Analyzing (Baseball Data) for Streaks Using MapReduce

Ross Anderson

# Introduction

For my MapReduce Final Project, I elected to study baseball data to attempt to find streaks (in a similar method to how Elias Sports Bureau, among others perform the task). While the scope of this project will not come anywhere near the streaks/statistical correlations that are frequently provided by ESPN’s SportsCenter, this was an attempt to start down the path of how it could be done. It should also be keep in mind, that as the purpose of the course is in regards to Hadoop MapReduce, the intent of this project was an application of the course material, not necessarily ground breaking research in baseball history.

The source code for the project is available in GitHub at the following address: <https://github.com/ande8331/BigDataFinalProject>. The files are organized into the following directories:

lib - Libraries

scripts - Python script for parsing data

src - Java files for project

run - Jar file/hdfs files not readily available from web

writeup - This document

# Data Source

Data for this project was obtained from retrosheet.org. The data consists of historical play by play records from Major League Baseball games. The data goes as far back as 1940, however, prior to 1974, the records may be missing one or more games per year, so the records cannot be considered 100% complete (or accurate as a missing game could have been one to break a streak, or could have extended it). Also, as the data does not go back to the start of Major League Baseball, they cannot be considered “official” and any records could not be considered official due to the data discrepancies.

# Data Pre-Processing Required

The data obtained from Retrosheet consists of an entire year’s worth or records in a zip file. The files consist of:

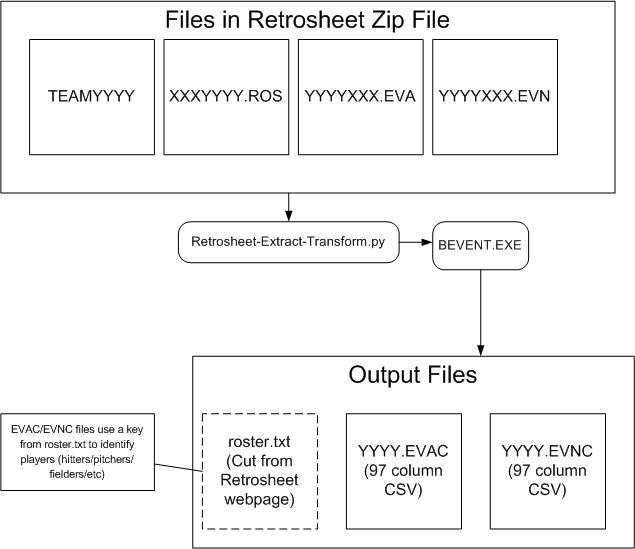
* One Event file for each ballpark (contains all of the game records that occurred in that ballpark)
  + File is named <4 digit year><3 Letter Team Abbreviation>.EV<League>
    - File extension “.EVA” for American League Teams
    - File extension “.EVN” for National League Teams
* One file for Translating Team Abbreviations to League and Team Name (Not used at this time)
  + File is named TEAM<4 digit year>
* One Roster file for each name for translating Player Ids to Player Names, as well as some characteristics of each player (Batting/Throwng hand, position)
  + File is named <Team Abbreviation><4 digit year>.ROS

To transform the event files provided into a CSV format that expands the records out to a more table oriented format, the utility BEVENT has been provided by retrosheet.org. For ease of use, I created a python script (in GitHub directory scripts/Retrosheet-Extract-Transform.py). This script unzips each year file, runs the BEVENT command and creates a <4 digit year>.EVAC file for American League teams, and <4 digit year>.EVNC file for National League teams.

After all of the files have been converted into the .EVAC/.EVNC files, the total disk space is approximately 4.1GB (80 MB/year for later years), and a total of just over 1 million records. Earlier years consume less due to less games being played in a season, fewer teams, missing records, and statistics that weren’t tracked at the time (for instance, pitch sequence).

# Data Description and Schema

After running the BEVENT command to expand the data, the data consists of a 97 column CSV file. The 97 columns represent batter/pitcher/each fielding position, score, outs, and numerous flags to indicate if the event was designated as an “at-bat”, if a double play occurred, if the record represents an plate appearance (steals, pickoffs, balks, and wild-pitches cause the plate appearance to be repeated to reflect the movement of runners). Full details of the column identifiers, and descriptions of how to use them are available at this path: <http://www.retrosheet.org/datause.txt> (See the BEVENT heading).



