==========

Distributed database transparency features

DDBMS transparency features have the common property of allowing the end user to feel like the database's only user. In other words, the user believes that(s) he is working with centralized DBMS; all complexities of a distributed database are hidden, or transparent to the user.

ميزات شفافية قاعدة البيانات الموزعة

تتمتع ميزات شفافية DDBMS بخاصية مشتركة وهي السماح للمستخدم النهائي بالشعور بأنه المستخدم الوحيد لقاعدة البيانات. بمعنى آخر ، يعتقد المستخدم أنه (ق) يعمل مع نظم إدارة قواعد البيانات المركزية ؛ جميع تعقيدات قاعدة البيانات الموزعة مخفية أو شفافة للمستخدم.

The DDBMS transparency features are:ميزات الشفافية DDBMS هي:

1. **Distribution transparency, which** allows a distributed database to be treated as a single logical database. If a DDBMS exhibits distribution transparency , the user does not need to know:

1. شفافية التوزيع ، والتي تتيح التعامل مع قاعدة البيانات الموزعة كقاعدة بيانات منطقية واحدة. إذا أظهر DDBMS شفافية التوزيع ، فلن يحتاج المستخدم إلى معرفة:

- That the data are partitioned _ meaning the table's rows and columns are split vertically or horizontally and stored among multiple sites.
- That the data can be replicated at several sites.
- The data location.

- أن البيانات مقسمة _ بمعنى أن صفوف الجدول وأعمدته يتم تقسيمها رأسياً أو أفقيًا وتخزينها بين مواقع متعددة.
- أن البيانات يمكن تكرارها في عدة مواقع.
- موقع البيانات.

2. **Transaction transparency**, which allows a transaction to update data at more than one network site Transaction transparency ensures that the transaction will be either entirely completed or aborted. Thus, maintaining database integrity.
3. **Failure transparency**, which ensures that the system will continue to operate in the event of a node failure. Functions that were lost because of the failure will be picked up by another network node.
4. **Performance transparency,** which allows the system to perform as if it were a centralized DBMS. The system will not suffer any performance degradation due to its use on a network or due to the network's platform differences. Performance transparency also ensures that the system will find the most cost-effective path to access remote data.

2. شفافية المعاملة ، التي تسمح للمعاملة بتحديث البيانات في أكثر من موقع شبكة واحد ، تضمن شفافية المعاملة أن المعاملة ستكتمل بالكامل أو تم إحباطها. وبالتالي ، الحفاظ على سلامة قاعدة البيانات.
3. فشل الشفافية ، والذي يضمن استمرار النظام في العمل في حالة فشل العقدة. سيتم التقاط الوظائف التي تم فقدها بسبب الفشل بواسطة عقدة شبكة أخرى.
4. شفافية الأداء ، والتي تسمح للنظام بالعمل كما لو كان نظام DBMS مركزيًا. لن يعاني النظام من أي تدهور في الأداء بسبب استخدامه على الشبكة أو بسبب اختلافات النظام الأساسي للشبكة. تضمن شفافية الأداء أيضًا أن النظام سيجد المسار الأكثر فعالية من حيث التكلفة للوصول إلى البيانات البعيدة.

==========

5. **Heterogeneity transparency**, which allows the integration of several different local DBMSs (relational. network, and hierarchical) under a common, or global, schema. The DDBMS is responsible for translating the data requests from the global schema to the local DBMS schema.

5. شفافية عدم التجانس ، والتي تسمح بدمج العديد من نظم إدارة قواعد البيانات (DBMSs) المحلية المختلفة (العلائقية ، والشبكة ، والتسلسل الهرمي) في إطار مخطط مشترك أو عالمي. DDBMS مسؤول عن ترجمة طلبات البيانات من المخطط العام إلى مخطط DBMS المحلي.

DISTRIBUTION TRANSPARENCY شفافية التوزيع

The level of transparency supported by the DDBMS varies from system to system. **Three** levels of distribution transparency are recognized:

يختلف مستوى الشفافية الذي يدعمه DDBMS من نظام إلى آخر. يتم التعرف على ثلاثة مستويات من شفافية التوزيع:

1. **Fragmentation transparency** is the highest level of transparency. The end user or programmer does not need to know that a database is partitioned. Therefore, neither fragment names nor fragment locations are specified prior to data access.
2. **Location transparency** exists when the end user or programmer must specify the database fragment name but does not need to specify where those fragments are located.
3. **Local mapping transparency** exists when the end user or programmer must specify both the fragment names and their locations.

