Page Rank: Google Search

IN831: Auxiliar2 – Tarea1 – Parte 2

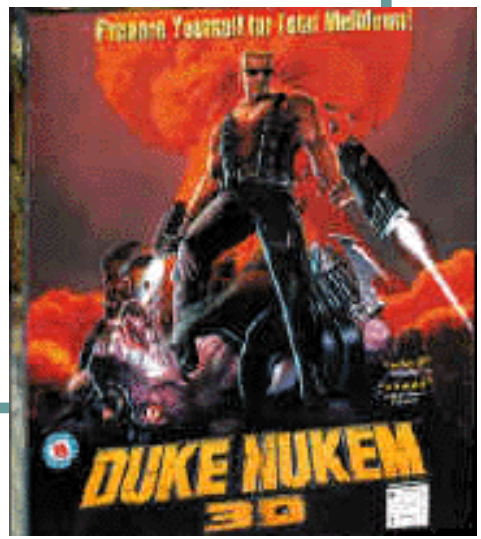
Pablo Roman

Tarea1: Googlito

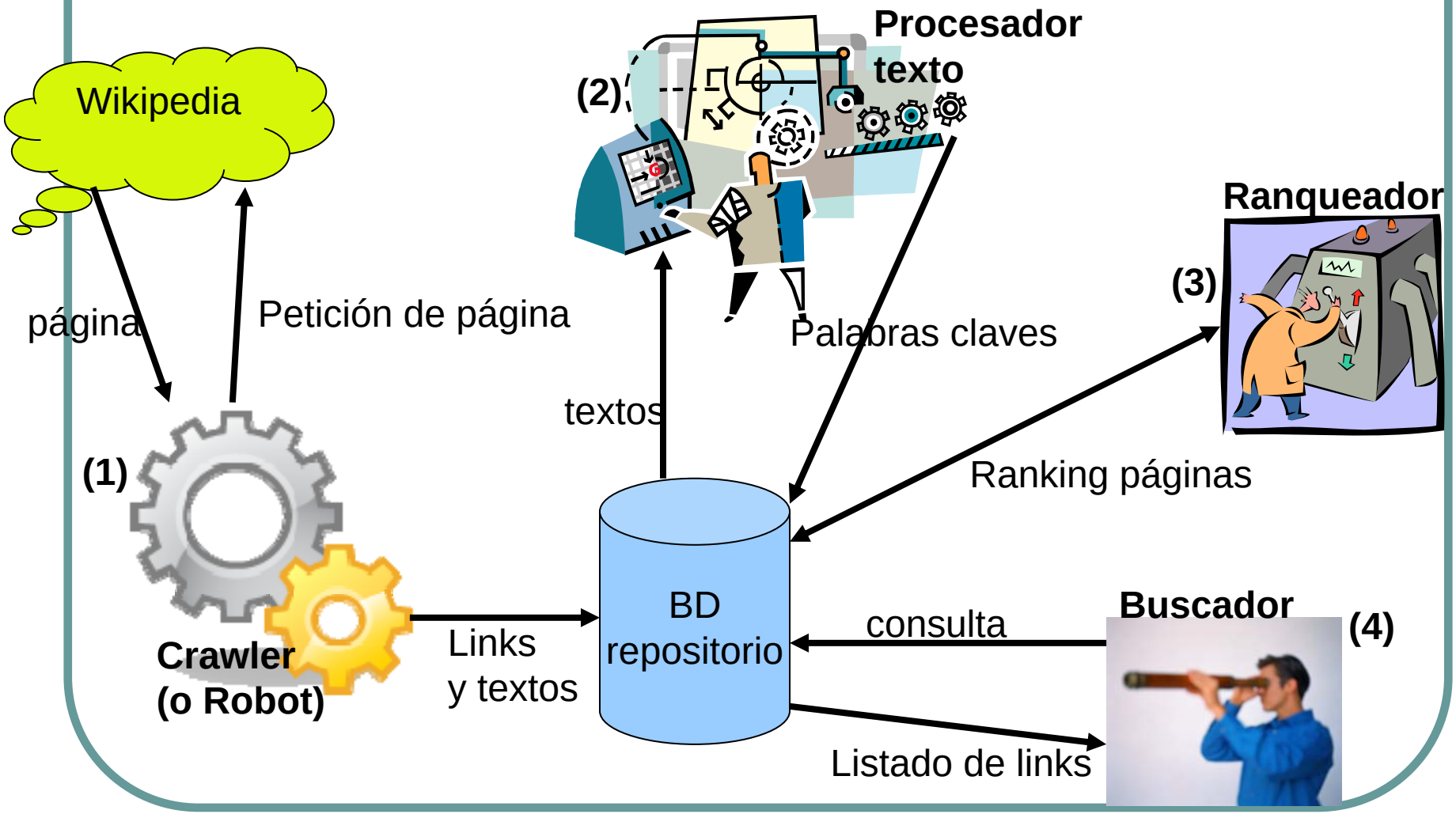
- Aplicación que:
  - Busque y “capture el texto” de paginas de Internet, datos que se almacenan en un repositorio.
  - Un buscador que dado una palabra clave liste las 10 paginas con mejor PageRank.