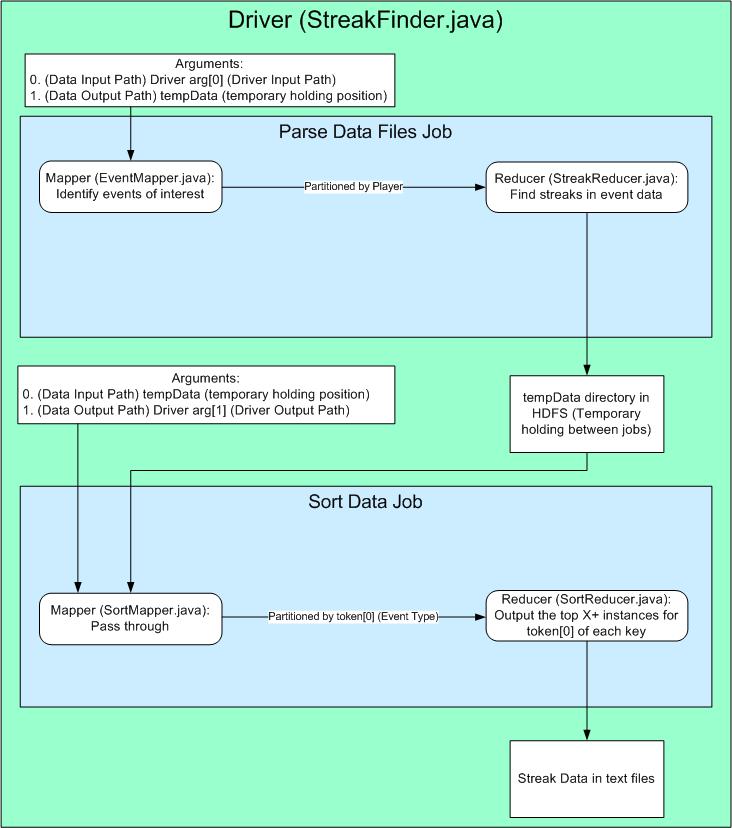
## Figure - Data Schema

# Bad Data Issues

The only observed issue was while extracting data using the BEVENT utility, 4 instances of “too many assists on play” were reported. I did research one instance of this, and its an example of how difficult it can be to model such data. On 9/15/11, in the 4th inning of the Kansas City Royals/Chicago White Sox game, a double play was recorded with what appears to be 6 fielders getting an assist on what one can only assume was a hot box created where the players rotated through before tagging the runner. This issue would have been flagged as the resulting CSV file only has 5 columns for assists.

# MapReduce Algorithm/Architecture

The MapReduce architecture used consists of two MapReduce jobs. The first job is responsible for extracting data that could contribute to a statistical streak, and then analyzing the data for a streak. The second mapper sorts the streaks to obtain the top ones. A temporary directory is created to hold data between the two jobs. The directory is automatically deleted once completed.



## Figure - MapReduce High Level Architecture

## Find Streaks Job - Mapper (EventMapper.java)

Each record is read in (as mentioned earlier, the files are formatted as such that all events that took place in a American League park are in one file, all events that took place in a National League park are in a different file. As interleague play is possible in the years since 1997, the events need to be checked against both files for continuity of streaks, and to handle player moves). As each record is read, its confirmed to be the actual plate appearance. If it is the plate appearance, it will check for different types of events (Walks, Strikeouts, Hits, Extra Base Hits, Hit into Double Play, etc). Currently this only tracks a batters statistics, but a simple extension could include pitching, fielding or base running data. Any noted event is sent to the mapper, as well as each plate appearance. This is important to send, as if no plate appearance in a game for a given player recorded a particular event occurring, that would signal the end of the streak. The data is sent to the reducer as a composite key. The next paragraph details the composite key. The value sent to the reducer is a simple text key of the statistical event (Strikeout, Walk, Hit, etc).

## Find Streaks Job - Composite Key (gameEventWritable.java)

To align with how the Reducer is setup, all of the records sent to the reducer need to be processed in sequential order. To accomplish this, and to assist in maintenance, a composite key was used. The composite key class consists of the Game Identifier, Player Id, Inning, and Score (Although currently unused, a unique value would need to be included to handle situations of batting around the order, the score would be sufficient to distinguish one plate appearance from another in the same inning). To ensure the records are sorted correctly for the reducer to find a streak, a custom compareTo() was defined to sort by: Player Id, then Game Id, then Inning, then Score.