1. تجزئة الشفافية هي أعلى مستوى من الشفافية. لا يحتاج المستخدم النهائي أو المبرمج إلى معرفة أن قاعدة البيانات مقسمة. لذلك ، لم يتم تحديد أسماء الأجزاء أو مواقع التجزئة قبل الوصول إلى البيانات.
2. توجد شفافية الموقع عندما يتعين على المستخدم النهائي أو المبرمج تحديد اسم جزء قاعدة البيانات ولكن لا يحتاج إلى تحديد مكان تلك الأجزاء.
3. توجد شفافية الخرائط المحلية عندما يجب على المستخدم النهائي أو المبرمج تحديد كل من أسماء الأجزاء ومواقعها.

Transparency features are summarized in the following Table:

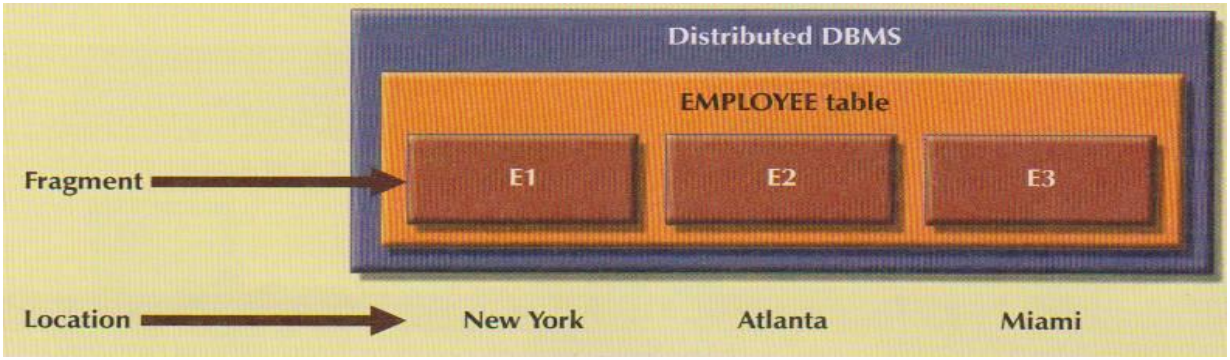
يتم تلخيص ميزات الشفافية في الجدول التالي:

FRAGMENT NAME?	LOCATION NAME?	THEN THE DBMS SUPPORTS	LEVEL OF DISTRIBUTON TRANSPARENCY
Yes	Yes	Local mapping	Low
Yes	No	Location transparency	Medium
No	No	Fragmentation transparency	High

there is no reference to a situation in which the fragment name is "No" and the location name is “Yes.” The reason for not including that scenario is simple: you cannot have a location name that fails to reference an existing fragment. See the following figure:

لا توجد إشارة إلى حالة يكون فيها اسم الجزء هو "لا" واسم الموقع هو "نعم". سبب عدم تضمين هذا السيناريو بسيط: لا يمكنك الحصول على اسم موقع يفشل في الإشارة إلى جزء موجود. انظر الشكل التالي:

==========



Now suppose the end user wants to list all employees with a date of birth prior to January 1, 1960. To focus on the transparency issues, also suppose the EMPLOYEE table is fragmented and each fragment is unique. Assume that no portion of the database is replicated at any other site on the network.

لنفترض الآن أن المستخدم النهائي يريد إدراج جميع الموظفين بتاريخ ميلاد قبل 1 يناير 1960. للتركيز على قضايا الشفافية ، افترض أيضًا أن جدول الموظف مجزأ وأن كل جزء فريد. افترض أنه لا يتم نسخ أي جزء من قاعدة البيانات في أي موقع آخر على الشبكة.

