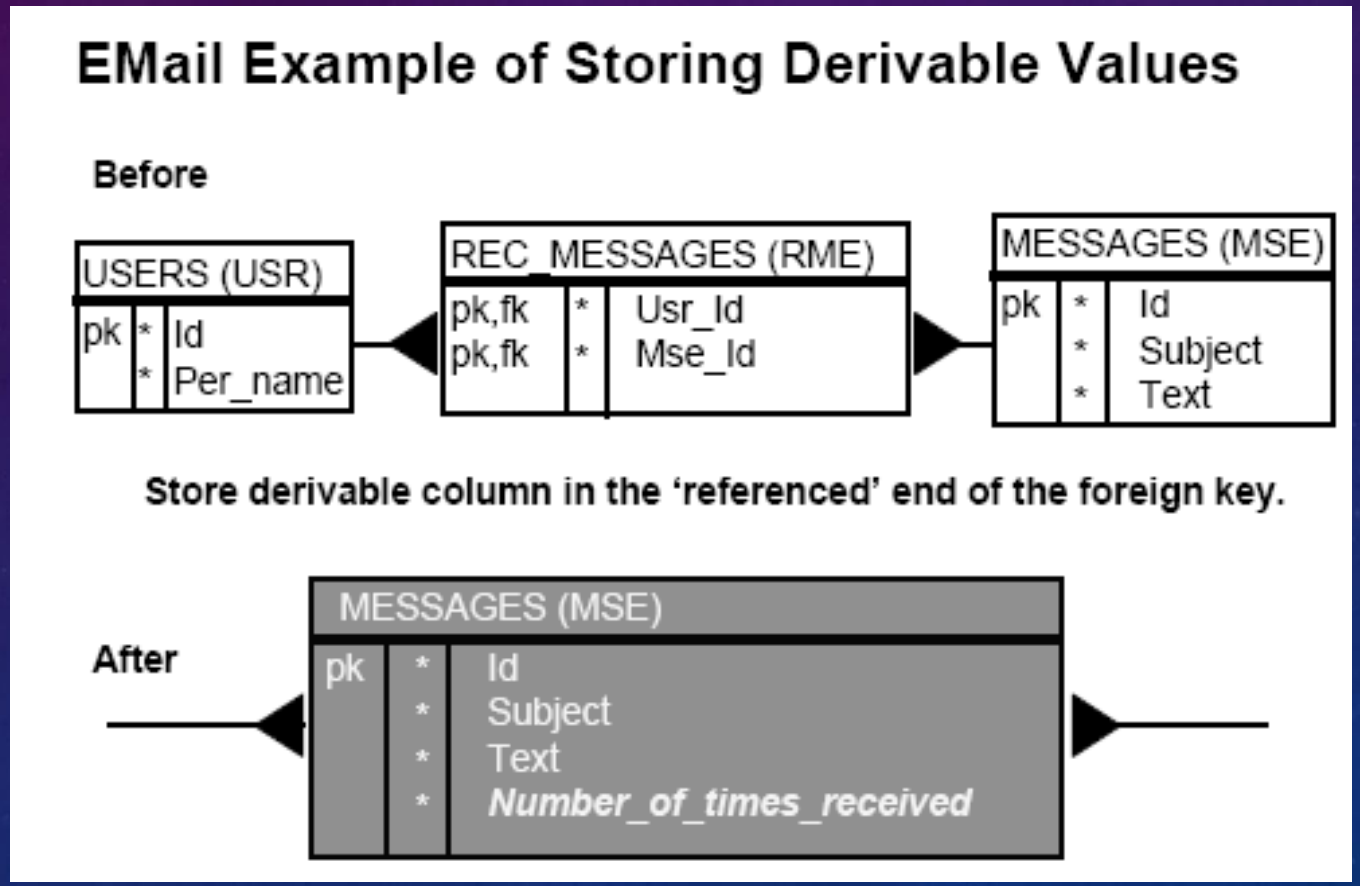
STORING DERIVABLE VALUES

- When a calculation is frequently executed during queries, it can be worthwhile storing the results of the calculation.
- If the calculation involves detail records, then store the derived calculation in the master table.
- Make sure to write application code to re-calculate the value, each time that DML is executed against the detail records.