## Find Streaks Job - Partitioner (gameEventPartitioner.java)

To align with how the Reducer is setup, either all records would need to go to a single reducer, or records would need to be dispersed in such a way that all of the records for a given player were at the same reducer. To provide scalability, and usage of available resources, a partitioner was created to hash the Player Id field of the composite key output by the Mapper. The hash is modded by the number of reduce tasks, and helps to provide a more even distribution of the workload.

## Find Streaks Job - Reducer (StreakReducer.java)

The reducer of the streaks operates on the basis that all records arrive sorted by player, then game event. Each record compares the current player against the last player. If there is a difference, all of the actively tracked streaks need to be flushed as there will be no more records for the previous player. Then, each record compares the current game event to the last game event. If there is a difference, the previous game is no more, and each actively tracked streak needs to check for continuity. If the previous game does not match the last recorded event for the streak, that’s the flag to indicate the streak is over, and should be recorded. After establishing the player/game combination, it’s a matter of walking through each record for the combination, and updating any streaks. This leaves the last record processed by each reducer at risk of not being included/reported. To combat this, the cleanup() method was populated to flush any streaks that were active at the end of record processing.

The reducer was structured in such a way it uses the Value as the tracker. Any type of event the mapper can extract and place into the Value, the reducer can handle. This provides some flexibility for future growth in the reducer could be left as is, while the mapper could be extended to handle other items.

Once the mapper outputs a streak, its output as a Text key, and a NullWritable Value. The text key consists of <Event Name> : <Count of occurrences, 0 padded to 5 digits> : <Start Date - Finish Date> : Player Id. It should also be noted, that more than one occurrence had to be tracked for event to be output.

## Sort Streaks Job - Mapper (SortMapper.java)

The sort streaks mapper is fairly straightforward. It is configured to take the incoming data as the TextInputFormat class. This is fitting as the Find Streaks Job Reducer output data as a Key only. As an alternative, a custom class could have been output into a sequence file, and processed here. As the data was viewed to be straightforward, and not highly susceptible to change, it was left as just Text. The net effect of the mapper is to take the values passed in, and move them to the Key, with the Value being NullWritable.

## Sort Streaks Job - Partitioner (SortPartitioner.java)

To distribute the workload of outputting the streaks, a partitioner was employed to extract and hash the Event Name field, causing all of same events to be sent to the same partitioner. The necessity of using this partitioner vs., using a single reducer is debatable. There are currently about 1.3 million records processed by this job, and with 48 reducers, the reduce portion of the job appears to take less than 2 seconds. It should also be noted that even though there are 48 reducers available, the use of the partitioner on 14 different types of events, drives a maximum of 14 reducers to be used, largely defeating the purpose of using reducers as most are left to idle. If more stats were to be tracked, this may become more important, but at the current time, this is probably excessive for the amount of work accomplished.

## Sort Streaks Job - Custom Text Comparator (TextComparatorInverted.java)

The way the Reducer was structured was to operate on the basis that the longest streak should be the first record. The default sort will not accomplish this as its in ascending order (recall from the Find Streaks Job - Reducer, the key starts with <Event Name>:<Streak length 0 padded>…). To get the sort, a custom text comparator was created. This comparator simply runs the default Text comparator, but inverts the result by multiplying it by -1. As the streak length was 0 padded, this works out. If it weren’t, either a composite key would need to be used, or the custom comparator could be extended to extract the integer value and sort by that.

## Sort Streaks Job - Reducer (SortReducer.java)

With records coming into the Reducer sorted so the longest streaks are on the top, the reducer just needs to walk through the received values, and output the largest ones. To handle this, local variables are used to track the last type of event that was received, and how many of that event were output. For each incoming record, the event type is checked against the last processed event type. If they are the same, then check how many times the event has been output, and if the last value output is the same as the current value. The check for how many times acts as a “top” type of statement, while the check of the last value allows a streak like: 10, 9, 8, 8, 8… . If the top 3 were requested, one of the 8 values would be output, but the others would not be output.

### Use of Distributed Cache

As the records at the reducer still use the Player ID provided by Retrosheet, this isn’t ideal for a list intended to be human readable as the user would either have to guess based on the first four characters of the players lastname, first letter of their first name, and if more than one has ever played or coached, the index based on who played first with the same designation.