Case 1: The Database Supports Fragmentation Transparency

The query conforms to a non-distributed database query format; that is, it does not specify fragment names or locations. The query reads:

الحالة 1: قاعدة البيانات تدعم شفافية التجزئة

يتوافق الاستعلام مع تنسيق استعلام قاعدة البيانات غير الموزع ؛ أي أنه لا يحدد أسماء الأجزاء أو المواقع. يقرأ الاستعلام:

```
SELECT    *
FROM      EMPLOYEE
WHERE     EMP_DOB<'01-JAN-1960';
```

Case 2: The Database Supports Location Transparency

Fragment names must be specified in the query, but fragment location is not specified. The query reads:

```
SELECT    *
FROM      E1
WHERE     EMP_DOB < ‘Ol-JAN-1960’;
UNION
SELECT    *
FROM      E2
WHERE      EMP_DOB <‘Ol-JAN-1960’;
UNION
SELECT     *
FROM      E3
WHERE     EMP_DOB <‘Ol-JAN-1960’;
```

Case 3: The Database Supports Local Mapping Transparency

Both the fragment name and location must be specified in the query. Using pseudo-SQL:

الحالة 3: قاعدة البيانات تدعم شفافية الخرائط المحلية

يجب تحديد كل من اسم الجزء والموقع في الاستعلام. باستخدام pseudo-SQL:

==========

```
SELECT    *
FROM      El NODE NY
WHERE     EMP_DOB < ‘O1-JAN-1960;
UNION
SELECT    *
FROM      E2 NODE ATL
WHERE     EMP_DOB < O1-JAN-1960;
UNION
SELECT    *
FROM      E3 NODE MIA
WHERE     EMP_DOB <‘O1-JAN-1960;
```

Distribution transparency is supported by a **distributed data dictionary (DDD)**. or a **distributed data catalog(DDC)**. The DDC contains the description of the entire database as seen by the database administrator. The database description, known as the **distributed global schema**, is the common database schema used by local TPs to translate user requests into sub queries (remote requests) that will be processed by different DPs. The DDC is itself distributed and it is replicated at the network nodes. Therefore, the DDC must maintain consistency through updating at all sites.

يتم دعم شفافية التوزيع بواسطة قاموس البيانات الموزعة (DDD). أو كتالوج البيانات الموزع (DDC). يحتوي DDC على وصف قاعدة البيانات بأكملها كما يراها مسؤول قاعدة البيانات. وصف قاعدة البيانات ، المعروف باسم المخطط العام الموزع ، هو مخطط قاعدة البيانات الشائع المستخدم من قبل TPs المحلية لترجمة طلبات المستخدم إلى استعلامات فرعية (الطلبات البعيدة) التي ستتم معالجتها بواسطة موانئ دبي المختلفة. يتم توزيع DDC نفسه ويتم تكراره في عقد الشبكة. لذلك ، يجب أن تحافظ DDC على الاتساق من خلال التحديث في جميع المواقع.

Keep in mind that some of the current DDBMS implementations impose limitations on the level of transparency support. For instance, you might be able to distribute a database, but not a table, across multiple sites. Such a condition indicates that the DDBMS supports location transparency but not fragmentation transparency.

ضع في اعتبارك أن بعض تطبيقات DDBMS الحالية تفرض قيودًا على مستوى دعم الشفافية. على سبيل المثال ، قد تتمكن من توزيع قاعدة بيانات ، وليس جدول ، عبر مواقع متعددة. يشير هذا الشرط إلى أن DDBMS يدعم شفافية الموقع ولكن ليس شفافية التجزئة.

TRANSACTION TRANSPARENCY

Distributed database systems require complex mechanisms to manage transactions and to ensure the consistency and integrity. To understand how the transactions are managed, you should know the basic concepts governing ***remote requests***, ***remote transactions***, ***distributed transactions***, and ***distributed requests***.

شفافية المعاملات

تتطلب أنظمة قواعد البيانات الموزعة آليات معقدة لإدارة المعاملات ولضمان الاتساق والنزاهة. لفهم كيفية إدارة المعاملات ، يجب أن تعرف المفاهيم الأساسية التي تحكم الطلبات عن بُعد والمعاملات عن بُعد والمعاملات الموزعة والطلبات الموزعة.

