Errata and Updates For Database System Concepts, 6th Edition Silberschatz, Korth, and Sudarshan

Last updated: October 4, 2014

We list below errors, clarifications, and recent updates. NOTE: The third printing of the book will be released in the US market in late 2014; all errata shown below have been fixed in this printing. Some of these errata have been fixed in the 2nd printing of the book. These are tagged as "(1st *pr.*)".

If you own an international edition, note that these editions follow a different correction schedule, so your copy may still have errata that have been fixed in the US edition.

Check your copy for the errata noted here, and ignore those that have been fixed in your copy.

Errata for Part 1: Relational Databases, Chapters 1 to 6

CHAPTER 2

1. Page 47. Figure 2.8: Delete the line between *section.time_slot_id* and *time_slot.time_slot_id.* Although there is a referential integrity constraint from *section.time_slot_id* to *time_slot.time_slot_id* it is not a foreign key constraint since *time_slot.time_slot_id* is not a primary key of *time_slot.* The schema diagram notation does not currently allow us to specify referential integrity constraints that are not foreign key constraints.

2. (1st *pr.*) Page 54, Exercise 2.9: Add the following to the first line: "Assume that branch names and customer names uniquely identify branches and customers, but loans and accounts can be associated with more than one customer."

CHAPTER 3

1. (1st *pr.*) Page 61, at the end of the 3rd para (just before the bullet for "not null"): add the sentence: "(Some databases such as MySQL require an alternative syntax, of the form "**foreign key** (*dept _name*) **references** *department*(*dept_name*)", where the referenced attributes in the referenced table are explicitly listed.)"
   (Reported by: Cam Hong Tran)

2. (1st *pr.*) Page 69, Figure 3.6: The department name and salary of all instructors except Einstein are wrong (and have been copied incorrectly from Einstein's values). The correct values can be found in the *instructor* relation shown in Figure 2.1, Page 40, which should be: (Comp. Sci., 65000) for Srinivasan, (Finance, 90000) for Wu, and (Music, 40000) for Mozart; the same value should occur in all rows for that instructor.

   Also, the ID of the instructor of FIN-201 has been shown as 10101 in multiple lines in the table, in the column preceding FIN-201; the ID should be 12121.
   (Reported by: Celine Kuttler)

[1]**Errors reported by**: Deepak Aggrawal, G. Aishwarya, Jameel Al-Aziz, Scot Anderson, Yahui Chang, David Chiu, Jonghoon Chun, Matt Cremeens, Dona Dungan, Pham Nguyen Duc Duong, Helena Galhardas, Eric Gossett, Ravindra Guravannavar, Leon Ho, Pranav Jain, Jevitha K. P., Cheqing Jin, Minhua Kang, Celine Kuttler, Daniel Sadoc Menasche, Thimas Nielsen, Linda Null, Judi Paige, Donnie Pinkston, Subhasish Saha, Vemireddy Satish, Stan Thomas, Cam Hong Tran, Duc Tran, Daniel Vieira, and a few others. Their help, and in particular that of Daniel Sadoc Menasche, is deeply appreciated. Also thanks to Juha Haaga for suggestions on improvements for future editions.

3. (1st *pr.*) Page 73, Footnote 3: Replace the current footnote which states "As a consequence, it is not possible to use attribute names containing the original relation names, for instance *instructor.name*or *teaches.course_id*, to refer to attributes in the natural join result; we can, however, use attribute names such as *name* and *course_id*, without the relation names."
   →

   > As a consequence, it may not be possible in some systems to use attribute names containing the original relation names, for instance *instructor.name*or *teaches.course_id*, to refer to attributes in the natural join result. While some systems allow it, others don't, and some allow it for all attributes except the join attributes (that is, those that appear in both relation schemas. We can, however, use attribute names such as *name* and *course_id*, without the relation names.

4. (1st *pr.*) Page 80, paragraph after first query: "... Fall 2010 ..." → "... Spring 2010 ..."

5. Page 81, first line: "... Fall 2010 ..." → "... Spring 2010 ..."
   (Reported by: Jameel Al-Aziz)

6. (1st *pr.*) Page 85, para 2: "The average balance is ..." → "The average salary is ...".
   (Reported by: Daniel Vieira)

7. (1st *pr.*) Page 88, Figure 3.17: in the second column header: "*avg(avg_salary)*" → "*avg_salary*"

8. (1st *pr.*) Page 94, top of page: in "**select distinct** *S.ID, S.name*", the use of **distinct** is not required, although it is not incorrect.
   (Reported by: Jonghoon Chun)

9. (1st *pr.*) Page 95, In the query at the top of the page:
   "**where** 1 <= (**select count**(*R.course_id*) ..."
   →
   "**where** 1 >= (**select count**(*R.course_id*) ..."
   (Reported by: Duc Tran)

10. (1st *pr.*) Page 96, Para 3:
    "However, some SQL implementations, notably Oracle, do not support renaming of the result relation in the **from** clause."
    →
    "Note that some SQL implementations require that each subquery result relation be given a name, even if the name is never referenced; Oracle allows a subquery result relation to be given a name (with the keyword **as** omitted) but does not allow renaming of attributes of the relation."

