Distributed Database Management System

نظام ادارة قواعد البيانات الموزعة

العملية Transaction:

- **العملية الجارية**: هي التي يجري تنفيذها حالياً.
- **العملية الجارية المحلية**: في قاعدة البيانات الموزعة، هي التي تتطلب التعامل مع البيانات المخزنة في الموقع الذي تنشأ فيه العملية الجارية فقط.
- **العملية الجارية الشاملة**: في قاعدة البيانات الموزعة، هي التي تتطلب التعامل مع البيانات الموجودة في موقع واحد أو أكثر غير محلي لتحقيق الطلب.

نظام ادارة قاعدة البيانات الموزعة DDBMS

- لكي توجد قاعدة بيانات موزعة يجب أن يكون هناك نظام إدارة لقواعد البيانات الموزعة والذي ينسق الإتصال بالبيانات في المواقع المختلفة

- ورغم أنه يمكن أن يوجد نظام DBMS في كل موقع لإدارة قاعدة البيانات المحلية الموجودة في هذا الموقع فيلزم نظام DBMS أيضاً لتنفيذ **الوظائف** التالية:

DDBMS Functions- DDBMS وظائف

- حفظ أين توجد البيانات في قاموس البيانات الموزعة.
- تحديد المواقع التي يتم إسترجاع البيانات المطلوبة منها.
- إذا إقتضى الأمر، ترجمة الطلب عند أحد المواقع بإستخدام DBMS محلي الى طلب مناسب لموقع أخر يستخدم DBMS مختلف ونموذج بيانات مختلف، وإعادة البيانات الى الموقع المرسل في صورة يقبلها هذا الموقع.
- توفير وظائف إدارة البيانات مثل الأمن، التزامن، مراقبة الورطة و الإسترجاع.

DDBMS Objectives

- **شفافية الموقع Location Transparency**: رغم إنتشار البيانات جغرافياً يمكن العمل كما لو كانت كل البيانات موجودة في عقدة واحدة.
- **شفافية التكرار Replication Transparency**: رغم إمكانية تكرار عنصر معين في العديد من المواقع في الشبكة فيمكن العمل على عنصر البيانات كما لو كان عنصر بيانات واحد في موقع واحد فقط.
- **شفافية الفشل Failure Transparency**: مقدرة النظام على إكتشاف الفشل وإسترجاع البيانات وإعادة تشكيل النظام.
- **شفافية التزامن Concurrency Transparency**: مقدرة النظام على ضمان صحة البيانات بحيث لايتسبب تعامل أكثر من مستخدم مع نفس عنصر البيانات في نفس الوقت في عدم صحة البيانات. عندما يحدث تشغيل متزامن للعديد من العمليات الجارية يجب أن تكون النتائج هي نفس النتائج كما لو كانت كل عملية جارية يحدث لها تشغيل على التوالي.

معمارية توزيع البيانات في قواعد البيانات الموزعة:

- بفرض أنه هناك مجموعة من قواعد البيانات التقليدية وأردنا توزيعها على مجموعة من المواقع Sites المكونة لشبكة حواسيب مترابطة فيما بينها بغرض توفير هذه البيانات والتشارك عليها، يمكن لنظم إدارة قواعدة البيانات الموزعة المتواجدة على هذه الأجهزة أن تعمل بطرق مختلفة ضمن فضاء توزيع البيانات والذي يتألف من ثلاثة أبعاد وهي:
- **1/ الاستقلالية Autonomy:**
- **2/ التوزيع Distribution:**
- **3/التغاير (عدم التجانس) Heterogeneity:**

1/ الاستقلالية Autonomy:

- وهي تعني **استقلالية نظام إدارة قاعدة البيانات DBMS وليس استقلالية البيانات نفسها**، يحدد هذا المؤشر فيما إذا كان من الممكن لنظام إدارة قاعدة بيانات معين أن يعمل بشكل مستقل والذي يعتمد على مجموعة من العوامل مثل:
  - تنفيذ العمليات الداخلية في نظام إدارة قاعدة البيانات المستقل يجب أن لا يؤثر على البيانات المخزنة في قواعد البيانات الأخرى.
  - معالجة وتنفيذ الاستعلامات Queries في قاعدة البيانات المستقلة يجب أن لا يؤثر على معالجة وتنفيذ الاستعلامات الخارجية في قواعدة البيانات الأخرى.
  - تماسك البيانات Data Consistency في قاعدة البيانات المستقلة يجب أن لا يتأثر عند الارتباط Join أو فك الارتباط بين قاعدة البيانات المستقلة وقواعدة البيانات الأخرى.