Tarea1: Googlito

- Sitio Wikipedia (en.wikipedia.org)
  - Video Game
  - Toy
- Máximo de 300-1000 páginas
- Debe por ejemplo poder encontrar “Duke Nukem”.



Googlito: Estructura muy general



Segunda Fase del proyecto

- Tenemos un crawler que almacena páginas, contenido de texto (palabras) y links entre páginas.
- Tenemos un buscador que dadas las palabras claves encuentra las paginas asociadas.

Segunda Fase del proyecto: Que falta?

- Calcular el ranking de las páginas, de acuerdo al algoritmo PageRank.
- Mejorar el resultado de las búsquedas incorporando algún tipo de score a las palabras encontradas y mejorar el buscador.
- Comparar con google: haciendo búsquedas como “Duke Nukem site:en.wikipedia.org”.
- Informe con el proyecto, evaluación del sistema, algoritmos e instalación.

Algoritmo PageRank© for beginners

- Matrices estocásticas
- Teoría de cadenas de markov
- Cálculo (de verdad) de vectores propios de matrices estocásticas de 1000x1000

Pagerank: El valor de un algoritmo

- Patentado Brin&Page 1998
- Valor numérico de importancia de una página respecto de otras.
- Esta claro que es parte del algoritmo real que usa Google, el verdadero algoritmo tiene un valor incalculable.
- Motivación de mas para mejorar la búsqueda que hace su tarea.

Pagerank: Intuición

- Una página recomienda a otra si tiene un link hacia ella.
- Páginas más recomendadas por otras son más importantes, valor de importancia de página i: Π_i
- Una página al recomendar a otra le agrega una fracción de su valor de importancia, página k recomienda a página i que recibe: $H_{ki}\,\Pi_k$, $H_{ki}<1$.
- $\Pi_i = \Sigma_k\, \Pi_k\, H_{ki}$ -> $\Pi^t = \Pi^t\, H$, Π es vector propio de H.
- Las fracciones de importancia deben sumar 1, $\Sigma_i\, H_{ki}=1$
- H: es entonces una matriz estocástica.
- En la teoría de las cadenas de markov las matrices estocásticas son probabilidades de transicion.
- ¿CÓMO ENTENDEMOS ESTAS PROBABILIDADES H_{ki}?

Pagerank: ¿Cómo entendemos H? The random surfer

- Necesitamos un modelo del comportamiento de navegación de los usuarios.
- Consideremos un personaje muy simple:
  - navega siguiendo los links entre páginas y tiene una probabilidad “d” (aprox 0.85) de NO dejar de seguir links.
  - Si deja de seguir links entonces empieza en otro cualquiera equiprobable.
- Elije el próximo links en una página de manera equiprobable.

Pagerank: H la matriz de probabilidad de transición del random surfer.

- Cada nodo es una pagina.
- Hay un nodo adicional: El nodo “salida”
- Cada transición es un link.
- Hay transiciones adicionales: las transiciones a la salida, con probabilidad “d”.
- Todos los estados son transientes. H es irreducible y aperiódica.
- **El tiempo que pasa el random surfer en cada nodo será proporcional a la importancia de la página.**
- Π_k : la probabilidad de estar en k
- $\Pi_k = P(k|\text{no salir})P(\text{no salir}) + P(\text{recomenzar en k})(1-P(\text{no salir}))$
  - $P(\text{no salir}) = d$
  - $P(\text{recomenzar en i}) = 1/n$ (n: el número de páginas)
  - $P(k|\text{no salir}) = \Sigma_k \Pi_k S_{ki}$
- $H_{ki} = d*S_{ki} + (1-d)/n$
- S_{ki}: probabilidad de transición de k->i = 1/(número de links que salen de k).