STORING DERIVABLE VALUES

- Advantages:
  - Source values do not need to be looked up every time the derivable value is required.
  - The calculation does not need to be performed during a query or report.
- Disadvantages:
  - DML against the source data will require recalculation or adjustment of the derivable data.
  - Data duplication introduces the possibility of data inconsistencies.

STORING DERIVABLE VALUES



STORING DERIVABLE VALUES

- When a message is delivered to a recipient, the user only receives a pointer to that message, which is recorded in RECEIVED_MESSAGES.
- The reason for this, of course, is to prevent the mail system from storing a hundred copies of the same message when one message is sent to a hundred recipients.
- Then, when someone deletes a message from their account, only the entry in the RECEIVED_MESSAGES table is removed.

STORING DERIVABLE VALUES

- We could consider adding a de-normalized column to the MESSAGES table to keep track of the total number of RECEIVED_MESSAGES that are still kept for a particular message.
- Then each time users delete a row in RECEIVED_MESSAGES, in other words, they delete a pointer to the message, the Number_of_times_received column can be decremented.
- When the value of the de-normalized column equals zero, then we know the message can also be deleted from the MESSAGES table.

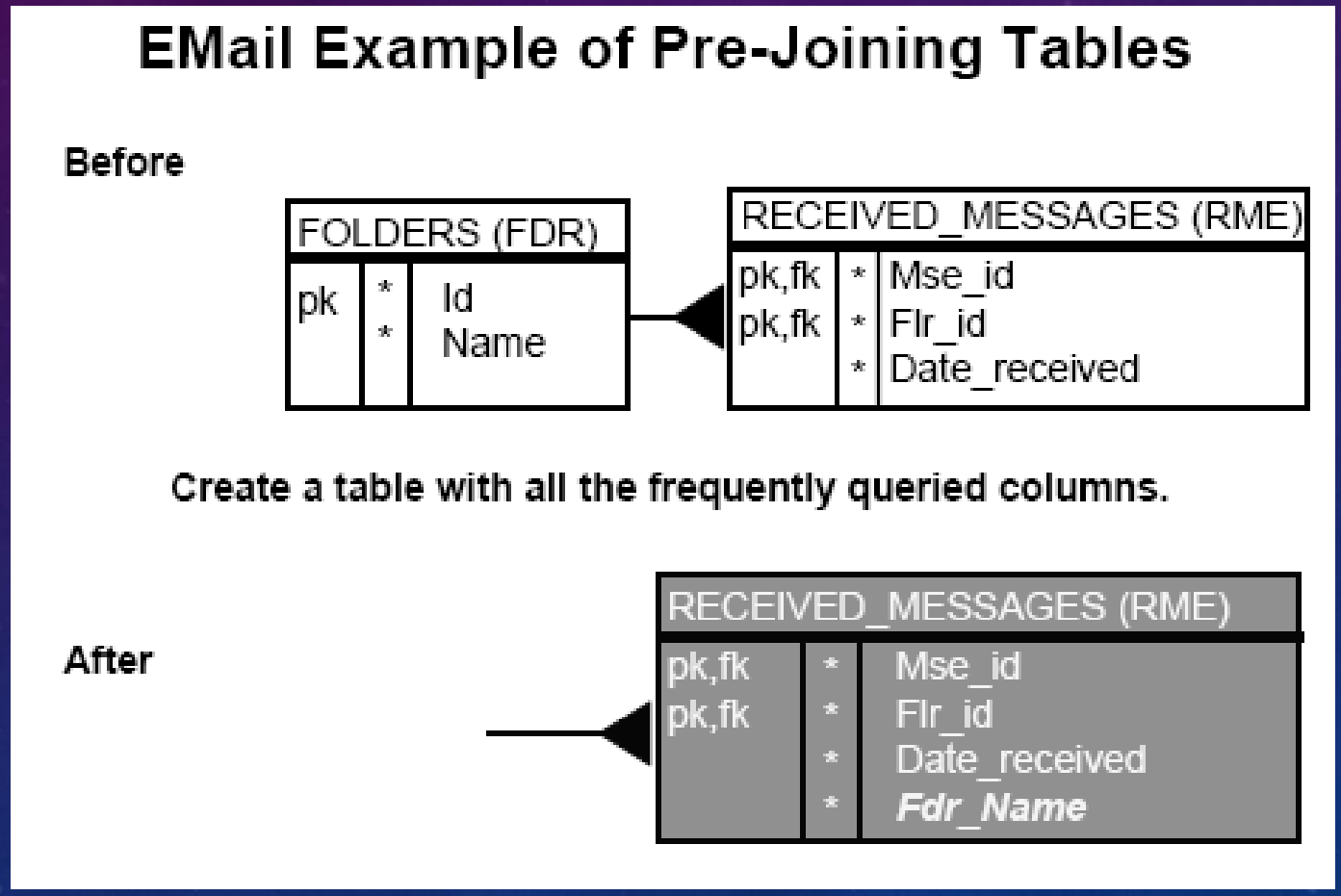
PRE-JOINING TABLES

- You can pre-join tables by including a non-key column in a table, when the actual value of the primary key, and the foreign key, has no business meaning.
- By including a non-key column that has business meaning, you can avoid joining tables, thus speeding up specific queries.
- We must include application code that updates the denormalized column, each time the “master” column value changes in the referenced record.

PRE-JOINING TABLES

- Advantages:
- Time-consuming joins can be avoided.
- Disadvantages:
- Extra DML needed to update original non-denormalized column.
- Extra column and possibly larger indices require more working space and disk.

PRE-JOINING TABLES



PRE-JOINING TABLES

- **Example**
  - Suppose users often need to query RECEIVED_MESSAGES, using the name of the folder where the received message is filed.
  - In this case it saves time when the name of the folder is available in the RECEIVED_MESSAGES table.
  - Now, if a user needs to find all messages in a particular folder, only a query on RECEIVED_MESSAGES is needed.