- ومن جانب أخر فإن الاستقلالية تعتمد على **النقاط** التالية:
  - **استقلالية التصميم Design Autonomy:**
    - DBMS يختار **التقنية** المناسبة لإدارة ومعالجة البيانات والعمليات Transactions.
  - **استقلالية الاتصال Communication Autonomy:**
    - DBMS يختار **البيانات** التي يمكن أن يوفرها لأنظمة إدارة قواعد البيانات الأخرى التي تتصل به.
  - **استقلالية التنفيذ Execution Autonomy:**
    - DBMS يختار **الطريقة المناسبة لتنفيذ العمليات** الموكلة له تنفيذها من أنظمة إدارة قواعد البيانات الأخرى.

2/ التوزيع Distribution:

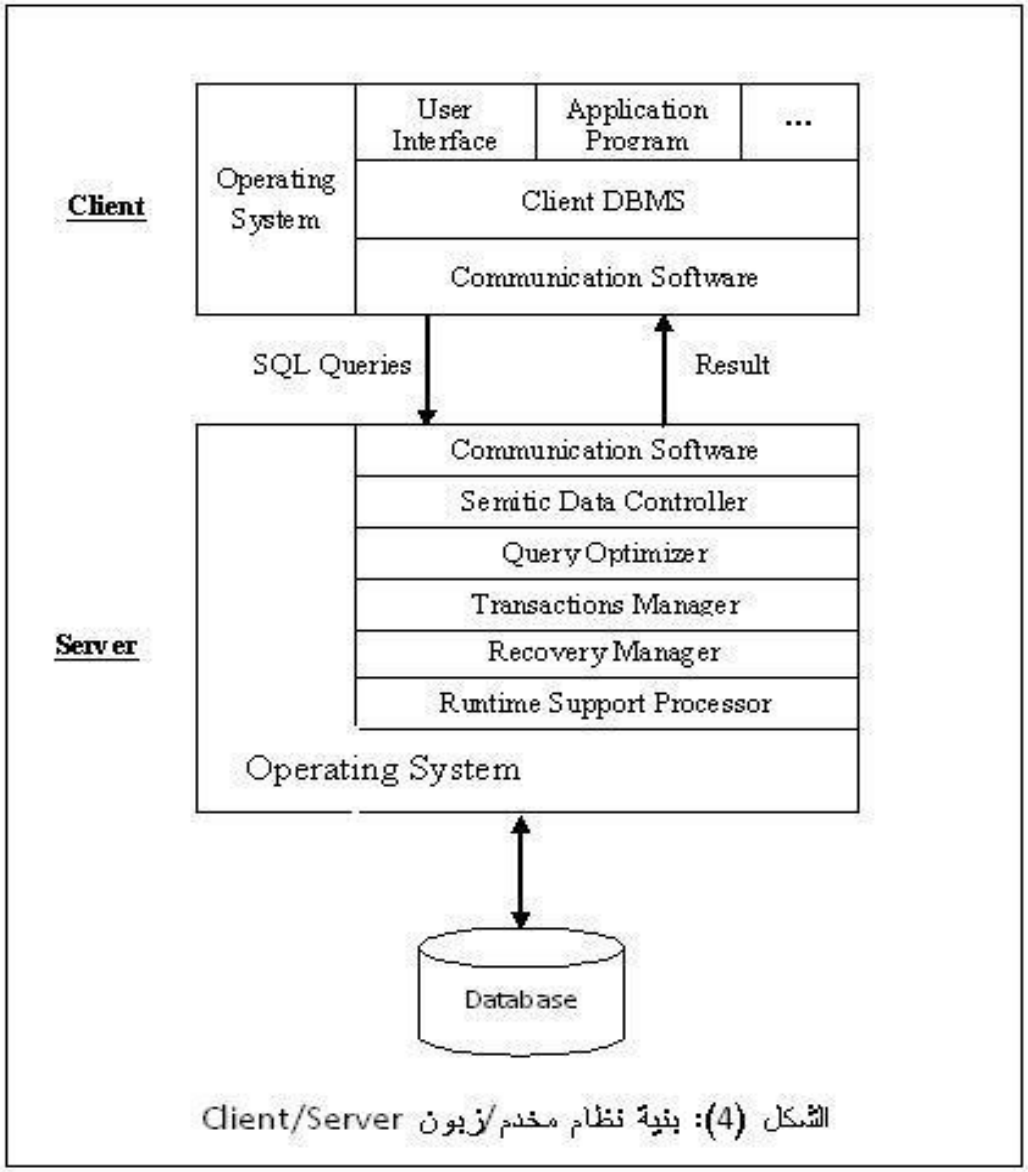
- **نظام المخدم/الزبون Client/Server**: في هذا النظام تتركز مهام إدارة وتخزين البيانات على المخدم Server، بينما ينحصر دور ومهام أنظمة الزبائن Clients على توفير الإتصال مع المخدم وتوفير البيانات للأنظمة البرمجية وواجهات المستخدم. يمثل نظام المخدم/الزبون المحاولة الأولى لتجسيد فكرة قواعد البيانات الموزعة
- **مخدم وحيد - متعدد الزبائن Multiple Clients-Single Server**،
- **متعدد المخدمين – متعدد الزبائن Multiple Clients- Multiple Servers**.

