CS340 Final

Databases

08/18/2017

# Bridge Inspection Database

By Kevin Lewis



Live Website:

http://flip1.engr.oregonstate.edu:2334/bridge

Bridge Inspection Database Outline

The Federal Highway Administration through the National Bridge Inspection (NBI) program requires that a safety inspection is performed for all bridges once every two years at a minimum. This is to ensure that bridges are maintained such that they are safe to use by the public. In practice. This entails documenting any deficiencies in the bridge, example a crack in a beam, so that the owners, such as the state government, can identify what maintenance will needs to be performed.

The inspection process is imperative for what is known as Fracture Critical bridges. These are typically bridges with two or less load carrying members such that if one of the members fails, the entire bridge will collapse. An excellent example is a through truss bridge, see photo below. The top and bottom horizontal members are called the top and bottom chord. If either of those fails on one side of the bridge, then the entire bridge will fail. Because Fracture Critical bridges are prone to sudden failure, they are required to be inspected once a year.

Bridge inspection project management is a complex task that involves juggling personnel and equipment to ensure that bridges are inspected on-time and on-budget while producing a high-quality deliverable for the client. Currently, at my company it is difficult for a project manager to take a snap-shot of the project at any point. He or she must rely on employee timesheet input that eventually percolates up to the project manager, who then needs to parse the information to determine the status of a bridge. However, bridge inspection work can sometimes take place over several weeks, and can give a false indication that a bridge is below budget. The purpose of the database is to give more tools to the project manager to better track the status of a bridge’s inspection.

Database Outline

The following is a detailed description of the various elements within the Bridge Inspection Database.

Employee:

* Description: This entity are the people at the company that will take part in managing the program, inspecting the bridges, and writing the reports.
* Properties:
  + Employee ID Primary Key
  + First Name
  + Last Name
  + Hourly Rate
* Relationship:
  + Assigned to an inspection (many-to-many): Each employee can be assigned to an inspection. An inspection means the person is either planning or going out to the bridge. Within the inspection assignment, the employee will have a role, such as team leader, inspector, inspection manager. Employees are not required to be assigned to any inspections.
  + Assigned to a report (many-to-many): Like the inspection assignment, each employee can be work on a report associated with a report associated with an inspection. Each employee will be associated with a task within the relationship
  + Has-an Employee ID (one-to-one): Each employee has one employee ID

Office:

* Description: Each office location where the employee works.
* Properties:
  + ID (Primary Key)
  + Name
  + Zip Code
* Relationship:
  + Has employees (one-to-Many): Has at least one employee who works there

Bridge:

* Description: Each bridge to be inspected.
* Properties:
  + Bridge ID
  + Type
  + Span
  + Length
  + Zipcode
* Relationship:
  + Has a report (one-to-many): Each bridge has one to many reports. Typically it will only be one, but there could be different inspection (special vs routine) which would require two separate inspections.
  + Has an inspection (one-to-many): Like the reports, each bridge has one to many inspections.
  + Is associated with an LOA (Many to One): Each bridge is only associated with one LOA

Report:

* Description: The report that are associated with the bridges and typically paired with an inspection.
* Properties:
  + ID
  + Type : Special vs Routine, etc.
  + Budget
  + Status (text)
  + Percent complete
* Relationship:
  + Has a bridge/LOA (many-to-one): Each report has only one bridge and one LOA.
  + Many employees work on it (one to Many)

Inspection:

* Description: Inspection associated with the bridge
* Properties:
  + ID (Primary Key)
  + Type: Special vs Routine, etc.
  + Budget
  + Status (text)
  + Percent complete
* Relationship:
  + Has a bridge (many-to-one): Each inspection has only one bridge.
  + Many employees work on it (one to Many)

LOA (Letter of Authorization):

* Description: The LOA is the overall section of the project that the bridges are associated with. It is at this level that Project Managers mostly see if a project is over budget or not.
* Properties:
  + ID
  + Number
* Relationship:
  + Has at least one report / inspection (one-to-many): The loa has several reports or inspections, but each inspection / report must have one loa.