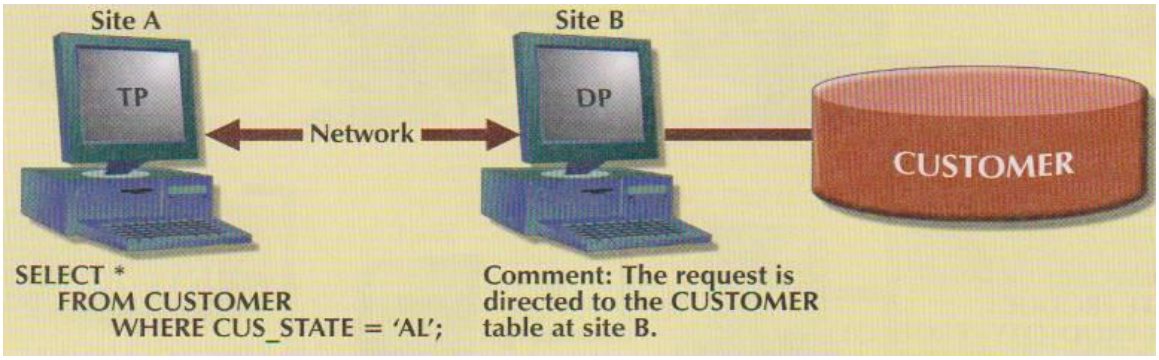
==========

DISTRIBUTED REQUESTS AND DISTRIBUTED TRANSACTIONS

A **remote request**, lets a single SQL statement access the data that are to be processed by a single remote database processor. In other words, the SQL statement (or request) can reference data at only one remote site.

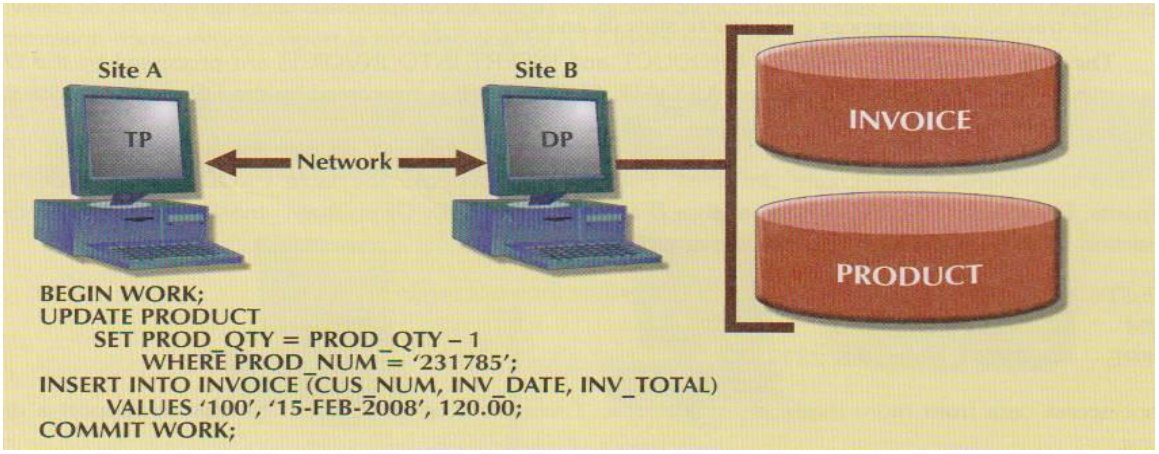
الطلبات الموزعة والمعاملات الموزعة

يتيح الطلب عن بُعد لعبارة SQL واحدة الوصول إلى البيانات التي ستتم معالجتها بواسطة معالج قاعدة بيانات واحد بعيد. بمعنى آخر ، يمكن أن تشير عبارة SQL (أو الطلب) إلى البيانات في موقع بعيد واحد فقط.



remote transaction, composed of several requests, accesses data at a single remote site.

المعاملات البعيدة ، المكونة من عدة طلبات ، تصل إلى البيانات في موقع واحد بعيد.



The following remote transaction features:

- The transaction updates the PRODUCT and INVOICE tables (located at site B).
- The remote transaction is sent to and executed at the remote site B.
- The transaction can reference only one remote DP.
- Each SQL statement (or request) can reference only one (the same) remote DP at a time, and the entire transaction can reference and be executed at only one remote DP.