Pagerank: H la matriz de probabilidad de transición del random surfer.

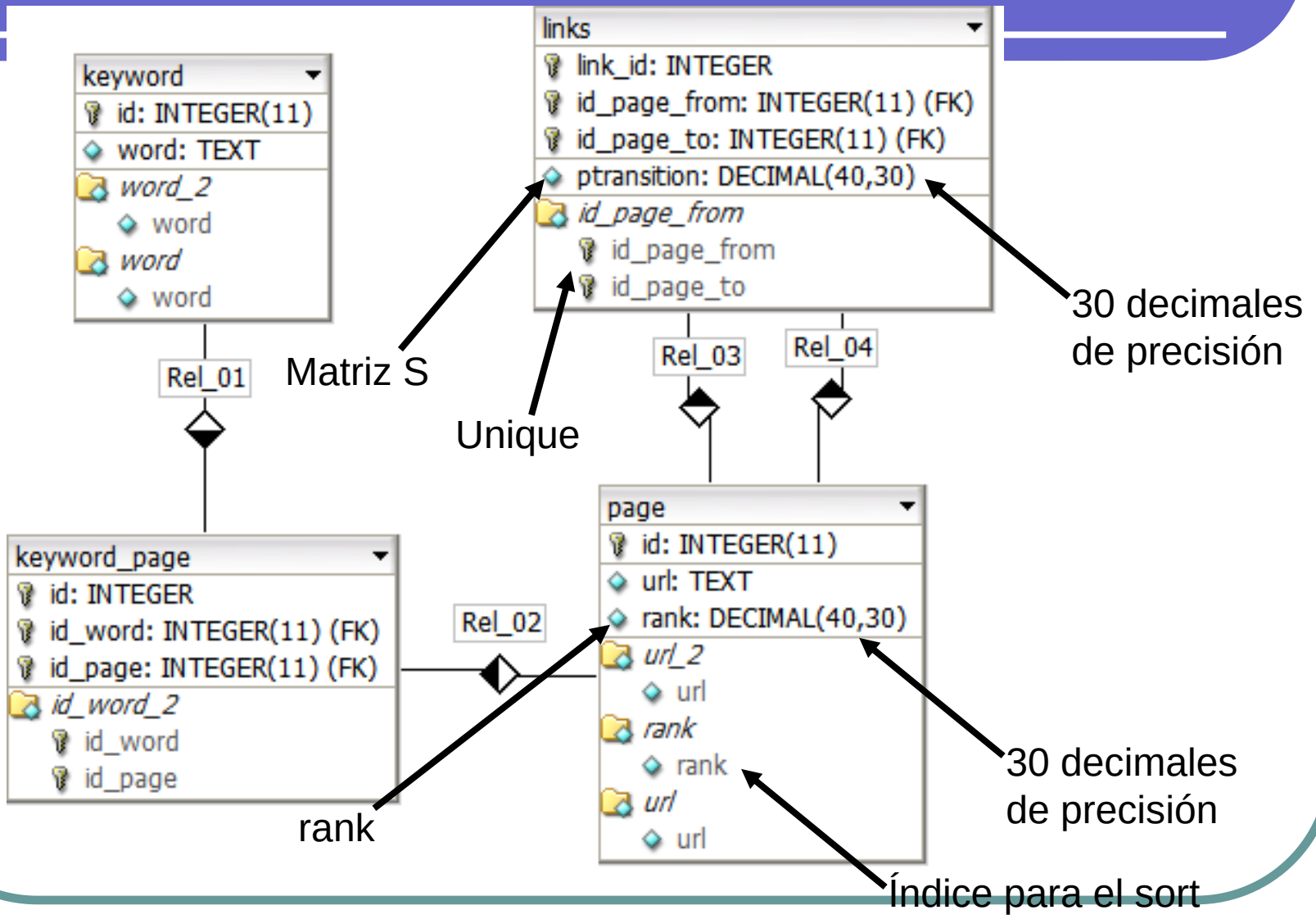
- $\Pi^t = \Pi^t H, \Sigma_i \Pi_i = 1$
- Π_i: probabilidad estacionaria o probabilidad de estar en i a largo plazo o la proporción del tiempo que paso el random surfer en el nodo i.
- $\Pi_i = lim_{n\to\infty} (H^{(n)})_{ki}$