The mapping is trivial, and with it just being Player Id->Player Name, it’s a good fit for the map data structure. As the only point these values matter is the final output, the Sort Reducer is the ideal place to map these names in as not nearly as many lookups would need to be made.

After a great deal of frustration, I was able to get this feature working, although the implementation is less than ideal. The method I used to get it to work was to manually load the file to hdfs. Then add an addCacheFile call to the driver to code to specify the filename/path. Then, in the setup command of the Reducer, load the file (assuming it was the one and only file put into the distributed cache) and populate the map.

After reviewing the book, and the web, I feel this area may still be lacking in the Hadoop structure as there seems to be a heavy dependency on filename/path used in the code lining up with the filename/path used when the file was added to HDFS. The book did suggest using the “-files” parameter from the command line. I was unable to get this to work, and wasn’t 100% sure why. I suspect the job may not have been setup the way the book was suggesting, but everything I read was less than clear, and the single example left a fair amount up in the air.

# 3rd Party Libraries / Tools

The resulting CSV files from the BEVENT utility included a double quote qualifier on each non-numeric field. After pondering this, I decided a regular expression wasn’t going to handle this robustly, so I searched the web, and found opencsv which is an open source CSV library that can handle such qualifiers (http://opencsv.sourceforge.net/).

For development, I used the ClouderaVM, and the standard eclipse installation in that environment.

# Output Description

The resulting output is a series of text files. As it works out now, each type of event is in a different text file, but this is highly dependent on the number of reducers in use, as well as how many events are being tracked (the reducer an event type is sent to is determined by hashing the event type. A change in the number of reducers, or addition of events could cause event type hashes to overlap). A sample of the output for hitting streaks:

Hit 00056 : 19410515-19410716:Joe DiMaggio

Hit 00044 : 19780614-19780731:Pete Rose

Hit 00039 : 19870716-19870825:Paul Molitor

Hit 00038 : 20050823-20060405:Jimmy Rollins

Hit 00035 : 20060623-20060803:Chase Utley

Hit 00035 : 20020508-20020621:Luis Castillo

Hit 00034 : 19870825-19871002:Benito Santiago

Hit 00033 : 20110705-20110813:Dan Uggla

Hit 00031 : 19990727-19990826:Vladimir Guerrero

Hit 00031 : 19750927-19760527:Ron LeFlore

Hit 00031 : 19700408-19700515:Rico Carty

Hit 00031 : 19690801-19690903:Willie Davis

Hit 00031 : 19650903-19660418:Vada Pinson

Hit 00031 : 19490629-19490807:Dom DiMaggio

Hit 00030 : 20110402-20110506:Andre Ethier

Hit 00030 : 20090408-20090512:Ryan Zimmerman

Hit 00030 : 20060727-20060827:Willy Taveras

Hit 00030 : 19990411-19990518:Luis Gonzalez

Hit 00030 : 19980712-19980815:Eric Davis

Hit 00030 : 19960827-19980331:Hal Morris

Hit 00030 : 19890721-19890820:Jerome Walton

Hit 00030 : 19800718-19800818:George Brett

# Verification of Output

Where possible, streaks were compared against a Wikipedia page that has compiled various streaks (<http://en.wikipedia.org/wiki/List_of_Major_League_Baseball_individual_streaks>). In particular, the following were reviewed:

* Consecutive games with a hit (56): Joe DiMaggio
* Consecutive games hitting a home run (8): Don Mattingly/Ken Griffey (Note: Dale Long was omitted from the output. As noted above, records prior to 1974 are not complete. After reviewing the list of missing games, several of the games that occurred during Dale Long’s streak are on Retrosheet’s “Most Wanted Games”).
* Consecutive games reaching base (78): Ted Williams (Note: Official record is 84, however multiple games are missing in the date range this occurred).
* Extra Base Hit (14): Chipper Jones (Note: Paul Wagner is also credited with this feat, however it occurred in 1927, and current data available does not cover that year).

Another source was used for reviewing consecutive games with a strikeout (http://www.57hits.com/top-ten-longest-strikeout-streaks). Surprisingly, it should be noted, that since this blog posting two and a half seasons ago, there are two new entrants to the top five.

Streaks could also be analyzed by loading the data into Excel, and manually looking through the rows in question to confirm any findings.