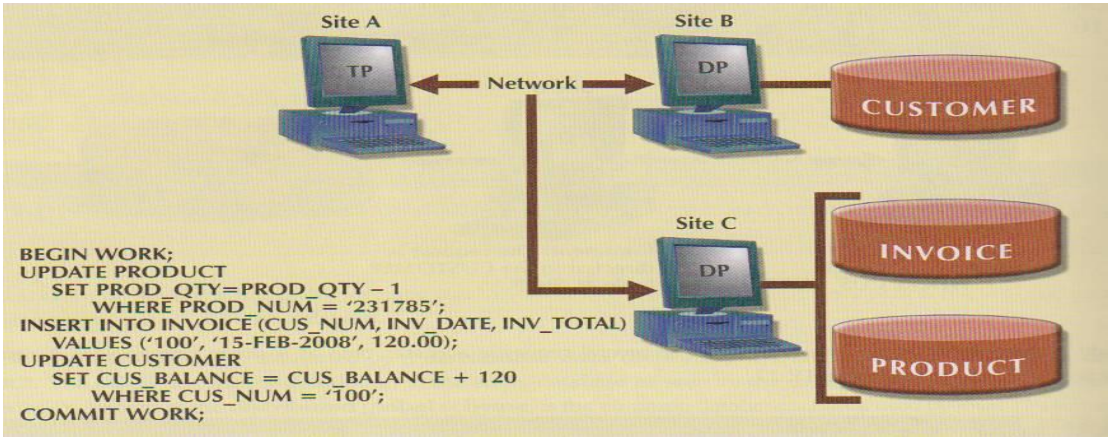
ميزات المعاملات عن بعد التالية:

- تقوم المعاملة بتحديث جدولي "المنتج" و "الفاتورة" (الموجودان في الموقع ب).
- يتم إرسال المعاملة عن بُعد وتنفيذها في الموقع البعيد B.
- يمكن أن تشير المعاملة إلى DP واحد فقط عن بعد.
- يمكن لكل عبارة (أو طلب) SQL أن تشير فقط إلى DP عن بعد واحد (نفس) في كل مرة ، ويمكن أن تشير المعاملة بأكملها إلى DP واحد فقط ويتم تنفيذها.

==========

A **distributed transaction** allows a transaction to reference several different local or remote DP sites. Although each single request can reference only one local or remote DP site, the transaction as a whole can reference DP sites because each request can reference a different site.

تسمح المعاملة الموزعة للمعاملة بالإشارة إلى العديد من مواقع DP المحلية أو البعيدة. على الرغم من أن كل طلب يمكن أن يشير إلى موقع DP محلي واحد أو بعيد ، إلا أن المعاملة ككل يمكن أن تشير إلى مواقع DP لأن كل طلب يمكن أن يشير إلى موقع مختلف.



Note the following features

- The transaction references two remote sites (B and C).
- The first two requests (UPDATE PRODUCT and INSERT INTO INVOICE) are processed by the DP a: remote site C, and the last request (UPDATE CUSTOMER) is processed by the DP at the remote site B.
- Each request can access only one remote site at a time.

لاحظ الميزات التالية

- تشير المعاملة إلى موقعين نائيين (B و C).
- الطلبان الأولين (تحديث المنتج وإدراج الفاتورة) هما
  تمت معالجته بواسطة DP a: الموقع البعيد C ، والطلب الأخير (UPDATE CUSTOMER)
  تتم معالجتها بواسطة DP في الموقع البعيد B.
- يمكن لكل طلب الوصول إلى موقع بعيد واحد فقط في كل مرة.

The third characteristic may create problems. For example, suppose the table PRODUCT is divided into two fragments, PRODI and PROD2, located at sites B and C, respectively.

الصفة الثالثة قد تخلق مشاكل. على سبيل المثال ، افترض أن الجدول PRODUCT مقسم إلى جزأين ، PRODI و PROD2 ، الموجودين في الموقعين B و C ، على التوالي.