1. Iterar potencias de la matriz H (muy costoso !!) o
2. O resolver el sistema lineal o
3. Iterar $\Pi^{(l)}$:
   - $\Pi_i^{(l+1)} = (1-d)/n + d*\Sigma_k \Pi_k^{(l)} S_{ki}$
   - $\Pi_i^{(0)} = 1/n$ <- *Ergodicidad nos permiten partir de cualquier vector.*
   - *HAY QUE NORMALIZAR EN CADA ITERACION.*

Implementación

- Almacenar matrices de tamaño nxn y multiplicarlas es muy costoso.
- Guardaremos el vector Π con la tabla page.
- Guardaremos la matriz S con la tabla de links
- La base de datos efectuará la iteración:
  - Maneja mejor la memoria.
  - Esta construida para efectuar estos tipos de cálculos.
  - OJO: Precisión numérica, implica usar tipo de dato DECIMAL(n,m).

Implementación: Tablas



Implementación: Precisión numérica en MySQL

- Valores numéricos “normales” efectúa aproximaciones burdas.
- Debe indicarse al motor que se esta trabajando con datos con gran precisión numérica.
- usar DECIMAL(n,m) : indica precisión de n dígitos numéricos totales y m decimales (m<n).
- Ejemplo: para hacer “1/n” usar CONVERT

CONVERT(1, DECIMAL(40,30))/CONVERT(n, DECIMAL(40,30))

Los datos del crawler: links

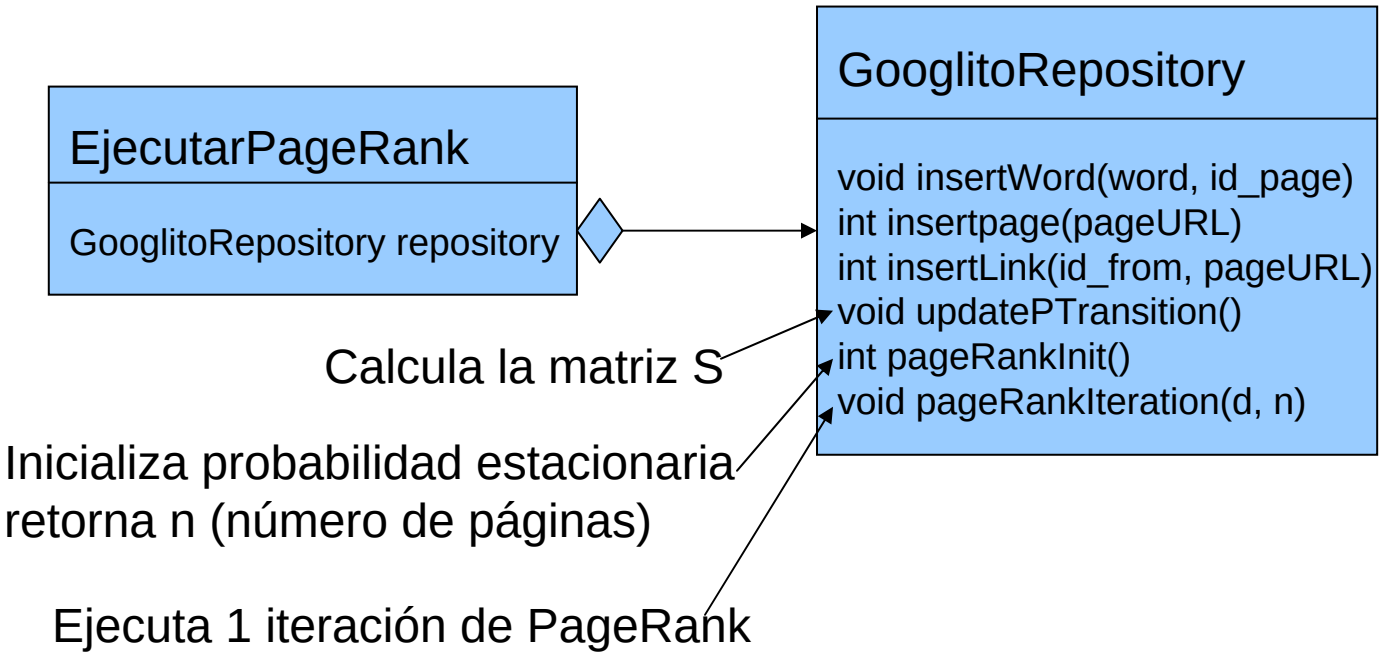
- En cada visita a página i
- Busca cada link j de la pagina
- Si shouldVisit(j) es true
- Trato de insertar j como nueva page
- Inserto (i,j) en link

- Resultado: Para video_games 241.418 links.