ER Diagram

Schema

**Data Definition Queries**

CREATE TABLE employee (

eid INT AUTO\_INCREMENT NOT NULL,

first\_name VARCHAR(255) NOT NULL,

last\_name VARCHAR(255) NOT NULL,

hourly\_rate FLOAT,

fk\_officeID INT,

PRIMARY KEY(eid),

CONSTRAINT fl\_employee UNIQUE (first\_name, last\_name),

FOREIGN KEY (fk\_officeID) REFERENCES office(id)

) ENGINE=InnoDB

CREATE TABLE office (

id INT AUTO\_INCREMENT NOT NULL,

name VARCHAR(255) NOT NULL,

zipcode INT,

PRIMARY KEY(id),

UNIQUE (name)

) ENGINE=InnoDB

CREATE TABLE bridge (

id INT AUTO\_INCREMENT NOT NULL,

length INT,

type varchar(255),

zipcode INT NOT NULL,

spans INT,

NBI\_ID INT NOT NULL,

PRIMARY KEY (id),

UNIQUE (NBI\_ID)

) ENGINE=InnoDB

CREATE TABLE LOA (

id INT AUTO\_INCREMENT NOT NULL,

number INT NOT NULL,

PRIMARY KEY (id),

UNIQUE (number)

) ENGINE=InnoDB

CREATE TABLE inspection (

insp\_id INT AUTO\_INCREMENT NOT NULL,

type varchar(255),

budget FLOAT NOT NULL,

pcomp INT,

fk\_bridgeID INT NOT NULL,

fk\_LOA\_num INT NOT NULL,

PRIMARY KEY (insp\_id),

FOREIGN KEY (fk\_bridgeID) REFERENCES bridge(id) ON DELETE CASCADE,

FOREIGN KEY (fk\_LOA\_num) REFERENCES LOA(id)

) ENGINE=InnoDB

CREATE TABLE report (

rep\_id INT AUTO\_INCREMENT NOT NULL,

type varchar(255),

budget FLOAT NOT NULL,

pcomp INT,

fk\_bridgeID INT NOT NULL,

fk\_LOA\_num INT NOT NULL,

PRIMARY KEY (rep\_id),

FOREIGN KEY (fk\_bridgeID) REFERENCES bridge(id) ON DELETE CASCADE,

FOREIGN KEY (fk\_LOA\_num) REFERENCES LOA(id)

) ENGINE=InnoDB

CREATE TABLE insp\_assign(

eid int NOT NULL,

insp\_id int NOT NULL,

hours\_worked int,

PRIMARY KEY (eid, insp\_id),

FOREIGN KEY (eid) REFERENCES employee(eid),

FOREIGN KEY (insp\_id) REFERENCES inspection(insp\_id)

) ENGINE=InnoDB

CREATE TABLE rep\_assign(

eid int NOT NULL,

rep\_id int NOT NULL,

hours\_worked int,

PRIMARY KEY (eid, rep\_id),

FOREIGN KEY (eid) REFERENCES employee(eid),

FOREIGN KEY (rep\_id) REFERENCES report(rep\_id)

) ENGINE=InnoDB

**Data Manipulation Queries**

**SQL = BLUE**

**Variables = RED**

Report Status Site:

SELECT s\_rep.rep\_id AS elem\_id, s\_rep.NBI\_ID, s\_rep.bridgeID, l.number, IFNULL(s\_rep.total\_rep,0) AS total\_spent, IFNULL(s\_rep.budget,0) AS total\_budget, IF( IFNULL(s\_rep.total\_rep,0) - IFNULL(s\_rep.budget,0) < 0,

'Over Budget', 'Under Budget') AS cur\_status FROM LOA l

INNER JOIN (SELECT r.rep\_id, b.NBI\_ID, b.id AS bridgeID, r.fk\_LOA\_num as LOA\_num, SUM(e.hourly\_rate \* ra.hours\_worked) AS total\_rep, SUM(r.budget) as budget FROM report r

LEFT JOIN rep\_assign ra ON ra.rep\_id = r.rep\_id

LEFT JOIN employee e ON e.eid = ra.eid

INNER JOIN bridge b ON b.id = r.fk\_bridgeID GROUP BY r.rep\_id)

AS s\_rep ON s\_rep.LOA\_num = l.number

GROUP BY elem\_id [HAVING cur\_status = "Over Budget", HAVING cur\_status = "Under Budget", HAVING l.number = ?, HAVING bridgeID = ?, HAVING elem\_id = ?]

Query Comments: The above queries overall goal is to determine if an reoprt is over-budget. It starts by pulling the information that we need from the report by joining with the various tables. The larger inner selection that it joins with determines the total amount spent on each project by summing the hours x hourly wage for each employee working on the report.

The user is given several options of filtering the data (see the final group by statement) including the status, loa number, bridge NBI and report number.

