**Lausn á Dæmi 1: Employees on Projects**

Greining á Vandamálum:

Upprunalega taflan, project\_assignments, er með samsettan frumlykil (project\_id, employee\_id). Hönnunin þjáist af alvarlegu offramboði gagna (

*data redundancy*) sem leiðir til þekktra frávika (*anomalies*)1111:

1. **Innsetningarfrávik (Insertion Anomaly):** Ekki er hægt að skrá nýtt verkefni nema að minnsta kosti einn starfsmaður sé skráður á það, því employee\_id er hluti af frumlyklinum2.
2. **Uppfærslufrávik (Update Anomaly):** Ef verkefni skiptir um nafn (project\_name) eða ef starfsmaður skiptir um deild (employee\_department) þarf að uppfæra upplýsingarnar í mörgum röðum. Ef gleymist að uppfæra eina röð verða gögnin ósamkvæm3.
3. **Eyðingarfrávik (Deletion Anomaly):** Ef síðasti starfsmaðurinn er fjarlægður úr verkefni eyðist röðin og þar með allar upplýsingar um verkefnið sjálft, svo sem nafn þess og fjárhagsáætlun4.

Vandamálið er brot á **annarri staðalformi (2NF)**. Dálkar eins og project\_name eru aðeins háðir project\_id og dálkar eins og employee\_name eru aðeins háðir employee\_id. Þeir eru því háðir aðeins

*hluta* af frumlyklinum, sem kallast hlutaháð (*partial dependency*)5.

Stöðluð Hönnun (3NF):

Til að laga þetta þarf að brjóta töfluna upp í þrjár aðskildar töflur: eina fyrir verkefni, eina fyrir starfsmenn og eina samskeytatöflu (

*junction table*) til að tengja þær saman6.

* projects(project\_id, project\_name, project\_budget)
* employees(employee\_id, employee\_name, employee\_department)
* project\_assignments(project\_id, employee\_id)

**CREATE kóði fyrir nýjar töflur:**

SQL

-- Tafla fyrir verkefni

CREATE TABLE projects (

project\_id INT PRIMARY KEY,

project\_name VARCHAR(255) NOT NULL,

project\_budget DECIMAL(10, 2)

);

-- Tafla fyrir starfsmenn

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

employee\_name VARCHAR(100) NOT NULL,

employee\_department VARCHAR(100)

);

-- Samskeytatafla sem tengir starfsmenn og verkefni

CREATE TABLE project\_assignments (

project\_id INT,

employee\_id INT,

PRIMARY KEY (project\_id, employee\_id),

FOREIGN KEY (project\_id) REFERENCES projects(project\_id) ON DELETE CASCADE,

FOREIGN KEY (employee\_id) REFERENCES employees(employee\_id) ON DELETE CASCADE

);

Rökstuðningur:

Þessi nýja hönnun er í þriðju staðalformi (3NF). Hún leysir öll frávikin með því að tryggja að hver staðreynd sé aðeins geymd á einum stað. Upplýsingar um verkefni eru eingöngu í projects töflunni og upplýsingar um starfsmenn eru eingöngu í employees töflunni. project\_assignments taflan geymir aðeins frumlyklana tvo til að skrá sambandið. Nú er hægt að stofna verkefni án starfsmanna og öfugt, og uppfærsla á nafni eða deild þarf aðeins að eiga sér stað á einum stað.

**Lausn á Dæmi 2: Student Course Registration**

Greining á Vandamálum:

Taflan

course\_enrollments brýtur í bága við **fyrstu staðalform (1NF)**7. Ástæðan er sú að dálkurinn

enrolled\_courses inniheldur ekki atómísk gildi; hann geymir lista af mörgum gildum í einum reit8. Þetta veldur mörgum vandamálum:

* **Erfitt að leita:** Það er flókið og óskilvirkt að leita að öllum nemendum í ákveðnu námskeiði með SQL.
* **Erfitt að uppfæra:** Ef námskeiðsnúmer breytist þarf að framkvæma flókna textaleit og -breytingu í mörgum röðum.
* **Engin heilleikakvilla:** Ekki er hægt að nota erlendan lykil til að tryggja að námskeiðsnúmerin séu til í raun og veru.