    Oracle does allow renaming of result relations (although it does not require it), but as in other kinds of renaming in Oracle, the keyword **as** should be omitted.

11. Page 99, last para, line 3: "all tuples that fail the test" → "all tuples that pass the test".
    (Reported by: Pranav Jain)

12. (1st *pr.*) Page 101, third SQL query:
    select *student*
    from *student*

    →

    select *ID*
    from *student*

13. (1st *pr.*) Page 105, Practice Exercise 3.1, Parts e, f, g: "Autumn" → "Fall"

14. (1st *pr.*) Page 109, Exercise 3.12, Part b: "Autumn" → "Fall"

CHAPTER 4

1. (1st *pr.*) Page 126, Figure 4.7: In the last row of the *department* relation, change 'Painter' to 'Taylor'.
2. (1st *pr.*) Page 130, Section 4.4.3: "...form a candidate key..." $\rightarrow$ "...form a superkey...", and "However, candidate key attributes..." $\rightarrow$ "However attributes declared as unique..."
   (Reported by: Cheqing Jin)
3. (1st *pr.*) Page 131, para 5: After the 1st sentence of this paragraph (which begins "By default, in SQL, ..."), add the sentence:

   For example, the foreign key declaration for the *course* relation can be specified as:
   foreign key (*dept_name*) **references** *department*(*dept_name*)
4. (1st *pr.*) Page 148, Para 1: "*branch_name* of the *branch* relation" $\rightarrow$ "*dept_name* of the *department* relation"
   (Reported by: Daniel Sadoc Menasche)
5. (1st *pr.*) Page 155, Question 4.11: "Music" $\rightarrow$ "Taylor". (We need a building name, not a department name here.)

CHAPTER 5

1. (1st *pr.*) Page 163, Para 1 (Java expression):

   ```
   " ' + dept_name + " ', " ' balance + ")"
   ```

   $\rightarrow$

   ```
   " "' + dept_name + " ', " + salary + ")"
   ```

   (Reported by: Daniel Sadoc Menasche)
2. (1st *pr.*) Page 167, Figure 5.4, in the printf statement: "depthname" $\rightarrow$ "deptname"
3. (1st *pr.*) Page 174, first query of Section 5.2.1: "**from** *instructor*" $\rightarrow$ "**from** *department*".
4. (1st *pr.*) Page 175, Figure 5.6, Line 1: "*instructors_of*" $\rightarrow$ " *instructor_of*"
5. (1st *pr.*) Page 184, Figure 5.10: "**update on** *takes*" $\rightarrow$ "**update of** *takes*"
6. (1st *pr.*) Page 185, Figure 5.11, first line: "amount" $\rightarrow$ "level".
7. (1st *pr.*) Page 191 Fig 5.15:
   (a) Change all 5 occurrences of *c_prereq* $\rightarrow$ *rec_prereq*;
   (b) "**select** *prereq.prereq_id*, *c_prereq.course_id*"
   $\rightarrow$
   "**select** *rec_prereq.course_id*, *prereq.prereq_id*"
8. (1st *pr.*) Page 194, Section 5.5.1: "**select** *ID*, *GPA*)" $\rightarrow$ "**select** *ID*, *GPA*"
9. (1st *pr.*) Page 200, Figure 5.18, in the cell for "white" "dress": "8" $\rightarrow$ "5"
10. (1st *pr.*) Page 212, Exercise 5.8:
    "for each owner of the account, check if the owner has any remaining accounts, and if she does not, delete her from the *depositor* relation."
    $\rightarrow$
    "for each depositor of the account, check if the depositor has any remaining accounts, and if she does not, delete her from the *customer* relation."

Errata and Updates For Database System Concepts, 6^{th} Edition Silberschatz, Korth, and Sudarshan

Last updated: October 4, 2014

We list below errors, clarifications, and recent updates. NOTE: The third printing of the book will be released in the US market in late 2014; all errata shown below have been fixed in this printing. Some of these errata have been fixed in the 2nd printing of the book. These are tagged as "(1^{st} *pr.*)".

If you own an international edition, note that these editions follow a different correction schedule, so your copy may still have errata that have been fixed in the US edition.

Check your copy for the errata noted here, and ignore those that have been fixed in your copy.