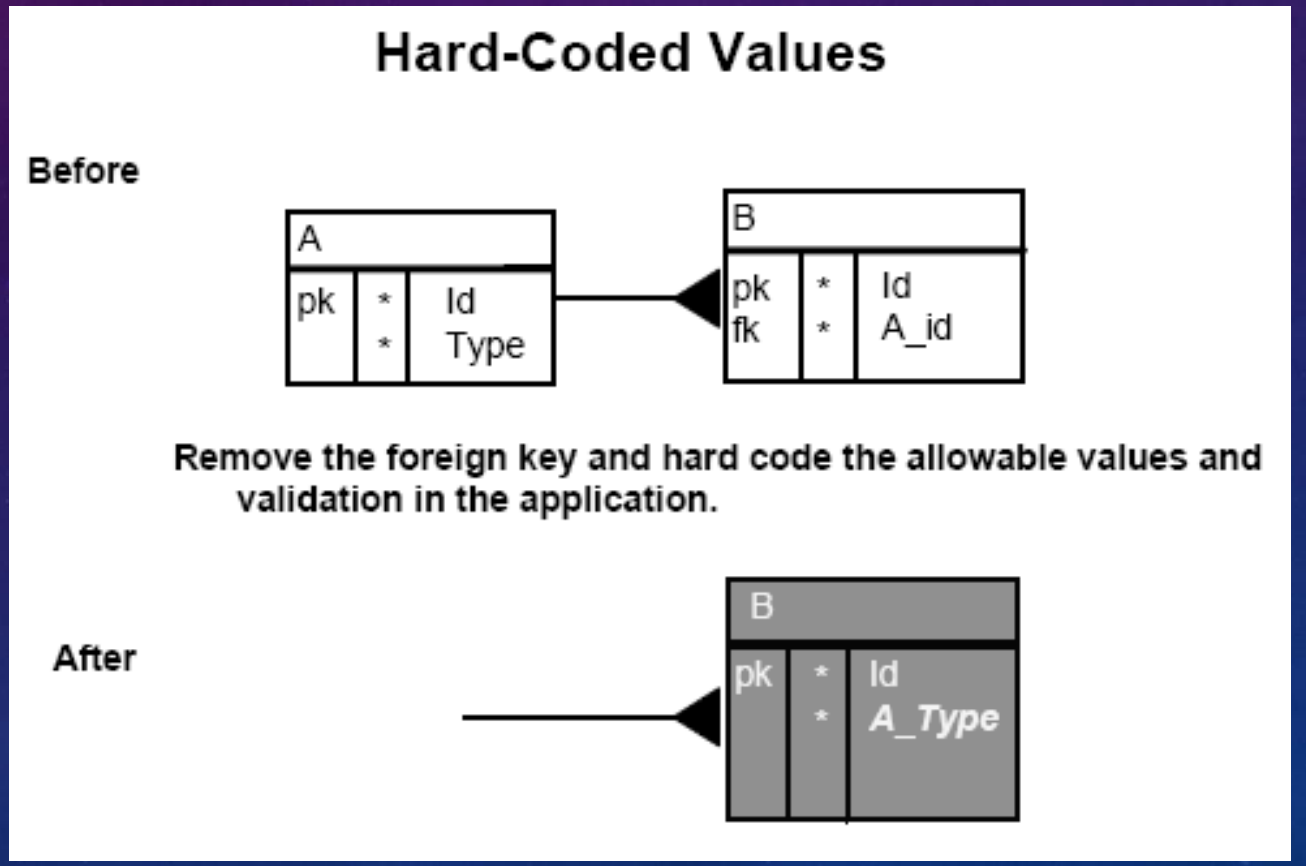
HARD-CODED VALUES

- If a reference table contains records that remain constant, then you can consider hard- coding those values into the application code.
- This will mean that you will not need to join tables to retrieve the list of reference values.
- This is a special type of denormalization, when values are kept outside a table in the database.

HARD-CODED VALUES

- Advantages:
  - Avoids implementing a look-up table.
  - Avoids joins to a look-up table.
- Disadvantages:
  - Changing look-up values requires recoding and retesting.