==========

A **distributed request** lets a single SQL statement reference data located at several different local or remote DP sites. Because each request (SQL statement) can access data from more than one local or remote DP site, a transaction access several sites. The ability to execute a distributed request provides fully distributed database processing capabilities because of the ability to:

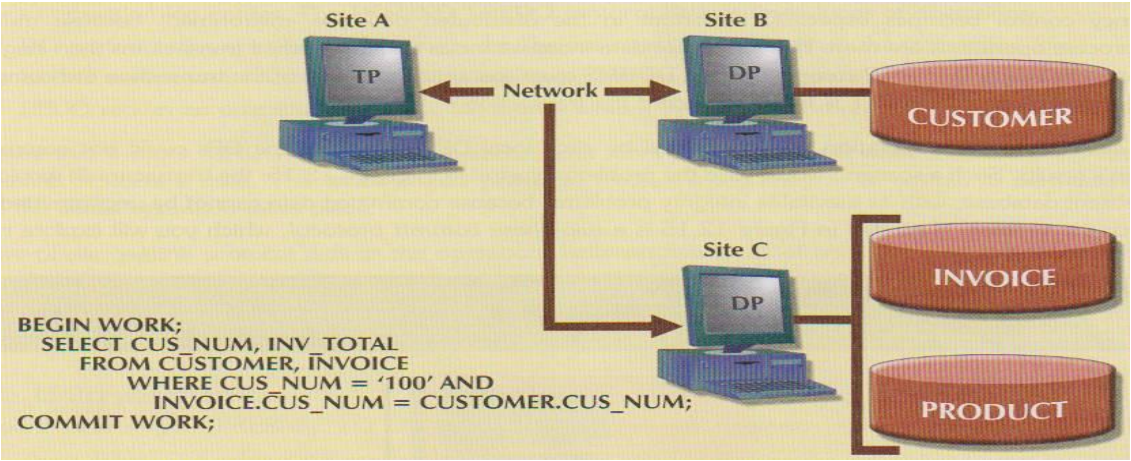
يتيح الطلب الموزع الحصول على بيانات مرجعية لعبارة SQL واحدة موجودة في العديد من مواقع DP المحلية أو البعيدة. نظرًا لأن كل طلب (عبارة SQL) يمكنه الوصول إلى البيانات من أكثر من موقع محلي أو بعيد DP ، فإن المعاملة تصل إلى عدة مواقع. توفر القدرة على تنفيذ الطلب الموزع إمكانات معالجة قاعدة البيانات الموزعة بالكامل بسبب القدرة على:

- Partition a database table into several fragments.
- Reference one or more of those fragments with only one request. In other words, there is fragmentation transparency.

- تقسيم جدول قاعدة البيانات إلى عدة أجزاء.
- قم بالإشارة إلى جزء أو أكثر من هذه الأجزاء بطلب واحد فقط. بعبارات أخرى، هناك شفافية التجزئة.

The location and partition of the data should be transparent to the end user. In the following Figure note that the transaction uses a single SELECT statement to reference two tables, CUSTOMER and INVOICE. The two tables are located at two different sites, B and C.

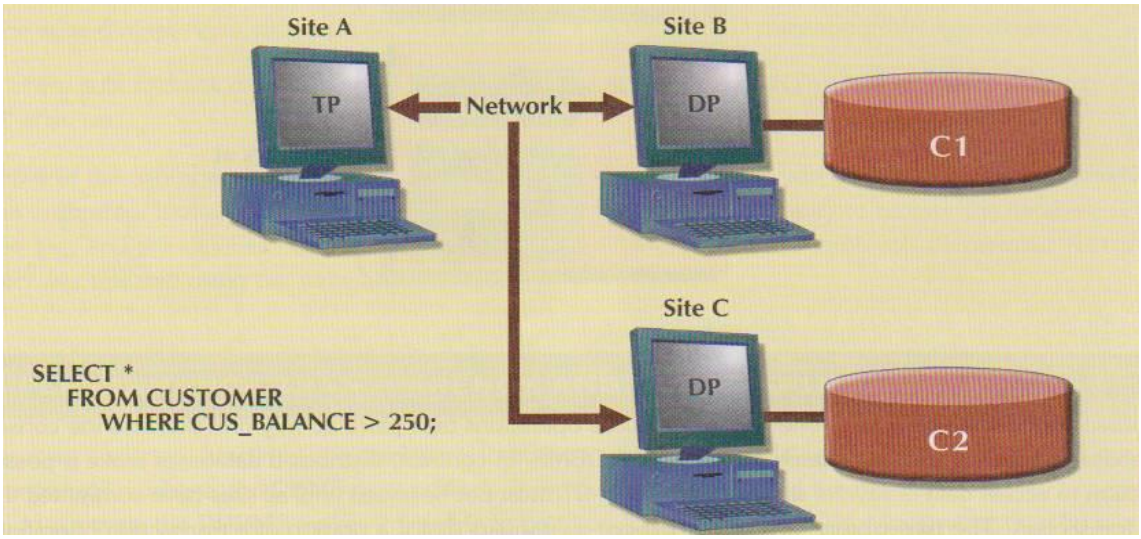
يجب أن يكون موقع البيانات وتقسيمها شفافين للمستخدم النهائي. في الشكل التالي ، لاحظ أن المعاملة تستخدم عبارة SELECT واحدة للإشارة إلى جدولين ، CUSTOMER و INVOICE. يوجد الجدولان في موقعين مختلفين ، B و C.