Errata for Part 2: Database Design, Chapters 7 to 9

CHAPTER 7

1. (1^{st} *pr.*) Page 284, paras 2 and 5: Change the two occurrences of "*middle_name*" to "*middle_initial*".
2. (1^{st} *pr.*) Page 289, Line 2: "every entity a in the entity set B" $\rightarrow$ "every entity a in the entity set A"
   (Reported by: Jevitha K. P.)
3. (1^{st} *pr.*) Page 319, Exercise 7.22: "... customers (who ship items) and customers (who receive items); "
   $\rightarrow$
   "... customers who ship items and customers who receive items;"

CHAPTER 8

1. (1^{st} *pr.*) Page 331, 2nd bullet: "in every legal instance of $r(R)$ it satisfies the functional dependency."
   $\rightarrow$
   "every legal instance of $r(R)$ satisfies the functional dependency."
   (Reported by: Daniel Sadoc Menasche)
2. (1^{st} *pr.*) Page 355, last paragraph before Section 8.6: "Thus, in case we are not able to get a dependency-preserving BCNF decomposition, it is generally preferable to opt for BCNF"
   $\rightarrow$
   "In summary, even if we are not able to get a dependency-preserving BCNF decomposition, it is still preferable to opt for BCNF"
   (Reported by: Daniel Sadoc Menasche)
3. (1^{st} *pr.*) Page 360, first bullet, 3rd line: " This decomposition is lossless of R" $\rightarrow$
   " This decomposition of R is lossless".
   (Reported by: Daniel Sadoc Menasche)

[1]**Errors reported by**: Deepak Aggrawal, G. Aishwarya, Jameel Al-Aziz, Scot Anderson, Yahui Chang, David Chiu, Jonghoon Chun, Matt Cremeens, Dona Dungan, Pham Nguyen Duc Duong, Helena Galhardas, Eric Gossett, Ravindra Guravannavar, Leon Ho, Pranav Jain, Jevitha K. P., Cheqing Jin, Minhua Kang, Celine Kuttler, Daniel Sadoc Menasche, Thimas Nielsen, Linda Null, Judi Paige, Donnie Pinkston, Subhasish Saha, Vemireddy Satish, Stan Thomas, Cam Hong Tran, Duc Tran, Daniel Vieira, and a few others. Their help, and in particular that of Daniel Sadoc Menasche, is deeply appreciated. Also thanks to Juha Haaga for suggestions on improvements for future editions.

4. Page 367: “bank database” $\rightarrow$ “university database”
   (Reported by: Thomas Nielsen)

CHAPTER 9

1. (1st *pr.*) Page 383, last para, lines 1 and 2: “... extracts values of the parameter’s type and number ...” $\rightarrow$
   “... extracts values of the parameters persontype and name ...”.
   (Reported by: Daniel Sadoc Menasche)
2. (1st *pr.*) Page 384, Figure 9.08: “String number” $\rightarrow$“String name”.
   (Reported by: Daniel Sadoc Menasche)

Errata and Updates For Database System Concepts, 6^{th} Edition Silberschatz, Korth, and Sudarshan

Last updated: October 4, 2014

We list below errors, clarifications, and recent updates. NOTE: The third printing of the book will be released in the US market in late 2014; all errata shown below have been fixed in this printing. Some of these errata have been fixed in the 2nd printing of the book. These are tagged as "(1^{st} *pr.*)".

If you own an international edition, note that these editions follow a different correction schedule, so your copy may still have errata that have been fixed in the US edition.

Check your copy for the errata noted here, and ignore those that have been fixed in your copy.

Errata for Part 3: Data Storage and Querying, Chapters 10 to 13

CHAPTER 10

1. (1^{st} *pr.*) Page 445, RAID level 3: The sentences "Since reads and writes of a byte are spread out over multiple disks, ... since every disk has to participate in every I/O request." are technically correct, but somewhat confusing. Replace these sentences by:

   "Since reads and writes of a byte are spread out over multiple disks, with N-way striping of data, the transfer rate is N times faster than a RAID level 1 organization using N-way striping, for reading or writing a single block. However, RAID level 3 has no transfer rate benefit for writes of multiple blocks. Further, RAID level 3 has a higher access time, and supports a lower number of I/O operations per second, since every disk has to participate in every I/O request."

CHAPTER 11

1. (1^{st} *pr.*) Page 486, first para of Section 11.3.1: After $K_i < K_j$ add "(we assume for now that there are no duplicate key values)."
2. (1^{st} *pr.*) Page 487, 5th para: Delete the sentence: "We have shown instructor names abbreviated to 3 characters in order to depict the tree clearly; in reality, the tree nodes would contain the full names."
3. (1^{st} *pr.*) Page 487, last para: Delete the sentence: "As before, we have abbreviated instructor names only for clarity of presentation."
4. (1^{st} *pr.*) Page 488, Figure 11.9: The line from the left-most leaf to the record for Crick should come from the space between Califieri and Crick, not from the space after Crick.
   (Reported by: Minhua Kang)
5. (1^{st} *pr.*) Page 488, first para of Section 11.3.2: "Suppose that we wish to find records with a search-key value of V. Figure 11.11 presents pseudocode for a function `find()` to carry out this task." $\rightarrow$
   " Suppose that we wish to find a record with a search-key value of V. Figure 11.11 presents pseudocode for a function `find()` to carry out this task, assuming there are no duplicates."