HARD-CODED VALUES



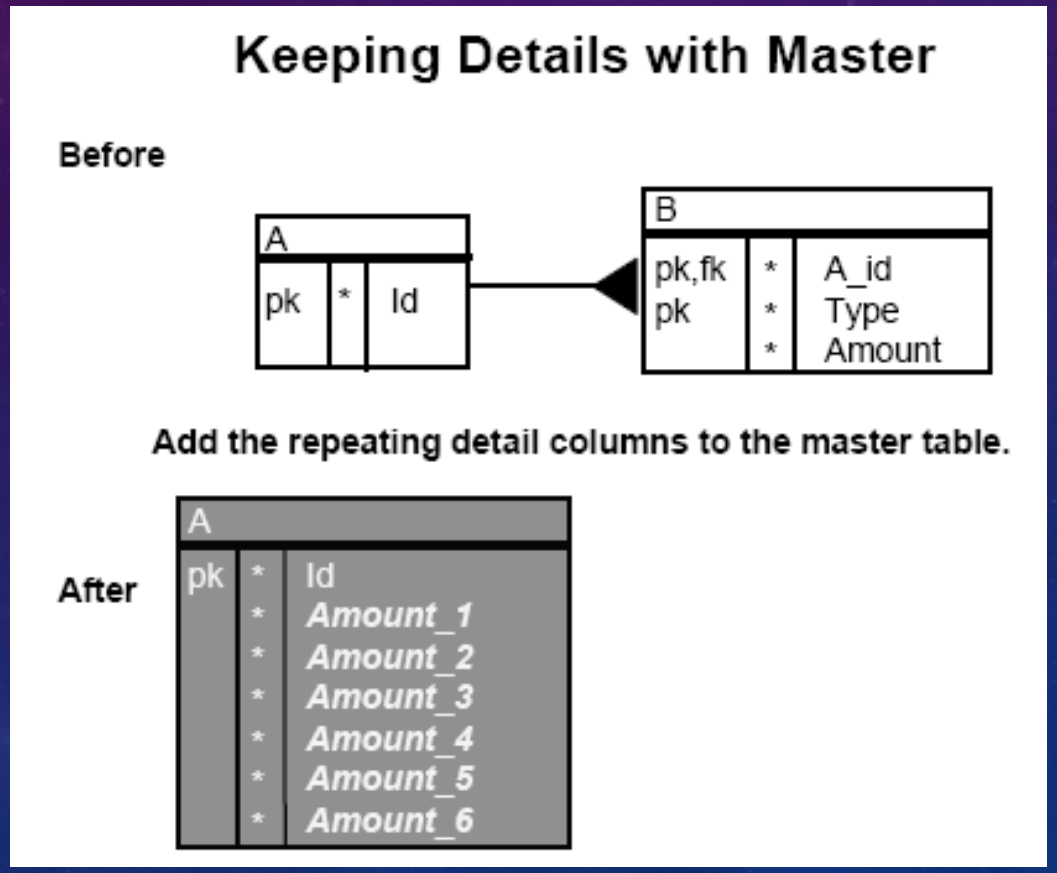
KEEPING DETAILS WITH MASTER

- In a situation where the number of detail records per master is a fixed value (or has a fixed maximum) and where usually all detail records are queried with the master, we may consider adding the detail columns to the master table.
- This denormalization works best when the number of records in the detail table are small.
- This way we can reduce the number of joins during queries.

KEEPING DETAILS WITH MASTER

- Advantages:
  - No joins are required.
  - Saves space, as keys are not propagated.
- Disadvantages:
  - Increases complexity of data manipulation language (DML) and SELECTs across detail values.
  - Checks for Amount column must be repeated for Amount1, Amount2 and so on.

