Lecture 06 (revised version) : Database updates

Outline

- ACID transactions
- Update anomalies
- General integrity constraints
- Problems with data redundancy
- A simple language for transactions
- Reasoning about transactions.

Transactions — The ACID abstraction

ACID

Atomicity Either all actions are carried out, or none are

- logs needed to undo operations, if needed

Consistency If each transaction is consistent, and the database is initially consistent, then it is left consistent

- This is very much a part of applications design.

Isolation Transactions are isolated, or protected, from the effects of other scheduled transactions

- Serializability, 2-phase commit protocol

Durability If a transactions completes successfully, then its effects persist

- Logging and crash recovery

Should be review from Concurrent and Distributed Systems so we will not go into the details of how these abstractions are implemented.

Bad design

Big Table

sid	name	college	course	part	term_name
yy88	Yoni	New Hall	Algorithms I	IA	Easter
uu99	Uri	King's	Algorithms I	IA	Easter
bb44	Bin	New Hall	Databases	IB	Lent
bb44	Bin	New Hall	Algorithms II	IB	Michaelmas
zz70	Zip	Trinity	Databases	IB	Lent
zz70	Zip	Trinity	Algorithms II	IB	Michaelmas

Data anomalies

Insertion anomalies

How can we tell if an inserted record is consistent with current records? Can we record data about a course before students enroll?

Deletion anomalies

Will we wipe out information about a college when last student associated with the college is deleted?

Update anomalies

Change New Hall to Murray Edwards College

- Conceptually simple update
- May require locking entire table.

General database integrity constraints

Just write predicates with quantifiers $\forall x \in Q, P(x)$ and $\exists x \in Q, P(x)$, where Q is a query in a relational calculus.

For a database assertion P, the notation $DB \models P$ means that P holds in the database instance DB.

Examples

Example. A key constraint for R:

$$\forall t \in R, \forall u \in R, t.\text{key} = u.\text{key} \rightarrow t = u$$

Example. A foreign key constraint (*key* is a key of S):

$$\forall t \in R, \exists u \in S, t.\text{key} = u.\text{key}$$

One goal of database schema design

Design a database schema so that almost all integrity constraints are key constraints or foreign key constraints.

One possible approach

- Suppose that C is some constraint we would like to enforce on our database.
- Let $Q_{\neg C}$ be a query that captures all violations of C.
- Enforce (somehow) that the assertion that is always $Q_{\neg C}$ empty.

```
create view C_violations as ....

create assertion check_C
       check not (exists C_violations)
```

A simple language for transactions?

Although the relational algebra or relational calculi are widely used, there seems to be no analogous formalism for database updates and transactions. So we invent one!

Transactions will have the form

$$\text{transaction} \;\; f(x_1, x_2, ..., x_k) = E$$

where

E	::=	skip	(do nothing)
	|	abort	(abort transaction)
	|	$\text{INS}(R,\ t)$	(insert tuple t into R)
	|	$\text{DEL}(R,\ p)$	(delete $\sigma_p(R)$ from R)
	|	$E_1;\ E_2$	(sequence)
	|	if P *then* E_1 *else* E_2	(P a predicate)

Hoare Logic for Database updates

We write

$$\{P\}\ E\ \{Q\}$$

to mean that if $DB \models P$ then $E(DB) \models Q$, where $E(DB)$ denotes the result of executing E in database DB.

One way to think about an integrity constraint C

For all transactions

$$\text{transaction } f(x_1, x_2, ..., x_k) = E$$

and all values $v_1, \ldots v_k$ we want

$$\{C\}\ f(v_1, v_2, ..., v_k)\ \{C\}$$

That is, constraint C is an *invariant* of for all transactions.

The weakest precondition

Defined the *weakest precondition of E with respect to Q*, $\mathrm{wpc}(E,\ Q)$, to be a database predicate such that if

$$P \rightarrow \mathrm{wpc}(E,\ Q),$$

then

$$\{P\}\ E\ \{Q\}.$$

That is, $\mathrm{wpc}(E,\ Q)$ is the weakest predicate such that

$$\{\mathrm{wpc}(E,\ Q)\}\ E\ \{Q\}.$$

In other words, if $DB \models \mathrm{wpc}(E,\ Q)$ then $E(DB) \models Q$.