[1]**Errors reported by**: Deepak Aggrawal, G. Aishwarya, Jameel Al-Aziz, Scot Anderson, Yahui Chang, David Chiu, Jonghoon Chun, Matt Cremeens, Dona Dungan, Pham Nguyen Duc Duong, Helena Galhardas, Eric Gossett, Ravindra Guravannavar, Leon Ho, Pranav Jain, Jevitha K. P., Cheqing Jin, Minhua Kang, Celine Kuttler, Daniel Sadoc Menasche, Thimas Nielsen, Linda Null, Judi Paige, Donnie Pinkston, Subhasish Saha, Vemireddy Satish, Stan Thomas, Cam Hong Tran, Duc Tran, Daniel Vieira, and a few others. Their help, and in particular that of Daniel Sadoc Menasche, is deeply appreciated. Also thanks to Juha Haaga for suggestions on improvements for future editions.

6. (1st *pr.*) Page 489, Figure 11.11, function find(): Change
   "/* Returns leaf node C and index i such that $C.P_i$ points to first record with search key value V */
   $\rightarrow$
   "/* Assumes no duplicate keys, and returns leaf node C and index i such that $C.P_i$ points to record with search key value V, if such a record exists */"

7. (1st *pr.*) Page 489, Figure 11.11, function printAll(): "Set (L, i) = find(V);"
   $\rightarrow$ "Set (L, i) = findFirst(V);"

8. (1st *pr.*) Page 490, Para 2: Replace this entire paragraph starting with "If there is at most one record .." with the following new paragraph:

   "The `find` function of Figure 11.11 needs to be modified to handle duplicates keys. With duplicate keys, for both leaf and internal nodes, if $i < j$, then $K_i < K_j$ may not hold, but certainly $K_i \leq K_j$ holds. Further, records in the subtree pointed to by P_i may contain values that are less than or equal to K_i (to understand why, consider two adjacent leaf nodes pointed to by P_i and P_{i+1} that both contain a duplicate key value v, in which case $K_i = v$). To fix this, we must modify the loop in the `find` function to set $C = C.P_i$, even if $V = C.K_i$. Further, the leaf node C reached thereby may contain only search keys less than V (even if V does exist in the tree); in this case the `find` procedure must set C = right sibling C, and recheck if C contains V. The modified `find` procedure, which we call `findFirst`, returns the first occurrence of value V in the tree."

9. (1st *pr.*) Page 490, Para 3, Line 2:
   "The procedure first steps through the remaining keys in the node L, to find other records with search-key value V."
   $\rightarrow$
   "The `printAll` procedure calls `findFirst` to find the node L with the first occurrence of V, and then steps through the remaining keys in the node L, to find other records with search-key value V."

10. Page 490, Para 4: Change:
    "To execute such queries, we can create a procedure `printRange`(L, U), whose body is the same as `printAll` except for these differences: `printRange` calls `find`(L), instead of `find`(V), and then steps through records as in procedure `printAll`, but with the stopping condition being that $L.K_i > U$, instead of $L.K_i > V$."
    $\rightarrow$
    "To execute such queries, we can create a procedure `printRange`(lb, ub), which does the following: it first traverses to a leaf in a manner similar to `find`(lb); the leaf may or may not actually contain value lb. It then steps through records in a manner similar to `printall`, but only returns records with key values $L.K_i$ s.t. $lb \leq L.K_i \leq ub$, and stops when $L.K_i > ub$."
    (Reported by: Deepak Aggrawal)

11. Page 494, Figure 11.15, procedure insert_in_leaf: Change
    "Let K_i be the highest value in L that is less than K
    $\rightarrow$
    "Let K_i be the highest value in L that is less than or equal to K

    and also change
    "Insert P, K into L just after $T.K_i$"
    $\rightarrow$
    "Insert P, K into L just after $L.K_i$"
    (Reported by: Donnie Pinkston)

12. Page 494, Figure 11.15, procedure insert_in_parent: Change:

    Copy $T.P_1 \ldots T.P_{\lceil n/2 \rceil}$ into P

Let $K'' = T.K_{\lceil n/2 \rceil}$
Copy $T.P_{\lceil n/2 \rceil +1} \ldots T.P_{n+1}$ into P'

$\rightarrow$

Copy $T.P_1 \ldots T.P_{\lceil (n+1)/2 \rceil}$ into P
Let $K'' = T.K_{\lceil (n+1)/2 \rceil}$
Copy $T.P_{\lceil (n+1)/2 \rceil +1} \ldots T.P_{n+1}$ into P'

NOTE: The change does not affect correctness, both versions are correct (i.e. it does not matter whether we take $\lceil n/2 \rceil$ or $\lceil (n+1)/2 \rceil$), but this change was done to ensure the procedure is consistent with the examples in the book.
(Reported by: Helena Galhardas)

13. (1st *pr.*) Page 496, line 3: "... the leaf node containg "Mozart"..." $\rightarrow$ "... the leaf node containg "Gold"...".
(Reported by: Jayvant Anantpur)

14. Page 504, last 2 paragraphs:
"11.13" $\rightarrow$ "11.09" in two places, and
"... "Califieri", "Einstein", "Gold", ... " $\rightarrow$"... , "Einstein", "Gold", ... "
(Reported by: Deepak Aggrawal)

15. Page 505, Caption of Figure 11.21: "11.13" $\rightarrow$ "11.09"
(Reported by: Deepak Aggrawal)

16. (1st *pr.*) Page 505, Figure 11.21: "... and soon for other records ..." $\rightarrow$
"... and so on for other records ...".