Multiple Clients-Single Server

- ، يمكننا تشبيه هذه البنية من أنظمة مخدم/زبون بنظام قاعدة البيانات المركزية Centralized Database لأنه يتم تخزين البيانات في قاعدة بيانات واحدة موجودة على المخدم والذي يتواجد عليه أيضاً نظام إدارة قاعدة البيانات DBMS.

Multiple Clients- Multiple Servers

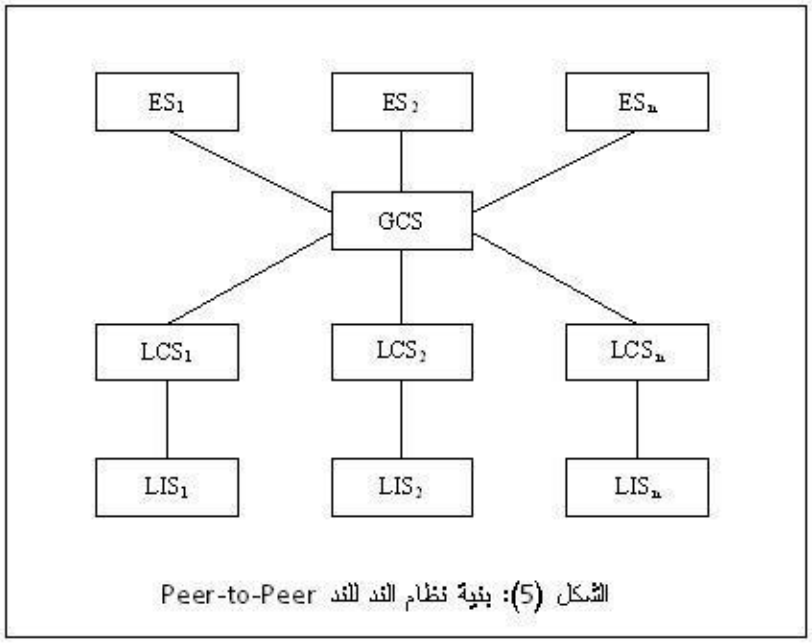
- هناك طريقتين للاتصال:
- الطريقة الأولى تتمثل بأن يقوم كل زبون بالاتصال بالمخدم المناسب لطلب البيانات التي يريدها، وفي هذه الطريقة **يتم تبسيط مهام المخدم في حين تزداد المسئولية على نظام الزبون**، تدعى هذه الطريقة بنظام الزبون الثقيل Heavy Client System.
- أما الطريقة الأخرى فتقتضي بأن يقوم كل زبون بالإتصال بمخدم واحد فقط لطلب البيانات منه ليقوم هذا المخدم بدوره بالاتصال ببقية المخدمات لطلب وتجميع البيانات التي لا تتواجد لديه وتوفيرها لذلك الزبون، وبهذا **تزداد وظائف المخدم وتنخفض أعباء الزبون** والذي يسمى بهذه الحالة بالزبون الخفيف Light Client



الشكل (4): بنية نظام مخدم/زبون Client/Server

Peer-to-Peer

- في هذا النوع من الأنظمة لن يكون هناك أي تفريق بين المخدم والزبون، فكلا الكمبيوترين المتصلين بهذه التقنية يمتلكان نظام إدارة قاعدة البيانات بكامل وظائفه ويمكنه الاتصال بالكمبيوتر الأخر لتنفيذ العمليات Transactions وطلب الاستعلامات Queries. يمكننا تسمية هذا النظام بنظام التوزيع الكلي Full Distribution.