# Performance/Scale Characteristics

When run against the St Thomas cluster, this was a relatively quick job completing in just under a minute. Even if more statistics were to be tracked, the Find Streaks Mapper portion of the job should continue to run in about the same time as the same amount of data is processed, just some additional string comparisons would be added to check for other events occurring. This would drive up work at the Find Streaks Reducer portion of the job as it would need to be run that many more times, and would surely drive up the level of work at the Sort job as more items will need to be processed. However, I would not expect any addition of work to significantly drive up the time it takes to process. Also given the data source, it will very slowly grow over time (approximately 80 MB/year at this time), so future implementations should be predictable.

# Lessons Learned

* Start Earlier
  + Not jumping on this project immediately didn’t provide adequate “think time”, to sort out various issues ahead of time, forcing me to go into panic mode when things didn’t work right.
* Google Is NOT Your Friend For Hadoop
  + As pointed out in lecture, most of the Hadoop content in Google is aimed at the older API, and is misleading or just outright wrong for the new API. Unfortunately there are those cases where the book doesn’t cover items in enough detail, and there aren’t a lot of places to turn…
* Git Is Your Friend
  + Having source control at my fingertips was invaluable as it was a place I knew I could put anything, and get it back if something went wrong. In the past it happened far too often where I was hesitant to make a change for feat I’d make things worse, and not even be able to get back to where I was. With Git/GitHub, this isn’t a concern.
* Pick A Better Topic
  + Although I don’t think this was the worst topic to cover, this isn’t anything new. In retrospect, with access to the data I used, I could have picked something that was unexplored, or at least something that was always an open question in my mind. However, this exercise did force me to implement several of the items covered in class, which is probably more important than the actual outcome of the project.
* Excel Doesn’t Play Well With Big Data
  + When it comes to pasting in 400K records or more, Excel can become completely unresponsive, or throw an exception and quit. Found the best way to bring in large pieces of data was to compile them in a text editor, then paste them in as one large operation rather than several smaller pastes. But even after that, fitting data, and just trying to browse through it just doesn’t work…

# Conclusions

Hadoop can be used very effectively to track streaks/sequences of events when setup properly. The setup does require a fair number of things to be setup to work with Hadoop, however I didn’t not find them particularly troubling to setup, nor cumbersome to manage with the knowledge I have from class.

Overall, this project was a good method to apply course material:

* Two jobs were run back to back to successfully get to the end result.
* Data input was in the several GB range to demonstrate scalability.
* Partitioners were successfully used to avoid running a single reducer to help with scalability issues.
* Composite Key’s were used to use Hadoop’s built in sort, in addition to implementing a software engineering best practice of loosely coupled data.
* The Distributed Cache was successfully used to provide lookups of static data.

Running Map Reduce Job

1. Load Data Input Files (.EVAC/.EVNC) into hdfs. For the purposes of this explanation, the files will be placed in the hdfs directory “RetroData”.
2. Load the Roster file into hdfs. File must be named “\_roster.txt”, and should be placed in the hdfs root directory.
3. Run the job. Example command line input:

[ande8331@hc ~]$ hadoop jar finalProject.jar StreakFinder RetroData Streaks

1. Wait for job to complete.
2. Once job has completed, the records found can be output (if more records were to be tracked, a different means would be suggested as this could be resource/time intensive for this call):