17. (1st *pr.*) Page 513, Para 2:
"The form of hash structure that we have just described is sometimes referred to as closed hashing. Under an alternative approach called open hashing . . ."

$\rightarrow$

"The form of hash structure that we have just described is called closed ad- dressing (or, less commonly, closed hashing). Under an alternative approach called open addressing (or, less commonly, open hashing) . . . "

Also in the same para: change all occurrences of "open hashing" $\rightarrow$ "open addressing" and "closed hashing" $\rightarrow$ "closed addressing'.

Why this change?: The form of hashing which we refer to as **closed hashing** is referred to more commonly as **closed addressing**, while the form of hashing we refer to as **open hashing** is referred to more commonly as **open addressing**.

But many sources also use the term "closed hashing" synonymously with "open addressing", and "open hashing" synonymously with "closed addressing", which is the exact opposite of our notation. To avoid this confusion, we suggest using the term closed addressing and open addressing, instead of closed hashing and open hashing.
(Reported by: Donnie Pinkston)

18. (1st *pr.*) Page 519, Figure 11.29: Change salary value of Srinivasan from 90000 $\rightarrow$ 65000.

19. Page 521, Figure 11.33: The value in the box above the top bucket (just above 15151) should be 1, not 2.
(Reported by: Vemireddy Satish)

20. (1st *pr.*) Page 533, Question 11.11: "Outline the steps in ..." $\rightarrow$ "Assuming the availability of the above bitmap index on *salary*, and a bitmap index on *dept_name*, outline the steps in ...".

21. Page 534, Exercise 11.14 (b) iii: "... a n-page block" $\rightarrow$ "... an n-block unit".

22. (1^{st} *pr.*) Page 534, exercise 11.19: "closed and open hashing" $\rightarrow$ "closed and open addressing".

CHAPTER 12

1. (1^{st} *pr.*) Page 543, Figure 12.3: In row A3, "$h_i * (t_T + t_S) + b * t_T$" $\rightarrow$ "$h_i * (t_T + t_S) + t_S + b * t_T$"

2. (1^{st} *pr.*) Page 543, Figure 12.3: In row A5, "$h_i * (t_T + t_S) + b * t_T$" $\rightarrow$ "$h_i * (t_T + t_S) + t_S + b * t_T$"
(Reported by: Daniel Sadoc Menasche, G. Aishwarya and Subhasish Saha)

3. Page 544, bullet for A5: "first tuple in the file that has a value of $A = v$" $\rightarrow$ "first tuple in the file that has a value of $A \geq v$".

4. (1^{st} *pr.*) Page 549, paragraphs 1 and 2: Some of the formulae here assume runs are read in one block at a time, and others assume that runs are read in b_b blocks at a time, to reduce the number of seeks. To restore consistency, make the following changes:

   (a) (1^{st} *pr.*) Replace the contents of Paragraph 1 starting from: "Since the number of runs decreases bny a factor of $M - 1$ in each merge pass ..." till the end of the paragraph by:

   > During the merge pass, reading in each run one block at a time leads to a large number of seeks; to reduce the number of seeks, a larger number of blocks, denoted b_b, are read or written at a time, requiring b_b buffer blocks to be allocated to each input run and to the output run. Then, $\lfloor M/b_b \rfloor - 1$ runs can be merged in each merge pass, decreasing the number of runs by a factor of $\lfloor M/b_b \rfloor - 1$. The total number of merge passes required is $\lceil \log_{\lfloor M/b_b \rfloor - 1}(b_r/M) \rceil$. Each of these passes reads every block of the relation once and writes it out once, with two exceptions. First, the final pass can produce the sorted output without writing its result to disk. Second, there may be runs that are not read in or written out during a pass—for example, if there are $\lfloor M/b_b \rfloor$ runs to be merged in a pass, $\lfloor M/b_b \rfloor - 1$ are read in and merged, and one run is not accessed during the pass. Ignoring the (relatively small) savings due to the latter effect, the total number of block transfers for external sorting of the relation is:
   >
   > $$b_r(2\lceil \log_{\lfloor M/b_b \rfloor - 1}(b_r/M) \rceil + 1)$$
   >
   > Applying this equation to the example in Figure 12.4, with b_b set to 1, we get a total of $12 * (4 + 1) = 60$ block transfers, as you can verify from the figure. Note that the above numbers do not include the cost of writing out the final result.