Stöðluð Hönnun (3NF):

Lausnin er að brjóta þetta upp í þrjár töflur sem mynda tvö one-to-many sambönd í gegnum samskeytatöflu. Þetta er rétta leiðin til að útfæra many-to-many samband milli nemenda og námskeiða.

* students(student\_id, student\_name, email)
* courses(course\_id, course\_name)
* enrollments(student\_id, course\_id)

**CREATE kóði fyrir nýjar töflur:**

SQL

-- Tafla fyrir nemendur

CREATE TABLE students (

student\_id INT PRIMARY KEY,

student\_name VARCHAR(100) NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL

);

-- Tafla fyrir námskeið

CREATE TABLE courses (

course\_id VARCHAR(10) PRIMARY KEY, -- Using VARCHAR for course codes like 'CS101'

course\_name VARCHAR(255) NOT NULL

);

-- Samskeytatafla fyrir skráningar

CREATE TABLE enrollments (

student\_id INT,

course\_id VARCHAR(10),

PRIMARY KEY (student\_id, course\_id),

FOREIGN KEY (student\_id) REFERENCES students(student\_id) ON DELETE CASCADE,

FOREIGN KEY (course\_id) REFERENCES courses(course\_id) ON DELETE RESTRICT

);

Rökstuðningur:

Með þessari hönnun er hvert gagn atómískt og geymt í viðeigandi töflu. Hver skráning nemanda í námskeið er nú sérstök röð í enrollments töflunni. Það er nú einfalt að finna alla nemendur í námskeiði með JOIN fyrirspurn og uppfærsla á nafni námskeiðs þarf aðeins að gerast á einum stað í courses töflunni. Gagnaintegritet er tryggt með erlendum lyklum9.

**Lausn á Dæmi 3: Web Store Orders**

Greining á Vandamálum:

Þetta er skýrt dæmi um brot á annarri staðalformi (2NF). Frumlykillinn er samsettur úr (order\_id, product\_id). Vandamálið er að dálkarnir

product\_name og price\_per\_item eru ekki háðir *öllum* lyklinum10. Þeir eru aðeins háðir

product\_id. Þetta er hlutaháð (

*partial dependency*) 11 og veldur:

* **Innsetningarfráviki:** Ekki er hægt að bæta nýrri vöru við kerfið nema hún sé hluti af pöntun.
* **Uppfærslufráviki:** Ef nafn vöru breytist þarf að uppfæra það í öllum pöntunarlínum þar sem varan kemur fyrir, sem er uppskrift að ósamkvæmni12.

Stöðluð Hönnun (3NF):

Við leysum þetta með því að skipta töflunni í tvennt:

* products(product\_id, product\_name, current\_price)
* order\_lines(order\_id, product\_id, quantity, price\_at\_purchase)

**CREATE kóði fyrir nýjar töflur:**

SQL

-- Tafla fyrir vörur

CREATE TABLE products (

product\_id INT PRIMARY KEY,

product\_name VARCHAR(200) NOT NULL,

current\_price DECIMAL(8, 2) NOT NULL

);

-- Tafla fyrir pöntunarlínur

CREATE TABLE order\_lines (

order\_id INT,

product\_id INT,

quantity INT NOT NULL,

price\_at\_purchase DECIMAL(8, 2) NOT NULL, -- Storing the price at the time of order

PRIMARY KEY (order\_id, product\_id),

-- Assuming an 'orders' table exists: FOREIGN KEY (order\_id) REFERENCES orders(order\_id),

FOREIGN KEY (product\_id) REFERENCES products(product\_id) ON DELETE RESTRICT

);

Rökstuðningur:

Nú eru upplýsingar um vörur geymdar á einum stað í products töflunni. order\_lines taflan inniheldur aðeins upplýsingar sem eru háðar bæði pöntun og vöru (magn) og geymir afrit af verðinu eins og það var þegar pöntunin var gerð. Þetta uppfyllir minnisregluna: "Every non-key attribute must provide a fact about...