Los datos del crawler: page

- Solo se bajan 1000 páginas con todos sus links.
- Pero se registran 93.483 páginas en la tabla.
- La mayoría son solo apuntadas.
- El calculo de pagerank debe hacerce solo a las 1000 que se bajaron y que son apuntadas.
- Por lo que en la tabla links hay varios links que tampoco son útiles para el calculo.
- Truco: en page se actualizo con rank=0 a aquellas paginas que no se encuentran en el dominio.
- De haber querido cargar el total de las 90000 paginas se hubieran necesitado unos 100G de espacio en disco y 2-3 días de carga. Además de modificar la configuración de MySQL.

La carga de pagerank



EjecutarPageRank

```
public static void main(String[] args) {
        int n;
        System.out.println("Inicializando rank ...");
        // Inicializo los valores de page rank
        n=repository.pageRankInit();

        System.out.println("Inicializando matriz S ...");
        // Actualizo la matriz S que esta implicita en la tabla links.
        repository.updatePTransition();


        System.out.println("Ejecutando PageRank ...");

        for (int k=1; k<=100; k++) {
                System.out.println("Iteracion " + k);
                // iteracion de PageRank
                repository.pageRankIteration(0.85, n);
        }
   }
```

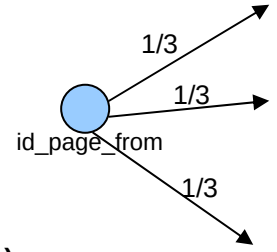
Actualizando la matriz S

Cuento cuantas salen de un mismo origen

Actualizo a 1/c la prob

```
public synchronized void updatePTransition() {
        Statement stmt1, stmt2;
        ResultSet rs1;
        int id_page_from, c;
        String query="";
        try {
                stmt1 = conn.createStatement();

                query = "Select id_page_from, count(*) as c from links group by id_page_from";
                rs1=stmt1.executeQuery(query);
                while (rs1.next()) {
                        id_page_from=rs1.getInt("id_page_from");
                        c=rs1.getInt("c");
                        stmt2 = conn.createStatement();
                        query="update links set ptransition=CONVERT(1,
DECIMAL(40,30))/CONVERT("+c+", DECIMAL(40,30)) where id_page_from="+id_page_from;
                        stmt2.execute(query);
                        stmt2.close();
                        stmt2=null;
                }
                stmt1.close();
                stmt1=null;
                rs1.close();
                rs1=null;
        } catch (SQLException ex) {
                System.out.println("SQLException: " + ex.getMessage());
                System.out.println("SQLState: " + ex.getSQLState());
                System.out.println("VendorError: " + ex.getErrorCode());
                System.out.println("SQL: " + query);
        }
        notifyAll();
}
```



Inicializo la probabilidad estacionaria

```
public synchronized int pageRankInit() {
        …
        try {
                    stmt = conn.createStatement();
                    // obtenemos el numero real de paginas crawleadas
                    query = "Select count(distinct id_page_from) as n from links";
                    rs=stmt.executeQuery(query);
                    if (rs.next()) {n=rs.getInt("n"); }
                    // actualizamos todo en 0
                    query="update page set rank=0";
                    stmt.execute(query);
                    //Iteramos sobre las paginas validas
                    stmt2 = conn.createStatement();
                    query = "Select distinct id_page_from as i from links";
                    rs=stmt.executeQuery(query);
                    while (rs.next()) {
                                i=rs.getInt("i");
                                // actualizamos el rank de las paginas validas en 1/n
                                query="update page set rank=CONVERT(1, DECIMAL(40,30))/CONVERT("+n+",
DECIMAL(40,30)) where id="+i;
                                stmt2.execute(query);
                    }
                                stmt.close(); stmt=null;
                    rs.close(); rs=null;
                    stmt2.close(); stmt2=null;
        } catch (SQLException ex) {
                    …
        }
        notifyAll();
        return(n);
}
```