The distributed request feature also allows a single request to reference a physically partitioned table. For example. Suppose a CUSTOMER table is divided into two fragments, Cl and C2. located at sites B and C, respectively.

تتيح ميزة الطلب الموزع أيضًا طلبًا واحدًا للإشارة إلى جدول مقسم فعليًا. علي سبيل المثال. افترض أن جدول العميل مقسم إلى جزأين ، Cl و C2. تقع في الموقعين B و C على التوالي.

==========



Transaction

Is any action that reads from and/or writes to a database. A transaction may consist of a simple SELECT statement to generate a list of table contents; it may consist of series of INSERT statements to add rows to one or more tables.

عملية تجارية
هو أي إجراء يقرأ من قاعدة بيانات و / أو يكتب إليها. قد تتكون المعاملة من عبارة SELECT بسيطة لإنشاء قائمة بمحتويات الجدول ؛ قد تتكون من سلسلة من عبارات INSERT لإضافة صفوف إلى جدول واحد أو أكثر.

TRANSACTION PROPERTIES

Each individual transaction must display atomicity, consistency, isolation, and durability. Let's look briefly at each of the properties.

خصائص المعاملات
يجب أن تعرض كل معاملة فردية الذرية والاتساق والعزلة والمتانة. دعونا نلقي نظرة مختصرة على كل خاصية.

• **Atomicity** requires that all operations (SQL requests) of a transaction be completed; if not, the transaction is aborted. If a transaction T1 has four SQL requests, all four requests must be successfully completed; otherwise, the entire transaction is aborted. In other words. a transaction is treated as a single, indivisible, logical unit work.

• تتطلب الذرية إتمام جميع العمليات (طلبات SQL) للمعاملة ؛ إذا لم يكن كذلك ، يتم إحباط المعاملة. إذا كانت المعاملة T1 تحتوي على أربعة طلبات SQL ، فيجب إكمال الطلبات الأربعة بنجاح ؛ وإلا ، يتم إحباط المعاملة بالكامل. بعبارات أخرى. تعامل المعاملة كوحدة منطقية واحدة غير قابلة للتجزئة
الشغل.

• **Consistency** indicates the permanence of the database's consistent state (**consistent database state** is one in which all data integrity constraints are satisfied). A transaction takes a database from one consistent state to another consistent state. When a transaction is completed, the database must be in a consistent state; if any of the transaction parts violates an integrity constraint, the entire transaction is aborted.

• يشير الاتساق إلى استمرار حالة الاتساق لقاعدة البيانات (حالة قاعدة البيانات المتسقة هي حالة يتم فيها استيفاء جميع قيود تكامل البيانات). تأخذ المعاملة قاعدة بيانات من حالة متسقة إلى حالة متسقة أخرى. عند اكتمال المعاملة ، يجب أن تكون قاعدة البيانات في حالة متسقة ؛ إذا كان أي جزء من أجزاء المعاملة ينتهك قيود التكامل ، يتم إحباط المعاملة بالكامل.

==========

• **Isolation** means that the data used during the execution of a transaction cannot be used by a second transaction until the first one is completed. In other words, if a transaction T_1 is being executed and is using the data item X, that data item cannot be accessed by any other transaction (T_2 ... T_n) until T_1 ends. This property is particularly useful in multi-user database environments because several users can access and update the database at the same time.

