Databases
Lectures 4, 5, and 6

Timothy G. Griffin

Computer Laboratory
University of Cambridge, UK

Databases, Lent 2009
Lecture 04: Database Updates

Outline

- Transactions
- Short review of ACID requirements
### Transactions — ACID properties

#### Should be review from Concurrent Systems and Applications

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Atomicity</strong></td>
<td>Either all actions are carried out, or none are</td>
</tr>
<tr>
<td></td>
<td>- logs needed to undo operations, if needed</td>
</tr>
<tr>
<td><strong>Consistency</strong></td>
<td>If each transaction is consistent, and the database is initially consistent, then it is left consistent</td>
</tr>
<tr>
<td></td>
<td>- This is very much a part of applications design.</td>
</tr>
<tr>
<td><strong>Isolation</strong></td>
<td>Transactions are isolated, or protected, from the effects of other scheduled transactions</td>
</tr>
<tr>
<td></td>
<td>- Serializability, 2-phase commit protocol</td>
</tr>
<tr>
<td><strong>Durability</strong></td>
<td>If a transactions completes successfully, then its effects persist</td>
</tr>
<tr>
<td></td>
<td>- Logging and crash recovery</td>
</tr>
<tr>
<td>Outline</td>
<td></td>
</tr>
<tr>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>Update anomalies</td>
<td></td>
</tr>
<tr>
<td>Functional Dependencies (FDs)</td>
<td></td>
</tr>
<tr>
<td>Normal Forms, 1NF, 2NF, 3NF, and BCNF</td>
<td></td>
</tr>
</tbody>
</table>
Transactions from an application perspective

Main issues

- Avoid update anomalies
- Minimize locking to improve transaction throughput.
- Maintain integrity constraints.

These issues are related.
Update anomalies

How can we tell if an insert record is consistent with current records?

Can we record data about a course before students enroll?

Will we wipe out information about a college when last student associated with the college is deleted?
Redundancy implies more locking ...

... at least for correct transactions!

Big Table

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>college</th>
<th>course</th>
<th>part</th>
<th>term_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>yy88</td>
<td>Yoni</td>
<td>New Hall</td>
<td>Algorithms I</td>
<td>IA</td>
<td>Easter</td>
</tr>
<tr>
<td>uu99</td>
<td>Uri</td>
<td>King’s</td>
<td>Algorithms I</td>
<td>IA</td>
<td>Easter</td>
</tr>
<tr>
<td>bb44</td>
<td>Bin</td>
<td>New Hall</td>
<td>Databases</td>
<td>IB</td>
<td>Lent</td>
</tr>
<tr>
<td>bb44</td>
<td>Bin</td>
<td>New Hall</td>
<td>Algorithms II</td>
<td>IB</td>
<td>Michaelmas</td>
</tr>
<tr>
<td>zz70</td>
<td>Zip</td>
<td>Trinity</td>
<td>Databases</td>
<td>IB</td>
<td>Lent</td>
</tr>
<tr>
<td>zz70</td>
<td>Zip</td>
<td>Trinity</td>
<td>Algorithms II</td>
<td>IB</td>
<td>Michaelmas</td>
</tr>
</tbody>
</table>

- Change New Hall to Murray Edwards College
  - Conceptually simple update
  - May require locking entire table.
Redundancy is the root of (almost) all database evils

- It may not be obvious, but redundancy is also the cause of update anomalies.
- By redundancy we do not mean that some values occur many times in the database!
  - A foreign key value may be have millions of copies!
- But then, what do we mean?
Functional Dependency

Functional Dependency (FD)

Let $R(X)$ be a relational schema and $Y \subseteq X$, $Z \subseteq X$ be two attribute sets. We say $Y$ functionally determines $Z$, written $Y \rightarrow Z$, if for any two tuples $u$ and $v$ in an instance of $R(X)$ we have

$$u.Y = v.Y \rightarrow u.Z = v.Z.$$  

We call $Y \rightarrow Z$ a functional dependency.

A functional dependency is a **semantic** assertion. It represents a rule that should always hold in any instance of schema $R(X)$. 

T. Griffin (cl.cam.ac.uk)  Databases Lectures 4, 5, and 6  DB 2009 9 / 1
### Example FDs

#### Big Table

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>college</th>
<th>course</th>
<th>part</th>
<th>term_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>yy88</td>
<td>Yoni</td>
<td>New Hall</td>
<td>Algorithms I</td>
<td>IA</td>
<td>Easter</td>
</tr>
<tr>
<td>uu99</td>
<td>Uri</td>
<td>King’s</td>
<td>Algorithms I</td>
<td>IA</td>
<td>Easter</td>
</tr>
<tr>
<td>bb44</td>
<td>Bin</td>
<td>New Hall</td>
<td>Databases</td>
<td>IB</td>
<td>Lent</td>
</tr>
<tr>
<td>bb44</td>
<td>Bin</td>
<td>New Hall</td>
<td>Algorithms II</td>
<td>IB</td>
<td>Michaelmas</td>
</tr>
<tr>
<td>zz70</td>
<td>Zip</td>
<td>Trinity</td>
<td>Databases</td>
<td>IB</td>
<td>Lent</td>
</tr>
<tr>
<td>zz70</td>
<td>Zip</td>
<td>Trinity</td>
<td>Algorithms II</td>
<td>IB</td>
<td>Michaelmas</td>
</tr>
</tbody>
</table>