Todo queda en 0

Actualiza solo para aquellas paginas que fueron leídas, a 1/n

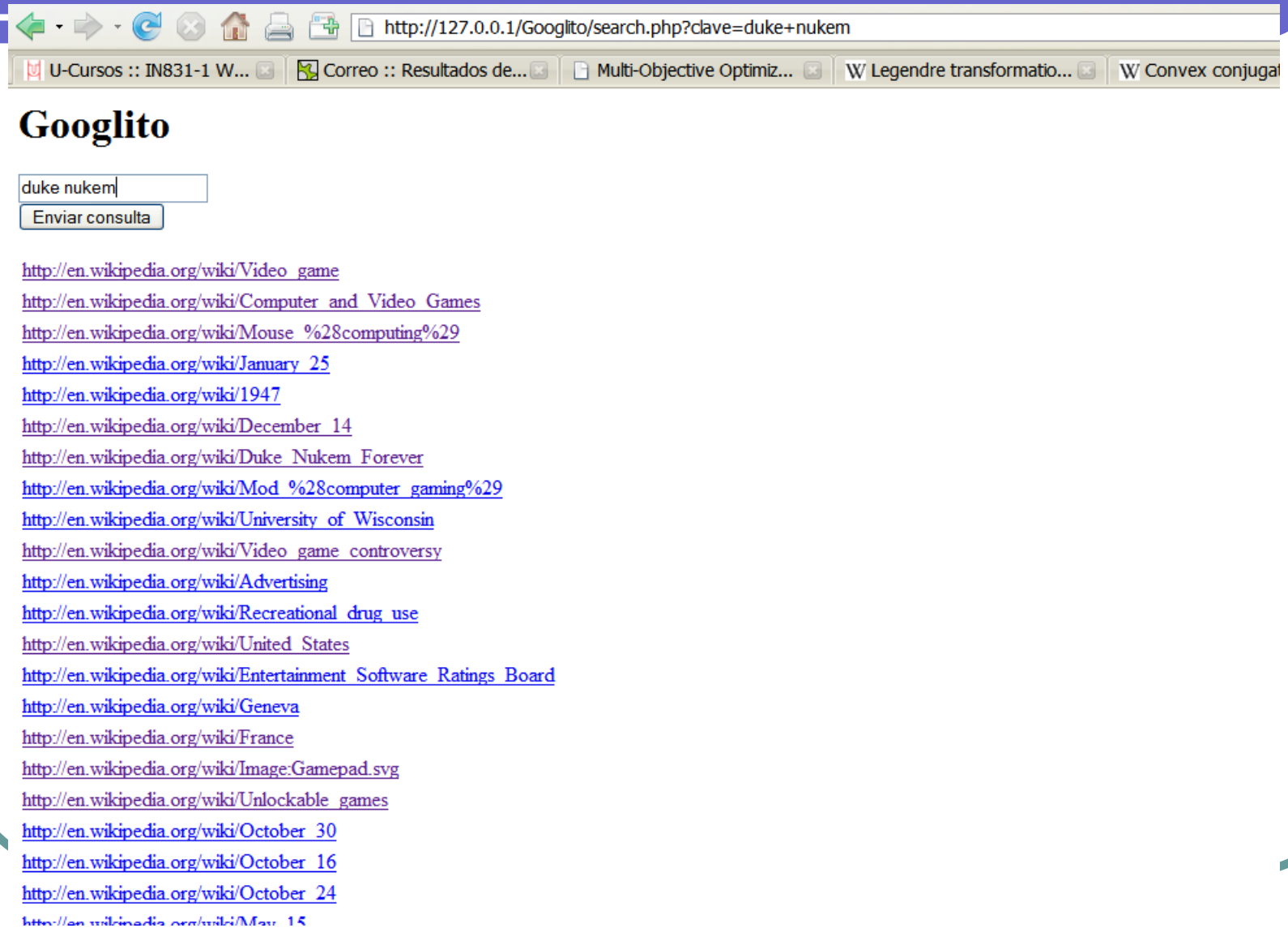
Iteración Pagerank

```
public synchronized void pageRankIteration(double d /*es la probabilidad de salir*/, int n /*numero de paginas*/)
{
        …
        try {
                    stmt1 = conn.createStatement();
                    // la iteracion
                    query = "Select id_page_from, (1-CONVERT("+d+",
DECIMAL(40,30)))/CONVERT("+n+",DECIMAL(40,30)) + CONVERT("+d+", DECIMAL(40,30))*sum(ptransition*rank) as r from
links, page where page.id=id_page_to and page.rank!=0 group by id_page_from";
                    rs1=stmt1.executeQuery(query);
                    while (rs1.next()) {
                                i=rs1.getInt("id_page_from");
                                r=rs1.getDouble("r");
                                stmt2 = conn.createStatement();
                                query="update page set rank="+r+" where id="+i;
                                stmt2.execute(query);
                                stmt2.close();
                                stmt2=null;
                    }
                    // vamos a normalizar
                    query = "Select sum(rank) as s from page where page.rank!=0";
                    rs1=stmt1.executeQuery(query);
                    if (rs1.next()) {s=rs1.getDouble("s"); }
                    query = "update page set rank=rank/CONVERT("+s+", DECIMAL(40,30)) where page.rank!=0";
                    stmt1.execute(query);
                    stmt1.close();          stmt1=null;
        rs1.close(); rs1=null;
        } catch (SQLException ex) {
                    …
        }
        notifyAll();
}
```