The following is used to fill out tables on the webpage:

SELECT l.id, l.number FROM LOA l

INNER JOIN inspection i ON i.fk\_LOA\_num = l.id

SELECT DISTINCT b.id, b.NBI\_ID FROM bridge b

INNER JOIN inspection i ON i.fk\_bridgeID = b.id

SELECT rep\_id AS elem\_id FROM report

Inspection Status Site:

SELECT tmp.insp\_id AS elem\_id, tmp.NBI\_ID, tmp.bridgeID, l.number, IFNULL(tmp.total\_insp,0) AS total\_spent, IFNULL(tmp.budget,0) AS total\_budget, IF(IFNULL(tmp.total\_insp,0) - IFNULL(tmp.budget,0) < 0, 'Over Budget', 'Under Budget') AS cur\_status FROM LOA l

INNER JOIN (SELECT i.insp\_id, b.NBI\_ID, b.id AS bridgeID, i.fk\_LOA\_num as LOA\_num, SUM(e.hourly\_rate \* ia.hours\_worked) AS total\_insp,

SUM(i.budget) as budget FROM inspection i

LEFT JOIN insp\_assign ia ON ia.insp\_id = i.insp\_id

LEFT JOIN employee e ON e.eid = ia.eid INNER JOIN bridge b ON b.id = i.fk\_bridgeID GROUP BY i.insp\_id)

AS tmp ON tmp.LOA\_num = l.number GROUP BY elem\_id [HAVING cur\_status = "Over Budget", HAVING cur\_status = "Under Budget", HAVING l.number = ?, HAVING bridgeID = ?, HAVING elem\_id = ?]

Query Comments: Following a similar structure as the report, the status of the inspections are determined.

The user is given several options of filtering the data (see the final group by statement) including the status, loa number, bridge NBI and report number.

The following is used to fill out tables on the webpage:

SELECT l.id, l.number FROM LOA l

INNER JOIN inspection i ON i.fk\_LOA\_num = l.id

SELECT DISTINCT b.id, b.NBI\_ID FROM bridge b

INNER JOIN inspection i ON i.fk\_bridgeID = b.id

SELECT insp\_id AS elem\_id FROM inspection

LOA Status Site:

SELECT l.number, IFNULL(s\_insp.total\_insp,0) + IFNULL(s\_rep.total\_rep,0) AS total\_spent, IFNULL(s\_insp.budget,0) + IFNULL(s\_rep.budget,0) AS total\_budget, IF((IFNULL(s\_insp.total\_insp,0) + IFNULL(s\_rep.total\_rep,0)) - IFNULL(s\_insp.budget,0) + IFNULL(s\_rep.budget,0) < 0, 'Over Budget', 'Under Budget') AS cur\_status FROM LOA l

LEFT JOIN (SELECT i.insp\_id, i.fk\_LOA\_num as LOA\_num,

SUM(e.hourly\_rate \* ia.hours\_worked) AS total\_insp, SUM(i.budget) as budget FROM inspection i

INNER JOIN insp\_assign ia ON ia.insp\_id = i.insp\_id

INNER JOIN employee e ON e.eid = ia.eid GROUP BY i.insp\_id)

AS s\_insp ON s\_insp.LOA\_num = l.number

LEFT JOIN (SELECT r.rep\_id, r.fk\_LOA\_num as LOA\_num, SUM(e.hourly\_rate \* ra.hours\_worked) AS total\_rep, SUM(r.budget) as budget FROM report r

INNER JOIN rep\_assign ra ON ra.rep\_id = r.rep\_id

INNER JOIN employee e ON e.eid = ra.eid GROUP BY r.rep\_id)

AS s\_rep ON s\_rep.LOA\_num = l.number GROUP BY l.number [HAVING cur\_status = "Over Budget", HAVING cur\_status = "Under Budget", HAVING l.number = ?]

Query Comments: See the comments from the Inspection and Report statuses. This will combine both the total spent and the total budget on the inspections and reports. The totals are compared to determine if an loa is over or under budget.

The user is given several options of filtering the data (see the final group by statement) including the status, loa number, bridge NBI and report number.

The following is used to fill out tables on the webpage:

SELECT \* FROM LOA

Add Employee Site:

**Data Manipulation**

UPDATE employee SET hourly\_rate = [?] WHERE eid = [?]

INSERT INTO employee (eid, first\_name, last\_name, hourly\_rate, fk\_OfficeID) VALUES (?, ?, ?, ?, ?)

**Gets data for the site**

SELECT e.eid, e.first\_name, e.last\_name, e.hourly\_rate, o.name AS office\_name FROM employee e

INNER JOIN office o ON e.fk\_Officeid = o.id ORDER BY e.eid

SELECT \* FROM office

Add Report Site:

**Data Manipulation**

INSERT INTO report (type, budget, pcomp, fk\_bridgeID, fk\_LOA\_num) VALUES (?, ?, ?, ?, ?)