[ande8331@hc ~]$ hadoop fs -cat Streaks/\*

Output From Map Reduce Job

[ande8331@hc ~]$ hadoop fs -cat Streaks/\*

cat: `Streaks/\_logs': Is a directory

Intentional Walk 00006 : 20040919-20040925:Barry Bonds

Intentional Walk 00005 : 20040813-20040817:Barry Bonds

Intentional Walk 00005 : 20040603-20040607:Barry Bonds

Intentional Walk 00005 : 20020914-20020918:Barry Bonds

Intentional Walk 00005 : 19880702-19880706:Will Clark

Intentional Walk 00005 : 19870905-19870909:Tim Raines

Extra Base Hit 00014 : 20060626-20060716:Chipper Jones

Extra Base Hit 00011 : 20060927-20070411:Alex Rodriguez

Extra Base Hit 00011 : 20050507-20050518:Bobby Abreu

Extra Base Hit 00011 : 19850817-19850827:Jesse Barfield

Extra Base Hit 00010 : 20110720-20110730:Justin Upton

Extra Base Hit 00010 : 20100823-20100902:Carlos Gonzalez

Extra Base Hit 00010 : 20090504-20090514:Johnny Damon

Extra Base Hit 00010 : 20010722-20010801:Paul O'Neill

Extra Base Hit 00010 : 20010712-20010721:Cliff Floyd

Extra Base Hit 00010 : 20000903-20000912:Richard Hidalgo

Extra Base Hit 00010 : 19930719-19930729:Ken Griffey

Extra Base Hit 00010 : 19870707-19870719:Don Mattingly

Extra Base Hit 00010 : 19550913-19560417:Willie Mays

Extra Base Hit 00010 : 19540523-19540603:Willie Mays

Extra Base Hit 00010 : 19480722-19480806:Stan Musial

Single 00028 : 20060727-20060825:Willy Taveras

Single 00028 : 19690802-19690901:Willie Davis

Single 00026 : 19980711-19980807:Jose Offerman

Single 00024 : 20070508-20070601:Ichiro Suzuki

Single 00023 : 20020508-20020605:Luis Castillo

Single 00023 : 19910615-19910712:Brett Butler

Single 00023 : 19790903-19790925:Pete Rose

Walk 00020 : 20020909-20030401:Barry Bonds

Walk 00017 : 20070504-20070525:Barry Bonds

Walk 00017 : 20030416-20030503:Nick Johnson

Walk 00017 : 19800918-19810411:Willie Randolph

Walk 00016 : 20040620-20040707:Barry Bonds

Walk 00016 : 19990819-19990905:Chipper Jones

Walk 00016 : 19870718-19870810:Jack Clark

Walk 00016 : 19850420-19850511:Toby Harrah

Hit By Pitch 00006 : 20080807-20080814:Carlos Quentin

Hit By Pitch 00004 : 20120901-20120904:Jon Jay

Hit By Pitch 00004 : 20060913-20060918:Shawn Green

Hit By Pitch 00004 : 19760923-19760927:Don Baylor

Hit By Pitch 00004 : 19750518-19750521:Richie Hebner

Hit By Pitch 00004 : 19680927-19690407:Ron Hunt

Hit By Pitch 00004 : 19610528-19610601:Joe Cunningham

Bunt 00012 : 20120506-20120518:Tony Campana

Bunt 00011 : 20040523-20040604:Alex Sanchez

Bunt 00010 : 19880824-19890410:Orel Hershiser

Bunt 00008 : 20050415-20050425:Nook Logan

Bunt 00008 : 20030524-20030601:Ramon Santiago

Bunt 00008 : 19980906-19980913:Neifi Perez

Bunt 00008 : 19930615-19930726:Wally Whitehurst

Bunt 00008 : 19900824-19900928:Dwight Gooden

Bunt 00008 : 19800519-19800703:Burt Hooton

Home Run 00008 : 19930720-19930728:Ken Griffey

Home Run 00008 : 19870708-19870718:Don Mattingly

Home Run 00007 : 20060421-20060428:Kevin Mench

Home Run 00007 : 20020625-20020703:Jim Thome

Home Run 00006 : 20120926-20121002:Chris Davis

Home Run 00006 : 20100606-20100612:Carlos Pena

Home Run 00006 : 20060905-20060911:Frank Thomas

Home Run 00006 : 20060522-20060528:Jason Bay

Home Run 00006 : 20060415-20060421:Morgan Ensberg

Home Run 00006 : 20050918-20050923:Travis Hafner

Home Run 00006 : 20010929-20011005:Jose Cruz

Home Run 00006 : 20010517-20010522:Barry Bonds

Home Run 00006 : 20010412-20010418:Barry Bonds

Home Run 00006 : 19840811-19840822:Graig Nettles

Home Run 00006 : 19760718-19760723:Reggie Jackson

Home Run 00006 : 19680512-19680518:Frank Howard

Home Run 00006 : 19650816-19650822:Willie Mays

Home Run 00006 : 19610811-19610816:Roger Maris