   (b) (1^{st} *pr.*) Page 549: Replace the contents of Paragraph 2, starting from "During the merge phase ..." till the end of the paragraph by:

   > Each merge pass the requires around $\lceil b_r/b_b \rceil$ seeks for reading data.[4] Although the output is written sequentially, if it is on the same disk as the input runs the head may have moved away between writes of consecutive blocks. Thus we would have to add a total of $2\lceil b_r/b_b \rceil$ seeks for each merge pass, except the final pass (since we assume the final result is not written back to disk).
   >
   > $$2\lceil b_r/M \rceil + \lceil b_r/b_b \rceil (2\lceil \log_{\lfloor M/b_b \rfloor - 1}(b_r/M) \rceil - 1)$$
   >
   > Applying this equation to the example in Figure 12.4, we get a total of $8 + 12 * (2 * 2 - 1) = 44$ disk seeks if we set the number of buffer blocks per run, b_b to 1.

   (Reported by: Leon Ho)

5. Page 556, bullet 2: "1164" $\rightarrow$ "1168" and "1264" $\rightarrow$ "1268".
(Reported by: Deepak Aggrawal)

6. (1^{st} *pr.*) Page 561, Section 12.5.5.4, paragraph following the 2nd bullet: For the case with recursive partitioning, some formulae assume data is read in b_b blocks at a time, while others do not. For consistency, the following changes are required:

(a) Replace the text in this paragraph starting from "Each pass reduces the size of each of the ..." by:

> Again we assume that b_b blocks are allocated for buffering each partition. Each pass then reduces the size of each of the partitions by an expected factor of $\lfloor M/b_b \rfloor - 1$; and passes are repeated until each partition is of size at most M blocks. The expected number of passes required for partitioning s is therefore $\lceil \log_{\lfloor M/b_b \rfloor - 1}(b_s/M) \rceil$.

(b) Replace the bullet immediately after the paragraph, starting with "Since, in each pass, every block ..." by:

> Since, in each pass, every block of s is read in and written out, the total block transfers for partitioning of s is $2b_s \lceil \log_{\lfloor M/b_b \rfloor - 1}(b_s/M) \rceil$. The number of passes for partitioning of r is the same as the number of passes for partitioning of s, therefore the join is estimated to require:
>
> $$2(b_r + b_s)\lceil \log_{\lfloor M/b_b \rfloor - 1}(b_s/M) \rceil + b_r + b_s$$
>
> block transfers.

(c) In the following bullet:
"$2(\lceil b_r/b_b \rceil + \lceil b_s/b_b \rceil)\lceil \log_{M-1}(b_s) - 1 \rceil$"
$\rightarrow$
"$2(\lceil b_r/b_b \rceil + \lceil b_s/b_b \rceil)\lceil \log_{\lfloor M/b_b \rfloor - 1}(b_s/M) \rceil$"

7. (1^{st} *pr.*) Page 562, first para: "There is enough memory to allocate 3 buffers for the input and each of the 5 outputs during partitioning ..."
$\rightarrow$
"There is enough memory to allocate 3 buffers for the input and each of the 5 outputs during partitioning (that is, $b_b = 3$) ..."

8. (1^{st} *pr.*) Page 566, Section 12.6.5, 2nd para: "*branch_name*" $\rightarrow$ "*dept_name*".
(Reported by: Daniel Sadoc Menasche)

9. (1^{st} *pr.*) Page 572, para 1: "Thus, the join of r_1 with s_0, and s_0 with r_1, ..." $\rightarrow$ "Thus, the join of r_1 with s_0, and s_1 with r_0, ...".
(Reported by: G. Aishwarya and Subhasish Saha)

CHAPTER 13

1. Page 580, second line: "ten attributes" $\rightarrow$ "nine attributes"
(Reported by: Deepak Aggrawal)

2. (1^{st} *pr.*) Page 584, Rule 3: At the end of the rule, add the line: " where $L_1 \subseteq L_2 \subseteq \ldots \subseteq L_n$."
(Reported by: Daniel Sadoc Menasche)

3. (1^{st} *pr.*) Page 585, Rule 12: At the end of the rule: " The projection operation distributes over the union operation

$$\Pi_L(E_1 \cup E_2) = \;(\Pi_L(E_1)) \cup (\Pi_L(E_2))$$

"

add the condition: "provided E_1 and E_2 have the same schema."
(Reported by: G. Aishwarya and Subhasish Saha)

4. Page 588, query just before Section 13.2.3: Add an extra ")" just after the relation *teaches*.

5. Page 589 1st para of 13.2.4:
"If an expression, say E_i, of any subexpression e_i of E_i (which could, as a special case, be E_i itself) matches one side of an equivalence rule, the optimizer generates a new expression where e_i is transformed to match the other side of the rule."
$\rightarrow$

"If a subexpression e_j of any expression $E_i \in EQ$ (as a special case, e_j could be E_i itself) matches one side of an equivalence rule, the optimizer generates a copy E_k of E_i, in which e_j is transformed to match the other side of the rule, and adds E_k to EQ."
(Reported by: Deepak Aggrawal)

6. (1^{st} *pr.*) Page 595, Line 4: "$n(r)$" $\rightarrow$ "n_r"

7. (1^{st} *pr.*) Page 601-602: "We have already seen equivalence rules with aggregation operation, and equivalence rules can also be created for outer joins."
$\rightarrow$
"Equivalence rules can also be defined for the aggregation and outer join operations, as illustrated in Practice Exercises 13.1 and 13.2."
(Reported by: Ravindra Guravannavar)

Errata and Updates For Database System Concepts, 6^{th} Edition Silberschatz, Korth, and Sudarshan

Last updated: October 4, 2014

We list below errors, clarifications, and recent updates. NOTE: The third printing of the book will be released in the US market in late 2014; all errata shown below have been fixed in this printing. Some of these errata have been fixed in the 2nd printing of the book. These are tagged as "(1^{st} *pr.*)".