الشكل (5): بنية نظام الند للند Peer-to-Peer

3/التغاير (عدم التجانس) Heterogeneity:

- يندرج ضمن مفهوم التغاير مجموعة من الحالات التي يبدو فيها الاختلاف في نقطة معينة أو جانب معين مثل:
- الإختلاف في التجهيزات المادية Hardware للأجهزة المتصلة مع بعضها البعض.
- الإختلاف في بروتوكولات الاتصال المستخدمة للتواصل بين الأجهزة المتصلة فيما بينها ونقل وتبادل البيانات.
- الإختلاف بطرق معالجة وتخزين البيانات في نظم إدارة قواعد البيانات.
- الإختلاف بطرق كتابة الاستعلامات Queries واللغات المستخدمة فيها والبنى والتراكيب التي تستخدمها هذه اللغات، مثل الإختلافات بين لغة الاستعلامات البنيوية Structured Query Language (SQL) المستخدمة في أنظمة إدارة قواعد البيانات المختلفة مثل Oracle و SQL Server و FoxPro وبقية الأنظمة الأخرى.

Database Concurrency

- من صفات منهج قاعدة البيانات السماح لعدة مستخدمين بإستخدام قاعدة البيانات في نفس الوقت، ومن الممكن أن يكون هناك أكثر من مستخدم يتنازعون على نفس عنصر البيانات في نفس الوقت. قد يتسبب هذا التنازع في عدم صحة البيانات. يعرف **التزامن في قواعد البيانات بأنه تنازع عدد من الإجراءات transactions على مورد معين في نفس الوقت مما يتسبب في عدم صحة البيانات**. ومن الحلول المستخدمة لحل مشكلة التزامن:
- **الإغلاق الثنائي Binary Lock**
- **إغلاق المشاركة والحصر Shared Exclusive Lock**

Binary Lock

- عندما يطلب إجراء transaction(T) عنصر بيانات item(X) يقوم الإجراء بإغلاق عنصر البيانات (X) بحيث لا يتمكن أي إجراء اخر من القيام بأي عملية عليه حتى ينتهي الإجراء (T) من العملية على عنصر البيانات (X) ثم يقوم بإطلاقه. الإجراءات الأخرى تقف في صف الإنتظار الى حين إطلاق عنصر البيانات (X) ثم يقوم إجراء واحد منها بإغلاق عنصر البيانات (X) وهكذا الى أن تنتهي العمليات على عنصر البيانات (X). ويتم إستخدام خوارزميتين هما **خوارزمية الإغلاق lock_item** و**خوارزمية الإطلاق unlock_item**.

- **lock_item(X)**:
- B: if lock(X) = 0 (“ item is unlocked “)
- then lock(X) ← 1 (“ lock the item”)
- else begin
- wait (until lock(X) = 0 AND the lock manager wakes up the transaction)
- goto B
- end

- **unlock_item(X)**:
- lock(X) ← 0 (“ unlock the item “)
- if any transactions are waiting then wakeup one of the waiting transactions

إغلاق المشاركة والحصر

- هنا يتم السماح بمشاركة الإجراءات في **عملية القراءة فيمكن السماح لعددغير محدد من الإجراءات بالقراءة من عنصر بيانات واحد في نفس الوقت**. أما **عملية الكتابة فتكون حصرية على إجراء واحد فقط** يقوم بإغلاق عنصر البيانات الى أن تنتهي عملية الكتابة ثم الإطلاق. ويتم إستخدام ثلاث خوارزميات **خوارزمية القراءة read_lock** ، **خورارزمية الكتابة write_lock** وخوارزمية **الإطلاق unlock**.

- **read_lock(X)**:
- B: if lock(X) = “unlocked”
- then begin
- lock(X) ← “read-locked”
- no_of_reads(X) ← 1
- end
- else if lock(X) = “read-locked”
- then no_of_reads(X) ← no_of_reads(X) + 1
- else begin
- wait (until lock(X) = “unlocked” AND the lock manager wakes up the transaction)
- goto B
- end

- **write_lock(X)**:
- B: if lock(X) = “unlocked”
- then lock(X) ← “write-locked”
- else begin
- wait (until lock(X) = “unlocked” AND the lock manager wakes up the transaction)
- goto B
- end

- **unlock(X)**:
- if lock(X) = “write-locked”
- then begin
- lock(X) ← “unlocked”
- wakeup one of the waiting transactions, if any
- end
- else if lock(X) = “read-locked”
- then begin
- no_of_reads(X) ← no_of_reads(X) – 1
- if no_of_reads(X) = 0
- then begin
- lock(X) = “unlocked”
- wakeup one of the waiting transactions, if any
- end
- end