So, for C to be an invariant of f we want for all $v_1, v_2, ..., v_k$,

$$C \rightarrow \mathrm{wpc}(f(v_1, v_2, ..., v_k),\ C).$$

The weakest precondition

For simplicity we ignore abort ...

$$
\begin{aligned}
\mathrm{wpc}(\mathrm{skip},\ Q) &= Q \\
\mathrm{wpc}(\mathrm{INS}(R,\ t),\ Q) &= Q[R \cup \{t\}/R] \\
\mathrm{wpc}(\mathrm{DEL}(R,\ p),\ Q) &= Q[\{t \in R \mid \neg p(t)\}/R] \\
\mathrm{wpc}(E_1;\ E_2,\ Q) &= \mathrm{wpc}(E_1,\ \mathrm{wpc}(E_2,\ Q)) \\
\mathrm{wpc}(\text{if } T \textit{ then } E_1 \textit{ else } E_2,\ Q) &= (T \rightarrow \mathrm{wpc}(E_1,\ Q)) \wedge \\
&\quad\ (\neg T \rightarrow \mathrm{wpc}(E_2,\ Q))
\end{aligned}
$$

Example (a foreign key constraint, *key* is a key of S)

$$
\begin{aligned}
Q &= \forall t \in R, \exists u \in S, t.\text{key} = u.\text{key} \\
E &= \text{INS}(R,\ v);\ \text{INS}(S,\ w)
\end{aligned}
$$

$$
\begin{aligned}
& \text{wpc}(E,\ Q) \\
= \quad & \text{wpc}(\text{INS}(R,\ v),\ \text{wpc}(\text{INS}(S,\ w),\ Q)) \\
= \quad & \text{wpc}(\text{INS}(R,\ v),\ \forall t \in R, \exists u \in S \cup \{w\}, t.\text{key} = u.\text{key}) \\
= \quad & \forall t \in R \cup \{v\}, \exists u \in S \cup \{w\}, t.\text{key} = u.\text{key} \\
\leftrightarrow \quad & \forall t \in R \cup \{v\}, (t.\text{key} = w.\text{key}) \vee (\exists u \in S, t.\text{key} = u.\text{key}) \\
\leftrightarrow \quad & ((v.\text{key} = w.\text{key}) \vee (\exists u \in S, v.\text{key} = u.\text{key})) \\
& \wedge \forall t \in R, (t.\text{key} = w.\text{key}) \vee \exists u \in S, t.\text{key} = u.\text{key} \\
\leftarrow \quad & ((v.\text{key} = w.\text{key}) \vee (\exists u \in S, v.\text{key} = u.\text{key})) \wedge Q
\end{aligned}
$$

Example (a foreign key constraint, *key* is a key of S)

Conclude that the integrity constraint

$$Q \quad = \quad \forall t \in R, \exists u \in S, t.\text{key} = u.\text{key}$$

is an invariant of the following transaction.

transaction $f(v,\ w) =$
  if $(v.\text{key} = w.\text{key}) \vee (\exists u \in S, v.\text{key} = u.\text{key}$
  then $\text{INS}(R,\ v);\ \text{INS}(S,\ w)$
  else skip

Example : key constraint

In a similar way, we can show that the transaction

$$\begin{array}{l}\text{transaction } \mathrm{insert}(R,\ t) = \\ \quad \text{if } \forall u \in R, u.\mathsf{key} \neq t.\mathsf{key} \\ \quad \text{then } \mathrm{INS}(R,\ t) \\ \quad \text{else } \mathrm{skip}\end{array}$$

has invariant

$$Q = \forall t \in R, \forall u \in R, t.\mathsf{key} = u.\mathsf{key} \rightarrow t = u.$$

Exercise: Show that

$$Q \rightarrow \mathrm{wpc}(\mathrm{insert}(R,\ t),\ Q).$$

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?
- We will model logical redundancy with *functional dependencies* (next lecture).