**the whole key**..."13.

**Lausn á Dæmi 4: Instructor Directory**

Greining á Vandamálum:

Þessi tafla er í 2NF (þar sem hún hefur ekki samsettan frumlykil) en brýtur í bága við

**þriðju staðalform (3NF)**14. Ástæðan er sú að hún inniheldur yfirfærsluháð (

*transitive dependency*)15. Frumlykillinn er

instructor\_id. Við höfum eftirfarandi háð:

instructor\_id → department\_name

department\_name → department\_head

Hér ákvarðar dálkur sem ekki er lykill (

department\_name) annan dálk sem ekki er lykill (department\_head)16. Þetta veldur:

* **Uppfærslufráviki:** Ef deild fær nýjan deildarstjóra þarf að uppfæra department\_head í röð hvers einasta kennara í þeirri deild.
* **Eyðingarfráviki:** Ef síðasti kennarinn yfirgefur deild glatast allar upplýsingar um deildina, þar með talið hver stýrði henni17.

Stöðluð Hönnun (3NF):

Við skiptum töflunni í tvennt til að fjarlægja yfirfærsluháðið:

* departments(department\_name, department\_head)
* instructors(instructor\_id, instructor\_name, department\_name)

**CREATE kóði fyrir nýjar töflur:**

SQL

-- Tafla fyrir deildir

CREATE TABLE departments (

department\_name VARCHAR(150) PRIMARY KEY,

department\_head VARCHAR(100)

);

-- Tafla fyrir kennara

CREATE TABLE instructors (

instructor\_id INT PRIMARY KEY,

instructor\_name VARCHAR(100) NOT NULL,

department\_name VARCHAR(150),

FOREIGN KEY (department\_name) REFERENCES departments(department\_name) ON DELETE SET NULL

);

Rökstuðningur:

Núna eru upplýsingar um deildir (nafn og stjórnandi) aðeins geymdar á einum stað. instructors taflan vísar aðeins í deildina með erlendum lykli. Þetta uppfyllir minnisregluna: "...and

**nothing but the key**."18. Staðreyndir um deildina eru nú í deildartöflunni, ekki í kennaratöflunni.

**Lausn á Dæmi 5: Library Loans**

Greining á Vandamálum:

Þessi tafla, loans\_unnormalized, þjáist af bæði offramboði og yfirfærsluháði, og brýtur því í bága við bæði 2NF og 3NF (þótt tæknilega séð sé það 3NF sem er brotið þar sem lykillinn er einfaldur).

1. **Offramboð:** member\_name og member\_address eru endurtekin fyrir hvert einasta lán sem sami lánþeginn tekur. Þetta veldur uppfærslu- og innsetningarfrávikum.
2. **Yfirfærsluháð (Transitive Dependency):** Við höfum keðjuna loan\_id → isbn → book\_title. book\_title er staðreynd um bókina (isbn), ekki um lánið (loan\_id).

Stöðluð Hönnun (3NF):

Við þurfum þrjár aðskildar töflur til að geyma upplýsingar um hverja einingu (entity) fyrir sig.

* books(isbn, book\_title)
* members(member\_id, member\_name, member\_address)
* loans(loan\_id, isbn, member\_id, loan\_date)

**CREATE kóði fyrir nýjar töflur:**

SQL

-- Tafla fyrir bækur

CREATE TABLE books (

isbn VARCHAR(13) PRIMARY KEY,

book\_title VARCHAR(255) NOT NULL

);

-- Tafla fyrir lánþega

CREATE TABLE members (

member\_id INT PRIMARY KEY,

member\_name VARCHAR(100) NOT NULL,

member\_address VARCHAR(255)

);

-- Tafla fyrir útlán

CREATE TABLE loans (

loan\_id INT PRIMARY KEY,

isbn VARCHAR(13) NOT NULL,

member\_id INT NOT NULL,

loan\_date DATE NOT NULL,

FOREIGN KEY (isbn) REFERENCES books(isbn) ON DELETE RESTRICT,

FOREIGN KEY (member\_id) REFERENCES members(member\_id) ON DELETE CASCADE

);

Rökstuðningur:

Með því að aðskilja gögnin í þrjár töflur er hver staðreynd nú geymd nákvæmlega einu sinni. loans taflan virkar sem tenging milli members og books og inniheldur aðeins gögn sem eiga við lánið sjálft (eins og loan\_date). Þetta útilokar offramboð og leysir öll frávikin. Ef lánþegi flytur þarf aðeins að uppfæra heimilisfangið á einum stað í members töflunni. Hönnunin fylgir nú stöðlunarreglum að fullu.