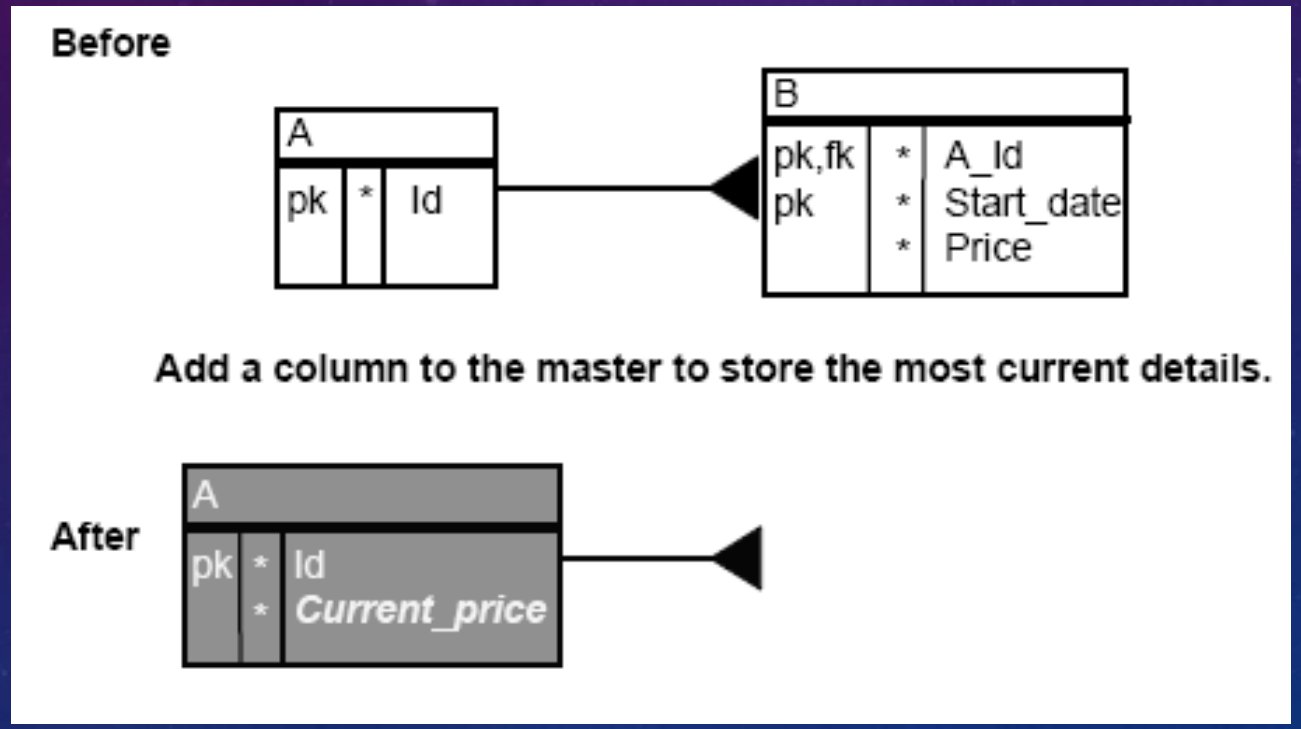
KEEPING DETAILS WITH MASTER



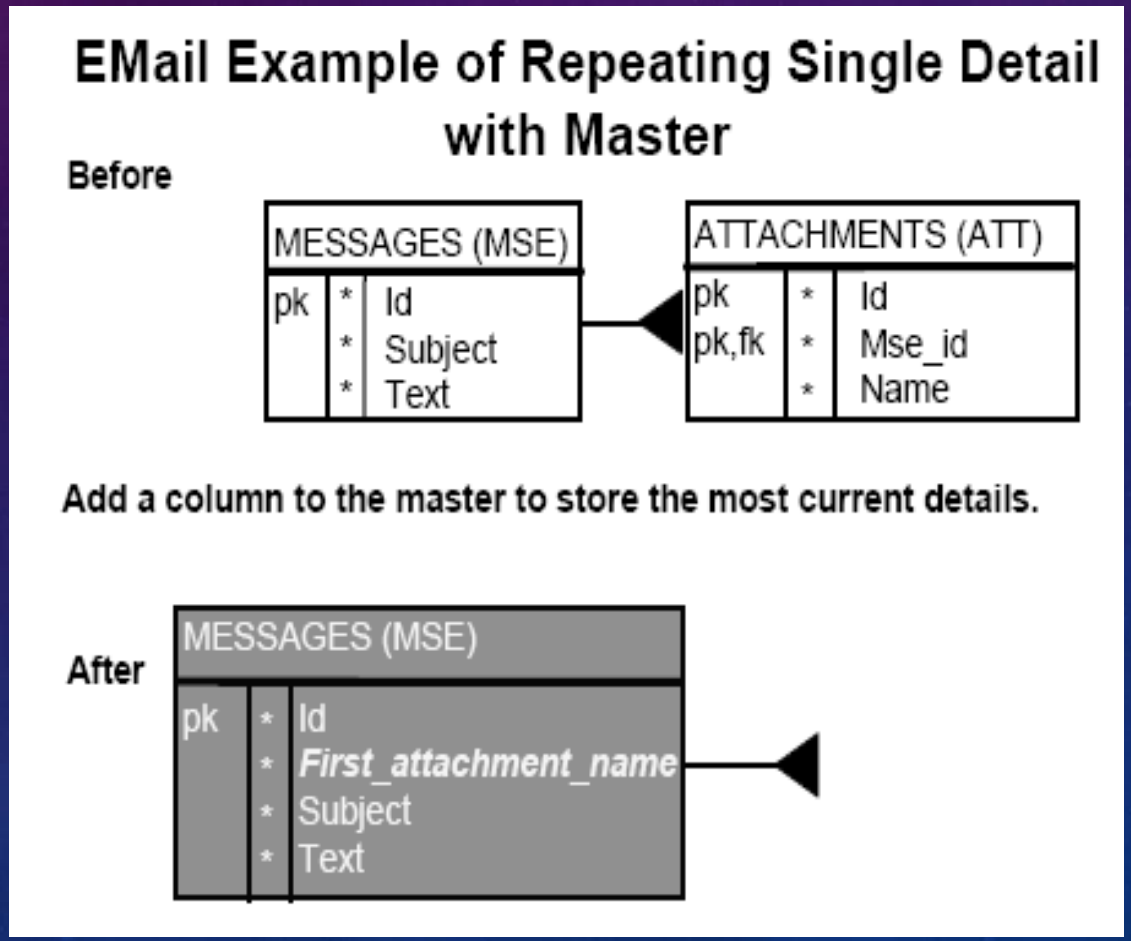
REPEATING SINGLE DETAIL WITH MASTER

- Often when the storage of historical data is necessary, many queries require only the most current record.
- We can add a new foreign key column to store this single detail with its master.
- Make sure you add code to change the denormalized column any time a new record is added to the history table.