• العزل يعني أن البيانات المستخدمة أثناء تنفيذ الصفقة لا يمكن استخدامها من قبل معاملة ثانية حتى يتم الانتهاء من المعاملة الأولى. بمعنى آخر ، إذا كانت المعاملة T1 قيد التنفيذ وتستخدم عنصر البيانات X ، فلا يمكن الوصول إلى عنصر البيانات هذا بأية معاملة أخرى (T2 ... T n) حتى تنتهي T1. هذه الخاصية مفيدة بشكل خاص في بيئات قواعد البيانات متعددة المستخدمين لأن العديد من المستخدمين يمكنهم الوصول إلى قاعدة البيانات وتحديثها في نفس الوقت.

• **Durability** ensures that once transaction changes are done (committed). they cannot be undone or lost, even in the event of a system failure.

• تضمن المتانة أنه بمجرد إجراء تغييرات المعاملة (الالتزام). لا يمكن التراجع عنها أو فقدها ، حتى في حالة فشل النظام.

• **Serializability** ensures that the schedule for the concurrent execution of the transactions yields consistent results. This property is important in multi-user and distributed databases, where multiple transactions are likely to be executed concurrently. Naturally, if only a single transaction is executed, Serializability is not an issue.

• تضمن القابلية للتسلسل أن يؤدي الجدول الزمني للتنفيذ المتزامن للمعاملات إلى نتائج متسقة. هذه الخاصية مهمة في قواعد البيانات متعددة المستخدمين والموزعة ، حيث من المحتمل أن يتم تنفيذ معاملات متعددة بشكل متزامن. بطبيعة الحال ، إذا تم تنفيذ معاملة واحدة فقط ، فإن قابلية التسلسل ليست مشكلة.

THE TRANSACTION LOG

A DBMS uses a transaction log to keep track of all transactions that update the database. The information stored in this log is used by the DBMS for a recovery requirement triggered by a ROLLBACK statement, a program abnormal termination, or a system failure such as a network discrepancy or a disk crash.

While the DBMS executes transactions that modify the database, it also automatically updates the transaction log. The transaction log stores:

سجل المعاملات

يستخدم DBMS سجل المعاملات لتتبع جميع المعاملات التي تقوم بتحديث قاعدة البيانات. يتم استخدام المعلومات المخزنة في هذا السجل بواسطة نظام إدارة قواعد البيانات (DBMS) لمتطلبات الاسترداد التي يتم تشغيلها بواسطة عبارة ROLLBACK أو إنهاء غير طبيعي للبرنامج أو فشل في النظام مثل تعارض في الشبكة أو تعطل القرص.

أثناء تنفيذ DBMS للمعاملات التي تعدل قاعدة البيانات ، يقوم أيضًا بتحديث سجل المعاملات تلقائيًا. يخزن سجل المعاملات:

• A record for the beginning of the transaction.
• For each transaction component (SQL statement):
  - The type of operation being performed (update, delete, insert).
  - The names of the objects affected by the transaction (the name of the table).
  - The “before’ and “after” values for the fields being updated.
  - Pointers to the previous and next transaction log entries for the same transaction.
• The ending (COMMIT) of the transaction.

• سجل لبداية المعاملة.
• بالنسبة لكل مكون معاملة (عبارة SQL):
- نوع العملية التي يتم إجراؤها (تحديث ، حذف ، إدخال).
- أسماء الأشياء المتأثرة بالمعاملة (اسم الجدول).
- قيمتا "قبل" و "بعد" للحقول التي يتم تحديثها.
- مؤشرات إلى إدخالات سجل المعاملات السابقة والتالية لنفس المعاملة.
• إنهاء (COMMIT) الصفقة.

TRL_ ID	TRX_ NUM	PREV PTR	NEXT PTR	OPERATION	TABLE	ROW ID	ATTRIBUTE	BEFORE VALUE	AFTER VALUE
341	101	Null	352	START	****Start Transaction				
352	101	341	363	UPDATE	PRODUCT	1558-QW1	PROD_QOH	25	23
363	101	352	365	UPDATE	CUSTOMER	10011	CUST_ BALANCE	525.75	615.73
365	101	363	Null	COMMIT	**** End of Transaction				

TRL_ID = Transaction log record ID
TRX_NUM = Transaction number
(*Note*: The transaction number is automatically assigned by the DBMS.)

PTR = Pointer to a transaction log record ID