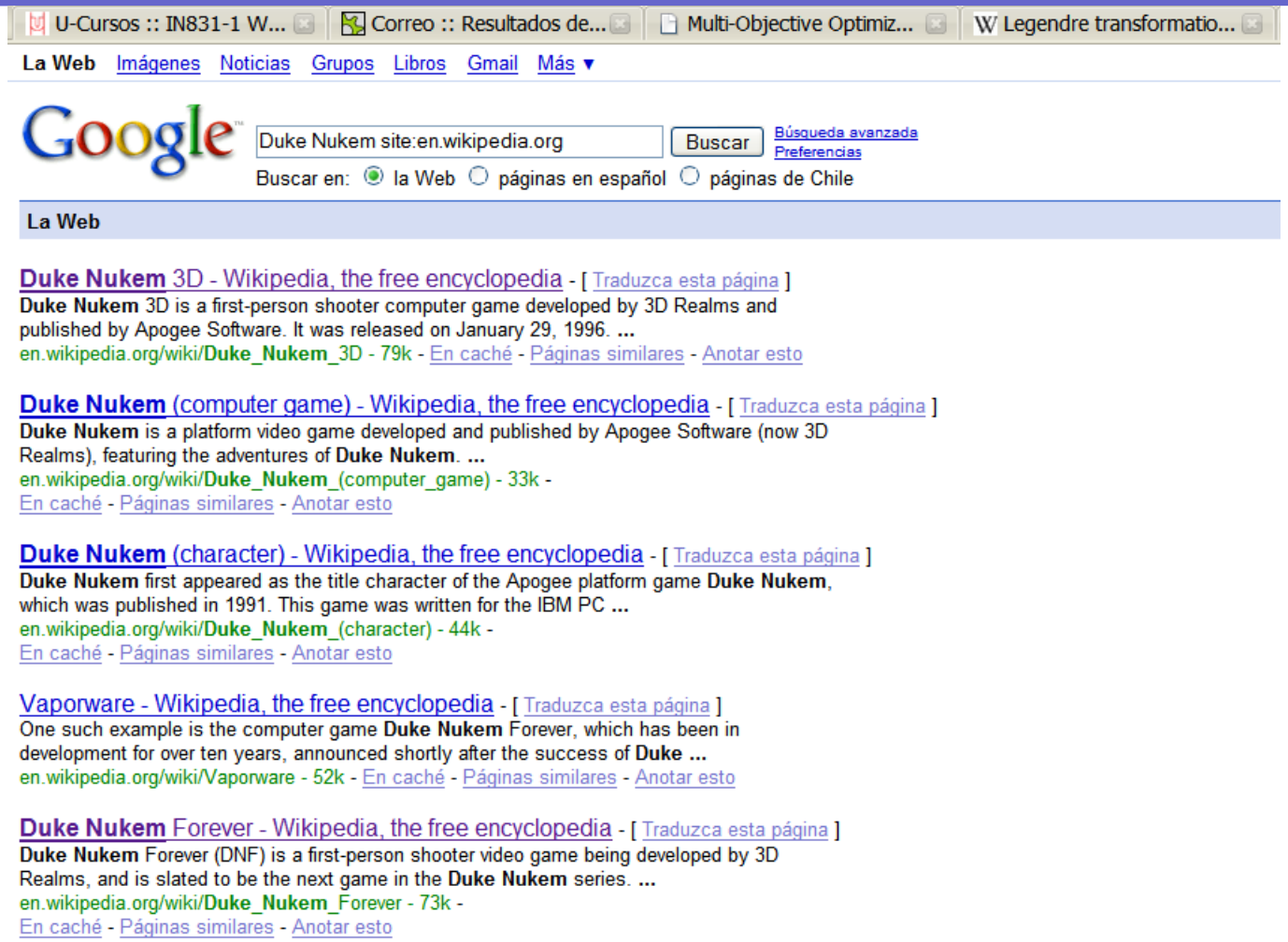
$$\Pi_i^{(l+1)} = (1-d)/n + d*\Sigma_k \Pi_k^{(l)} S_{ki}$$