Home Run 00006 : 19570728-19570803:Roy Sievers

Home Run 00006 : 19470807-19470816:Ralph Kiner

Home Run 00006 : 19440604-19440612:Phil Weintraub

Home Run 00006 : 19310828-19310901:Lou Gehrig

Reached Base 00078 : 19490701-19490927:Ted Williams

Reached Base 00074 : 19410514-19410802:Joe DiMaggio

Reached Base 00063 : 20060425-20060706:Orlando Cabrera

Reached Base 00062 : 19950916-19960618:Mark McGwire

Reached Base 00060 : 20020728-20030405:Jim Thome

Hit Into Double Play 00006 : 20080406-20080412:David Ortiz

Hit Into Double Play 00006 : 19560528-19560609:Hal Smith

Hit Into Double Play 00005 : 20100629-20100703:Kevin Kouzmanoff

Hit Into Double Play 00005 : 20100506-20100510:Wilson Valdez

Hit Into Double Play 00005 : 20091004-20100409:Pablo Sandoval

Hit Into Double Play 00005 : 20070506-20070515:Jeff Cirillo

Hit Into Double Play 00005 : 20060906-20060912:Mark DeRosa

Hit Into Double Play 00005 : 20060620-20060625:Kendrys Morales

Hit Into Double Play 00005 : 20040728-20040803:Mike Lowell

Hit Into Double Play 00005 : 20030909-20030913:Angel Berroa

Hit Into Double Play 00005 : 20030712-20030719:Carlos Lee

Hit Into Double Play 00005 : 19980915-19980920:Greg Norton

Hit Into Double Play 00005 : 19980604-19980608:Nomar Garciaparra

Hit Into Double Play 00005 : 19920730-19920803:Tom Pagnozzi

Hit Into Double Play 00005 : 19870724-19870729:Bill Doran

Hit Into Double Play 00005 : 19850808-19850813:Butch Wynegar

Hit Into Double Play 00005 : 19850430-19850505:Butch Wynegar

Hit Into Double Play 00005 : 19820814-19820820:Ken Singleton

Hit Into Double Play 00005 : 19820702-19820705:Jim Rice

Hit Into Double Play 00005 : 19770905-19770907:Rusty Staub

Hit Into Double Play 00005 : 19730908-19730916:Joe Lis

Hit Into Double Play 00005 : 19660424-19660503:Jerry Grote

Hit Into Double Play 00005 : 19610903-19610906:Roger Maris

Hit Into Double Play 00005 : 19500912-19500917:Sam Mele

Hit Into Double Play 00005 : 19490725-19490730:Joe Gordon

Hit Into Double Play 00005 : 19310727-19310803:Dib Williams

Double 00008 : 20070423-20070502:Derrek Lee

Double 00007 : 20130604-20130612:Yadier Molina

Double 00007 : 20040809-20040815:Brian Roberts

Double 00007 : 20000722-20000729:Carlos Delgado

Double 00007 : 19990422-19990429:Jeff Kent

Double 00007 : 19980819-19980826:Gary Disarcina

Double 00007 : 19980422-19980502:Todd Walker

Double 00007 : 19860508-19860514:Jim Presley

Hit 00056 : 19410515-19410716:Joe DiMaggio

Hit 00044 : 19780614-19780731:Pete Rose

Hit 00039 : 19870716-19870825:Paul Molitor

Hit 00038 : 20050823-20060405:Jimmy Rollins

Hit 00035 : 20060623-20060803:Chase Utley

Hit 00035 : 20020508-20020621:Luis Castillo

Pinch Hitter 00207 : 19650615-19671001:Smoky Burgess

Pinch Hitter 00134 : 19600721-19611001:Elmer Valo

Pinch Hitter 00122 : 19770722-19820901:Manny Mota

Pinch Hitter 00115 : 19600419-19610824:Bob Hale

Pinch Hitter 00110 : 19830624-19840731:Rusty Staub

RBI 00015 : 20000614-20000702:Mike Piazza

RBI 00014 : 20090929-20100415:Jorge Cantu

RBI 00013 : 19990623-19990704:Mike Sweeney

RBI 00013 : 19310708-19310723:Al Simmons

RBI 00013 : 19310615-19310702:Babe Ruth

Strikeout 00037 : 19710430-19720421:Bill Stoneman

Strikeout 00036 : 20110924-20120509:Adam Dunn

Strikeout 00035 : 19670721-19680818:Bob Veale

Strikeout 00034 : 19710409-19710912:Vida Blue

Strikeout 00031 : 20120902-20130415:Chris Carter

Triple 00004 : 20030531-20030604:Nomar Garciaparra

Triple 00004 : 19940514-19940517:Sammy Sosa

Triple 00004 : 19810420-19810425:Tom Herr

Triple 00004 : 19630508-19630511:Billy Williams

Triple 00004 : 19490427-19490501:Dale Mitchell