**Gets data for the site**

SELECT r.rep\_id, r.type, r.budget, r.pcomp, LOA.number, b.NBI\_ID FROM report r

INNER JOIN LOA ON r.fk\_LOA\_num = LOA.id

INNER JOIN bridge b ON b.id = r.fk\_bridgeID ORDER BY LOA.number

SELECT \* FROM LOA ORDER BY number

SELECT \* FROM bridge ORDER BY NBI\_ID

Add Inspection Site:

**Data Manipulation**

INSERT INTO inspection (type, budget, pcomp, fk\_bridgeID, fk\_LOA\_num) VALUES (?, ?, ?, ?, ?)

**Gets data for the site**

SELECT i.insp\_id, i.type, i.budget, i.pcomp, LOA.number, b.NBI\_ID FROM inspection i

INNER JOIN LOA ON i.fk\_LOA\_num = LOA.id

INNER JOIN bridge b ON b.id = i.fk\_bridgeID ORDER BY LOA.number

SELECT \* FROM LOA ORDER BY number

SELECT \* FROM bridge ORDER BY NBI\_ID

Add LOA Site:

**Data Manipulation**

INSERT INTO LOA (number) VALUES (?)

**Gets data for the site**

SELECT \* FROM LOA ORDER BY number

Add Bridge Site:

**Data Manipulation**

INSERT INTO bridge (nbi\_id, type, length, spans, zipcode) VALUES (?, ?, ?, ?, ?)

**Gets data for the site**

SELECT \* FROM bridge ORDER BY NBI\_ID

Add Office Site:

**Data Manipulation**

INSERT INTO office (name, zipcode) VALUES (?, ?)

**Gets data for the site**

SELECT \* FROM office ORDER BY name

Assign Report Site:

**Data Manipulation**

UPDATE rep\_assign SET hours\_worked = ? WHERE eid = ? AND rep\_id = ?

INSERT INTO rep\_assign (eid, rep\_id, hours\_worked) VALUES (?, ?, ?)

DELETE FROM rep\_assign WHERE eid = ? AND rep\_id = ?

**Gets data for the site**

SELECT e.eid, e.first\_name, e.last\_name, e.hourly\_rate, o.name AS office\_name FROM employee e

INNER JOIN office o ON e.fk\_Officeid = o.id ORDER BY e.eid

SELECT r.rep\_id, r.type, r.budget, r.pcomp, LOA.number, b.NBI\_ID FROM report r

INNER JOIN LOA ON r.fk\_LOA\_num = LOA.id

INNER JOIN bridge b ON b.id = r.fk\_bridgeID ORDER BY r.rep\_id"

SELECT e.eid, e.first\_name, e.last\_name, r.rep\_id, r.type, b.NBI\_ID, l.number, ra.hours\_worked FROM rep\_assign ra

INNER JOIN employee e ON e.eid = ra.eid

INNER JOIN report r ON r.rep\_id = ra.rep\_id

INNER JOIN bridge b ON b.id = r.fk\_bridgeID

INNER JOIN LOA l ON l.id = r.fk\_LOA\_num ORDER BY e.eid

Assign Inspection Site:

**Data Manipulation**

UPDATE insp\_assign SET hours\_worked = ? WHERE eid = ? AND insp\_id = ?

INSERT INTO insp\_assign (eid, insp\_id, hours\_worked) VALUES (?, ?, ?)

DELETE FROM insp\_assign WHERE eid = ? AND insp\_id = ?

**Gets data for the site**

SELECT e.eid, e.first\_name, e.last\_name, e.hourly\_rate, o.name AS office\_name FROM employee e

INNER JOIN office o ON e.fk\_Officeid = o.id ORDER BY e.eid

SELECT i.insp\_id, i.type, i.budget, i.pcomp, LOA.number, b.NBI\_ID FROM inspection i

INNER JOIN LOA ON i.fk\_LOA\_num = LOA.id

INNER JOIN bridge b ON b.id = i.fk\_bridgeID ORDER BY i.insp\_id

SELECT e.eid, e.first\_name, e.last\_name, i.insp\_id, i.type, b.NBI\_ID, l.number, ia.hours\_worked FROM insp\_assign ia

INNER JOIN employee e ON e.eid = ia.eid

INNER JOIN inspection i ON i.insp\_id = ia.insp\_id

INNER JOIN bridge b ON b.id = i.fk\_bridgeID

INNER JOIN LOA l ON l.id = i.fk\_LOA\_num ORDER BY e.eid"