If you own an international edition, note that these editions follow a different correction schedule, so your copy may still have errata that have been fixed in the US edition.

Check your copy for the errata noted here, and ignore those that have been fixed in your copy.

Errata for Part 4: Transaction Management, Chapters 14 to 16

CHAPTER 14

1. (1^{st} *pr.*) Page 631, bullet item 2: "Information about the updates carried out by the transaction and written to disk is sufficient to enable ..."
   $\rightarrow$
   "Information about the updates carried out by the transaction is written to the disk, and such information is sufficient to enable ..."
   (Reported by: Ravindra Guravannavar)

2. (1^{st} *pr.*) Page 646, Para 2: "addition and subtraction are commutative." $\rightarrow$ "the increment and decrement operations are commutative".
   (Reported by: Ravindra Guravannavar)

3. (1^{st} *pr.*) Page 653, line 2: "Oracle and PostgreSQL implement .." $\rightarrow$ "Oracle and PostgreSQL versions prior to PostgreSQL 9.1 implement .." $\rightarrow$

CHAPTER 15

1. (1^{st} *pr.*) Page 664, Figure 15.4: "concurreny" $\rightarrow$ "concurrency"

2. (1^{st} *pr.*) Page 671, Figure 15.10: "1" $\rightarrow$"I" in the following places: 17 $\rightarrow$ I7, 123 $\rightarrow$ I23, 1912 $\rightarrow$ I912, 14 $\rightarrow$ I4, and 144 $\rightarrow$ I44.

3. (1^{st} *pr.*) Page 675, Section 15.2.1, in the para for "wait-die":
   "If T_{24} requests...", $\rightarrow$ "If T_{16} requests..."
   (Reported by: Ravindra Guravannavar)

4. (1^{st} *pr.*) Page 679, last para, line 5 and 8: Change both occurrences of "F_c" $\rightarrow$ "F_b". (This change is required for consistency with the following paragraph on page 680.)
   (Reported by: Linda Null)

[1]**Errors reported by**: Deepak Aggrawal, G. Aishwarya, Jameel Al-Aziz, Scot Anderson, Yahui Chang, David Chiu, Jonghoon Chun, Matt Cremeens, Dona Dungan, Pham Nguyen Duc Duong, Helena Galhardas, Eric Gossett, Ravindra Guravannavar, Leon Ho, Pranav Jain, Jevitha K. P., Cheqing Jin, Minhua Kang, Celine Kuttler, Daniel Sadoc Menasche, Thimas Nielsen, Linda Null, Judi Paige, Donnie Pinkston, Subhasish Saha, Vemireddy Satish, Stan Thomas, Cam Hong Tran, Duc Tran, Daniel Vieira, and a few others. Their help, and in particular that of Daniel Sadoc Menasche, is deeply appreciated. Also thanks to Juha Haaga for suggestions on improvements for future editions.

5. (1^{st} *pr.*) Page 696, line 3: "phantom phenomenon" → "phantom phenomenon, which is described later in Section 15.8.3"

6. (1^{st} *pr.*) (Update) Page 697, 1st para, last line: "In Oracle and PostgreSQL, the *serializable* isolation level offers only snapshot isolation."
   →
   "In Oracle and PostgreSQL versions prior to 9.1, the *serializable* isolation level offers only snapshot isolation. Since version 9.1, PostgreSQL implements a technique called "serializable snapshot isolation", which provides true serializability."

7. Page 697, 3rd bullet in Section 15.8.1: In " If I_i comes before I_j, T_i will have a logical error." change $T_i \rightarrow T_j$.
   (Reported by: Deepak Aggrawal)

8. (1^{st} *pr.*) (Update) Page 701, paragraph just prior to Section 15.9: "PostgreSQL (as of version 8.1) ..." → "PostgreSQL versions prior to 9.1 ..."

9. (1^{st} *pr.*) Page 704, last paragraph of Section 15.9, line 11: In the line "... regular transactions validation using version numbers is very useful for such ...", add a ";" between "transactions" and "validation".

10. (1^{st} *pr.*) Page 715, 5th line from bottom: "time," → "time."

CHAPTER 16

1. (1^{st} *pr.*) Page 724, Line 2: Change
   "replaces the content of the first block with the value of the second. This recovery procedure ..."
   →
   "can either replace the content of the first block with the value of the second, or replace the content of the second block with the value of the first. Either way, the recovery procedure ..."

2. (1^{st} *pr.*) Page 725: Last para: "The system then performs ..." → "The transaction then performs...".