- $\text{sid} \rightarrow \text{name}$
- $\text{sid} \rightarrow \text{college}$
- $\text{course} \rightarrow \text{part}$
- $\text{course} \rightarrow \text{term}_\text{name}$
Keys, revisited

Candidate Key

Let $R(X)$ be a relational schema and $Y \subseteq X$. $Y$ is a candidate key if

1. The FD $Y \rightarrow X$ holds, and
2. for no proper subset $Z \subseteq Y$ does $Z \rightarrow X$ hold.

Prime and Non-prime attributes

An attribute $A$ is prime for $R(X)$ if it is a member of some candidate key for $R$. Otherwise, $A$ is non-prime.

Database redundancy roughly means the existence of non-key functional dependencies!
First Normal Form (1NF)

We will assume every schema is in 1NF.

1NF

A schema \( R(A_1 : S_1, A_2 : S_2, \cdots, A_n : S_n) \) is in First Normal Form (1NF) if the domains \( S_1 \) are elementary — their values are atomic.

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>name</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timothy George Griffin</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>first_name</strong></th>
<th><strong>middle_name</strong></th>
<th><strong>last_name</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Timothy</td>
<td>George</td>
<td>Griffin</td>
</tr>
</tbody>
</table>
Second Normal Form (2NF)

A relational schema $R$ is in 2NF if for every functional dependency $X \rightarrow A$ either

- $A \in X$, or
- $X$ is a superkey for $R$, or
- $A$ is a member of some key, or
- $X$ is not a proper subset of any key.
3NF and BCNF

Third Normal Form (3CNF)
A relational schema $R$ is in 3NF if for every functional dependency $X \rightarrow A$ either
- $A \in X$, or
- $X$ is a superkey for $R$, or
- $A$ is a member of some key.

Boyce-Codd Normal Form (BCNF)
A relational schema $R$ is in BCNF if for every functional dependency $X \rightarrow A$ either
- $A \in X$, or
- $X$ is a superkey for $R$. 

Inclusions

Clearly BCNF ⊆ 3NF ⊆ 2NF. These are proper inclusions:

<table>
<thead>
<tr>
<th>In 2NF, but not 3NF</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R(A, B, C) ), with ( F = {A \rightarrow B, B \rightarrow C} ).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>In 3NF, but not BCNF</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R(A, B, C) ), with ( F = {A, B \rightarrow C, C \rightarrow B} ).</td>
</tr>
</tbody>
</table>
  * This is in 3NF since \( AB \) and \( AC \) are keys, so there are no non-prime attributes
  * But not in BCNF since \( C \) is not a key and we have \( C \rightarrow B \). |
Given a relational schema $R(X)$ with FDs $F$:

- **Reason about FDs**
  - Is $F$ missing FDs that are logically implied by those in $F$?
- **Decompose each** $R(X)$ **into smaller** $R_1(X_1)$, $R_2(X_2)$, … $R_k(X_k)$, **where each** $R_i(X_i)$ **is in the desired Normal Form.**

Are some decompositions better than others?
Desired properties of any decomposition

Lossless-join decomposition

A decomposition of schema $R(X)$ to $S(Y \cup Z)$ and $T(Y \cup (X - Z))$ is a lossless-join decomposition if for every database instances we have $R = S \Join T$.

Dependency preserving decomposition

A decomposition of schema $R(X)$ to $S(Y \cup Z)$ and $T(Y \cup (X - Z))$ is dependency preserving, if enforcing FDs on $S$ and $T$ individually has the same effect as enforcing all FDs on $S \Join T$.

We will see that it is not always possible to achieve both of these goals.
Lecture 06: Reasoning about FDs

Outline

- Implied dependencies (closure)
- Armstrong’s Axioms
Semantic Closure

Notation

\[ F \models Y \rightarrow Z \]

means that any database instance that that satisfies every FD of \( F \), must also satisfy \( Y \rightarrow Z \).

The semantic closure of \( F \), denoted \( F^+ \), is defined to be

\[ F^+ = \{ Y \rightarrow Z \mid Y \cup Z \subseteq \text{atts}(F) \text{ and } \land F \models Y \rightarrow Z \} . \]

The membership problem is to determine if \( Y \rightarrow Z \in F^+ \).
Reasoning about Functional Dependencies

We write $F \vdash Y \rightarrow Z$ when $Y \rightarrow Z$ can be derived from $F$ via the following rules.