Normalizo

Resultado


http://127.0.0.1/Googlito/search.php?clave=duke+nukem
U-Cursos :: IN831-1 W...
Correo :: Resultados de...
Multi-Objective Optimiz...
Legendre transformatio...
Convex conjugat
Googlito
duke nukem
Enviar consulta
http://en.wikipedia.org/wiki/Video_game
http://en.wikipedia.org/wiki/Computer_and_Video_Games
http://en.wikipedia.org/wiki/Mouse_%28computing%29
http://en.wikipedia.org/wiki/January_25
http://en.wikipedia.org/wiki/1947
http://en.wikipedia.org/wiki/December_14
http://en.wikipedia.org/wiki/Duke_Nukem_Forever
http://en.wikipedia.org/wiki/Mod_%28computer_gaming%29
http://en.wikipedia.org/wiki/University_of_Wisconsin
http://en.wikipedia.org/wiki/Video_game_controversy
http://en.wikipedia.org/wiki/Advertising
http://en.wikipedia.org/wiki/Recreational_drug_use
http://en.wikipedia.org/wiki/United_States
http://en.wikipedia.org/wiki/Entertainment_Software_Ratings_Board
http://en.wikipedia.org/wiki/Geneva
http://en.wikipedia.org/wiki/France
http://en.wikipedia.org/wiki/Image:Gamepad.svg
http://en.wikipedia.org/wiki/Unlockable_games
http://en.wikipedia.org/wiki/October_30
http://en.wikipedia.org/wiki/October_16
http://en.wikipedia.org/wiki/October_24

Comparación con Google



¿Por qué no es igual?

- Algoritmo real es secreto: PageRank no toma en cuenta los textos para rankear!!
- Faltan páginas que cargar
- Se sabe que el algoritmo secreto:
  - Toma en cuenta la relevancia de las palabras en el texto.
  - Toma en cuenta el comportamiento de usuarios (varias formas de capturarlo)

Tarea 1 – Parte 2

- Deben proponer mejoras al sistema:
  - Mejorar ajuste de la salida a google.
  - Mejorar presentación del search, que se parezca a google.
- Grupo con mejor ajuste a google en 5 primeras salidas tendrá 1 punto de bonus.

Tarea: Hints

- En tabla keyword_page agregar un número (keyword_score) que indique la relevancia de la palabra en la página.
  - Si esta en el titulo -> keyword_score +=T
  - Si aparece n veces en el texto -> keyword_score += n*c
- Modificar el search para que ordene según una función score=score(keyword_score, rank)
- Quizás podrían almacenar ese score en la tabla keyword_page y ordenar los resultados según SUM(keyword_page) para todos los key que se consultaron para esa pagina….

Tarea: Hints

- Mejorar el crawler
  - Afinar búsquedas relativas a Toy y Video_Games: que no busque en otros lados.
  - Que guarde paginas irrelevantes.
  - Afinar selección de palabras a guardar: que no guarde palabras no útiles.
- Quizás mejorar la iteración para que indique convergencia.
- Cargar desde varios orígenes para cubrir mejor los temas indicados.

Entrega

- 5 Mayo: Entrega proyecto completo.
- 2 personas.
- Informe completo.
- Códigos
- Base de datos
- proman@ing.uchile.cl

Corrección

- Informe: Igual que todos los informes de proyectos.
- Código:
  - Que compile.
  - Que tenga la intención de hacer lo que se dice en el informe.
  - Que instalación (BD y código) funcione como lo dice el informe.
  - Que se ejecute sin errores.
- Sistema: Se tomaran varios consultas a Google relativas al tema y se compararan los 5 primeros resultados. Si hay grupos empatados se agregan mas consultas hasta llegar a 1 solo grupo.