3. Page 725: Last line: "final write to X" → "final write to x_i".
   (Reported by: Deepak Aggrawal)

4. (1^{st} *pr.*) Page 765, Exercise 16.20: "just after before" → "just before".

Errata and Updates For Database System Concepts, 6^{th} Edition Silberschatz, Korth, and Sudarshan

Last updated: October 4, 2014

We list below errors, clarifications, and recent updates. NOTE: The third printing of the book will be released in the US market in late 2014; all errata shown below have been fixed in this printing. Some of these errata have been fixed in the 2nd printing of the book. These are tagged as "(1^{st} *pr.*)".

If you own an international edition, note that these editions follow a different correction schedule, so your copy may still have errata that have been fixed in the US edition.

Check your copy for the errata noted here, and ignore those that have been fixed in your copy.

Errata for Parts 5 to 8, Chapters 17 to 26

CHAPTER 20

1. (1^{st} *pr.*) Page 897, definitions of Gini and Entropy: "$\sum_{i-1}^{k}$" $\rightarrow$ "$\sum_{i=1}^{k}$"
2. (1^{st} *pr.*) Page 898, definition of Information_content: "$\sum_{i-1}^{r}$" $\rightarrow$ "$\sum_{i=1}^{r}$"
   (Reported by: Daniel Sadogh Menasche)
3. Page 898, 2nd para: "All of this leads to a definition: The **best split** for an attribute is the one that gives the maximum **information gain ratio**, defined as: "
   $\rightarrow$
   "The **information gain ratio** of a split is then defined as:"
   Also add the following after the definition of Information_gain:

   "Classifiers usually choose the split that gives the maximum information gain ratio. However, for binary splits, many classifiers choose the split that gives the maximum information gain, instead of the maximum information gain ratio. This is because the number of partitions is fixed, and moreover if a split results in most values being in one partition, its information content is very low, which can result in a high information gain ratio even with a small information gain."
4. Page 898, para before the last para: "We then compute the information gain obtained by splitting at each value."
   $\rightarrow$
   "We then compute the desired metric (information gain or information gain ratio) obtained by splitting at each value."
5. Page 898, 8 lines from page bottom: " The best binary split for the attribute is the split that gives the maximum information gain."
   $\rightarrow$
   "The best split is the one that maximizes the desired metric."

[1] **Errors reported by**: Deepak Aggrawal, G. Aishwarya, Jameel Al-Aziz, Scot Anderson, Yahui Chang, David Chiu, Jonghoon Chun, Matt Cremeens, Dona Dungan, Pham Nguyen Duc Duong, Helena Galhardas, Eric Gossett, Ravindra Guravannavar, Leon Ho, Pranav Jain, Jevitha K. P., Cheqing Jin, Minhua Kang, Celine Kuttler, Daniel Sadoc Menasche, Thimas Nielsen, Linda Null, Judi Paige, Donnie Pinkston, Subhasish Saha, Vemireddy Satish, Stan Thomas, Cam Hong Tran, Duc Tran, Daniel Vieira, and a few others. Their help, and in particular that of Daniel Sadoc Menasche, is deeply appreciated. Also thanks to Juha Haaga for suggestions on improvements for future editions.

6. Page 899, para 1, line 3: "that results in the maximum information-gain ratio" $\rightarrow$ "that maximizes the information gain, or information gain ratio."
(Reported by: All above points related to best splits reported by Scot Anderson)

CHAPTER 21

1. (1st *pr.*) Page 918, line 1: "... where $n(d)$ denotes the number of terms in the document ..." $\rightarrow$ "... where $n(d)$ denotes the number of term occurrences in the document ..."

CHAPTER 23

1. (1st *pr.*) Page 987, Figure 23.5, 3rd line from the bottom: "coursr_id" $\rightarrow$ "course_id"
2. (1st *pr.*) Page 992, Figure 23.10: Delete the ">" symbol at the end of "IID ID #REQUIRED >"
3. (1st *pr.*) Page 995, Figure 23.12: <xs:complexType> and <xs:sequence> elements should be added to enclose the subelements of the course element, in the same way as the department and instructor elements.
4. Page 1007: "local:inst_dept_courses($i/IID)" $\rightarrow$ "local:dept_courses($i/IID)"
(Reported by: Stan Thomas)
5. (1st *pr.*) Page 1011, Section 23.6.2, first paragraph: "A tuple inserted in the *nodes* relation ..." $\rightarrow$ "A tuple is inserted in the *nodes* relation ...".
6. (1st *pr.*) Page 1014, Figure 23.15, line preceding last line: "<course>" $\rightarrow$ "</course>".

CHAPTER 24

1. (1st *pr.*) Page 1058, Exercise 24.2(c): "... assuming your database supports index-only plans ..." $\rightarrow$ "... assuming your database supports covering indices (that is, indices that store extra attributes, in addition to the search key, at the leaf nodes) and index-only plans ..."
2. Page 1063, Figure 25.1: In the 2nd and 5th row, replace the value in the *to* column by "*".
(Reported by: Thomas Nielsen, who pointed out that none of the rows has a "" value.)*