### Armstrong’s Axioms

<table>
<thead>
<tr>
<th>Rule</th>
<th>Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reflexivity</td>
<td>If $Z \subseteq Y$, then $F \vdash Y \rightarrow Z$.</td>
</tr>
<tr>
<td>Augmentation</td>
<td>If $F \vdash Y \rightarrow Z$ then $F \vdash Y, W \rightarrow Z, W$.</td>
</tr>
<tr>
<td>Transitivity</td>
<td>If $F \vdash Y \rightarrow Z$ and $F \models Z \rightarrow W$, then $F \vdash Y \rightarrow W$.</td>
</tr>
</tbody>
</table>
Logical Closure (of a set of attributes)

Notation

\[
\text{closure}(F, X) = \{ A \mid F \vdash X \rightarrow A \}
\]

Claim 1

If \( Y \rightarrow W \in F \) and \( Y \subseteq \text{closure}(F, X) \), then \( W \subseteq \text{closure}(F, X) \).

Claim 2

\( Y \rightarrow W \in F^+ \) if and only if \( W \subseteq \text{closure}(F, Y) \).
Soundness and Completeness

**Soundness**

\[ F \vdash f \implies f \in F^+ \]

**Completeness**

\[ f \in F^+ \implies F \vdash f \]
Proof of Completeness (soundness left as an exercise)

Show $\neg (F \models f) \iff \neg (F \models f)$:

- Suppose $\neg (F \models Y \rightarrow Z)$ for $R(X)$.
- Let $Y^+ = \text{closure}(F, Y)$.
- $\exists B \in Z$, with $B \notin Y^+$.
- Construct an instance of $R$ with just two records, $u$ and $v$, that agree on $Y^+$ but not on $X - Y^+$.
- By construction, this instance does not satisfy $Y \rightarrow Z$.
- But it does satisfy $F$! Why?
  - let $S \rightarrow T$ be any FD in $F$, with $u.[S] = v.[S]$.
  - So $S \subseteq Y^+$. and so $T \subseteq Y^+$ by claim 1,
  - and so $u.[T] = v.[T]$
Consequences of Armstrong’s Axioms

Union  If $F \models Y \rightarrow Z$ and $F \models Y \rightarrow W$, then $F \models Y \rightarrow W, Z$.

Pseudo-transitivity  If $F \models Y \rightarrow Z$ and $F \models U, Z \rightarrow W$, then $F \models Y, U \rightarrow W$.

Decomposition  If $F \models Y \rightarrow Z$ and $W \subseteq Z$, then $F \models Y \rightarrow W$.

Exercise: Prove these using Armstrong’s axioms!
Proof of the Union Rule

Suppose we have

\[ F \models Y \rightarrow Z, \]
\[ F \models Y \rightarrow W. \]

By augmentation we have

\[ F \models Y, Y \rightarrow Y, Z, \]
that is,

\[ F \models Y \rightarrow Y, Z. \]

Also using augmentation we obtain

\[ F \models Y, Z \rightarrow W, Z. \]

Therefore, by transitivity we obtain

\[ F \models Y \rightarrow W, Z. \]
Example application of functional reasoning.

Heath’s Rule

Suppose \( R(A, B, C) \) is a relational schema with functional dependency \( A \rightarrow B \), then

\[
R = \pi_{A,B}(R) \bowtie_A \pi_{A,C}(R).
\]
Proof of Heath’s Rule

We first show that $R \subseteq \pi_{A,B}(R) \bowtie_A \pi_{A,C}(R)$.

- If $u = (a, b, c) \in R$, then $u_1 = (a, b) \in \pi_{A,B}(R)$ and $u_2 = (a, c) \in \pi_{A,C}(R)$.
- Since $\{(a, b)\} \bowtie_A \{(a, c)\} = \{(a, b, c)\}$ we know $u \in \pi_{A,B}(R) \bowtie_A \pi_{A,C}(R)$.

In the other direction we must show $R' = \pi_{A,B}(R) \bowtie_A \pi_{A,C}(R) \subseteq R$.

- If $u = (a, b, c) \in R'$, then there must exist tuples $u_1 = (a, b) \in \pi_{A,B}(R)$ and $u_2 = (a, c) \in \pi_{A,C}(R)$.
- This means that there must exist a $u' = (a, b', c) \in R$ such that $u_2 = \pi_{A,C}(\{(a, b', c)\})$.
- However, the functional dependency tells us that $b = b'$, so $u = (a, b, c) \in R$. 

Closure Example

$R(A, B, C, D, D, F)$ with

\[
\begin{align*}
A, B & \rightarrow C \\
B, C & \rightarrow D \\
D & \rightarrow E \\
C, F & \rightarrow B
\end{align*}
\]

What is the closure of $\{A, B\}$?

\[
\begin{align*}
\{A, B\} & \xrightarrow{A, B \rightarrow C} \{A, B, C\} \\
& \xrightarrow{B, C \rightarrow D} \{A, B, C, D\} \\
& \xrightarrow{D \rightarrow E} \{A, B, C, D, E\}
\end{align*}
\]

So $\{A, B\}^+ = \{A, B, C, D, E\}$ and $A, B \rightarrow C, D, E$. 