REPEATING SINGLE DETAIL WITH MASTER



REPEATING SINGLE DETAIL WITH MASTER



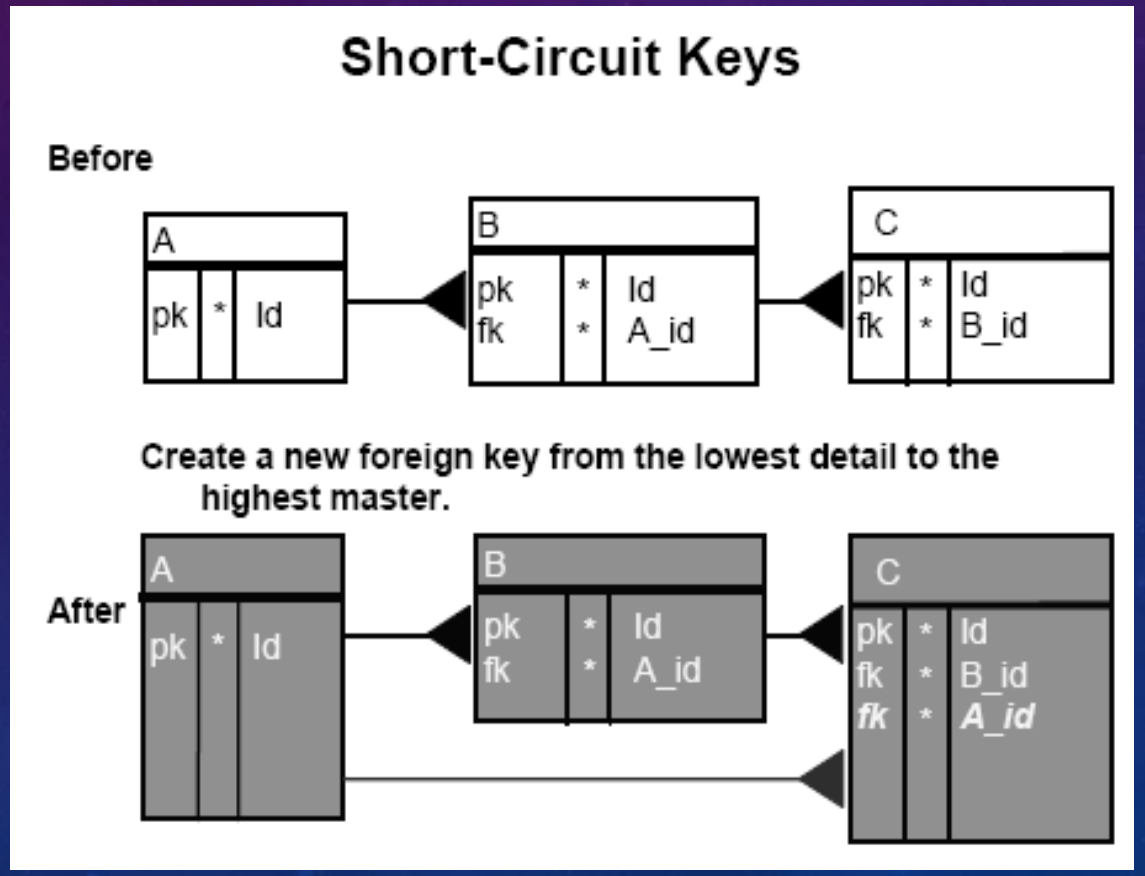
REPEATING SINGLE DETAIL WITH MASTER

- Advantages:
  - No join is required for queries that only need the specific single detail.
- Disadvantages:
  - Detail value must be repeated, with the possibility of data inconsistencies.
  - Additional code must be written to maintain the duplicated single detail value at the master record.

SHORT-CIRCUIT KEYS

- For database designs that contain three (or more) levels of master detail, and there is a need to query the lowest and highest level records only, consider creating short-circuit keys.
- These new foreign key definitions directly link the lowest level detail records to higher level grandparent records.
- The result can produce fewer table joins when queries execute.

SHORT-CIRCUIT KEYS



SHORT-CIRCUIT KEYS

- Advantages:
  - Queries join fewer tables together
- Disadvantages:
  - Extra foreign keys are required.
  - A_id is consistent with the value you would find after a join with table B.

RISKS OF DENORMALIZATION

- Disk space: This is expected, as we'll have duplicate data.
- Data anomalies: We have to be very aware of the fact that data now can be changed in more than one place. We must adjust every piece of duplicate data accordingly.
- Documentation: We must properly document every denormalization rule that we have applied. If we modify database design later, we'll have to look at all our exceptions and take them into consideration once again.

END OF SLIDES